A Cross-modal Approach for Karaoke Artifacts Correction

Wei-Qi Yan[1] and Mohan S Kankanhalli[2]

[1] University of California, Irvine, CA 92697-3425
[2] National University of Singapore, Singapore 117543

Abstract. Karaoke singing is a popular form of entertainment in several parts of the world. Since this genre of performance attracts amateurs, the singing often has artifacts related to scale, tempo, and synchrony. We have developed an approach to correct these artifacts using cross-modal multimedia streams information. We first perform adaptive sampling on the user's rendition and then use the original singer's rendition as well as the video caption highlighting information in order to correct the pitch, tempo and the loudness. A method of analogies has been employed to perform this correction. The basic idea is to manipulate the user's rendition in a manner to make it as similar as possible to the original singing. A pre-processing step of noise removal due to feedback and huffing also helps improve the quality of the user's audio. The results are described in the paper which shows the effectiveness of this multimedia approach.

Key Words: Adaptive Sampling, Artifacts Handling, Karaoke.

1 Introduction

A multimedia environment $\Pi(t)$ usually consists of a multiplicity of correlated data streams $\Pi_i(t)$ with $(n \geq 2)$:

$$\Pi(t) = \{\Pi_i(t), i = 0, 1, 2, \cdots, n; t \in (0, +\infty)\} \tag{1}$$

The correlations $\mathcal{R}$ among them can be expressed as:

$$\mathcal{R} = \{\sim: \Pi_p(t) \sim \Pi_q(t), 0 \leq p, q \leq n\} \tag{2}$$

Karaoke ("missing orchestra" in Japanese) is an example of such a multimedia environment.This Japanese entertainment form features a live singer with pre-recorded accompaniment. The user sings into a microphone while the stored music is played simultaneously. Karaoke is tremendously popular in eastern Asia as an avenue for recreation and entertainment. Some of the distinctive characteristics of karaoke are:

– It encourages artistry in which users try to emulate the original singer in terms of timbre and expression;

– The dynamic highlighting of the song caption along with the music provides synchronization cues to the user;
– Its appeal lies in the immediate feedback of the experience to the user;
– It is a vicarious means of experiencing concert singing;
– Users usually have a portfolio of songs to sing and they tend to improve the quality of rendition with practice.

The karaoke system is usually based on a CD player. The CD multimedia data includes one video stream and one audio stream which is the musical accompaniment. The system has two audio channels by which the live singing and the accompaniment channels are mixed and played through the speakers. The original rendering of the singing is usually available on the CD.
Given that most users are not professionally trained singers, their rendition often includes some artifacts related to pitch, tempo and tune. There are several reasons for the cause of artifacts [17]:

– Amateurs sometimes cannot sing a song in the proper key and they jump from one scale to another freely. In high key singing, exhaustion often leads to artifacts.
– Some karaoke users are not able to maintain a regular beat, which can vary from being very fast to being too slow.
– The timbre of the singing may be very different from that of the original singer. This is because a singer needs to be trained so as to properly exercise the vocal cords in order to produce a rich timbre. Moreover, accents can also affect the quality of the rendition.
– There can be noise artifacts due to sensitive pickups and feedback. A huffing sound is produced by a sensitive microphone which amplifies the breathing sounds. A piercing screech can sometimes be heard due to amplifier feedback especially when the speakers are close to the microphone.

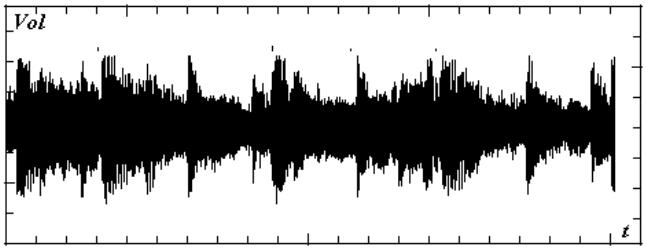


Fig. 1. A professional singer's waveform with music accompaniment

Such artifacts, which are annoying to the karaoke users and watchers, occur frequently as shown in Fig. 1 and Fig. 2. Our goal of karaoke artifacts handling

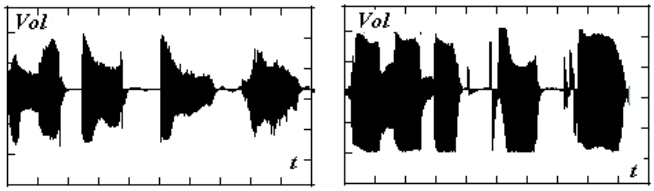


Fig. 2. Two amateur singers' waveforms without music accompaniment

is to remove or mitigate the effects of the artifacts in the karaoke audio. The key idea is to use the original singer's rendition as a guide to correct the artifacts. Given that there is correlation and redundancy among audio and video streams, we exploit the cross-modal information in order to perform artifact removal. We will thus judiciously employ all of the available information in order to perform the correction.

In this paper, we combine adaptive sampling in conjunction with video analogies (VA) to correct the audio stream in the karaoke environment $\kappa = \{\kappa(t) : \kappa(t) = (U(t), K(t)), t \in (t_s, t_e)\}$ where t_s and t_e are start time and end time respectively, $U(t)$ is the user multimedia data. We employ multiple streams from the karaoke data $K(t) = (K_V(t), K_M(t), K_S(t))$, where $K_V(t)$, $K_M(t)$ and $K_S(t)$ are the video, musical accompaniment and original singer's rendition respectively along with the user multimedia data $U(t) = (U_A(t), U_V(t))$ where $U_V(t)$ is the user video captured with a camera and $U_A(t)$ is the user's rendition of the song. We analyze the audio and video streaming features $\Psi(\kappa) = \{\Psi(U(t), K(t))\} = \{(\Psi(U(t)), \Psi(K(t)))\} = \{\Psi_U(t), \Psi_K(t)\}$, to produce the corrected singing, namely output $U'(t)$, which is made as close as possible to the original singer's rendition. Note that Ψ represents any kind of feature processing.

Fig. 3 summarizes the research problem tackled in this paper. In a Karaoke system, the input is video stream $K_V(t)$, music stream $K_M(t)$ and user audio stream $U_A(t)$. After multiple stream segmentation, the tune, tempo and loudness of audio stream are adjusted aligning to the audiovisual information extracted from the original performance. The corrected audio stream is mixed with the music stream together as the output.

Although an audio clip has a plurality of features, not all of them are useful for our purpose. In this paper, we use three features – *pitch*, *tempo* and *loudness* for removing artifacts in order to produce a rendition as close to the original as possible. We have selected pitch, tempo and loudness as features since they are the primary determinants of the quality of a rendition. Moreover, they are relatively easy to compute and manipulate (which is what we need to do in order to remove the artifacts).

This research is a part of our overall program in multimedia (video, audio and photographs) artifacts handling. We detect and correct those artifacts generated by limitations of either handling skills or consumer-quality equipment. The basic

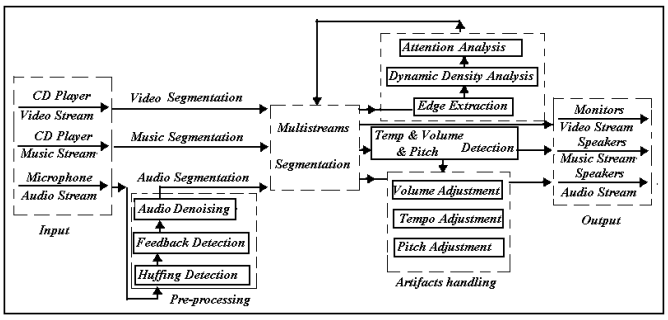


Fig. 3. Flowchart for the cross-modal karaoke correction

idea is to perform multimedia analysis in order to attenuate the effect of annoying artifacts by feature alteration [13].

2 Related work

Given the popularity of karaoke, there has been a lot of work concerning pitch correction, key scoring, gender-shifting, spatial effects, harmony, duet and tempo & key control [5][6] [7][8]. What is noteworthy is that most of these techniques work in the analog domain and are thus not applicable in the digital domain.

