

# Enriching Music with Synchronized Lyrics, Images and Colored Lights

Gijs Geleijnse, Dragan Sekulovski, Jan Korst,  
Steffen Pauws, Bram Kater, Fabio Vignoli  
Philips Research  
High Tech Campus 34  
Eindhoven, the Netherlands  
firstname.lastname@philips.com

## ABSTRACT

We present a method to synchronize popular music with its lyrics at the stanza level. First we apply an algorithm to segment audio content into harmonically similar and/or contrasting progressions, i.e. the stanzas. We map the stanzas found to a sequence of labels, where stanzas with a similar progression are mapped to the same label. The lyrics are analyzed as well to compute a second sequence of labels. Using dynamic programming, an optimal match is found between the two sequences, resulting in a stanza-level synchronization of the lyrics and the audio. The synchronized lyrics can be used to compute a synchronized slide show to accompany the music, where the images are retrieved using the lyrics. For an additional enrichment of the experience, colored light effects are synchronized with the music that are computed from the sets of images. The song segmentation can be done reliably, while the mapping of the audio segments and lyrics gives encouraging results.

## Categories and Subject Descriptors

H.3.1 [Information Storage and Retrieval]: Content Analysis and Indexing; H.3.3 [Information Search and Retrieval]: Query formulation; H.5.5 [Information Interfaces and Presentation]: Sound and Music Computing

## General Terms

Algorithms, Multimedia, Languages, Experimentation

## Keywords

Ambience Creation, Lyrics Synchronization, Term Identification, Color extraction, Images, Search engines, World Wide Web

## 1. INTRODUCTION

Traditionally, cd-booklets contain the lyrics of the accompanying cd. Music distributed through the Internet however is not accompanied with its lyrics. The popularity of the large lyrics sites on the

Web<sup>1</sup> underlines the interest of users in the lyrics of the music they listen to.

In this work we present a method to synchronize music and lyrics at the stanza level. A synchronization on this level enables people to read along with the words and directly grasp the meaning of the text. Moreover, by coupling the lyrics to the music, people can easily browse through a song. For example, by selecting a chorus in the lyrics, a user can directly hear the chorus of the song.

Light has been used only in conjunction with other media to enrich the media experience. From the disco lights with music, to the ambiLight<sup>TM</sup> television, light effects prove to enhance the experience. The effect is pronounced the most when the light effects are directly connected and time synchronized to the other media. Atmosphere creation with lighting however, is possible without the media it was derived from. We do so by using an intermediate medium, i.e. the lyrics of the song.

The features that are commonly used in atmosphere creation for music are low or mid level, for example volume, pitch, harmonic progression, etc. Using such features provides a good temporal correlation between the light effects and the music, but often fail to establish a semantic connection. In [8] a method is presented to enrich music with automatically computed light effects. The CIE [6, 22] lightness, chroma and hue of the light produced by light units is based on the volume, complexity of the sound and the pitch of the audio played. The results from the user study in [8] show that the users found the lightness and chroma well connected to the music. The hue of the lights, when automatically produced, was seen as arbitrary. The users preferred to change the hue manually to account for semantics in the music. It is clear that people perceive a semantic connection between hues and the music due to the lyrics. For example, when the Police sing *Roxanne put on the red light* one would also expect red light effects. When Wham's joyful *Club Tropicana* is played, different colors are appropriate than when playing R.E.M.'s *Everybody Hurts*.

Higher level semantics of the lyrics of the music play an important role of the color associated with the music. This inspired us to create a method to drive the color selection based on the lyrics. Work in [19] describes a method to compute a sequence of images and colored lights from a text. Terms within the text are used to query a large image repository. The retrieved images are on the one hand presented in a slide show and on the other hand used to

---

<sup>1</sup>e.g. lyricsfreak.com, azlyrics.com

compute colors to be displayed through the colored lights.

Using the lyrics synchronization at the stanza level, we can accomplish an approximate synchronization of colors and slide show with the running of the audio of the music. Hence our aim is to accomplish an enjoyable additional experience while listening to music. Dependent on the occasion and the device, the music can be accompanied with its lyrics and/or images displayed on a screen and colored lights.

The contributions of this work are the following.

- A novel lyrics synchronization algorithm at the stanza level is presented.
- Using terms within the lyrics, we present a method to automatically create a synchronized slide-show.
- Computing the representative color from a set of images and hence enabling to create synchronized meaningful light effects.

This work is organized as follows. After having discussed related work in Section 2, we discuss the general outline of the lyrics synchronization method in Section 3. In Section 3.1 we discuss the segmentation of the audio, and in Section 3.2 the labelling of the lyrics. The method to align the two is presented in Section 3.3. Synchronizing music with relevant images and colors is presented in Section 4. Finally, experimental results are given in Section 6 and we conclude in Section 7.

## 2. RELATED WORK

LyricAlly [21] is a system where music and lyrics are being aligned at the lyrics line level. However, the song structure of *verse-chorus-verse-chorus-bridge-outro* is pre-assumed. This simplifies both the tasks of categorizing the audio and the lyrics. We present a more generalized lyrics synchronization method, where we discover both the structure of the audio and the lyrics before aligning the two. This work is continued in [7], where a synchronization on the syllable level is achieved using a dynamic programming table of the lyrics and the texts recognized by a speech recognizer. Contrary to our approach, this work is language dependent, as the speech recognizer needs to be trained for a specific language.

