Syllabic Level Automatic Synchronization of Music Signals and Text Lyrics

Denny Iskandar
Institute for Infocomm Research
21 Heng Mui Keng Terrace
Singapore 119613
idenny@i2r.a-star.edu.sg

Ye Wang Min-Yen Kan
School of Computing
National University of Singapore
Singapore 177543
{wangye, kanmy}@comp.nus.edu.sg

Haizhou Li
Institute for Infocomm Research
21 Heng Mui Keng Terrace
Singapore 119613
hli@i2r.a-star.edu.sg

ABSTRACT
We present a framework to synchronize pop music to corresponding text lyric. We refine line level alignment achievable by existing work to syllabic level by using a dynamic programming process. Our main contribution is using music knowledge to constrain the dynamic programming search. This is done by modeling (1) non-uniform note length distribution and (2) a note length distribution for each section type (for example intro, chorus, and bridge). These reduce alignment error by 6.4% and improve time efficiency by a factor of 2.2.

Categories and Subject Descriptors
H.5.5 [**Information Interfaces and Presentation**]: Sound and Music Computing – Methodologies and Techniques; H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing

General Terms
Algorithms, Design, Experimentation.

Keywords
Music structure, voice alignment, dynamic programming, hidden Markov model.

1. INTRODUCTION
A current key focus of multimedia research is the integration of information from different modalities. One such application is in vocal music processing in which the acoustic music signal and textual lyrics constitute two correlated views of the same source. In karaoke, we may want to synchronize the lyrics to the music, so that a user can sing along. However, previous works only considered synchronization to individual lyric lines; they have not considered the fine-grained syllable level alignment needed to make automatic karaoke systems feasible. We address this final step here by extending previous work. Our formal problem statement is thus:

Given input: A line of lyric text and its corresponding music
Output: Start times for each syllable in the lyric line

Our key contribution is to use training data to impose both *local* line-level as well as *global* song-level constraints in such a dynamic programming (DP) search. We demonstrate a proof-of-concept on a small dataset of three pop songs and show that the techniques employed reduce alignment error by 6.4% on average at run time by a factor of 2.2.

MM'06, October 23–27, 2006, Santa Barbara, California, USA.
ACM 1-59593-447-2/06/0010.

Related Work: To our knowledge, no work has previously addressed this specific problem; however other works on vocal and music have informed our technique. Closest in spirit to our work are [2] and [4]. However, [2] considers only pure singing voice signals and assumes no background music, and largely focuses on real-time performance. In contrast, we consider the more difficult problem of alignment in pop songs, which have strong distortion in the vocal signal from the background music and drum beats. This scenario is identical to the scenario in the latter work, LyricAlly [4]. However, this alignment methodology relies on the song structure at the section-level and line-level alignment. In this work, we extend line level alignment to the syllable level.

A few other works are similar to ours in terms of music analysis with respect to karaoke but are not comparable as the source signal differs. In [1], synchronization of the audio file and text is done with the help of alignment metadata that have been pre-stored in the audio file. [6] also considers the processing of karaoke media in which video and audio streams are already synchronized. Both works focus on utilizing embedded alignment metadata, rather than automatically computing it as in our case. Finally it should be noted that our problem is not one of lyric transcription [3] but one of alignment, as we assume the lyrics are known. This is reasonable because textual lyrics for pop songs are often freely available on the Internet.

2. SYNCHRONIZATION METHOD
Given a correct line level alignment with the corresponding music signal and text lyrics, a straightforward method is to perform forced alignment using a speech recognition system. We have implemented this as the baseline system, using the standard approach of dynamic programming search within a hidden Markov model (HMM) framework.

