A repetition-based framework for lyric alignment in popular songs

LUONG Minh Thang[*]and KAN Min Yen[†]
Department of Computer Science,
School of Computing,
National University of Singapore

ABSTRACT

We examine the problem of automatically aligning acoustic musical audio and textual lyric in popular songs. Existing works have tackled the problem using computationally-expensive audio processing techniques, resulting in solutions unsuitable for any real-time application. In contrast, our work features only lightweight signal processing and is capable of real-time alignment. We investigate in repetition-based techniques and alignment algorithms to obtain a baseline alignment. A key extension of our work is to derive and utilize additional segmentation knowledge on both modalities to significantly enhance alignment performance by 34.85% and 8.18% in start and duration time errors. We conclude by suggesting a new repetition-based framework for lyric alignment together with a modular system design, where each module is independent and feasibly-extendable to improve the overall performance.

1. INTRODUCTION

In this project, we tackle the task of aligning lyric and audio automatically in popular songs. Specifically, given the textual transcription of lyrics and the acoustic musical signal of a song, we seek to find the time stamps corresponding to the beginning and ending points of each line in that song. Solutions to the problem have been proposed by two groups of works: one deal with synthesized music (MIDI files, or no-background music) such as (Hu, Dannenberg, & Tzanetakis, 2003) and (Turetsky & Ellis, 2003); the other, close to ours in dealing with real-world music, employs audio processing techniques. In the latter group, LyricAlly (Wang, Kan, Nwe, Shenoy, & Yin, 2004) is the first system that performs automatic alignment in popular songs, and followed by subsequent works such as (Iskandar, Wang, Kan, & Li, 2006), or (Wong, Chi, Szeto, Wai, Wong, & Kin, 2007). These works, however, are inefficient in two aspects: either dealing with restricted audio and/or employing intense audio processing techniques. Realizing that inefficiency, we look at a different approach in handling real-world audio using repetition-based techniques such as self-similarity matrix (Foote, 1999), repetitive pattern, and alignment algorithms. We follow the use of chroma vector to represent music segment as in (Wang et al., 2004), and only employs audio processing under the form of chroma computation.

The structure of this report is organized as follows. Section 2 gives an overview of the system, and describes our methodology. Section 3 evaluates our techniques against a standard dataset and discusses the outcomes of the experiments at both a macro- and micro-level. We offer some concluding remarks in section 4.

2. METHODOLOGY

Figure 1 displays an overview of our system consisting of six modules: feature extractor, self-similarity matrix (SSM) generator, repetition pattern generator, automatic aligner, and

[*]Student
[†]Supervisor

segment generator. *Feature extractor* module decomposes the acoustic and textual input of stream form into sequences of chroma vectors and words. Chroma-vector extraction is obtained by employing the two components in LyricAlly (beat detection, and chroma-based feature extraction); where as, for textual input, words are obtained using simple tokenization operation. Detailed discussions on other modules are presented in later sections.

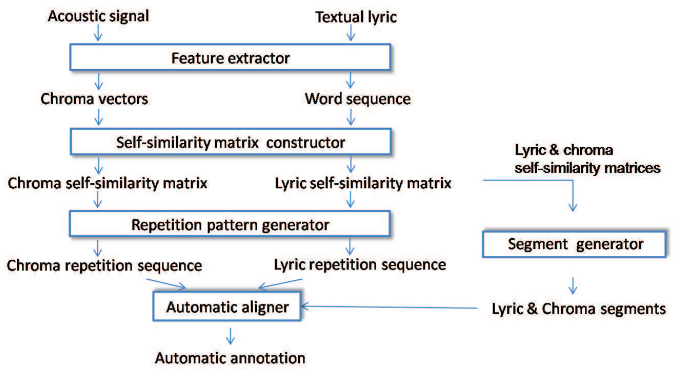
Acoustic signal
Textual lyric
Feature extractor
Chroma vectors
Word sequence
Self-similarity matrix constructor
Chroma self-similarity matrix
Lyric self-similarity matrix
Lyric & chroma self-similarity matrices
Repetition pattern generator
Segment generator
Chroma repetition sequence
Lyric repetition sequence
Automatic aligner
Lyric & Chroma segments
Automatic annotation