Chen et al. [2] use singing voice detection [15] to segment music. The vocals within the audio are processed by a speech recognizer. The output is then synchronized with the lyrics using forced alignment. Although the algorithm is evaluated only on six (Chinese) songs, the method is promising. However, it is currently unclear how the method performs on other music genres.

In [20] Shamma et al. present MusicStory, where they introduce the concept of music slide show creation and editing using lyrics. Our work differs from MusicStory in a number of aspects. We scan the lyrics for terms rather than single words, leading to more relevant images. Moreover, in MusicStory no synchronization of the images and lyrics is applied.

The creation of light effects is to our best knowledge only addressed in [8]. In this work however, the colors were selected on audio features rather than on the lyrics.

Earlier work on the retrieval of images relevant to music can be found in [18], where album covers are retrieved given an artist and album title. We do not focus specifically on album covers, although album covers typically appear among the images when the title is

one of the identified terms (e.g. *Yellow Submarine*).

## 3. GLOBAL OUTLINE OF THE LYRICS SYNCHRONIZATION

The stanza-level lyrics synchronization consists of three steps.

1. We segment the audio into stanzas (i.e. verses, choruses, bridges and the like) by analyzing the harmonic progressions. We label each of the identified stanzas, assigning the same label to stanzas with a similar harmonic progression.
2. We analyze the lyrics and label repeating stanzas as choruses, which are assigned with identical labels. To identify similar verses, we analyze the textual structure of the remaining stanzas. Two stanzas with similar structures are assigned the same label.
3. Having computed two sequences of labels, we compute a best match between the two using a dynamic programming table.

### 3.1 Identifying the structure of the audio

We want to find a segmentation of the audio into compositional elements like intro, verse, chorus, bridge and outro. To this end, we capitalize on musicological observations in rock music [3] informing us, for instance, that verses contain similar harmonic progressions that can be different from a chorus progression. Also, bridges have a progression that is contrasting to all other sections in a song.

The first music feature extraction step consists of identifying the beat period and the beat phase of the musical audio, similar to a technique described in [9]. This allows us to perform a beat-synchronous chroma vector analysis [17]. A chroma vector is a 12-element vector that summarizes the spectral content around the frequencies that correspond with the twelve pitches in a chromatic scale over a range of octaves within Western tonal music.

We now have computed a sequence of chroma vectors  $O_1^T = o_1, \dots, o_T$  that can serve as a harmonic progression representation of musical audio. A preliminary problem definition is formulated as segmenting this vector sequence  $O_1^T$  into  $L$  consecutive segments that represent either similar, say repeated, or contrasting harmonic progressions. For simplicity, we assume the number of segments  $L$  to be known. A sensible number is the number of stanzas found in the lyrics analysis stage (Section 3.2).

In order to measure the concept of two similar or two dissimilar chroma vectors, we use the *cosine similarity measure*  $s(t, k)$  between two vectors  $o_t$  and  $o_{t+k}$  at time index  $t$  and  $t+k$  and its *versed* version  $d(t, p)$ , respectively,

$$s(t, k) = \frac{o_t \cdot o_{t+k}}{|o_t| |o_{t+k}|}, \quad (1)$$

$$d(t, p) = 1 - s(t, p). \quad (2)$$

To extend this concept to chroma vector sequences, we use the mean value of similarities between all pairs of chroma vectors, with one vector from a given segment  $0 \leq l < L$  that extends from a

time index  $i$  to  $j$  and one vector that is a given number of time indices apart from the first one. In other words,

$$S(i, j, \hat{K}_l) = \frac{1}{(j-i) |\hat{K}_l|} \sum_{t=i}^j \sum_{k \in \hat{K}_l} s(t, k). \quad (3)$$

where  $\hat{K}_l$  is a set of time index differences (*lags*) between the chroma vectors in segment  $l$  and other parts in the chroma vector sequence. The set  $\hat{K}_l$  is selected such that all time index differences result in maximally similar chroma vector sequences and that all compared sequences are non-overlapping.

Likewise, we also compute harmonically dissimilar sequences by comparing all chroma vectors in a given segment  $l$  and those that are  $\hat{p}_l$  time indices apart. Thus,

$$D(i, j, \hat{p}_l) = \frac{1}{(j-i)^2} \sum_{t=i}^j \sum_{q=0}^{j-i} d(t, \hat{p}_l + q - t + i). \quad (4)$$

where  $\hat{p}_l$  is the time index difference that provides the maximally dissimilar (i.e. contrasting) and non-overlapping chroma vector sequences.

Finally, the problem can be formulated as finding the segment boundaries  $\{b_0, b_1, \dots, b_L\}$  by maximizing a linear combination of the harmonic similarities  $S(i, j, \hat{K}_l)$  and harmonic contrasts  $D(i, j, \hat{p}_l)$  over the complete chroma spectrum vector subsequence. This can be efficiently computed using a dynamic programming based on level building [14]. Additionally, at each iteration in building a single level, an optimization step is required to find the best  $\hat{K}_l$  and  $\hat{p}_l$ . The details on the audio segmentation algorithm are provided elsewhere [16].