Interestingly, most of the work has been published as patents. Also, they all attempt to adjust the karaoke output since most karaoke users are amateur singers. The patent [7] detects the actual gender of the live singing voice so as to control the voice changer to select either of the male-to-female and female-to-male conversions if the actual gender differs from the given gender so that the pitch of the live singing voice is shifted to match the given gender of the karaoke song. In the patent [5], a plurality of singing voices are converted into those of the original singers voice signals. In patent [8], the pitches of the user sound input and the music are extracted and compared in order to change it.

Textual lyrics [12] have been automatically synchronized with acoustic musical signals. The 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. Actually, this can be considered to be an elementary karaoke system with sentence level synchronization.

Our work is distinct from the past work in two ways. First, it works entirely on digital data. Second, we use correlated multimedia streams of both audio and video to effect the correction of artifacts. We believe that this approach of using multiple data streams for artifact removal has applications beyond that of karaoke.

3 Background

3.1 Adaptive sampling

Given the voluminous nature of continuous multimedia data, it is worth using sampling techniques to filter each media stream $\Pi_i = \{\pi_{i,j}, j = 0, 1, \cdots, m\}$, in order to produce relevant samples or frames $\pi_{i,j}$. We use a simplified version of the experiential sampling technique for doing adaptive sampling [4]. It utilizes $N_S(t)$ number of sensor samples $S(t)$ to deduce $N_A(t)$ number of attention samples $A(t)$ which are the relevant data. The advantage is that we can then focus only on the relevant parts of the stream: $\Pi_i = \{\pi_{i,j}, j = 0, 1, \cdots, m\}$ and ignore the rest. i.e.

$$T(N_s(t)_{i,j}, N_A(t)_{i,j}) \geq T_{es} \tag{3}$$

$T(\cdot)$ is the decision function defined by norm L_2 on the domain, such as temporal, spatial or frequency domain, and T_{es} is the sampling threshold. The final samples are obtained by re-sampling: $\Pi'_i = \{\pi'_{i,j}, j = 0, 1, \cdots, m'; m' \leq m\}$ which is precisely the relevant data. Adaptive sampling is primarily for the purpose of efficiency given the real-time requirement of the processing. Here is the concise definition of adaptive sampling:

If $\forall t \in [t_s, t_e]$, equation (3) is true, t_s and t_e are the start time and the end time respectively, then the set Π'_i=$\{\pi'_{i,j} : N_A(t)_{i,j} > 0; j = 0, 1, \cdots, m'; m' \leq m\}$ is the adaptively sampled stream of a multimedia stream $\Pi_i = \{\pi_{i,j}, j = 0, 1, \cdots, m\}$; and $\Pi'(t) = \{\Pi'_i(t), i = 0, 1, \cdots, n\}$ is the adaptively sampled multimedia environment $\Pi(t)$.
The adaptive sampling approach (algorithm 1) basically provides a solution for the detection of the dynamically changing data.

3.2 Video analogies

In automatic multimedia editing, we would like to process and transform the existing data into a better form. Video analogies [14] use a two-step operation involving learning and transfer of features: $\Psi(t) = \Psi(\Pi(t)) = \{\Psi(\Pi_i(t)), i = 0, 1, \cdots, n\} = \{\Psi_i(t), i = 0, 1, \cdots, n\}$. It learns the *ideal* from an exemplar and then *transform* the given data and emulates the exemplar as closely as possible. In order to set up the analogy, the given data and the exemplar data should have at least one common feature that is comparable.

Definition:(Media comparability) If $\overline{\Psi}_p(t)$= $\overline{\Psi}_q(t)$, $\forall \psi_{pr} \in \Psi_p(t)$, $\exists \psi_{q,s} \in \Psi_q(t)$, $d(\psi_{p,r}, \psi_{q,s}) = \|\psi_{p,s} - \psi_{q,t}\| < \varepsilon, \varepsilon > 0$, $r, s = 0, 1, \cdots, m$; then $\Psi_p \in \Psi(t)$ is *comparable* to $\Psi_q \in \Psi(t)$, $p, q = 0, 1, \cdots, n$; $t \in (-\infty, +\infty)$, denoted as $\Psi_p \approx \Psi_q$ where $\overline{\Psi}_p(t)$ and $\overline{\Psi}_p(t)$ are the rank of the sets. $\Re_{p,q}$ =$\{\approx |\Psi_p(t) \approx \Psi_q(t)$, $\Psi_p(t), \Psi_q(t) \in \Psi(t)$, $\overline{\Psi}_p(t)$= $\overline{\Psi}_q(t)$, $p, q = 0, 1, \cdots, n$; $t \in (-\infty, +\infty)\}$.
The underlying idea of video analogies (**VA**) is that given a source video Π_p and its feature Ψ_p, a target video Π_q and its feature Ψ_q, we seek feature correspondence between the two videos. This learned correspondence is then applied to generate a new video $\Pi'_q = \{\pi'_{q,j}, j = 0, 1, \cdots, m\}$. Our overall framework is succinctly captured by algorithm 2.

Input : Multimedia stream $\Pi_i(t)$

Output : Multimedia samples $\Pi'_{i,m'}$

Procedure:

Initialization: $t = 0$;
$N_s(t) = N_s(0)$;
$N_A(t) = N_A(0) = 0$;
$m' = 0$;
while $t \leq t_e - 1$ **do**
  for $i = 0, \ldots, n$ **do**
    $S_i(t) \leftarrow \Pi_i(t)$; //randomly sample one stream;
    $\omega_i(t) = \|\Pi_i(t) - \Pi_i(t-1)\|_{S_i}$; //Estimate samples;
    $\delta(t) = rand(t) > 0$; //Change the attention numbers, $rand(t)$ is a random number;
    if $\omega_i(t) > T_{es}$ **then**
      $N_{A_i}(t) \leftarrow N_{A_i}(t) + \delta(t)$;
    else
      $N_{A_i}(t) \leftarrow N_{A_i}(t) - \delta(t)$;
    end
    if $N_{A_i}(t) > 0$ **then**
      $A_i(t) \leftarrow (A_i(t-1), S_i(t))$; $\Pi'_{i,m'} \leftarrow \Pi_i(t)$; //perform resampling ;
      m'++; //Consider another media stream;
    else
      $N_{A_i}(t) = 0$;
    end
    GetTime(t); //Get current time for next iteration;
  end
end

Algorithm 1: Adaptive sampling

Input : Source video Π_p, target video Π_q

Output : The new target video Π'_q

Procedure:

$\Psi_p \leftarrow \Psi(\Pi_p)$;// extract features

$\Psi_q \leftarrow \Psi(\Pi_q)$;

$\forall c = 0, 1, \cdots, \Psi_p^{=}; \Psi_p^{=} = \Psi_q^{=}$

for $s = 0, 1, \cdots, m$ **do**

  for $k = 0, 1, \cdots, m$ **do**

    if $d(\psi_{p,s}^c, \psi_{q,k}^c) \leq d(\psi_{p,s}^c, \psi_{q,t}^c)$ **then**

      $\psi_{p,s}^c \approx \psi_{q,k}^c$;// select the comparable feature

    end

  end

end

$\Psi_p \approx \Psi_q \Leftarrow \psi_{p,s}^c \approx \psi_{q,k}^c$, $\forall s = 0, 1, \cdots, m$;//propagate the feature similarity

$\Pi_p \sim \Pi_q \Leftarrow \Psi_p \approx \Psi_q$;// comparison

$f \in \Re_{p,q}$; $g \in \Re_{p,q}$;//establish mapping functions

$f : \Psi_p \rightarrow \Psi_q$;

$g : \Psi_q \rightarrow \Psi'_p$;

$\Psi'_q = (g \circ f)(\Psi_p)$;

Ψ'_q and $\Pi_q \Rightarrow \Pi'_q$;// modify data to construct a new video

Algorithm 2: Video analogies

Video analogies have the propagation feature. If the analogy is denoted by $\Psi_p^k : \Psi_q^k :: \Psi_p^j : \Psi_q^j$, then $\Psi_1^k : \Psi_2^k : \cdots : \Psi_m^k :: \Psi_1^j : \Psi_2^j : \cdots : \Psi_m^j$ is true, '::' is the separator, ':' is the comparison symbol. In this paper, we propagate the video analogies onto the audio channel and use it to automatically correct the karaoke user's singing.

