LyricAlly: Automatic Synchronization of Acoustic Musical Signals and Textual Lyrics

Ye Wang Min-Yen Kan Tin Lay Nwe Arun Shenoy Jun Yin
Department of Computer Science, School of Computing
National University of Singapore, Singapore 177543
++ (65) 6874-2980
{wangye, kanmy, nwetl, arunshen, yinjun}@comp.nus.edu.sg

ABSTRACT
We present a prototype that automatically aligns acoustic musical signals with their corresponding textual lyrics, in a manner similar to manually-aligned karaoke. We tackle this problem using a multimodal approach, where the appropriate pairing of audio and text processing helps create a more accurate system. Our audio processing technique uses a combination of top-down and bottom-up approaches, combining the strength of low-level audio features and high-level musical knowledge to determine the hierarchical rhythm structure, singing voice and chorus sections in the musical audio. Text processing is also employed to approximate the length of the sung passages using the textual lyrics. Results show an average error of less than one bar for per-line alignment of the lyrics on a test bed of 20 songs (sampled from CD audio and carefully selected for variety). We perform holistic and per-component testing and analysis and outline steps for further development.

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
Audio/text synergy, music knowledge, vocal detection, lyric alignment, karaoke.

1. INTRODUCTION
We investigate the automatic synchronization between audio and text. Given an acoustic musical signal and corresponding textual lyrics, our system attempts to automatically calculate the start and end times for each lyric line. As this kind of an alignment is currently a manual process and a key step for applications such as karaoke, the system we propose here has a potential to automate the process, saving manual labor. Additionally, this information can also be used effectively in the field of music informational retrieval to facilitate random access to specific words or passages of interest.

In contrast to other work that has been restricted to either the symbolic domain (MIDI and score) or synthesized audio, LyricAlly has been developed to operate on real-world musical recordings (sampled from CD audio) to maximize its practical applicability.

We decompose the problem into two separate tasks, performed in series: alignment at a higher structural element level (*e.g.*, verse, chorus, etc.) , followed by a lower level per-line alignment level. In comparison to a simpler single-level alignment model, we feel that this cascaded architecture boosts system performance and allows for a more meaningful evaluation to be done.

The rest of this paper is organized as follows. In the next section, we review related work. We then introduce LyricAlly in Section 3 and define terms used throughout the paper. Section 4 details the audio processing components followed by the text processing component in Section 5. We then detail our two-level integration of components in Section 6. We analyze system performance and conclude with comments on current and future work.

2. RELATED WORK
To the best of our knowledge, there has been no published work on the problem addressed in this paper. The works closest to ours are briefly surveyed in this section. A framework to embed lyric time stamps inside MP3 files has been previously proposed [5]. This approach addressed the representation of such time stamps, but not how to obtain them automatically. [14] describes a large vocabulary speech recognizer employed for lyric recognition. The system deals with pure singing voice, which is more limiting as compared to real-world acoustic musical signals handled by our approach. Our experience shows that the transcription of lyrics from polyphonic audio using speech recognition is an extremely challenging task.

This difficulty has led us to re-examine the transcription problem. We recognize that transcription is often not necessary, as many lyrics are already freely available on the Internet. As such, we formulate the problem as one of lyric alignment rather than transcription. In a sense, our work is analogous to those in [1][4][12] which try to perform automatic alignment between audio recording and MIDI. However, their task is quite different from ours, as MIDI files provide some timing information which is not normally present in textual lyrics or in real-world acoustic environments.

MM'04, October 10-15, 2004, New York, New York, USA.

3. SYSTEM DESCRIPTION

In LyricAlly, the first round of integration performs a high-level alignment of the song's structural elements detected by the text and audio streams. A second round performs the lower-level line alignment. The points in which the two streams interchange intermediate calculations are thus sources of synergy, which are discussed later in Section 6. LyricAlly aligns these two modalities in a top-down approach, as shown in Figure 1.

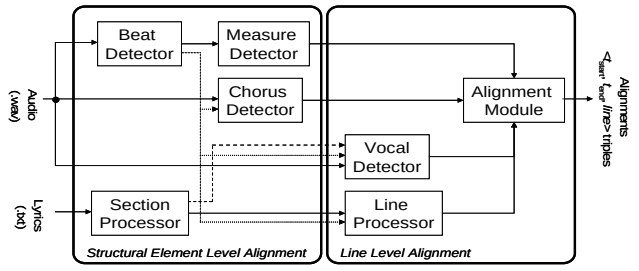

Figure 1: LyricAlly architecture.

We now define the structural elements of music (also referred to as *sections* in this paper) used in our work:

- Intro (I): The opening section that leads into the song, which may contain silence and lack a strong beat (arrhythmic).
- Verse (V): A section that roughly corresponds with a poetic stanza and is the preamble to a chorus section.
- Chorus (C): A refrain (lines that are repeated in music) section. It often sharply contrasts the verse melodically, rhythmically and harmonically, and assumes a higher level of dynamics and activity, often with added instrumentation.
- Bridge (B): A short section of music played between the parts of a song. It is a form of alternative verse which often modulates to a different key or introduces a new chord progression.
- Outro or Coda (O): A section which brings the music to a conclusion. For our purpose, an outro is a section that follows the bridge until the end of the song. This is usually characterized by the chorus section repeating a few times and then fading out.