Figure 1. System overview

2.1. Self-Similarity matrix (SSM) construction

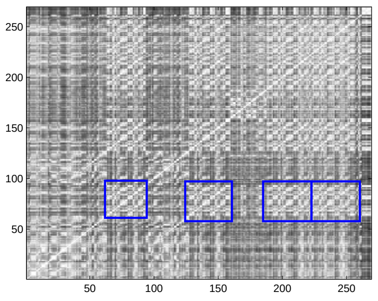

Figure 2. Chroma self-similarity matrix

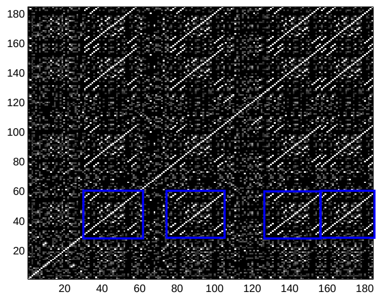

Figure 3. Lyric self-similarity matrix

We compute the similarity between two normalized chroma vectors using the *Euclidean distance*. For lyric, we decompose each word into a phoneme sequence, and allow partial similarity among words by compute the *longest common subsequence*. Figures 2 and 3 give an example of chroma and lyric SSMs for the same song "Paint my love". The lyric similarity exhibits clear pattern of diagonal lines of repeated sections, while that of chroma is richer in capturing the similarity information of audio. For example, six separate squares lie on the main diagonal line of the matrix indicates six different sections intro, chorus, verse, chorus, verse, and coda, and many small white squares within the same section show the intra-repetition. The four highlighted squares on both figures indicate the repetition of four chorus sections, and show the good correspondence between lyric and chroma SSMs.

2.2. Repetition pattern generation

The motivation in this module is to distinguish each unit in a sequence based on its occurrences together with other units around. For example, to differentiate the word "you" in the phrase "You should paint my love" with other "you", we count how many times the following subsequences are repeated "you", "you should", ..., "you should paint my love". The counts are averaged to give a *repetitive value* for that "you", and similarly for other words. The repetitive value of each word is used to distinguish itself. A key step in this module is to determine if two subsequences of the same length $L[i \ldots (i + k)]$, and $L[j \ldots (j + k)]$ are repeated. To decide that, we compute the similarity of the two sequence by summing up all values on the diagonal line of the SSM from (i, j) to (i +k, j+k). That sum divided by the subsequence length is compared against a *repetition threshold* to decide if the subsequences match. Experimentally, we use a threshold of 0.7 for chroma, and 0.9 for lyric. Figures 4 and 5 are examples of lyric and chroma repetition plots for the song "Paint my love". We could observe the corresponding four peaks on both figures, which corresponds to the four repeats of the chorus sections in the song, and shows the evidence of a comparable basis between lyric and chroma.