Analogy is a concept borrowed from reasoning. The main idea of an analogy is a metaphor, namely "doing the same thing". For an example, if a real bicycle Fig. 4(a) (from wikipedia) can be drawn as the traffic sign as shown in Fig. 4(b), can we similarly render a real bus Fig. 4(c) (from wikipedia) as the traffic sign as Fig. 4(d)? Similarly, in video analogies, if we have some desired feature in a source video, we can try to analogously transfer it to the target video.

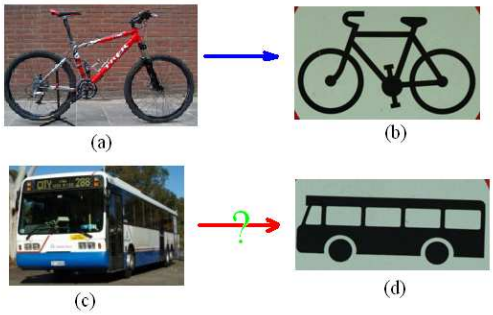


Fig. 4. An example of analogies

4 Our work

4.1 Adaptive sound adjustment

In this paper, our main idea is to emulate the performance of the professional singer in a karaoke audio. We simulate them from three key aspects: loudness, tempo and pitch. Although a perfect rendition is dependent upon many factors, these three features play a crucial role in a performance of karaoke song. Thus, we focus our artifact removal efforts on them.

Preprocessing: noise detection and removal Before we do adaptive audio adjustment for the loudness, tempo and pitch, we consider noise removal first. In a real karaoke environment, if the microphone is near the speakers, a feedback noise is often generated. Also, due to the extreme proximity of the microphone to the singer's mouth, a huffing sound is often generated.

For these two kinds of noise, we find that they have distinctive features after detecting the zero-crossing rate Eq.(4):

$$Z_0 = \frac{1}{2L}\{\sum_{l=1}^{L-1} |sgn[u_A(l)] - sgn[u_A(l+1)]|\} \cdot 100\% \tag{4}$$

where L is the window size for the processing and $sgn(n)$ is the sign function, $u_A(l)$ is the signal in a window, i.e.:

$$sgn(x) = \begin{cases} 1 & x \geq 0 \\ -1 & x < 0 \end{cases}$$

Normally, zero-crossing rate is the number of X-axis crossings for a signal and is employed to distinguish the vowels and the consonants. It is also used in audio and speech segmentation [2][3]. The zero-crossing rate of the two types of noise are shown in the following graphs (Fig. 5 and Fig. 6).

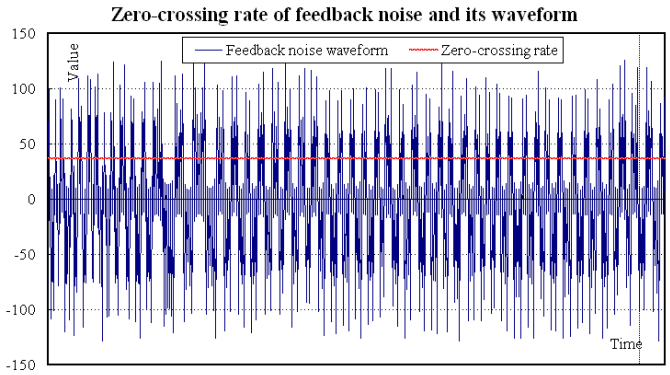


Fig. 5. Zero-crossing rate of feedback noise and its waveform

From the figure, we clearly see the zero-crossing rate of the feedback noise is a straight line since the piercing screeching sound is usually much higher-pitched than human voice. For the detection and removal of the huffing noise, we normally use the short term feature value (STFV). This value is the average in the current window Eq.(5):

$$STFV = \frac{1}{t_e^w - t_s^w} \int_{t_s^w}^{t_e^w} |u_A(t)|dt \tag{5}$$

where $u_A(t)$ is the audio signal in a window, $L = t_e^w - t_s^w$ is the windowing size. Because the huffing noise always has a high loudness, the average is high. The graphs for huffing and feedback noise are shown in Fig. 7 and Fig. 8.

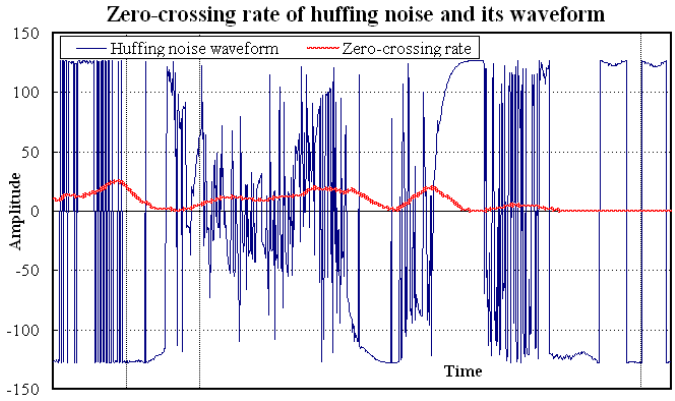


Fig. 6. Zero-crossing rate of huffing noise and its waveform

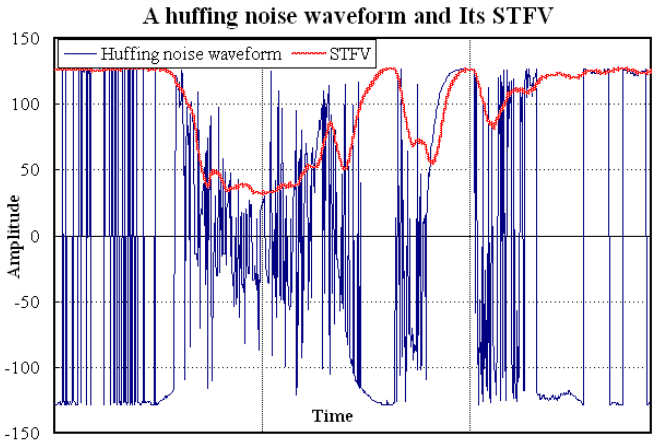


Fig. 7. The huffing noise waveform and its STFV

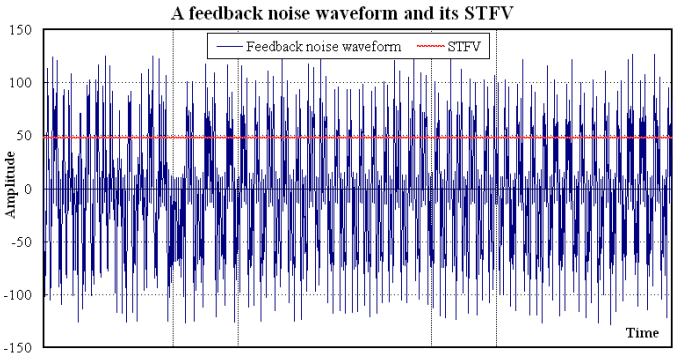


Fig. 8. The feedback noise waveform and its STFV

From Fig. 7, we see that the STFV has a high amplitude and it reflects the features of the huffing noise. What is interesting is that the short time feature value of the feedback noise is also a horizontal straight line in Fig. 8. This suggests that the feedback noise is symmetric in an arbitrary window. Using this feature, we replace the signals by silence, because most of time, people will stop singing at this moment.

Tempo handling We regard the karaoke video music K_M as our baseline for the new rendition. All the features of the new rendition should be aligned to this baseline. The reason is that music generated by instruments is usually more accurate in beat rate, scale and key than human singing. Thus we adaptively sample the accompaniment K_M and user audio input U_A first and they are synchronized as shown in Fig. 9.