Based on an informal survey of popular songs, we introduce the following heuristics into our system:

- Instrumental music sections occur throughout the song.
- Intro and bridge sections may or may not be present and may or may not contain sung vocals.
- Popular songs are strophic in form, with a usual arrangement of *verse-chorus-verse-chorus.* Hence the verse and chorus are always present and contain sung vocals.

LyricAlly was developed using music data from a wide variety of sung music, spanning many artists over many years. Our current prototype is limited to songs which have a structure comprising of no sung intros, two verses, two choruses, bridge and an outro (*i.e.*, "V_1-C_1-V_2-C_2-B-O"). This structure is the most common structure of popular songs based on our observations, accounting for over 40% of the songs we surveyed, and thus are not overly restrictive. As we detail the workings of the components in the next sections we will use a running example of a V_1-C_1-V_2-C_2-B-O song, *25 Minutes,* performed by the group *Michael Learns To Rock (MLTR).*

4. AUDIO PROCESSING

Audio processing in LyricAlly has three steps:

1. Determine the rhythm structure of the music. Knowledge of rhythm structure at the measure (bar level) helps to fine tune the time alignment of the other components.
2. Determine the rough location of the chorus (refrain) segments in the song. This serves as an anchor for subsequent line-level alignment in the chorus as well as the verse sections of the song.
3. Determine the presence of vocals in the song. This is needed for the alignment results at the line-level.

4.1 Hierarchical Rhythm Structure Determination

Our rhythm detector extracts rhythm information in real-world musical audio signals as a hierarchical beat-structure comprising the quarter-, half-, and whole-note (measure/bar) levels.

Rhythm can be perceived as a combination of strong and weak beats [11]. The beat forms the basic unit of musical time and in a meter of 4/4 (common time or quadruple time) there are four beats to a measure. The inter-beat interval (IBI) corresponds to the temporal length of a quarter note. We assume the meter to be 4/4, this being the most frequent meter of popular songs and the tempo of the input song is assumed to be constrained between 40-185 beats per minute (BPM) and almost constant. The audio signal is framed into beat-length segments to extract metadata in the form of quarter note detection of the music. The basis for this technique of audio framing is that within the quarter note, the harmonic description of the music expressed by musical chords can be considered as quasi-stationary. This is based on the premise that chord changes are more likely to occur on beat times in accordance with the rhythm structure than on other positions.

A system to determine the key of acoustical musical signals has been demonstrated in [11]. The key defines the diatonic scale that the song uses. In the audio domain, overlap of harmonic components of individual notes makes it difficult to determine the individual notes present in the scale. Hence this problem has been approached at a higher level by grouping individual detected notes to obtain the harmonic description of the music in the form of the 12 major and 12 minor triads (chords with 3 notes). Then based on a rule-based analysis of these chords against the chords present in the major and minor keys, the key is extracted.

As chords are more likely to change at the beginning of a measure than at other positions of beat times [6], we would like to incorporate this knowledge into the key system to determine the rhythm structure of the music. However, we observe that the chord recognition accuracy of the system is not sufficient to determine the hierarchical rhythm structure across the entire length of the music. This is because complexities in polyphonic audio analysis often results in chord recognition errors. We have thus enhanced this system with two post-processing stages that allow us to compute this task with good accuracy, as shown in Figure 2. The output of the beat detection is used to frame the audio into beat-length segments. This basic information is used by

all other modules in LyricAlly, including subsequent steps in this module.

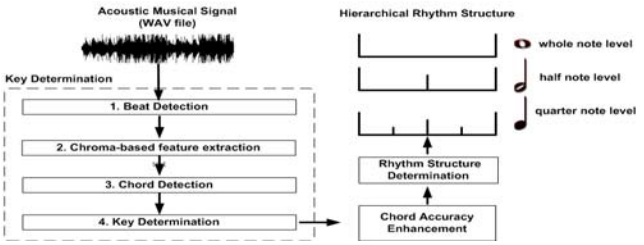

Figure 2: Hierarchical rhythm structure block flow diagram.

4.1.1 Chord Accuracy Enhancement

Knowledge of the detected key is used to identify the erroneous chords among the detected chords. We eliminate these chords based on a music theoretical analysis of the chord patterns that can be present in the 12 major and 12 minor keys.

4.1.2 Rhythm Structure Determination

We check for start of measures based on premise that chords are more likely to change at the beginning of a measure than at other positions of beat times [6]. Since there are four quarter notes to a measure, we check for patterns of four consecutive frames with the same chord to demarcate all the possible measure boundaries. However, not all of these boundaries may be correct. This is on account of errors in chord detection. The correct measure boundaries along the entire length of the song are determined as follows:

1. Along the increasing order on the time axis, obtain all possible patterns of boundaries originating from every boundary location that are separated by units of beat-spaced intervals in multiples of four. Select the pattern with the highest count as the one corresponding to the pattern of actual measure boundaries.
2. Track the boundary locations in the detected pattern and interpolate missing boundary positions across the rest of the song.

The result of our hierarchical rhythm detection is shown in Figure 3. This has verified against the measure information in commercially-available sheet music [8].