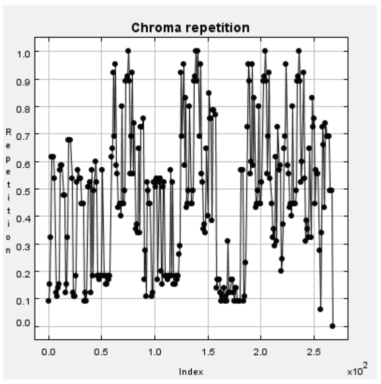

Figure 4. Chroma repetition plot

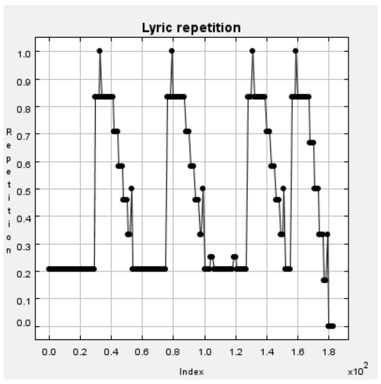

Figure 5. Lyric repetition plot

2.3. Lyric alignment

We employ the global alignment algorithm mentioned in (Gusfield, 1997) to align lyric and chroma repetition sequences, whose normalized values are denoted as L[1..n] and A[1..m]. The algorithm aligns the two sequences using *dynamic programming* technique (DP) by gradually compute V(i, j), the optimal aligning value of L[1..i] and A[1..j]. As scoring values and recurrence relations are indispensable in any alignment problem, we define the *matching score* s(L[i], A[j]) to be $1 - |L(i) - A(j)|$. We expect many skips on chroma side while no skip on lyric because of non-vocal sections in audio, so we set a *skip penalty* of 0 for chroma $s(\_, A[j])$, and -1 for lyric $s(L[i], \_)$. Recurrence formula is given in equation 1. As each chroma vector is associated with a time stamp recorded during chroma extraction process, the automatic annotation could be easily derived for each lyric line once alignment is obtained, by taking the time stamps of the beginning and ending words in that line.

As gaps (non-vocal sections) are often present in audio, the alignment algorithm encounters a challenge in identifying gap segments to skip over chroma vectors. In order to tackle that problem, we propose to use segmentation knowledge over lyric and chroma, which is provided by our segmentation module describe later. Based on the information, lyric and chroma sequences are divided into k segments each, and the alignment algorithm is used in aligning k pairs of segments to derive the final alignment.

$$V(i,j) = max \begin{cases} V(i-1,j-1) + s(L[i], A[j]) & \text{if i > 0, j > 0} \\ V(i-1,j) + s(L[i], \_) & \text{if i > 0} \\ V(i,j-1) + s(\_, A[j]) & \text{if j > 0} \end{cases} \qquad (1)$$

2.4. Segmentation

In this section, we will be using *local alignment* as an extension from the global alignment to find the two best aligned subsequences $L[n_1 \ldots n_2]$ and $A[m_1 \ldots m_2]$ instead of aligning the two sequences entirely (Gusfield, 1997). Similar DP formula for local alignment is obtained by adding a comparison to 0 into Eqn 1, which is the"heart" of the local alignment.

2.4.1. Segmentation using non-overlapping local alignment algorithm

We want to find the two most repeated segments in the sequence by self-aligning that sequence using local alignment with a constraint on the starting row of an alignment. Particularly, we restrict the best alignment path ending at (i, j) to have starting row of k where $j \leq k \leq i$. The modified DP formula is very similar to equation 1 except that another dimension k is added to V(i, j) together with conditions on i, j, and k (Kannan & Myers, 1993). As always, the crucial step in an alignment problem is to decide *matching score* s(x, y). In this problem, we decide $s(S[i], S[j])$ based on the SSM, and follow a general guideline to discourage two less-similar elements from matching by assigning negative score. As such, we have two choices of *scoring formula* for $s(S[i], S[j])$'s as below. Equation 2, "range shift", is to change the range of similarity value from [0, 1] to [-0.5, 0.5], while equation 3, "thresholding", is to filter low-similarity values with *mismatch penalty* θ. In our experiments, we assign θ to -1, which performs reasonable.

$$s(S[i], S[j]) = SMM[i][j] - 0.5 \quad (2) \qquad s(S[i], S[j]) = \begin{cases} SMM[i][j] & \text{if SMM[i][j]} > \delta \\ \theta & \text{if SMM[i][j]} \leq \delta \end{cases} \quad (3)$$

2.4.2. Refining segments using local alignment algorithm

From the two non-overlapped segments detected previously, we wish to detect other repeated segments, and have an efficient way so that those segments could naturally refine their boundaries. The idea is to use local alignment algorithm to compare between detected and undetected segments to discover other repeated segment as well as refine the boundaries:

Let S be the initial set of segments. Iteratively:

Step 1: Take the current section set $S = S_1, .., S_k$. These segments will result in unsegmented sections $U_1, .., U_t$ in the sequence.