By backtracking the optimal path in the dynamic programming table, one can easily find the segment boundaries  $\{b_0, b_1, \dots, b_L\}$ , but also the inter-segment similarities established by the linear combination of Equations 3 and 4, if they are stored in the table during the dynamic programming process. These inter-segment similarities are used in a clustering method based on singular value decomposition [4]. This allows us label the segments, resulting in sequences as ‘‘AABAABAABC’’, which resembles a contrasting verse-chorus form in rock music ended by an outro or modulated end chorus [3].

### 3.2 Identifying the structure of the lyrics

We retrieve lyrics  $l_s$  and assume that  $l_s$  is a sound transcription of the song  $s$ . In this work, we consider the lyrics to be given, for work on automatic lyrics retrieval we refer to [10] and [5].

We use the segmentation of  $l_s$  in stanzas – depicted by blank lines – to identify the structure of  $l_s$  [13]. To label each of the lyrics stanzas, we pose the following requirements.

- Identical stanzas (typically relating to multiple occurrences of the chorus) should be assigned to the same label.
- Stanzas that are textually different, but are similar in textual structure (i.e. the verses) should also be assigned to the same label.

We first identify *identical stanzas* by determining the longest common subsequence (lcs) for each pair  $(i, k)$  of stanzas. We allow minor variations, hence the length of the lcs of  $i$  and  $k$  should approach both the length of  $i$  and  $k$ . Each of such repeating stanzas is assigned the same free label, say  $A$ .

$$\begin{aligned} |i| - |\text{lcs}(i, k)| &\leq \epsilon \wedge |k| - |\text{lcs}(i, k)| \leq \epsilon \\ \implies l(i) &= l(k) \end{aligned} \quad (5)$$

Next we compare the structural properties of the remaining stanzas. We consider two stanzas to have a *similar textual structure* when they have the number of lines and number of syllables. Such stanzas may be verses with a similar harmonic progression in the audio. Structural similarity is based on determining for a given stanza the number of lines it contains and for each of these lines the number of syllables. For a given stanza  $i$ , let  $\text{nl}(i)$  denote the number of lines and let  $\sigma(i, j)$ , with  $1 \leq j \leq \text{nl}(i)$ , denote the number of syllables in its  $j$ -th line. Now, we consider two stanzas  $i$  and  $k$  to be structurally identical as follows.

$$\begin{aligned} \text{nl}(i) &= \text{nl}(k) \wedge \forall_j (|\sigma(i, j) - \sigma(k, j)| \leq u(i, j, k)) \\ \implies l(i) &= l(k) \end{aligned} \quad (6)$$

with

$$u(i, j, k) = c \cdot \max(s(i, j), s(k, j)) \quad (7)$$

for a given  $c$ .

In our experiments we used  $u(i, j, k) = 1$ . Note that for a given word, the number of syllables is not fixed, especially in music. Moreover, exceptions may occur where the syllable counting algorithm returns an erroneous number. We therefore allow differences between the numbers of syllables in the lines of the verses.

Finally, the stanzas that are not similar to any other stanza are mapped to a free label.

#### Computing the number of syllables

Focussing on lyrics in English, we used a rule-based approach to estimate the number of syllables per word. An alternative would be to use a dictionary to look up the phonetic transcription of words where the division of syllables is displayed. However not all words can be found in such dictionaries, for example given names. Since we only need an estimation of the number of syllables per stanza, we opt for a rule-based approach.

We implemented the algorithm proposed in [11]. Here, the basic idea is that each vowel can be associated with a syllable. A number of rules compensate for silent vowels (such as the  $u$  in *query*) or double vowels (such as  $ai$  in *pair*). The rule-based approach of syllable counting in [11] however does not cover every exception. To compensate for common words that are misinterpreted by the syllable counter, we use a small dictionary to look up the number of syllables of such exceptions.

### 3.3 Matching Lyrics and Music

We assume that lyrics stanzas are given as a sequence of labels, such as  $\mathcal{L} = (ABABCAC)$ , and similarly, that the audio segments are given as a sequence of labels, such as  $\mathcal{A} = (AAACAC)$ . For easy reference, the  $i$ -th label of  $\mathcal{L}$  and  $\mathcal{A}$  are called  $\mathcal{L}_i$  and  $\mathcal{A}_i$ , respectively. Note that the number of labels in  $\mathcal{L}$  and  $\mathcal{A}$  need not be identical, for the following reasons. As audio segments may be instrumental, some  $\mathcal{A}_k$  may not have a matching  $\mathcal{L}_i$ . Alternatively, it can be that multiple successive verses are not segmented in the audio domain. Also, in the lyrics transcription some blank lines may be missing, by which multiple stanzas appear as one. Hence, it may occur that  $|\mathcal{L}| < |\mathcal{A}|$  as well as  $|\mathcal{L}| > |\mathcal{A}|$ .