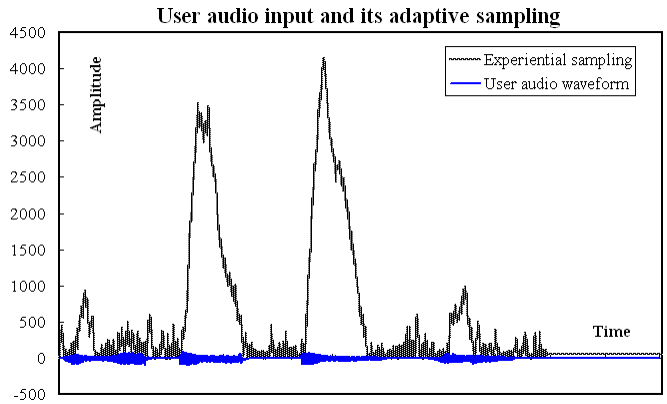


Fig. 9. User audio input and its adaptive sampling

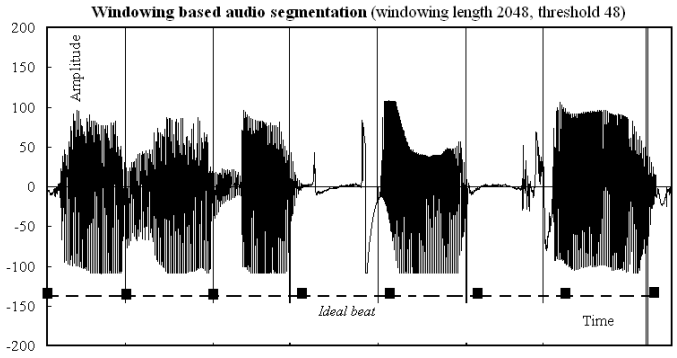
Then K_M and U_A are segmented again by the tempo or beat rate. The peak of the loudness will appear at constant intervals for a beat. The beat rate is fundamentally characterized by the peaks appearing at regular intervals. For $U_A = \{u_{a,j} > 0, j = 0, 1, 2, \cdots, m\}$, the start time $t_s^{U_A}$ and the end time $t_e^{U_A}$ are determined by the ends of the duration between two peaks. The peaks are defined by the two conditions shown in Fig. 10:

1. $u_{a,j} > \frac{1}{L}\sum_{l=j-L}^{j} u_{a,l}$, $L > 0$ is the windowing size.
2. $j \bmod L_b^{U_A} < \delta$, $L_b^{U_A}/3 > \delta > 0$, $L_b^{U_A} = t_e^{U_A} - t_s^{U_A}$ is the beat length.

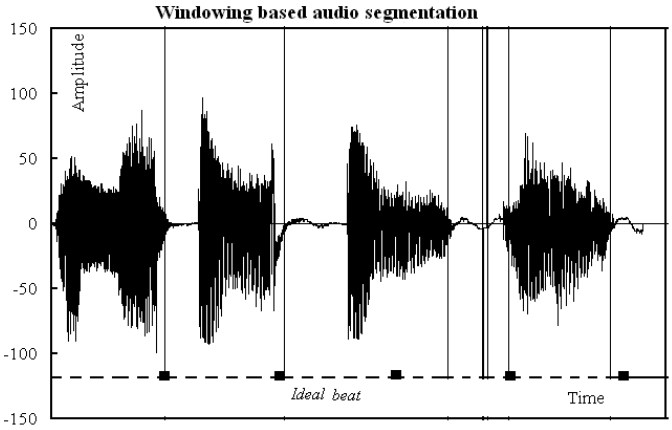
Correspondingly, for $K_M = \{k_{M_j}, j = 0, 1, \cdots, m\}$, the segmented beats are in the interval $[t_s^{K_M}, t_e^{K_M}]$ shown in Fig. 11. We can see there that the beat rate is fairly uniform.
For audio segmentation, the zero-crossing rate Eq.(4) is a powerful tool in the temporal domain. This can be seen from Fig. 12. The advantage of zero-crossing

computation is that it is computationally efficient. We compare the zero-crossing rate of the two singers' audio signals in Fig. 10.

After audio segmentation, the next step is to implement the karaoke audio correction based on analogies. Suppose the exemplar audio after segmentation is: $U_A^S(t) = \{u_A^S(i),\ i = 0, 1, \cdots, m\}$ and the user's audio after segmentation is $U_A^T(t) = \{u_A^T(j), j = 0, 1, \cdots, m\}$, thus our task is to obtain the following relationship: $U_A^T(0){:}U_A^T(1){:}\cdots{:}U_A^T(m) :: U_A^S(0){:}U_A^S(1){:}\cdots{:}U_A^S(m)$. For this, we build a mapping in the temporal domain. Subsequently, the centroid point $t_\tau^{U_A}$ should



(a) Person 1

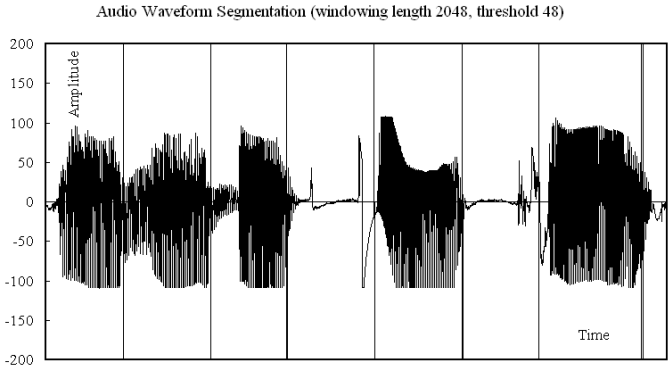


(b) Person 2

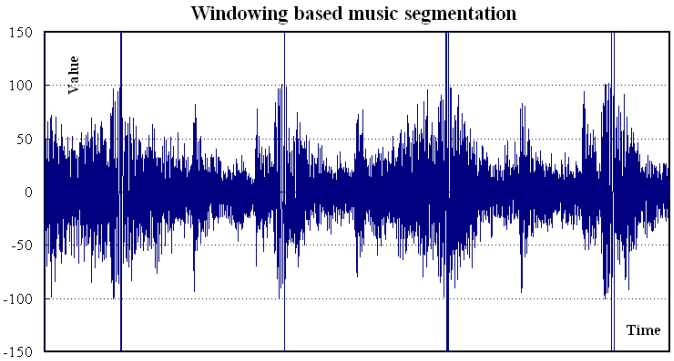
Fig. 10. Windowing based audio segmentation for different people

satisfy:

$$\int_{t_s^{U_A}}^{t_\tau^{U_A}} |u_a(t)|dt = \int_{t_\tau^{U_A}}^{t_e^{U_A}} |u_a(t)|dt \tag{6}$$



(a) Audio segmentation



(b) Music segmentation

Fig. 11. Windowing based music segmentation

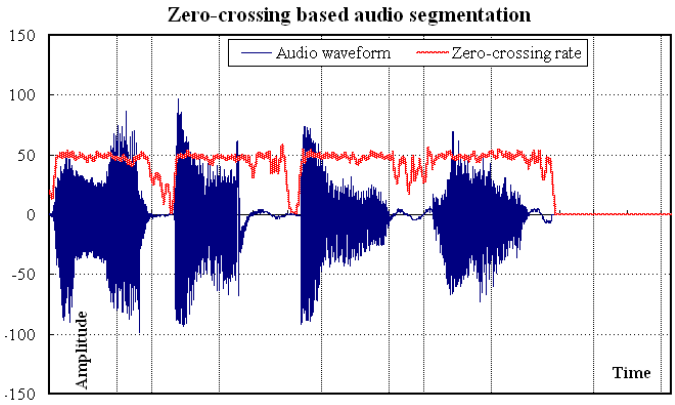


Fig. 12. Zero-crossing rate based audio segmentation

where $U_A(t) = \{u_a(t), t \in [t_s^{U_A}, t_e^{U_A}]\}$. The centroid point $t_\tau^{K_M}$ should satisfy:

$$\int_{t_s^{K_M}}^{t_\tau^{K_M}} |k_m(t)|dt = \int_{t_\tau^{K_M}}^{t_e^{K_M}} |k_m(t)|dt \tag{7}$$

where $K_M(t) = \{k_m(t), t \in [t_s^{K_M}, t_e^{K_M}]\}$. The corrected audio is then assumed to be:

$$U'_A(t) = \{u'_a(t), t \in [t_s^{K_M}, t_e^{K_M}]\} \tag{8}$$

We then cut the lagging and leading parts of the user audio input by:

$$\delta^- = \min(|t_\tau^{U_A} - t_s^{U_A}|, |t_\tau^{K_M} - t_s^{K_M}|) \geq 0 \tag{9}$$

$$\delta^+ = \min(|t_e^{U_A} - t_\tau^{U_A}|, |t_e^{K_M} - t_\tau^{K_M}|) \geq 0 \tag{10}$$