Step 2: Pairwise locally align(S_i, U_j) which gives two best locally-aligned segments S'_1, S'_2 to be added to S. If a new segment has a not-too-small size, and does not approximately equal any segment in S, it is added into S.

Step 3: remove redundant segments by first computing average segment size across all segments in S. Pairwise compare two segments in S. If they are overlapped, compare their sizes with the averaged one. Which one closer is kept, the other is removed.

3. EVALUATION

Our data set consists of 34 popular songs,taken from the LyricAlly project (Wang et al., 2004), which are in verse-chorus form and have 4/4 time. Each song is accompanied with a rich manually annotated alignment at a per-line as well as per-section levels.

3.1. Segmentation analysis

We focus our analysis toward *verse* and *chorus* sections which are main sections in general song structures. Observation reveals that verse section behaves differently in terms of repetition on lyric and chroma sequences. Particularly, *verse section on lyrics* is not necessarily repetitive. Some writers prefer to have their verses repeated with some minor variations, while others like to have markedly different verses for more story-telling purposes. In contrast, *verse section on chroma* does exhibit repetitive pattern. Even though the actual wordings are not the same, similar chord sequences are repeated at verse sections, and well-captured by chroma vectors. The interesting behavior of verse section results in *chorus* and *compound verse-chorus* sections being the longest repetitive sections on lyric and chroma sequences respectively. We refer to *chorus* and *compound verse-chorus* as *target sections* in our lyric and chroma analysis respectively.

	C	**Recall**
No iteration	27	39.71%
1 iteration	55	80.88%
2 iterations	59	86.76%

Table 1. Structural analysis on lyrics

	C	**Recall**
No conversion	35	51.47%
Range shift	48	70.56%
Threshold	50	73.53%

Table 2. Structural analysis on chroma

Tables 1 and 2 present our *structural measure* on lyric and chroma in which we evaluate how many detected sections (C) match the ideal target sections with matching portion greater than 70%. For lyric, non-overlapping alignment alone only obtain a recall of 39.71%. However, when refinement algorithm is executed several times, the system obtain a maximal significant recall of 86.76% after 2 iterations. For chroma, we present the analysis in another aspect, the choice of scoring formula mentioned in section 2.4.1. Table 2 has shown that the performance could be improved from 51.47% to 69.12% and 73.53% with “range shift” and “thresholding” (at 0.7) formula. This analysis is to demonstrate the important of scoring schemes in alignment.

3.2. Alignment analysis

	Start (sec)	Duration (sec)	Start (bar)	Duration (bar)
No segmentation	N(12.045, 9.62^2)	N(2.082, 0.91^2)	N(17.545, 14.03^2)	N(3.020, 1.33^2)
Automatic segmentation	N(7.399, 4.18^2)	N(1.897, 0.83^2)	N(11.430, 8.89^2)	N(2.773, 1.26^2)
Perfect segmentation	N(1.863, 1.00^2)	N(1.258, 0.43^2)	N(2.675, 1.36^2)	N(1.831, 0.70^2)

Table 3. Alignment analysis in normal distribution(sec and bar units)

We test our alignment module under three schemes: no segmentation, automatic segmentation, and perfect segmentation. The first scheme only invokes the alignment algorithm, while the others incorporate segmentation knowledge. For the *automatic segmentation* scheme, we divide songs into simple segments based on the chorus segments detected on lyric, as well as verse-chorus segments on chroma. As each song is in verse-chorus form, which implies a general structure of $-V_1C_1V_2C_2-$, we divide them into three segments at two anchor points, which are the end points of C_1 and C_2. Corresponding parts in lyric and chroma are aligned to derive the final annotation. For the last scheme *perfect segmentation*, we test our system performance using the perfect segmentation from our manual annotation, consisting of verse, chorus, and coda segments. Table 3 presents a significant performance improvement by 34.85% and 8.18% in start and duration time error when segmentation knowledge is utilized, as compared to the baseline. Moreover, with perfect segmentation, the system errors are kept effectively small at 2.675 and 1.831 bar for start and duration time errors. This gives evidence that our repetition-based method is potentially capable of accomplishing the alignment task.