As per standard speech recognition, a model needs to be trained based on feature vectors extracted from discrete time frames in the acoustic signal. We describe a triphone, a contextual phone unit, by a three-state left-to-right HMM. Each frame is represented as a feature set of 39-element MFCC vectors, similar to other standard speech recognition systems. As the identity of the lyrics is known, a standard left-to-right HMM chain is built, slightly modified to insert optional short pauses at word boundaries. These short pauses are added to handle potential short pauses in singing voice, which most likely contains background music accompaniment. Although this approach is suboptimal as

there is a wide variety in the instrumental accompaniment, we adopt it for simplicity. Given the constructed HMM and the feature frames, the best forced alignment is calculated using standard dynamic programming. This is a decoding process where the search space is limited to a known sequence representing the lyric words with optional pauses inserted between words. During the search, we find optimal transition times from one HMM state to the next. The search result is an alignment of states to timestamps, representing the start and end times of each syllable in the lyrics.

An alignment between two such sources can be pictorially represented on a grid, in which the **X** and **Y** axes each represent a source. Then an alignment is represented as a function that maps an **X** value to a corresponding **Y** value. In our case, the alignment between syllables (**X**s) and timestamps (**Y**s) must be monotonically increasing order. We can further normalize both axes to a [0…1] range, resulting in an alignment diagram as represented in Figure 1. Thus, a simple, monotonous alignment in which each syllable is mapped uniformly to 1/n time units would result in a 45° diagonal alignment. The optimal alignment can be found using standard Viterbi decoding, which finds the best alignment iteratively, based on partial results leading up to last iteration.

This baseline algorithm is a straightforward interpretation of the alignment problem but ignores the fact that the signal is a musical one. As music has recognizable structure and thus constraints, we can model this knowledge to refine the HMM search. These refinements have the dual benefit of 1) constraining the search space, increasing time efficiency and 2) rejecting optimal HMM alignments that conflict with music knowledge, increasing accuracy.

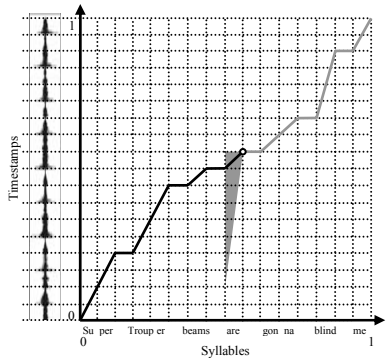


Figure 1. An example of ongoing dynamic programming search. The grayed portion of the alignment path has not been computed; the gray triangle represents all cases currently under consideration.

Our contribution in this work is injecting music-related information and knowledge to find such forced-alignment. We first observe that note length in music is not arbitrary; it is restricted to specific, discrete fractions of the bar (tempo) in each song. Second, notes of different lengths do not have a uniform probability distribution: $^1/_8$ and ¼ notes are more common than $^9/_{16}$ and $^{13}/_{16}$ notes. Finally, different music sections exhibit different note length distributions. Long sustained notes often belong to choruses rather than verses. In both cases, the DP search needs to be modified to model these observations. We review three modifications of the DP search in the following sections.

2.1 Employing Musical Bar Information

One key aspect in which our scenario is more constrained than in speech recognition is due to rhythm. In singing, rhythm is dictated by the music score. We reflect this in our implementation by using *time units* that are proportional to the tempo of each song (measured in bars) rather than a fixed absolute time interval. As such, we use a duration of $^1/_{16}$ bar (normally ranging from 110 ms to 160 ms) as our time unit, as musical notes of smaller bar fractions rarely occur and differences in alignment smaller than this are unlikely to be perceived by users. This essentially confines the search space in the time dimension to larger discrete chunks (compare the average 10ms speech frame to our 110-160 ms time units).

Next, we define a *note segment* as a sequence of zero or more time units. We approximate rhythmic mapping from music to lyric text as a one-to-one mapping between a note segment (from music) and a syllable (from lyric text). Note that because an optional silence can occur between words, note segments of zero length are allowed. Also, note lengths typically do not exceed 1 bar. We set this as the upper bound of the length of note segments, meaning that the DP search allocates between 0 to 16 time units for each note segment.