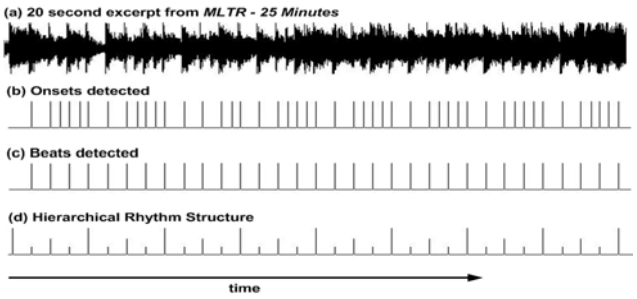

Figure 3: Hierarchical rhythm structure in *25 Minutes*.

4.2 Chorus detector

The audio chorus detector locates chorus sections and estimates the start and the end of each chorus. Our implementation is based on Goto's method [7], which identifies chorus sections as the most repeated sections of similar melody and chords by the use of chroma vectors.

We improve the original chorus detection algorithm by incorporating beat information obtained from the rhythm structure detector. This input allows us to significantly reduce the complexity of the algorithm. Since the chord is stable within an inter-beat interval, we extract chroma vectors from each beat, rather than for each 80ms frame as prescribed by the original algorithm. For a standard three-minute song at 100 BPM, our approach extracts only 300 vectors (as compared to 2250 in the original). As vectors are pairwise compared – a $O(n^2)$ operation – the savings scale quadratically. For an average song, our version uses only 2% of the time and space required by the original algorithm. We give an example of our chorus detector in Figure 4.

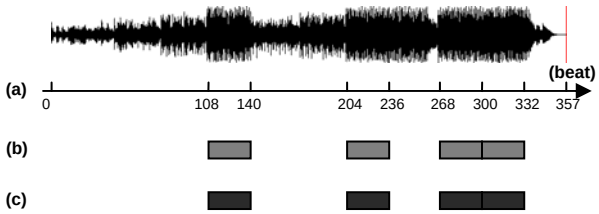

Figure 4: (a) the song *25 Minutes* (b) manually annotated chorus sections (c) automatically detected chorus sections.

4.3 Vocal detector

The vocal detector detects the presence of vocals in the musical signal. We use a Hidden Markov Model (HMM) classifier to detect vocals. In contrast to conventional HMM training methods which employ one model for each class, we create an HMM model space (multi-model HMMs) to perform vocal detection with increased accuracy. In addition, we employ an automatic bootstrapping process which adapts the test song's own models for increased classification accuracy.

Our assumption is that the spectral characteristics of different segments (pure vocal, vocal with instruments and pure instruments) are different. Based on this assumption, we extract feature parameters based on the distribution of energy in different frequency bands to differentiate vocal from non-vocal segments. The time resolution of our vocal detector is the inter-beat interval (IBI) described in the previous section.

We compute a subband based Log Frequency Power Coefficients (LFPC) [10] to form our feature vectors. This feature provides an indication of energy distribution among subbands. We first train our models using manually annotated songs, and then perform classification between vocal and non-vocal segments.

Most studies on vocals detection use statistical pattern classifiers [2][3][13]. To our knowledge, none have taken into account song structure information in modeling. An important observation is that vocal and non-vocal segments vary in their inter-song and intra-song signal characteristics. Signal strengths in different sections (verse, chorus, bridge and outro) usually differ. For example, chorus sections usually have stronger signal strength in comparison with verse sections since they may have busier drums, some additional percussion, a fuller string arrangement and additional melody lines [15]. The tempo and intensity of vocals are different among songs, accounting for inter-song

variation. These result in distinct single characteristics, as shown in Figure 5.

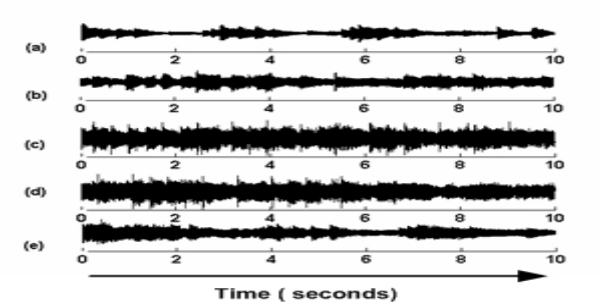

Figure 5: Waveforms of 10-second segments extracted from the (a) intro, (b) verse, (c) chorus, (d) bridge, and (e) outro sections of the song *25 Minutes*.

Due to this, we integrate song structure, inter-song and intra-song variation into our model. Training data are manually classified based on the section type, tempo and intensity and a model is created for each class. This process results in a total of 2 (vocal or non-vocal) × 5 (section types) × 2 (high or low tempo) × 2 (loud or soft intensity) = 40 distinct HMMs.

A test song is first segmented using these HMMs, then each section is classified with respect to section type. The probability distributions of the number of matched models for the test segments in $Verse_1$ and $Chorus_1$ sections are shown in Figure 6. As expected, segments from the verse section are more likely to match the verse models than others. In the same way, segments from the chorus section tend to match the chorus models better than others. With this step, we achieve about 80% classification accuracy.