To find a best match between both sequences, we aim to find a mapping  $m : \{1, 2, \dots, |\mathcal{L}|\} \rightarrow \{1, 2, \dots, |\mathcal{A}|\}$  that maps each label in  $\mathcal{L}$  to exactly one label in  $\mathcal{A}$ . A mapping  $m$  is called *feasible* if it satisfies the following constraints.

1. The mapping is order-preserving: for each pair  $i, j \in \{1, 2, \dots, |\mathcal{L}|\}$  it holds that  $i < j \implies m(i) \leq m(j)$ .
2. The mapping is consistent: for each pair  $i, j \in \{1, 2, \dots, |\mathcal{L}|\}$  it holds that  $\mathcal{L}_i = \mathcal{L}_j \implies \mathcal{A}_{m(i)} = \mathcal{A}_{m(j)}$ .

Note that the definition of an order-preserving mapping allows that multiple lyrics stanzas are assigned to the same audio segment.

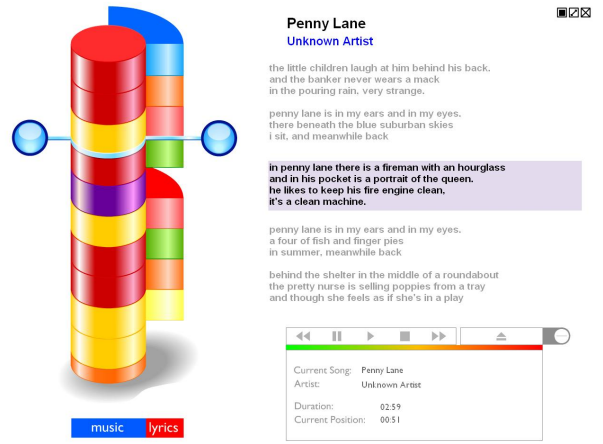
For given sequences  $\mathcal{L}$  and  $\mathcal{A}$ , we can generate all feasible mappings using a depth-first tree search strategy. The search tree consists of multiple levels, where the edges that branch from a node on level  $i$  correspond to the possible assignments of the  $i$ -th label of  $\mathcal{L}$ . An assignment made for a given node on the  $i$ -th level should be consistent with all assignments that are made on the path from the root-node to the given node. By definition, the resulting mappings will be order-preserving if  $m(i-1) \leq m(i)$ . If all generated edges satisfy both constraints, it is not difficult to see that all leaf-nodes on level  $|\mathcal{L}|$  correspond to feasible mappings.

The search tree will result in one of the following three situation:

1. There is no feasible mapping found.
2. There is exactly one feasible mapping found.
3. There are multiple feasible mappings found.

**One feasible mapping.** If there is exactly one feasible mapping found, then we consider that as the output of our algorithm.

**Multiple feasible mappings.** If there are multiple mappings, then we consider the following reduction. Although not all audio segments need to correspond to a lyrics stanza, it is generally good if the range of  $m$  under  $\mathcal{L}$  (or equivalently the image of  $\mathcal{L}$ ) is of maximum cardinality. Consequently, if there are multiple feasible mappings then we restrict ourselves to the feasible mappings of which the range is of maximum cardinality. This usually gives a considerably reduction of the feasible mappings without deleting the correct mapping.



**Figure 1: A screen shot of the lyrics synchronization application. By selecting a stanza within the lyrics, the corresponding stanza is being played.**

Of the remaining solutions with maximum cardinality, we compute the ratio of the number of syllables and the duration of the assigned audio segment. By applying an upper bound for this ratio solutions with many syllables assigned to audio segments with a short duration, will hence be filtered out. From the remaining set of solutions, we select the one that minimizes the variance of ratios.

**No feasible mappings.** If there is no feasible mapping found, then we look for a mapping that is still order-preserving, but we allow for a minimum number of inconsistencies. Given that there is no consistent, order-preserving mapping, we repeat the tree search, where we allow at most one assignment of an element from  $\mathcal{L}$  to an element  $\mathcal{A}$  that is inconsistent with earlier assignments (in the path from the root to the current node). Hence, in evaluating the search tree, we have to keep track of the partial mapping constructed so far as well as the number of inconsistencies that we allowed so far. If again no solution is found, then we can repeat the tree search allowing an additional inconsistency.

In the experiments, we used 1 as the maximal number of inconsistencies. When no mapping can be found with one inconsistency, we used a fall back mechanism where the lyrics stanzas were assigned to the audio proportionally to the number of syllables within the fragments. Hence, the audio segmentation is ignored when using the fall back mechanism.

The synchronization of music and lyrics on the stanza level enables users to browse through music using the lyrics. For example, the user can easily skip through to the chorus or the bridge (see Figure 1).

## 4. IDENTIFYING IMAGES AND COLORS

We are interested in the words in the lyrics that are the best candidates to generate relevant images. As objects depicted in images can in general be described by noun phrases (e.g. *little red corvette*, *crazy little thing called love*, *brown eyed girl*) and proper nouns (e.g. *Eindhoven*, *Prince Charles*, *Johnnie Walker*), we focus on these groups of words within the lyrics. We use these noun phrases

and proper nouns to query large image repositories<sup>2</sup>. The sizes of these image repositories enable querying these precise terms. Another benefit of querying noun phrases instead of single words is that it leads to disambiguation. For example the term *wiley, windy Moors*<sup>3</sup> reflects the moors in Yorkshire rather than the medieval Muslims in Iberia.