2.2 Local Constraint

Notes of different length have differing probability of observation. The standard DP search assigns a uniform probability distribution to all 17 possible note segment lengths. In actual songs, singers tend to use more short notes than long ones to satisfy the tempo constraints of each line.

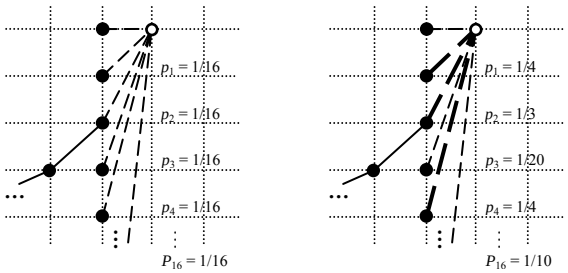


Figure 2. DP search with (left) uniform and (right) non-uniform weights.

We seek to weight the transition probabilities in the HMM by the actual frequency of the note lengths in the data. Figure 2 illustrates how non-uniform weights affect the DP search. The weighting is done as follows.

$$score_i' = (w_i - 1) \times \mathrm{C} + score_i,$$

where i is note segment length ($1 \le i \le 16$); w_i is the weight assigned to note segments of length i time units; C is a scaling constant; $score_i$ is the log probability of partial alignment between the current syllable with a note segment of length i time units; and $score_i$' is weighted log-probability score used in DP search.

We need to estimate only 16 weights–for note segment lengths 1 to 16 because we do not weight a zero length note segment–and the scaling constant C. To make the estimation search space manageable, we assume that the scaling constant C is independent from the weight values.

We first estimate the optimum value for scaling constant C by setting the 16 weights according to note length histogram computed from manual annotation. First we search for an appropriate order of magnitude: 1, 10, 100 and 1000. After we find the order of magnitude, we proceed with estimating an appropriate value at a smaller increment.

Then, we estimate the optimum set of values for the 16 weights. This is done through an iterative randomized search, in which the weight for parameters are optimized one at a time, with each iteration optimizing a particular transition probability weight to minimize alignment error. The weight is picked randomly from all possible 16 weights as we do not know which weights are more important than others. Further, each weight is picked exactly once. At each step, we consider weight values from 0.1 to 1.0 with increments of 0.1. While this procedure finds a local optimum and not necessarily a global one, we have chosen this approach due to time efficiency.

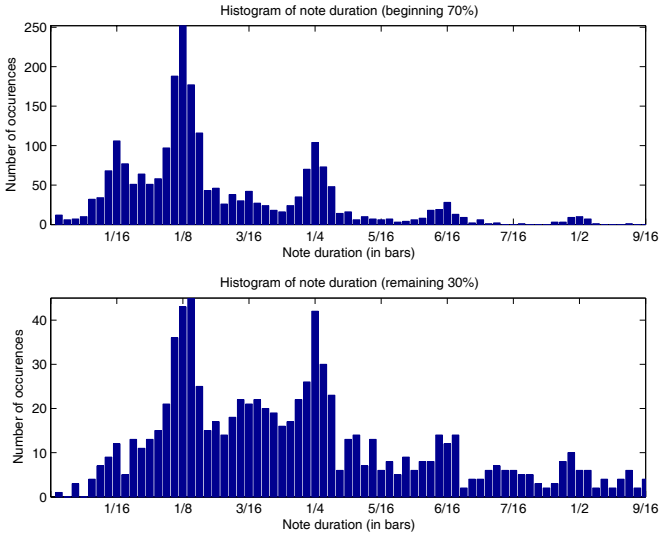


Figure 3. Note length histograms for the first 70% and remaining 30% of each lyric line (top and bottom, respectively). A portion of the histograms above ½ bar are truncated for clarity.