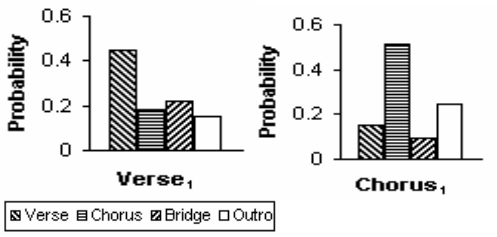

Figure 6: Probability distributions of the number of matched models for test segments in $Verse_1$ (l), $Chorus_1$ (r) sections.

We then use this initial classification to build song-specific vocal and non-vocal models of the test song with a bootstrapping process. This process allows us to use a song's own model for classification as shown in Figure 7. This bootstrapping process makes the algorithm adaptive, and achieves the vocal detection accuracy of 84%. An average improvement of 4% is demonstrated, which is statistically significant. Results of vocal detection by our vocal detector and manual annotation of the $Verse_1$ section in our test song are shown in Figure 8.

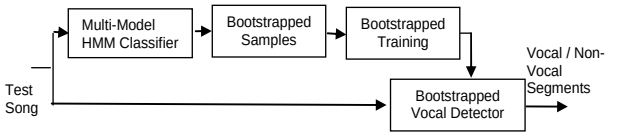

Figure 7: Bootstrapped training and segmentation process.

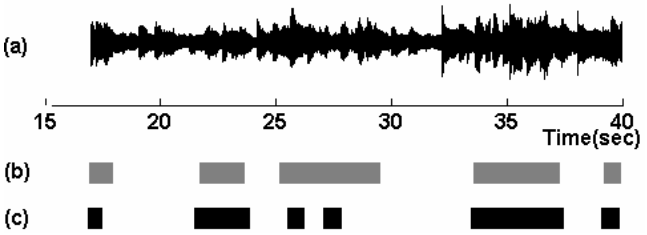

Figure 8: (a) The $Verse_1$ segment of *25 Minutes*. (b) Manually annotated and (c) automatically detected vocal segments.

5. TEXT PROCESSING

Text lyrics are analyzed in a two-stage cascade. The first phase labels each section with one of the five section types, and also calculates a duration for each section. A subsequent line processor refines the timings by using the hierarchical rhythm information to determine finer per-line timing. This simple model performs well for our system, and as such, other models (e.g., HMMs) have not been pursued.

5.1 Section Processor

The section processor takes as its sole input the textual lyrics. We assume that the input lyrics delimit sections with blank lines and that the lyrics accurately reflect the words sung in the song. Similar to the audio chorus detector described in Section 4.2, choruses are detected by their high level of repetition. Our model accounts for phoneme-, word- and line-level repetition in equal proportions. This model overcomes variations in words and line ordering that poses problems for simpler algorithms that use a variation of the longest common subsequence algorithm for detection. From music knowledge, we can further constrain candidate chorus sections to be interleaved with one or two other (e.g., verse and possible bridge) intervening sections and to be of approximately the same length in lines. The example song is classified in Figure 9.

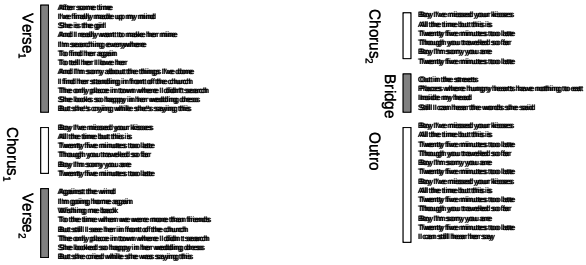

Figure 9: Section classification of *25 Minutes*. Automatic and manual annotations coincide.

An approximate duration of each section is also calculated. Each word in the lyrics is first decomposed into its phonemes based on the word's transcription in an inventory of 39 phonemes from the *CMU Pronouncing Dictionary* [16]. Phoneme durations are then looked up in a singing phoneme duration database (described next) and the sum total of all phoneme durations in a section is returned as the duration, which is used in the forced alignment discussed later in this paper.

As phoneme durations in sung lyrics and speech differ, we do not use information from speech recognizers or synthesizers. Rather, we learn durations of phonemes from annotated sung training data, in which each line and section are annotated with durations.

We decompose each line in the training data into its phonemes and parcel the duration uniformly among its phonemes. In this way, a phoneme can be modeled by the distribution of its instances. For simplicity, we model phoneme duration distributions as normal distributions, characterized by mean and variance. We represent section's duration by summing together the individual distributions that represent the phonemes present in the section.

Analysis of the induced phoneme database shows that phonemes do differ in duration in our training data: the average phoneme length is .19 seconds, but vary on the individual phoneme (max = .34, min = .12, σ = .04). The use of a per-phoneme duration model versus a uniform model (in which every phoneme is assigned the average duration) accounts for a modest 2.3% difference in estimated duration in a sample of lines, but is essential for future work on phoneme level alignment.

5.2 Line processor

The rhythm structure detector discussed in Section 4.1 provides bar length and offsets as additional input. This allows the text module to refine its estimates based on our observation that each line must start on the bar (discussed in more detail later). We start with the initial estimate of a line's duration calculated earlier and round it to the nearest integer multiple of bar. We calculate the majority bars per line for each song, and coerce other line durations to be either ½ or 2 times this value. For example, songs in which most lines take 2 bars of time may have some lines that correspond to 1 or 4 bars (but not 3 or 5). In our experience, this observation seems to increase system performance for popular music.