4. CONCLUSION AND FUTURE EXTENSIONS

We have shown that our repetition-based approach is feasible to achieve the solution with lightweight signal processing techniques. Even though not as accurate as the signal processing approach, we have made a main contribution in proposing a potential direction for lyric alignments with our well-defined system design. We end our paper by suggesting future extensions at module level for our system, which we belief critical and feasible to extend.

For *SSM construction*, an approach for a *newly-defined similarity matrix* could be adopted to extract more information from the SSM. A SSM of size 600×600 could be reduced to size 75×75 by grouping each N = 8 chroma vectors, called a measure, and the similarity between each pair of measures is computed as a sum of N cosine products. By varying N, we could capture similarity patterns both among sections, and within section. For *repetition module*, when comparing two subsequences $L[i \dots (i+k)]$, and $L[j \dots (j+k)]$, instead of considering only diagonal elements, we could utilize our global alignment to find the best path going from point (i, j) to (i+k, j+k) in the SSM, and decide matching based on the alignment value. This suggestion will add more flexibility in allowing minor error to be skipped for better alignment, and surely compute better repetition values. Lastly, for the *alignment module*, currently, our system only constrains on the alignment at both ends to account for non-vocal starting and ending sections in audio. However, more constraints could be further imposed by providing upper and lower boundaries on the region where the alignment path could go through, e.g. not allowing alignment path to cross the main diagonal towards lyric and skips rarely occur on lyric side. Such constrains both globally and locally are well-studied in (Iskandar et al., 2006) using musical knowledge.

5. ACKNOWLEDGEMENT

We would to extend our special appreciation to Dr. Kan Min Yen and Mr. Hendra Setiawan for their indispensable guidance on this project and many valuable comments on the paper.

6. REFERENCES

[1] Foote, J. (1999). Visualizing music and audio using self-similarity. *MULTIMEDIA '99: Proceedings of the seventh ACM international conference on Multimedia (Part 1)*, 1999.

[2] Gusfield, D. (1997). *Algorithms on strings, trees, and sequences: computer science and computational biology.* New York, NY, USA: Cambridge University Press.

[3] Hu, N., Dannenberg, R. B., & Tzanetakis, G. (2003). Polyphonic audio matching and alignment for music retrieval.

[4] Iskandar, D., Wang, Y., Kan, M.-Y., & Li, H. (2006). Syllabic level automatic synchronization of music signals and text lyrics. *MULTIMEDIA '06: Proceedings of the 14th annual ACM international conference on Multimedia* (pp. 659–662), New York, NY, USA, 2006: ACM.

[5] Kannan, S. K., & Myers, E. W. (1993). An algorithm for locating non-overlapping regions of maximum alignment score. *Proceedings of the 4th Annual Symposium on Combinatorial Pattern Matching*, no. 684 (pp. 74–86), 1993.

[6] Turetsky, R., & Ellis, D. (2003). Ground-truth transcriptions of real music from force-aligned midi syntheses.

[7] Wang, Y., Kan, M.-Y., Nwe, T. L., Shenoy, A., & Yin, J. (2004). Lyrically: automatic synchronization of acoustic musical signals and textual lyrics. *MULTIMEDIA '04: Proceedings of the 12th annual ACM international conference on Multimedia*, 2004.

[8] Wong, Chi, Szeto, Wai, Wong, & Kin (2007). Automatic lyrics alignment for cantonese popular music. *Multimedia Systems*, *12*(4-5), March, 2007, 307–323.