We identify terms within a text using a part of speech (PoS) tagger [1]. In this case we used QTag by Oliver Mason, since this package is provided with training data in English<sup>4</sup>. Since PoS taggers are typically trained on newspaper corpora, we first transform lyrics into somewhat more prosaic English. We consider each lyrics line as a sentence, so we capitalize the first letter and add punctuation marks where necessary. Then, we rewrite common slang and abbreviations. Hence, abbreviations like *gonna* and *kinda* are rewritten to *going to* and *kind of*. Words ending with *in*' are rewritten to *-ing*, thus *kiddin'* becomes *kidding*. Subsequently, we tag the adapted lyrics using QTag.

The tags are used to identify noun phrases and proper nouns. We use a regular expression  $R$  to identify the longest sequences of words that may be noun phrases or proper nouns.

$$R = (\mathbf{JJ} + \mathbf{JJR} + \mathbf{JJS})^* \cdot (\mathbf{NN} + \mathbf{NNS} + \mathbf{NP} + \mathbf{NPS})^+$$

Hence, we select the longest sequences of nouns possibly preceded by adjectives.

We send each term within the lyrics as a query to an image search engine. In our experiments we used both *Flickr.com* and *Google Images*<sup>5</sup>, where we selected the option to only download full-color images. For each term we retrieve up to  $n$  images and store them in the order as they are presented by the search engine. However, if the number of images found using a multi-word term does not surpass a threshold  $n$ , we broaden the query by removing the first words. For example, *little red corvette* can be broadened by querying first *red corvette* and consequently *corvette*. We remove duplicates by comparing the URLs of the images.

We have now obtained an annotated lyrics file, where each term is annotated with a list of at most  $n$  images.

After we have collected a set of images for term  $t$ , we identify the most representative color for this term using the images. The definition of a representative color is highly dependent on the context in which the extraction is done. The importance of an exact definition is diminished in our work because we extract a set of representative colors rather than one representative color. Most prior work on representative color extraction for single images uses the mean or the mode of the color distribution in an image, or a representative from a coherent part of the image after image color segmentation, as an estimate for the representative color. Prior work also describes use of perceptual effects like simultaneous color contrast as part of the representative color extraction.

We define a color to be in the set of representative colors if it is a lo-

<sup>2</sup>flickr.com, images.google.com

<sup>3</sup>Taken from *Wuthering Heights* by Kate Bush.

<sup>4</sup>www.english.bham.ac.uk/staff/omason/software/qtag.html

<sup>5</sup>http://flickr.com and http://images.google.com

cal maximum in the density, i.e. the local mode, of the distribution of colors found in an image or set of images and has a corrected frequency above a certain threshold.

The extraction is done in two stages. First, for each of the images in  $I_t$ , the representative colors are extracted. This is the set of colors which have a count larger than a threshold (the *histogram method*) or the set of local maxima with a small granularity (the *mean shift method*). In the case of extracting local maxima, we keep only the maxima that have a large enough contribution for the region they represent. The first step is done for two reasons: To take into account only the presence of a color, and not the relative contribution, something that in our experience would bias the distribution in the color set for all images even more; To account for the colors that are present only as noise.

Second, using the color set built previously, the overall set distribution is computed for all of the images in  $I_t$ . The distribution is then corrected to account for the non-uniformity of the distribution of colors in the set of all possible images (of which we build an estimate). After the correction, the set of representative colors is extracted by finding the local maxima of the distribution with a given granularity. The exact details depend on the algorithm that is selected. Beside extraction of the set, we also compute a relative contribution measure for every color in the set and filter the colors that have a significantly lower contribution than the one with the highest contribution in the set.

This phase resolves in an annotated lyrics file, where each term is annotated with a list of at most  $n$  images and a representative color expressed in RGB values.

## 5. SYNCHRONIZING IMAGES AND COLORS WITH THE MUSIC

The length of the stanzas is now expressed both in seconds and in the number of syllables. These parameters are used to estimate the moment the terms identified are being sung in the music. We assume that the syllables sung in the lyrics are of equal length in seconds. Hence, suppose a stanza has a length of  $s$  seconds and contains  $n$  syllables. Then the term starting at syllable  $m$  is assumed to be sung at  $\frac{s \cdot m}{n}$  seconds after the start of the stanza.

For the non-vocal stanzas, we display images and lights generated by querying both the name of the performing artist and the title of the song.

To enhance a relaxed, undisturbing atmosphere, we create smooth transitions of the colors.

## 6. EXPERIMENTAL RESULTS

As the main purpose of this work is to explore the technical feasibility of the system, we measure the performance of our algorithms in a number of separate aspects. In the next subsection, we focus on the evaluation of the audio segmentation. Section 6.2 handles the evaluation of the synchronization, while in Section 6.3 we focus on the computation of a representative color for a given term.

### 6.1 Segmenting the audio

We evaluate the audio segmentation on a ground truth collection of 60 manually segmented rock songs. We use two measures to evaluate the audio segmentation: *accuracy* assesses the locations of the computed boundaries and *precision/recall* to describe the degree

match/mismatch of the computed segments and the ground truth boundaries.