The text model developed thus far assumes that lyrics are sung from the beginning of the bar until the end of the bar, as shown in Figure 10(a). When lyrics are short, there can be a gap in which vocals rest before singing again on the next bar, as shown in Figure 10(b). Thus, an ending offset for each line is determined within its duration. For lines that are short and were rounded up, vocals are assumed to rest before the start of the following line. In these cases, the initial estimate from the derived database is used to calculate the ending offset. For lines that are long and rounded down in the coercion, we predict that the singing leads from one bar directly into the next, and that the ending offset is the same as the duration.

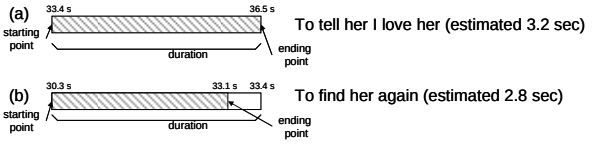

Figure 10: Finding ending offsets in *25 Minutes,* where the calculated bar length is 3.1 seconds: (a) case where the bar information overrides the initial estimate, (b) case in which the initial estimate are used.

6. SYSTEM INTEGRATION

In this section we integrate the audio and text components to align the audio file with its associated textual lyrics. Our alignment algorithm consists of two steps:

- Section level alignment, which uses the measure, chorus, vocal detector and section processor as input to demarcate the section boundaries.
- Line level alignment, which uses the vocal, measure and line processor as input to demarcate individual line boundaries.

6.1 Section level alignment

In section level alignment, boundaries of the verses are determined using the previously determined chorus boundaries. A key observation is that the detection of vocal segments is substantially easier than the detection of non-vocal ones. This is because both the audio and text processing can offer evidence for detecting and calculating the duration of vocal segments.

We use a statistical method to build a static gap model based on manual annotation of 20 songs from our testbed. The gap model (the normalized histogram) of all sections in our dataset is depicted in Figure 11. It can be seen that the duration between verse and chorus (V_1-C_1, V_2-C_2) is fairly stable in comparison to the duration of the sections themselves. This observation allows us to determine verse starting points using a combination of gap modeling and positions of the chorus or the song starting point.

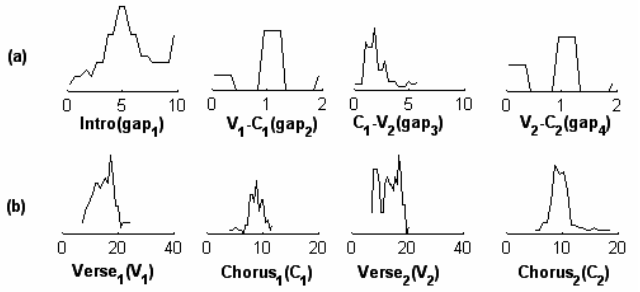

Figure 11: Duration distributions of (a) non-vocal gaps, (b) different sections of the popular songs with V_1-C_1-V_2-C_2-B-O structure. X-axis represents duration in bars.

This technique is embodied in LyricAlly in forward/backward search models, which use an anchor point to search for starting and ending points of other sections. For example, the forward search model uses the beginning of the song (time 0) as an anchor to determine the start of $Verse_1$. From Figure 11(a), we observe that the intro section is zero to ten bars in length. Over these ten bars of the music, we calculate the *Vocal to Instrumental Duration Ratio* (VIDR), which denotes the ratio of vocal to instrument probability in each bar, as detected by the vocal detector. To determine the beginning of a vocal segment, we select the global minimum within a window assigned by the gap models, as shown in Figure 12.

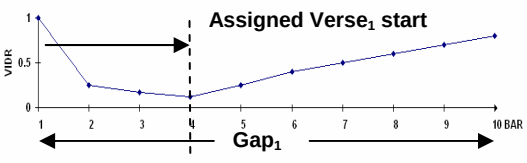

Figure 12: Forward search in Gap_1 to locate $Verse_1$ start.

This is based on two observations: first, the beginning of the verse is characterized by the strong presence of vocals that causes a rise in the VIDR over subsequent bars; second, as the vocal detector may errorneously detect vocal segments within the gap (as in bars 0-2), the beginning of the verse may also be marked by a decline in VIDR in previous bars.

In a similar manner, a backward search model is used to determine the end of a vocal segment. As an example, the end of $Verse_1$ is detected using the gap model and $Chorus_2$ starting point as an anchor provided by the chorus detector (Figure 13).

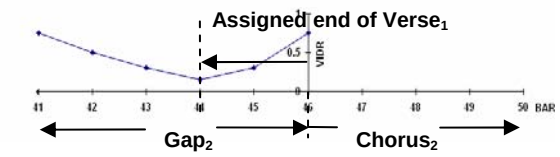

Figure 13: Backward search to locate the ending of a verse.

6.2 Line level alignment

The text processor is fairly accurate in duration estimation but is incapable of providing offsets. The vocal detector is able to detect the presence of vocals in the audio but not associate it with the line structure in the song. These complementary strengths are combined in line-level alignment.

First, we try to derive a one-to-one relationship between the lyric lines and the vocal segments. We use the number of text lines as the target number of segments to achieve. As such, there are three possible scenarios (Figure 14) in which the number of lyric lines is smaller, equal to or greater than the number of vocal segments. In the first and last cases, we need to perform grouping or partitioning before the final step of forced alignment.