Further, long notes are more likely to occur at the ends of a lyric line than at the beginning (especially when the last syllable of a lyric is drawn out to match the tempo). We model this by maintaining two separate distributions of note length: one for the beginning portion of the line and another for the remainder. A parameter *n* controls the cutoff point between the two histograms. We tried different values of *n* (from 0.1 to 0.9, again in increments of 0.1), and computed two note length histograms: the first from the *n*% of a line; the second for remaining (1-*n*)%. The optimal value of 0.7 was calculated for *n*, as this maximized the difference in the distributions between the two histograms in our training data, as shown in Figure 3. In the figure, one sees that the histogram at the bottom contains a markedly larger proportion of long sustained notes. This difference is effectively captured by our modeling.

We estimate the optimum weight values for beginning portion and the remainder separately. The weights for each set are optimized as described above. We optimize the weights for the beginning portion first, followed by the remainder portion. We use the same value of scaling constant C.

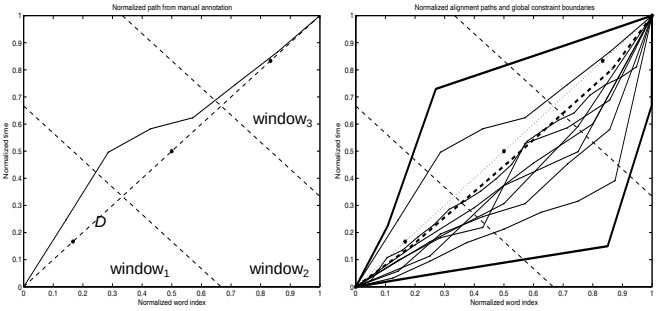
2.3 Global Constraint

The different distributional properties of note lengths in the previous section are local, specific to individual lines. However, distribution differences can also be observed at the song level. Each vocal section in a pop song can be categorized as one of three types: verse, chorus, or others (e.g., intro, outro or bridge). We observed that chorus sections usually contain more sustained notes than verses. Thus, we should also bias the DP search depending on section type. In our implementation, we achieve this by using global constraint boundaries and allowing only synchronization paths within the boundaries. With respect to Figure 1, the constraints define an ovular envelope when mapped to the grid, as all alignments must touch the lower left and upper right corners. HMM alignments that cross outside this envelope are discarded.

We compute these global constraint upper and lower boundaries from the statistics of alignment paths from manual annotation as follows. Figure 4a shows such a path for the line "Super Trouper beams are gonna blind me". The monotonous alignment, illustrated by diagonal line *D*, is shown as a guide.

We divide the normalized line duration into three uniformly-sized windows, for which the global constraints are obtained independently. Window boundaries are shown in Figure 4a as dashed lines perpendicular to *D*.

For each known syllable-to-timestamp alignment in the training data, we ascertain its section type (e.g., chorus) and which of the three windows the timestamp belongs to. We then calculate its displacement (both magnitude and direction) from *D*. Aggregating over all timestamps for a section type and window, we calculate the mean and standard deviation of all the displacements. This yields the *mean path* (bold dashed line in Figure 4b), from which we derive the upper (lower) boundary constraints by adding (subtracting) a multiple of the standard deviation to the mean path (the bold lines in Figure 4b). From experiments, this multiplication factor ranges from 3 to 7.



(a) (b)

Figure 4. (a) A path and diagonal alignment *D*. (b) A set of paths with computed mean path and upper and lower boundaries.

3. EXPERIMENTS

We use three pop songs to illustrate our concept: "YMCA" by Village People, "Super Trouper" by ABBA and "When You Say Nothing at All" by Ronan Keating. All songs were sampled at 11 KHz. This dataset consists of a total of 1843 syllables, 845

words, 104 lines. Four part-time polytechnic students manually annotated the ground truth word-level alignment for all three songs. Both the local and global constraints were estimated from the dataset.

As noted in Section 2, our technique requires a standard acoustic model for the HMM. To build the acoustic model, we train the HMM on regular speech data and then perform adaptation on music data. A window size of 25 ms and step size of 10 ms are used in MFCC calculation. The resulting HMM system has 1465 states shared among 10227 triphones. Each state has a triphone using 32 Gaussian mixtures. We use Wall Street Journal CSR datasets (WSJ 1 and 2) to build the acoustic model of speech. For music adaptation, we use a set of 20 pop songs which does not include any of the three songs above.