We align with the audio stream by using the following shift operation:

$$u'_a(tt) = u_a(t), tt = t_s^{K_M} + [t - (t_\tau^{U_A} - \delta^-)] \tag{11}$$

where $tt \in [t_s^{K_M}, t_s^{K_M} + \delta^- + \delta^+]$, $t \in [t_\tau^{U_A} - \delta^-, t_\tau^{U_A} + \delta^+]$.

$$u'_a(tt) = 0, tt \in (t_s^{K_M} + \delta^- + \delta^+, t_e^{K_M}] \tag{12}$$

The advantage of such cutting and shifting operations is that the most important audio information is retained and portions such as silences are cut. The basic idea is to automatically cut the redundant parts of the stream by using δ^+ and δ^-.

Tune handling Tune, as the basic melody of a piece of audio, is closely related to the amplitude of the waveform. Amateur singers easily generate a high key at the initial phase but the performance falters later due to exhaustion. To correct such artifacts in karaoke singing, we should adjust the tune gain by following the professional music and singer's audio.
From the last section, we know the $K_M(t) = \{k_m(t), t \in [t_s^{K_M}, t_e^{K_M}]\}$ and $U'_A(t) = \{u'_a(t), t \in [t_s^{K_M}, t_e^{K_M}]\}$. In order to reduce the tune artifact mentioned above, the average tune is calculated by:

$$A_{avr}^{K_M} = \frac{\int_{t_s^{K_M}}^{t_e^{K_M}} k_m(t)dt}{t_e^{K_M} - t_s^{K_M}} \tag{13}$$

$$A_{avr}^{U'_A} = \frac{\int_{t_s^{K_M}}^{t_e^{K_M}} u'_a(t)dt}{t_e^{K_M} - t_s^{K_M}} \tag{14}$$

Thus, a multiplicative factor is given by:

$$\sigma = \frac{A_{avr}^{K_M} - A_{avr}^{U'_A}}{2^{(channels \cdot 8)}} \tag{15}$$

where *channels* is the number of interleaved channels. Equation (15) is used to attenuate the high tune and amplify the low ones by using Eq.(16) for the compensation purpose:

$$u_a^*(t) = u_a'(t) * (1.0 - \sigma) + \Delta A \tag{16}$$

where $\Delta A = (A_{avr}^{K_M} - A_{avr}^{U_A'})$. We show the comparison of loudness for two pieces of audio (Fig. 13), which basically shows tune difference of two different people for the same song rendition. Our core idea for audio analogies based on beat

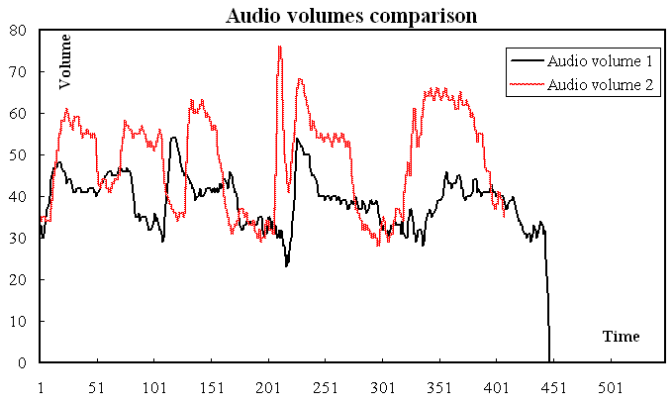


Fig. 13. Audio loudness comparison

and loudness correction algorithm is illustrated in Fig. 14. In this figure, the music waveform and the audio waveform in a beat are represented by the solid line (wave 1) and the dashed line (wave 2) respectively. We find the minimum effective interval for this beat $[t_\tau^{U_A} - \delta^-, t_\tau^{U_A} + \delta^+]$ so that the cropped audio can be aligned to the music track along the start point t_s. Simultaneously, the tune is amplified according to the equation (16).

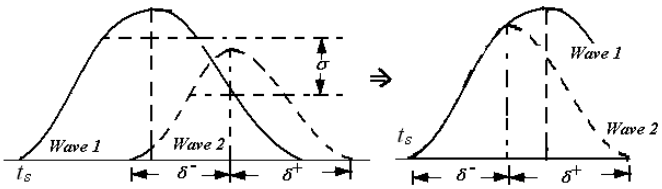


Fig. 14. Core idea for audio analogies based on beat and loudness correction

Pitch handling Pitch corresponds to the fundamental frequency in the harmonics of the sound. It is normally calculated by auto-correlation of the signal and Fourier transformation, but the auto-correlation is closely related to the windowing size. Thus, a more efficient way is to use the cepstral pitch extraction [2] [3]. In this paper, cepstrum is used to improve the audio timbre and pitch detection. Figure 15 illustrates music pitch processing. We see that the pitch using auto-correlation is not obvious while the pitch is prominent in the detection relying on cepstrum. The cepstrum is defined as the inverse discrete

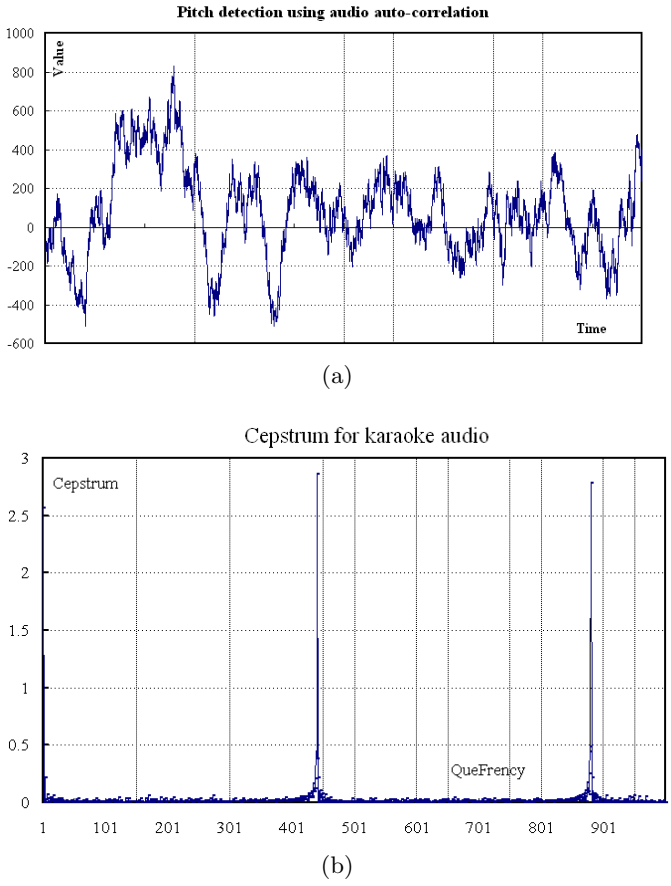


Fig. 15. Pitch detection using auto-correlation and cepstrum

Fourier transform of the log of the magnitude of the discrete Fourier transform

(DFT) of the input signal $U_A^*(x)$, $x = 0, 1, \cdots, N-1$. The DFT is defined as:

$$Y(\mu) = DFT(U_A^*(x)) = \sum_{x=0}^{N-1} U_A^*(x) e^{-j\frac{2\pi\mu x}{N}} \tag{17}$$

$Y(\mu)$ is a complex number, $\mu = 0, 1, \cdots, N-1$. The inverse Fourier transform (IDFT) is:

$$U_A^{DFT}(x) = IDFT(Y(\mu)) = \frac{1}{N} \sum_{\mu=0}^{N-1} Y(\mu) e^{j\frac{2\pi\mu x}{N}} \tag{18}$$

$x = 0, 1, \cdots, N-1$. The cepstrum $P(t)$ is:

$$P(t) = IDFT(log_{10}(|DFT(U_A^*(x))|)) \tag{19}$$

where t is defined as the quefrency of the cepstrum signal. Fig. 16 shows the spectrogram of a wave and its log spectrogram.