The *accuracy* of the audio segmentation on the 60 rock songs shows to be 80% within 4 beats, i.e. one musical measure, from both sides of the ground truth boundary. The accuracy is still 50% within two beats. The algorithm does not return completely wrong segmentation results as the accuracy increases rapidly towards full coverage.

The mean *precision* (focussing on the number of false boundaries) and *recall* (focussing on the missed boundaries) are 0.94 and 0.86 respectively for the set of 60 songs. For further details we refer to [16].

Hence, we conclude that the segmentation is robust enough for the intended application. In a semi-automatic setting, where computed results are manually fine-tuned, the algorithm can be of valuable assistance.

## 6.2 Aligning lyrics and audio

To evaluate the stanza-level lyrics synchronization algorithm, we applied it on 58 rock/pop songs. In order to properly evaluate the synchronization algorithm, the audio for these songs was manually segmented and labeled. We hence assume the audio segmentation and labeling to be correct. The lyrics were acquired from a lyrics site, where the partitioning in stanzas is assumed to be defined by the separating blank lines. The lyrics stanzas are automatically labeled and the number of syllables per line is automatically determined, as discussed in Section 3.2.

Giving these input data a mapping of lyrics stanzas to audio segments was constructed for each of the songs. Of all the 2157 lines appearing in these 58 songs, 1721 lines are assigned to the correct audio segment, i.e., to the audio segment in which the line is actually being sung. This means that 80% of the lines were assigned correctly.

Some of the errors that are still being made relate to the uncertainty between vocal and non-vocal audio segments. Others relate to situations where the heuristic is used to choose between multiple feasible mappings the one that results in the smallest variance in (number of syllables)/duration ratio.

The experimental results using the simple synchronization algorithm are encouraging. However, as a correct synchronization is crucial for linking terms, images and colors to the music, the current algorithm can be seen as an assistant for semi-automatic lyrics synchronization.

## 6.3 Identifying Colors for a Term

Given a term, we are interested whether the computed color is likely to be associated with the term.

In the first part of the experiment, we evaluate the computed colors for color terms like *red* and *blue* and compare the results with the same terms in a different language, Finnish. We expect these different terms, which lead to different sets of images, to generate similar colors.

In the second experiment on color computations, we compare the results using two different sources for the images: the social image site *Flickr.com* and the content found with *Google Images*.

## Color names

To subjectively assess the performance of the method, we present the colors which were computed as most relevant for a number of color names in English and Finnish in Table 1 and Figure 2. The presented results use the histogram method with 16 bins per channel and images from both Google and Flickr with a set of around 200 images for each image search engine.

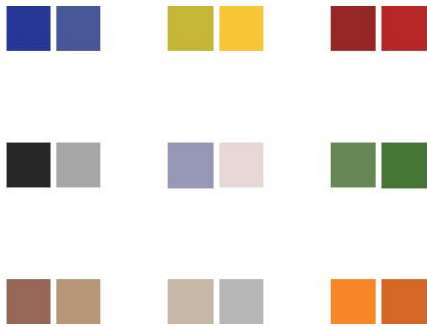
The colors with high expected chroma (blue, yellow, red, green, brown and orange) have also a high computed chroma value. Furthermore, the hue difference between the colors for the English and the Finnish terms is small. Visual inspection also shows that the results are acceptable as representatives of the colors which names are used in the method.

The colors with lower expected chroma (black, white and gray) have a chroma value in all cases smaller than the chroma values for the first group. The difference between the colors for the color names in English and Finnish is also bigger in this case. Note that for very low chroma values, the hue value is undefined. For each of the colors in this group, at least one of the colors is not acceptable as a representative for the color with the name used in the method. It has to be noted that in each of the cases, one of the other colors in the representative set was acceptable as a representative color.

	Lightness	Chroma	Hue
blue	27	59	280
sininen	38	38	282
yellow	74	63	94
keltainen	83	73	83
red	35	55	31
punainen	41	68	33
black	16	0.	315
musta	68	0.	315
white	63	17	286
valkoinen	87	6	19
green	53	28	130
vihreä	45	39	132
brown	48	25	43
ruskea	65	23	67
gray	75	11	69
harmaa	74	0.	315
orange	68	76	59
oranssi	57	69	52

**Table 1: Most representative colors for the color names in English and Finnish. The colors are given in the CIE Lch color space.**

To give a more objective assessment on the performance of the method and to give a basis for an automatic evaluation we used the results in [12], which give the acceptable range of CIE Lch hues for the colors blue, red, green and yellow. Table 2 gives an overview of the selected colors with the range of hue angles that correspond to each color. The hues for the most representative colors shown in Table 1 fall into the boundaries defined in Table 2. Furthermore, applying both the histogram with 8, 16 and 32 bins per channel and



**Figure 2: Most representative colors for nine color names in Table 1 in English and Finnish.**

the mean shift algorithm produced most representative colors with hues that fall into the boundaries given in Table 2. Taking only the images which were in the set of search results of one of the image search engines, either Google or Flickr, also produced results with hues inside the given boundaries.