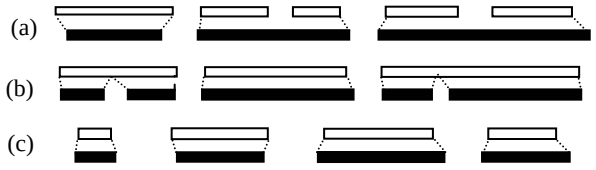

Figure 14: (a) Grouping, (b) partitioning and (c) forced alignment. White rectangles represent vocal segments and black rectangles represent lyric lines.

For the grouping process, all possible combinations of disconnected vocal segments are evaluated to find the best matching combination to the duration of the text lines. A similar approach is employed for the partitioning process, where the duration of the vocal segments and their combinations are compared with the estimated line duration from the text module.

Once an optimal forced alignment is computed, the system combines both the text and vocal duration estimates to output a single, final estimate. We start by calculating a section's duration given information from both detectors. We denote this as D_{TV}, which combines D_T and D_V, the total estimated duration of the section given by the text and vocal detectors, respectively.

$$D_{TV} = \max(D_T, D_V) - \alpha\left|D_T - D_V\right| \qquad (4)$$

Then, the durations of individual vocal segments $D'_v(i)$ are re-assigned:

$$D'_V(i) = \frac{D_T(i)}{D_T} \times D_{TV} \quad i = 1,2,\ldots, L \qquad (5)$$

where L is total number of lines. This is to force the ratio of the durations of the vocal segments to match those from the text, as we have found the text ratios to be more accurate. We assign a value for α such that the total duration of the vocal segments within each section is closest to the section estimates found earlier in Section 6.1. Example results of our automatic synchronization are shown in Figure 15.

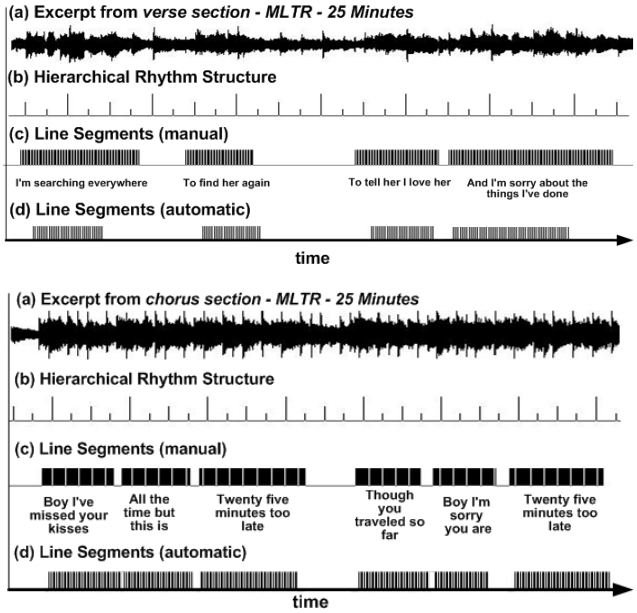

Figure 15: Line segment alignment.

7. EVALUATION

We perform both holistic and per-component evaluation of LyricAlly in order to assess its performance and understand its strengths and weaknesses. We evaluate over our dataset for which we manually annotated the songs with starting and ending time stamps of each lyric line.

Past work in audio alignment [4] used random sampling to compute an alignment error, given in seconds. The average and standard deviation of starting point and duration error are computed. In contrast, we evaluate our system over our entire dataset. These results are presented in column 3 of Table 1 (as normal distributions, characterized by mean and variance) for high-level section alignment and lower-level line alignment, respectively.

Alignment Error		Seconds	Bars
Section Level (n = 80)	Starting Point	N(0.80, 9.0)	N(0.30, 1.44)
	Duration	N(-0.50,10.2)	N(-0.14, 1.44)
Line Level (n = 565)	Starting Point	N(0.58, 3.6)	N(0.22, 0.46)
	Duration	N(-0.48,0.54)	N(-0.16, 0.06)

Table 1: Section- and line-level alignment error over 20 songs. Errors (in seconds) given as normal distributions: N (μ,σ^2).

Error given in seconds may not be ideal, as a one-second error may be perceptually different in songs with different tempos. We suggest measuring error in terms of bars as a more appropriate metric. Average error (in bars) is given in column 4.

Most importantly, starting point calculation is more difficult than duration estimation for individual lines. This is likely because the starting point is derived purely by audio processing, whereas the

text processing greatly assists in the duration calculation. We also see that durations of entire sections are more variable than single lines, as sections are larger units. On the other hand, starting point calculation performance does not vary significantly between lines and sections.

7.1 Error analysis of individual modules

As LyricAlly is a prototype based on an integration of separate modules, we also want to identify critical points in the system. Which components are bottlenecks in system performance? Does a specific component contribute more error in localization or in determining duration?

To answer these questions, we analyze each module's contribution to the system. Due to space constraints, we have simplified each of the four modules' performance to a binary feature (i.e., good performance on the target song or not). We re-analyze the system's performance over the same dataset and show our results in Table 2. As expected, the system works best when all components perform well, but performance degrades gracefully when certain components fail.