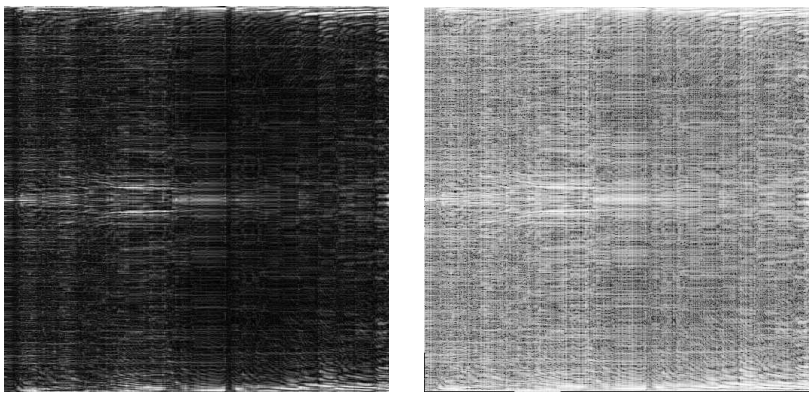
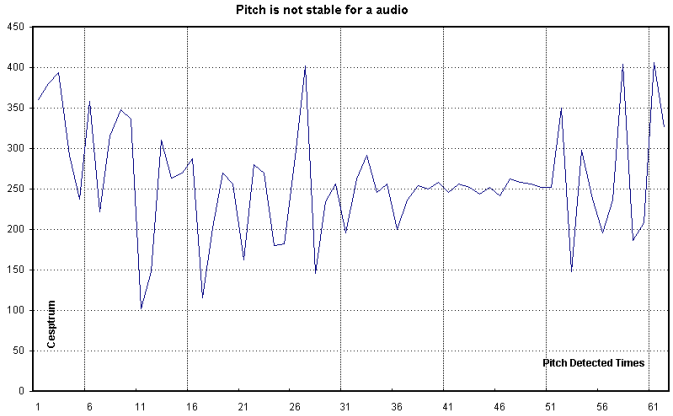


Fig. 16. Left: wave spectrogram; Right: its log spectrogram

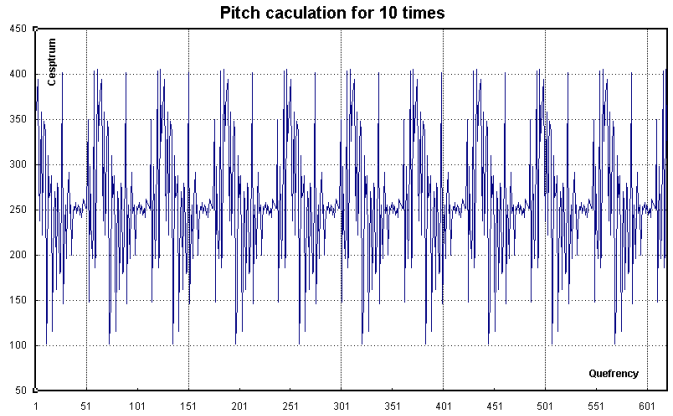
Normally, females and children have a high pitch while adult males have a low pitch. Pitch tracking is performed by median smoothing: Given windowing size $L > 0$, if

$$\frac{1}{L} \int_{-L/2+t_0}^{L/2+t_0} P(t)dt < P(t_0) \tag{20}$$

then t_0 is the pitch point. However the pitch is not stable throughout the duration of an audio clip. Pitch variations are normal as they reflect the melodic contour of the singing. Therefore we take the average pitch into account and compute the pitch over several windows as shown in Fig. 17(b).
Now we synthesize a new audio $U_A^\star(t)$ by utilizing the pitch $P_{U_A^S}(t)$ of $U_A^S(t) = \{u_a^S(t), t \in [t_s^{U_A^S}, t_e^{U_A^S}]\}$ and the pitch $P_{U_A^T}(t)$ of $U_A^T(t) = \{u_a^T(t), t \in [t_s^{U_A^T}, t_e^{U_A^T}]\}$.



(a)



(b)

Fig. 17. Pitch varies in a clip but is stable in each window

The pitch is modified by Eq.(21) [2]:

$$U_A^{\star}(x) = IDFT(Y(\mu - P_0^S + P_0^T)) \tag{21}$$

where $|Y(\mu)|$ is the amplitude of the $\mu - th$ harmonic, P_0^S and P_0^T are the pitch estimation at t_0, namely, P_0^S=$P_{U_A^T}(t_0^S)$, $P_0^T = P_{U_A^T}(t_0^T)$, $t_0^S = t_0^T = t_0$, $IDFT(\cdot)$ is the transformation by using equation (18), $U_A^{\star}(x)$ is the final audio after pitch correction. The expression (21) is visualized as the frequency response of the window, shifted in frequency to each harmonic and scaled by the magnitude of that harmonic.

4.2 Detection of highlighted video captions

Karaoke video highlighted caption is a significant cue for synchronizing the singing with the accompaniment and the video. In a karaoke environment, we play the video and accompanying music while a user is singing. The singer looks at the slow moving prompt on the captions with a salient highlight on the video so as to be in synchrony. Thus, the video caption provides a cue for a singer to catch up with the musical progression. Normally, human reaction is accompanied with a lag thus the singing is usually slightly behind the actual required timing. We therefore use the video caption highlighting as a cross-modal cue to perform better synchronization.

Although karaoke video varies in caption presentation, we assume the captions exist and have highlight on it. We detect the captions and their highlighting changes in the video frames by using the motion information in the designated region [10] [11] [16]. This is because a karaoke video is very dynamic – its shots are very short and the motion is rather fast. Also, the karaoke video usually is of a high quality with excellent perceptual clarity. We essentially compare the bold color highlighting changes of captions in each clip so as to detect the caption changes. By this segmentation based on caption changes, we can detect when the user should start or stop the singing.

We therefore segment [15] the karaoke video $K_V(t)$ =$\{k_v(x, y, t), x = 0, 1, \cdots, W-1;\ y = 0, 1, \cdots, H-1;\ t \in [t_s, t_e]\}$ first, where W and H are frame width and height respectively. Then, we detect the caption region. Since a caption consists of static characters of bold font, it is salient and distinguishable from the background. We extract the edges by using the Laplace operator Eq.(22).

$$\bigtriangledown k_v(x, y, t) = \frac{\partial k_v(x, y, t)}{\partial x} + \frac{\partial k_v(x, y, t)}{\partial y} \tag{22}$$

Normally, the first order difference is used in place of the partial derivative. With this operator, the image edges are easy to be extracted from a video frame [9]. The extracted edges are used to construct a new frame, we calculate the dynamic densities $I(\Omega, t)$ of those pixels in 8×8 blocks which are less than the threshold T:

$$I(\Omega, t) = \frac{1}{|\Omega|} \int_{\Omega} \Delta_T k_v(x, y, t) dx dy \tag{23}$$

where $\Delta_T k_v(x,y,t) = |k_v(x,y,t+\Delta t) - k_v(x,y,t)|$, Ω is the 8×8 block, $k_v(x,y,t)$ is the pixel value at position (x,y) and time t, $x = 0, 1, \cdots, W-1$; $y = 0, 1, \cdots, H-1$. The unions of these blocks are considered to be the caption region. This is also a form of adaptive sampling in video. Fig. 18 shows video captions and a detected caption region.
Finally, we detect the precise time of a caption appearance and disappearance. It is apparent that we can see a highlighted prompt moving from one side to the other clearly in a karaoke video, which reflects the progression of the karaoke. Thus, in the detected caption region, we calculate the dynamic changes of the two adjacent frames with the bright cursor moving along a straight line being considered the current prompt. The start time and the end time t are calculated by Eq.(24).

$$t = T_{k_v} \cdot \frac{Scale}{Rate} \tag{24}$$

where T_{k_v} is the $T-th$ video frame, $Scale$ is the time scale applicable for the entire video and $Rate$ is the video playing rate.