	Minimum $h$	Focal $h$	Maximum $h$
red	350	24	41
yellow	75	87	101
green	112	175	222
blue	223	246	287

**Table 2: Hue angles for color regions.**

### Search engine influence

Previous results presented take images from both image search engines. It is interesting to look at possible differences between the engines, specially because they use two different kinds of indexing. We expect that Flickr, which has manual tagging would overall perform better. To present the differences we use the KL divergence. The expectation that Flickr will provide better results can be translated to an expectation of a higher divergence for the distribution computed from the set of images acquired from Flickr.

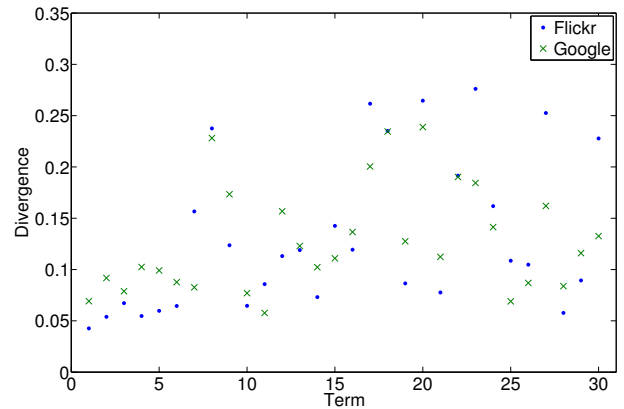
The distributions computed from the images acquired from Flickr had a mean divergence of 0.1769 with a variance of 0.1172. The Google based ones had a mean of 0.1315 and a variance of 0.0604. A paired t-test on the divergence values showed a significant effect at  $\alpha = 0.05$ .

On closer inspection of the data, we observed that the large variance in the Flickr data was due to a small number of terms (10 from the 60) which had a much higher divergence value than the others. The terms with this property were all in Finnish and the number of images acquired less than 50. On the other hand, all the Google image sets had 200 images, which was the limit used in the experiments. After these terms are filtered, the divergence distribution for the Google had a change of mean to the values of 0.1286 with a standard deviation of 0.0521. The new mean divergence for the Flickr set was 0.1325 with a standard deviation of 0.753. While the Google mean divergence changed 2.2%, the Flickr one changed 25%. A repeated paired t-test for equality of means showed no significant effect of the search engine at  $\alpha = 0.05$ .

Figure 3 shows the divergence values for the distributions com-

puted from the images acquired from Google and Flickr.

The above results show that contrary to our prior belief, for the set of terms we used we did not find a significant effect of the search engine used on the suitability of the produced image sets for representative color extraction. The number of images in the set, however, showed a significant effect.



**Figure 3: Divergence for images sets from different search engines.**

## 7. CONCLUSIONS AND FUTURE WORK

We presented a method to enrich music with synchronized lyrics, images and colored lights. We automatically synchronize the lyrics at the stanza level, providing easy access to the text of the song. As the results of the synchronization are promising, this work provides possibilities to enrich music with synchronized effects such as images and colors. With or without the retrieved images, the use of colored lights with music may be a promising new multimedia concept in the living room. The colors are computed based on the representative color associated with the terms in the lyrics. Hence, the colors are meaningful when it matters and are an enjoyable experience otherwise.

Experimental results from studies in audio segmentation, lyrics synchronization and color identification are promising. However, the system described in this work is open to further improvements. We mention the following issues for future work.

**Lyrics Selection.** Currently, we segment the lyrics of the song using blank lines. However, when we search for lyrics on the web, multiple versions – and multiple segmentations – may be found [10, 5]. Hence, in future work we plan to compute the most likely segmentation of the lyrics instead of manually selecting a well-segmented version.

**Stanza-level Lyrics Synchronization.** We currently assume the number of segments  $L$  in the audio to be given. In future work, we want to find the best possible  $L$  for segmenting the audio, the number of lyrics can be used to approximate this. Singing voice detection is helpful when mapping lyrics to audio fragments, as we currently cannot discriminate between vocal and non-vocal fragments.

**Towards Fine-grained Lyrics Synchronization.** As indicated in Section 2, recently [7, 2], work is done on a more fine-grained synchronization of lyrics and audio. We plan to investigate whether the

use of these methods is suitable for western popular music. A reliable synchronization on the syllable level will improve the timing of the images and colors as well.

**Language Dependencies.** The syllable counting algorithm as described in Section 3.2 is created for the English language. Alike the approaches in [7] and [2] our method is thus language dependent. Alternatives to the current syllable counting algorithm need to be found to solve this problem.

**User Evaluation.** The results of this work show that we can automatically or semi-automatically create a synchronized atmosphere using the lyrics of the song being played. In future work, a number of open questions are to be resolved by user experiments.

- **Perception of the slide show.** How do users experience the slide show? We currently select a fixed number of terms per stanza, but this might not be optimal. The tempo of the song might effect the best appreciated pace for the slide show. Moreover, we can vary on the number images to display.
- **Perception of the colored lights.** Do the users appreciate the connection of the lights to the lyrics? It is interesting to find out whether users recognize the connection between lyrics and light effects, without the images displayed.
- **Overall perception.** How is the overall user experience? User test need to point out whether the lights and images indeed enrich the listening experience.