Different modules are responsible for different errors. If we force starting point and duration calculations to be classified as either good or not, then we have four possible scenarios for a song's alignment, as exemplified in Figure 16.

Failure of the rhythm detector affects all modules as estimates are rounded to the nearest bar, but the effect is limited to beat length over the base error. Failure of the chorus detection causes the starting point anchor of chorus sections to be lost, resulting in cases such as Figure 16(c). When the vocal detector fails, both starting point and duration mismatches can occur, as shown in Figure 16(b, c and d). The text processor can only calculate duration, and its failure leads to less accurate estimations of the duration of sung lines, as in Figure 16(b).

8. DISCUSSION

These results indicate that each module contributes a performance gain in the overall system. Excluding any module degrades performance. If we weight starting point and duration errors equally, and equate minimizing the sum of squares of the per-line error as a performance measure, we can rank the modules in decreasing order of criticality:

Vocal > Measure > Chorus > Text

We believe that errors in starting point and duration are likely to be perceived differently. In specific, starting point errors are more likely to cause difficulties for karaoke applications in comparison to duration errors. When we weight starting point errors five times as important, a different ranking emerges:

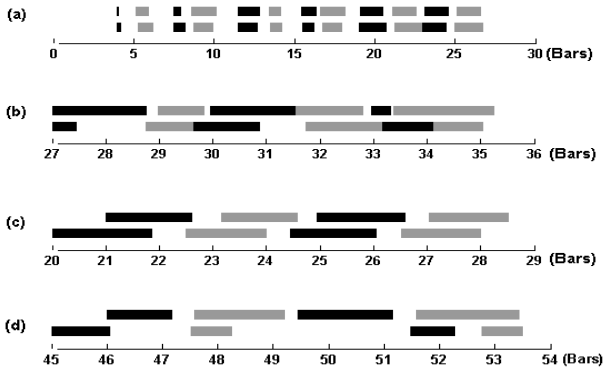

Figure 16: Alignment between manual (upper line) and automatic timings (lower line). (a) Perfect alignment, (b) Duration mismatch, (c) Starting point mismatch, (d) Both duration and starting point mismatches.

Chorus > Vocal > Text > Measure

We believe that this is a more realistic ranking of the importance of each of the modules. As the modules contribute differently to the calculation of starting point and duration calculation, their effect on the overall system is different.

As can be seen by integration strategy in LyricAlly, the accurate detection and alignment of chorus sections is paramount as it allows an anchor for the subsequent development of verse alignment. As our solution to this subtask has significant limitations at this point, we intend to invest our resources in solving this subtask.

We have emphasized error analysis in our evaluation, yet it is not the only criteria in assessing performance. Efficiency is also paramount, especially for applications that may be deployed in mobile devices. The text processing of the dataset requires magnitudes less computation to perform as compared to the audio components. It also helps to limit the problem for the audio processing: for example, knowing that there are two choruses in a song instead of three helps the chorus detector prune inconsistent hypotheses. As LyricAlly is scaled up to handle more complex song structures, we feel that the synergies between text and audio processing will play a larger role.

Songs	Systems do well	System Fails	Starting point (Sec)	Duration (Sec)	Starting point (Bar)	Duration (Bar)	Sample Song
6	A,B,C,D	--	N(-0.1, 0.49)	N(-0.1, 0.01)	N(-0.03, 0.09)	N(-0.04, 0.04)	[2001] Westlife – World of Our Own
2	B,C,D	A	N(-0.4, 1.21)	N(-0.3, >0.01)	N(-0.18, 0.16)	N(-0.09, >0.01)	[1996] Michael Learns to Rock – Sleeping Child
2	A,C,D	B	N(1.3, 1.00)	N(-0.2, >0.01)	N(0.6, 0.16)	N(-0.02, >0.01)	[1998] The Corrs - I never loved you anyway
2	A,B,D	C	N(0.7,5.76)	N(-0.5, 0.04)	N(0.3, 0.81)	N(-0.2, >0.01)	[2000] Leann Rimes - Can't fight the moonlight
2	A,B,C	D	N(-0.9, 0.04)	N(-0.8, 0.04)	N(-0.4, 0.01)	N(-0.3, 0.04)	[1996] R Kelly - I believe I can fly
6	*Other configurations*		N(1.4, 7.29)	N(-0.8, 1.44)	N(0.5, 0.81)	N(-0.2, 0.16)	[1997] Boyzone - Picture of you

A=Measure detector, B=Chorus detector, C=Singing voice detector, D=Duration calculation of text processor

Table 2: Average alignment error and standard deviation over all lines (n=565) in the 20 song dataset. Errors given as Nor (μ,σ^2).

9. CONCLUSION AND FUTURE WORK

We have outlined LyricAlly, a multimodal approach to automate alignment of textual lyrics with acoustic musical signals. It incorporates state-of-the-art modules for music understanding in terms of rhythm, chorus detection and singing voice detection. We leverage text processing to add constraints to the audio processing, pruning unnecessary computation and creating rough estimates for duration, which are refined by the audio processing. LyricAlly demonstrates that two modalities are better than one and furthermore, that the processing of acoustic signals on multiple levels places the solution for automatic synchronization of audio with lyrics problem in reach.