It should also be noted that for this proof-of-concept, we have used the three songs for both model parameter optimization and testing, in a closed test setup; current work is being done to assess performance on separate training and testing data.

Table 1. Word-level synchronization error with respect to tolerance *T* (in bars)

Experiment description	Sychronization error rate (%)		
	T=1/4	1/8	1/16
0. Baseline	20.7	34.2	49.9
1. Discretize to 1/16 bar frames	19.7	31.1	46.0
2. Local constraints (L)	19.4	29.0	43.2
3. Global constraints (G)	18.7	30.6	45.6
4. L+G	18.7	28.8	43.5

Table 1 shows the results of our prototype synchronization system. Although our method achieves syllable level synchronization, we evaluate at the word level because ground truth at the syllable level was difficult for annotators to reliably annotate. We evaluate our system using different levels of alignment error tolerance, as indicated by the *T* values (measured in bars) in the table. We call attention to the following aspects of our alignment performance.

Music structure constraints. Overall, we see that using 1/16 bar long frames does not affect alignment performance significantly. We also see that local and global constraints help to improve accuracy by 5.2% and 3.6% at 1/8 bar tolerance individually; but that their gains in performance largely overlap when employed together. This may be partially due to the fact that portions of models of the local constraints (two unequal portions of the line) and the global constraints (three equal portions of the line) partially overlap.

Time efficiency. Aside from improving accuracy, the music structure has a positive side effect on time efficiency. This is because the optimum global constraints cut down the search space by more than a half (approximately 55% on average) for each line. Note that time discretization does not affect the search space. This is because alignment between a note segment and a syllable is ultimately computed as alignment between a sequence of frames and a sequence of triphones.

4. CONCLUSION

We have presented a framework to synchronize pop music to the corresponding lyric text at syllabic-level. Our main contribution is using music knowledge to constrain dynamic programming search.

We are currently validating our approach on a clean training and testing set separation. For future work, we would like to investigate how we can reduce distortion from background music accompaniment as well as how we can exploit information from repeated lines in lyric text (e.g., in chorus sections).

5. REFERENCES

[1] Furini, M. and Alboresi, L. Audio-text synchronization inside MP3 files: a new approach and its implementation. In *Proceedings of the IEEE Consumer Communications & Networking 2004 (CCNC2004)*, Las Vegas, USA, 2004.

[2] Loscos, A., Cano, P., and Bonada, J. Low-Delay Singing Voice Alignment to Text. In *Proceedings of International Computer Music Conference*, Beijing, China, 1999.

[3] Wang, C.-K., Lyu, R.-Y., Chiang, Y.-C. An automatic singing transcription system with multilingual singing lyric recognizer and robust melody tracker. In *Proceedings of the 8th European Conference on Speech Communication and Technology* (*EUROSPEECH-2003)*, 1197-1200.

[4] Wang, Y., Kan, M.-Y., Nwe, T.L., Shenoy, A., and Yin, J. LyricAlly: automatic synchronization of acoustic musical signals and textual lyrics. In *Proceedings of the 12th ACM International Conference on Multimedia*, pp. 212-219, 2004.

[5] Yoshii, K., Goto, M., and Okuno, H. G. Automatic drum sound description for real-world music using template adaptation and matching methods. In *Proceedings of the 5th International Conference on Music Information Retrieval (ISMIR2004)*, Barcelona, Spain, 2004.

[6] Zhu, Y., Chen, K., and Sun, Q. Multimodal content-based structure analysis of karaoke music. In *Proceedings of ACM International Conference on Multimedia*, pp. 638-647, 2005.

[7] HTK Speech Recognition Toolkit. http://htk.eng.cam.ac.uk/