## 8. REFERENCES

- [1] E. Brill. A simple rule-based part-of-speech tagger. In *Proceedings of the third Conference on Applied Natural Language Processing (ANLP'92)*, pages 152 – 155, Trento, Italy, 1992.
- [2] K. Chen, S. Gao, Y. Zhu, and Q. Sun. Popular song and lyrics synchronization and its application to music information retrieval. In *Proceedings of the Thirteenth Annual Multimedia Computing and Networking Conference 2006*, San Jose, CA, January 2006.
- [3] J. Covach. Form in rock music. In D. Stein, editor, *Engaging Music: Essays in Music Analysis*, pages 65 – 76. 2005.
- [4] J. Foote and M. Cooper. Media segmentation using self-similarity decomposition. In *Proceedings of SPIE Storage and Retrieval for Multimedia Databases*, pages 167 – 175, 2003.
- [5] G. Geleijnse and J. Korst. Efficient lyrics extraction from the web. In R. Dannenberg, K. Lemström, and A. Tindale, editors, *Proceedings of the Seventh International Conference on Music Information Retrieval (ISMIR'06)*, pages 371 – 372, Victoria, Canada, 2006. University of Victoria.
- [6] International Commission on Illumination (CIE). website: <http://www.cie.co.at/cie/>.
- [7] D. Iskander, Y. Wang, M.-Y. Kan, and H. Li. Syllabic level automatic synchronization of music signals and text lyrics. In *MULTIMEDIA '06: Proceedings of the 14th annual ACM international conference on Multimedia*, pages 659 – 662, Santa Barbara, CA, 2006.
- [8] B. Kater. Music based light effects. Master's thesis, Eindhoven University of Technology, 2005.
- [9] A. Klapuri, A. Eronen, and J. Astola. Analysis of the meter of acoustic musical signals. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(1):342 – 355, 2006.
- [10] P. Knees, M. Schedl, and G. Widmer. Multiple Lyrics Alignment: Automatic Retrieval of Song Lyrics. In *Proceedings of 6th International Conference on Music Information Retrieval (ISMIR'05)*, pages 564 – 569, London, UK, September 2005.
- [11] H. Kucera, R. Sokolowski, and J. Rossum. Method and apparatus for text analysis, 1986. US Patent no. 4,773,009.
- [12] R. G. Kuehni. Focal color variability and unique hue stimulus variability. *Journal of Cognition and Culture*, 5(18):409–426, 2005.
- [13] J. P. G. Mahedero, Álvaro Martínez, P. Cano, M. Koppenberger, and F. Gouyon. Natural language processing of lyrics. In *MULTIMEDIA '05: Proceedings of the 13th annual ACM international conference on Multimedia*, pages 475 – 478, New York, NY, USA, 2005. ACM Press.
- [14] C. Myers and L. Rabiner. A level building dynamic time warping algorithm for connected word recognition. *IEEE Trans. on Acoustics, Speech and Signal Processing*, ASSP-29(2):284 – 297, 1981.
- [15] T. L. Nwe, A. Shenoy, and Y. Wang. Singing voice detection in popular music. In *MULTIMEDIA '04: Proceedings of the 12th annual ACM international conference on Multimedia*, pages 324–327, New York, NY, USA, 2004. ACM Press.
- [16] S. Pauws. Automatically discovering structure in popular music. In W. Verhaegh, E. Aarts, W. ten Kate, J. Korst, and S. Pauws, editors, *Proceedings Third Philips Symposium on Intelligent Algorithms (SOIA 2006)*, pages 73 – 84, Eindhoven, the Netherlands, December 2006.
- [17] S. Pauws. Extracting the key from music. In *Intelligent Algorithms in Ambient and Biomedical Computing*, Philips Research Book Series, pages 119 – 132. Springer, 2006.
- [18] M. Schedl, P. Knees, T. Pohle, and G. Widmer. Towards Automatic Retrieval of Album Covers. In *Proceedings of the 28th European Conference on Information Retrieval (ECIR'06)*, London, UK, April 2006.
- [19] D. Sekulovski, G. Geleijnse, B. Kater, J. Korst, S. Pauws, and R. Clout. Enriching text with images and colored light. In *Proceedings of the IS&T/SPIE 20th Annual Electronic Imaging Symposium*, San Jose, CA, 2008.
- [20] D. A. Shamma, B. Pardo, and K. J. Hammond. Musicstory: a personalized music video creator. In *MULTIMEDIA '05: Proceedings of the 13th annual ACM international conference on Multimedia*, pages 563 – 566, Singapore, 2005.
- [21] Y. Wang, M.-Y. Kan, T. L. Nwe, A. Shenoy, and J. Yin. Lyrically: automatic synchronization of acoustic musical signals and textual lyrics. In *MULTIMEDIA '04: Proceedings of the 12th annual ACM international conference on Multimedia*, pages 212 – 219, New York, NY, 2004.
- [22] G. Wyszecki and W. S. Styles. *Color Science: Concepts and Methods, Quantitative Data and Formulae (2nd ed.)*. John Wiley and Sons, Inc., New York, NY, 1982.