Our project has lead to several innovations in combined audio and text processing. In audio processing, we have demonstrated a new chord detection algorithm and applied it to hierarchical rhythm detection. We capitalize on rhythm structure to vastly improve the efficiency of a state-of-the-art chorus detection algorithm. We develop a new singing voice detection algorithm which combines multiple HMM models with bootstrapping to achieve higher accuracy. In our text processing models, we use a phoneme model based on singing voice to predict the duration of sung segments. To integrate the system, we have viewed the problem as a two-stage forced alignment problem. We have introduced gap modeling and used voice to instrument duration ratios as techniques to perform the alignment.

LyricAlly currently is limited to songs of a limited structure and meter. For example, our hierarchical rhythm detector is limited to 4/4 time signature. The performance of our chorus and vocal detectors is not yet good enough for real life applications. In our vocal detector, we could consider an implementation using mixture modeling or classifiers such as neural networks or support vector machines. These are two important areas in the audio processing module for future work. Furthermore, our observation shows that sung vocal are more likely to change at positions of half note intervals than at other positions of beat times. The starting time of each vocal line should be rounded to the nearest half note position detected by the rhythm detector. This will be implemented in the future version of LyricAlly.

To broaden its applicability, we have started to remove these limitations, most notably in the text processing module. The text module handles the classification and duration estimates of all five section types. Obtaining lyrics for use in the text analysis is a bottleneck in the system, as they are manually input. Our current focus for the text module is to find and canonicalize lyrics automatically through focused internet crawling.

Creating a usable music library requires addressing the description, representation, organization, and use of music information [8]. A single song can be manifested in a range of symbolic (e.g., score, MIDI and lyrics) and audio formats (e.g., mp3). Currently, audio and symbolic data formats for a single song exist as separate files, typically without cross-references to each other. An alignment of these symbolic and audio representations is definitely meaningful but is usually done in a manual, time-consuming process. We have pursued the alternative of automatic alignment for audio data and text lyrics, in the hopes of providing karaoke-type services with popular music recording.

10. ACKNOWLEDGMENTS

We thank the anonymous reviewers for their helpful comments in structuring the paper and Yuansheng Wu for helping us annotate the manual training data.

11. REFERENCES

[1] Arifi, V., Clausen, M., Kurth, F., and Muller, M. Automatic Synchronization of Music Data in Score-, MIDI- and PCM-Format. In *Proc. of Intl. Symp. on Music Info. Retrieval (ISMIR)*, 2003.

[2] Berenzweig, A. and Ellis, D.P.W. Locating singing voice segments within music signals. In *Proc. of Wrkshp. on App. of Signal Proc. to Audio and Acoustics (WASPAA)*, 2001.

[3] Berenzweig, A., Ellis, D.P.W. and Lawrence, S. Using voice segments to improve artist classification of music. In *Proc. of AES-22 Intl. Conf. on Virt., Synth., and Ent. Audio.* Espoo, Finland, 2002.

[4] Dannenberg, R. and Hu, N. Polyphonic Audio Matching for Score Following and Intellegent Audio Editor, In *Proc. of Intl. Computer Music Conf. (ICMC)*, Singapore, 2003.

[5] Furini, M. and Alboresi, L. Audio-Text Synchronization inside MP3 files: A new Approach and its Implementation. In *Proc. of IEEE Consumer Communication and Networking Conf.*, Las Vegas, USA, 2004.

[6] Goto, M. An Audio-based Real-time Beat Tracking System for Music With or Without Drum-sound. *J. of New Music Research*, 30(2):159-171, June 2001.

[7] Goto, M. A Chorus-Section detection Method for Musical Audio Signals. In *Proc. of IEEE Intl. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 2003.

[8] Minibayeva, N. and Dunn, J-W. A Digital Library Data Model for Music. In *Proc. of ACM/IEEE-CS Joint Conf. on Digital Libraries (JCDL)*, 2002.

[9] Musicnotes.com. Commercial sheet music resource. http://www.musicnotes.com.

[10] Nwe T.L., Wei, F.S. and De Silva, L.C. *Stress Classification Using Subband Based Features*, IEICE Trans.on Info. and Systems, E86-D (3), pp. 565-573, 2003.

[11] Shenoy, A., Mohapatra, R. and Wang, Y. Key Determination of Acoustic Musical Signals. In *Proc. of the Int'l Conf. on Multimedia and Expo (ICME)*, Taipei, Taiwan, 2004.

[12] Turetsky, R. J. and Ellis, D.P.W. Ground Truth Transcriptions of Real Music from Force-aligned MIDI Syntheses. In *Proc. of Intl. Symp. On Music Info. Retrieval (ISMIR)*, 2003.

[13] Tzanetakis, G. Song-specific bootstrapping of singing voice structure. In *Proc. of the Int'l Conf. on Multimedia and Expo (ICME)*, Taipei, Taiwan, 2004.

[14] Wang , C.K., Lyu, R.Y. and Chiang, Y.C. An Automatic Singing Transcription System with Multilingual Singing Lyric Recognizer and Robust Melody Tracker. In *Proc. of EUROSpeech*, Geneva, Switzerland, 2003.

[15] Waugh , I. Song Structure. *Music Tech Magazine*, 2003.

[16] Weide, R. CMU Pronouncing Dictionary (release 0.6, 1995). http://www.speech.cs.cmu.edu/cgi-bin/cmudict