The dynamic density of a video has been calculated and shown in Fig. 19. We would like to point out that in this paper, we only do the ends synchronization for the singing of each caption. However, a more fine-grained synchronization is possible if required.

4.3 Algorithm for Karaoke adjustment

Algorithm 3 describes the overall procedure for karaoke video adjustment. It is based on the fact that all the data streams in a karaoke are of professional quality except that of the user singing. Because most users are not trained singers, their input has a high possibility of having some artifacts. However, we use the cross-modal information from the video captions and professional audio in order to correct the user's input based on the pitch, tempo and loudness. The overall procedure has been summarized in algorithm 3.

5 Results

In this section, we present the results of cross-modal approach to karaoke artifacts handling. Fig. 20 shows an example for beat and loudness correction in a piece of segmented karaoke audio based on audio analogies. Their parameters in bytes are given in Table 1.
We have presented results of experiments for audio analogies in the form of four groups of audio comparisons in Table 2. We employ Peak Signal Noise Ratio (PSNR)(dB), Signal Noise Ratio (SNR)(dB), Spectral difference (SD) and correlation between two audio clips as quality measures. The comparison between the user's singing and the original singer's rendition (which is the exemplar) before (B.) and after (A.) correction is shown in Table 2.

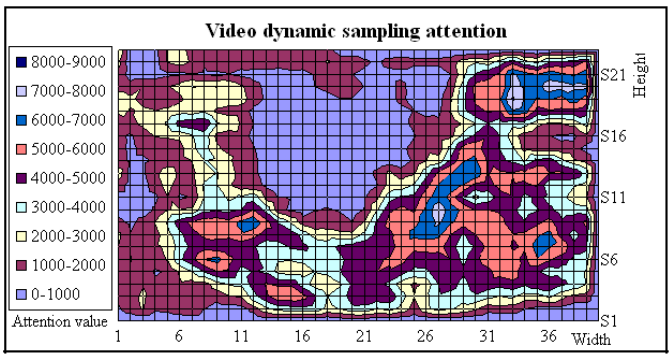
In order to understand the correspondence between numerical values (PSNR, SNR, Correlation) in Table 2 and users' subjective opinion about the quality of

(a)

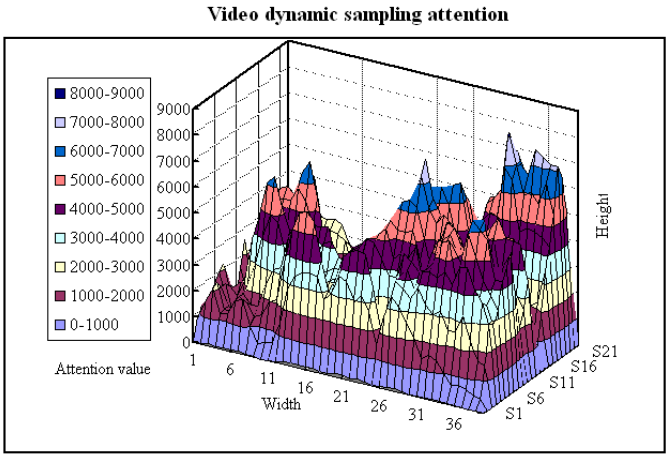
(b)

(c)

Fig. 18. A highlighted and a detected caption region



(a)



(b)

Fig. 19. 2D and 3D graphs of dynamic density for a video caption detection

Table 1. Audio parameters(Bytes) in analogies based loudness and tempo correction

Audio Parameter	Audio 1	Audio 2	Analogous Audio
Length	24998	32348	24998
Centroid	12480	16034	12480
δ^-	12480	16034	12480
δ^+	12518	16314	12518
BPS	8	8	8
σ	-	-	2.73%
Average Amplitude	41	48	46.87

Input : Karaoke Stream κ
Output : Corrected Karaoke Stream κ'

Procedure:
1. Initialize the system at $t = t_s < t_e$;
2. Input the karaoke stream $\kappa(t)$ consist of video stream $K_V(t)$, music stream $K_M(t)$ and the audio stream $U_A(t)$;
3. Denoise the input audio stream $U_A(t)$;
    3.1 Detect & remove huffing noise by using Eq.(5);
    3.2 Detect & remove feedback noise by using Eq.(4);
4. Segment the karaoke audio stream employing;
    4.1 Video segmentation [15];
    4.2 Video caption detection by using Eqs:(22)(23);
    4.3 Music tempo detection by using Eq.(5);
    4.4 Audio adaptive sampling by using Eq.(3);
    4.5 Audio segmentation by using Eqs.(4)(5);
5: Modify audio tempo using Eq.(11)(12);
6: Modify audio tune using Eq.(16);
7: Modify audio pitch using Eq.(21);
8: Output the video, music & corrected audio streams;

Algorithm 3: Karaoke artifacts handling

Table 2. Audio comparisons before (B.) and after (A.) analogies

No.	PSNR(B.)	PSNR(A.)	SNR(B.)	SNR(A.)	SD(B.)	SD(A.)	Correlation(B.)	Correlation(A.)
1	9.690989	17.22	-2.509843	-0.253588	0.022842	0.022842	0.003611	0.596143
2	9.581241	11.829815	-2.495654	-5.713023	0.014145	0.055127	0.0105338	0.023705
3	9.511368	15.53444	-2.311739	-0.266603	0.018469	0.023402	0.0161687	0.721914
4	9.581241	15.927253	-3.702734	0.044801	0.016865	0.038852	0.0105338	0.784130

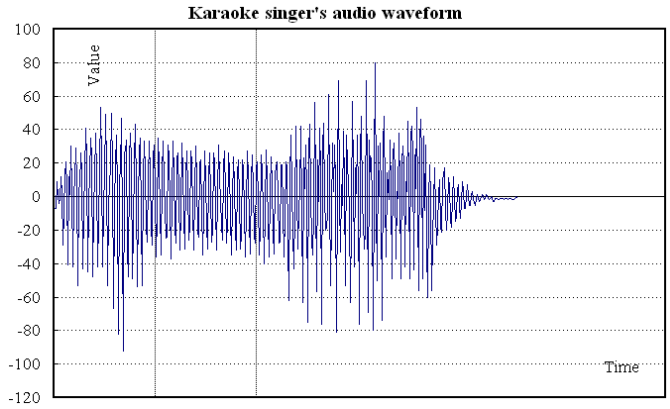
the results of audio analogies, we conducted a user study. We polled 11 subjects, with a mix of genders and expertise. The survey was administered by setting up an online site. The users had to listen to four karaoke signing renditions (performed by one child and three adults). The subjects were asked to listen to the original rendition as well as the corrected version using the proposed audio analogies technique. The subjects were asked to rate the quality of the corrected renditions using three numerical labels (corresponding to (1) no change, (2) sounds better & (3) sounds excellent). The mean opinion scores for all participants for the four audio clips were 1.63, 1.80, 1.55 and 1.55 respectively. This indicates that the subject perceived a moderate but definite improvement.
For pitch artifacts, our correction is based on the following analysis shown in Fig. 21. We can easily see that different people have a different pitch and the same person has less amount of variations in his or her pitch. After the pitch handling by audio analogies, the pitch is improved as shown in Fig. 22. The cepstrum of the corrected audio is between that of the original singer's audio and the user's audio.

6 Conclusions

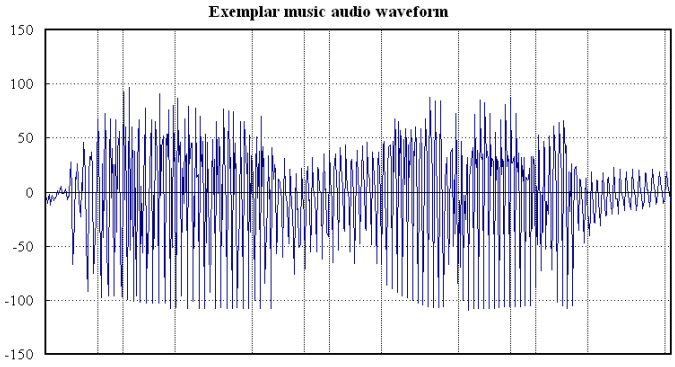
In this paper, we have presented a cross-modal approach to karaoke audio artifacts handling in temporal domain. Our approach uses adaptive sampling along with the video analogies approach for correcting the artifacts. The pitch, tempo and loudness of the user's singing are synchronized better with video by using audio cues (from original singer's rendition) as well as video cues (caption highlighting information is extracted to aid proper audio-video synchronization). We also perform the noise removal step prior to artifacts handling. In the future, we plan to extend this cross-modal approach for better video synthesis of karaoke video and for other multimedia applications such as on-line music tutoring with real-time feedback. There are also applications in active video editing area which can be considered [1].

References

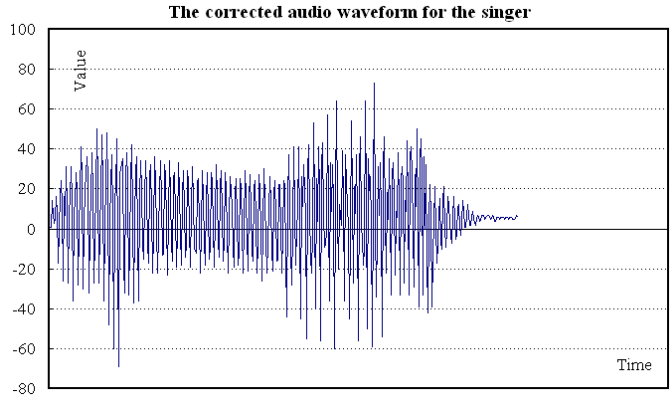
1. Marc Davis. Editing out video editing. *IEEE Multimedia*, pages 54–64, Apr.-Jun. 2003.
2. Randy Goldberg and Lance Riek. *A Practical Handbook of Speech Coders*. CRC Press, Floria U.S.A., 2000.
3. Jonathan Harrington and Steve Cassidy. *Techniques in Speech Acoustics*. Kluwer Academic Press, Dordrecht, The Netherlands, 1999.
4. Mohan S. Kankanhalli, Jun Wang, and Ramesh Jain. Experiential sampling in multimedia systems. *IEEE Transactions on Multimedia*, 8(5):937–946, Sep. 2006.
5. Hirokazu Kato. Karaoke apparatus selectively providing harmony voice to duet singing voices. U.S. Patent 6121531, Sep. 2000.
6. David Kumar and Subutai Ahmad. Method and apparatus for providing interactive karaoke enterainment. U.S. Patent 6692259, Dec. 2002.



(a)



(b)



(c)

Fig. 20. From up to down: karaoke singer's audio waveform, exemplar music audio waveform and the corrected audio waveform for the singer

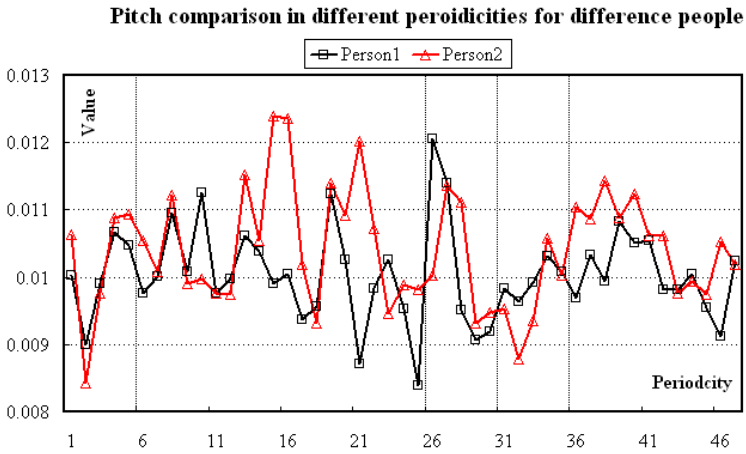


Fig. 21. Pitches for different people

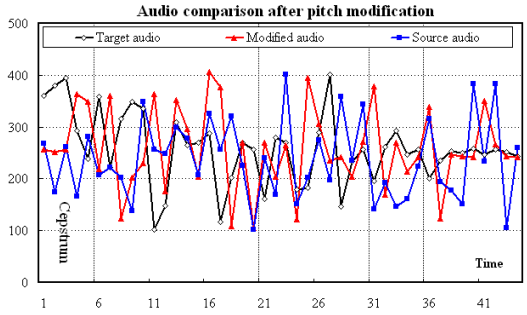


Fig. 22. Pitch comparison after audio analogies

7. Shuichi Matsumoto. Karaoke apparatus converting gender of singing voice to match octave of song. U.S. Patent 5889223, Mar. 1998.
8. Kenji Muraki and Katsuyoshi Fujii. Karaoke sound processor for automatically adjusting the pitch of the accompaniment signal. U.S. Patent 5477003, Dec. 1995.
9. Milan Sonka, Vaclav Hlavac, and Roger Boyle. *Image Processing, Analysis, and Machine Vision*. PWS Publishing, 1998.
10. Xiaou Tang, Xinbo Gao, Jianzhuang Liu, and Hongjiang Zhang. A spatial-temporal approach for video caption detection and recognition. *IEEE Transactions on Neural Netwroks*, 13(4):961–971, Jul. 2002.
11. Xiaou Tang, Bo Luo, Xinbo Gao, Edwige Pissaloux, Jianzhuang Liu, and Hongjiang Zhang. Video text extraction using temporal feature vectors. In *Proc. of IEEE ICME 2002*, pages 85–88, Lausanne, Switzerland, Aug. 2002.
12. Ye Wang, Min-Yen Kan, Tin-Lay Nwe, Arun Shenoy, and Jun Yin. Lyrically: Automatic synchronization of acoustic musical signals and textual lyrics. In *Proc. of ACM Multimedia 2004*, pages 212 – 219, New York, USA, Oct. 2004.
13. Wei-Qi Yan and Mohan S Kankanhalli. Detection and removal of lighting and shaking artifacts in home videos. In *Proc. of ACM Multimedia 2002*, pages 107–116, Juan Les Pins, France, Dec. 2002.
14. Wei-Qi Yan, Jun Wang, and Mohan S. Kankanhalli. Analogies based video editing. *ACM Multimedia Systems*, 11(1):3–18, 2005.
15. HongJiang Zhang, Atreyi Kankanhalli, and Stephen W. Smoliar. Automatic partitioning of full-motion video. *ACM/Springer Multimedia Systems*, 1(1):10–28, 1993.
16. Yi Zhang and Tat-Seng Chua. Detection of text captions in compressed domain video. In *Proc. of ACM Multimedia 2000*, pages 201–204, Marina Del Rey, CA USA, Aug. 2000.
17. Yong-Wei Zhu, Mohan S Kankanhalli, and Chang-Sheng Xu. Music scale modeling for melody matching. In *Proc. of ACM Multimedia 2003*, pages 359–362, Berkeley, U.S., Nov. 2003.