Chapter 17
Automated Music Video Generation Using Multi-level Feature-based Segmentation

Jong-Chul Yoon, In-Kwon Lee, and Siwoo Byun

Introduction

The expansion of the home video market has created a requirement for video editing tools to allow ordinary people to assemble videos from short clips. However, professional skills are still necessary to create a music video, which requires a stream to be synchronized with pre-composed music. Because the music and the video are pre-generated in separate environments, even a professional producer usually requires a number of trials to obtain a satisfactory synchronization, which is something that most amateurs are unable to achieve.

Our aim is automatically to extract a sequence of clips from a video and assemble them to match a piece of music. Previous authors [8, 9, 16] have approached this problem by trying to synchronize passages of music with arbitrary frames in each video clip using predefined feature rules. However, each shot in a video is an artistic statement by the video-maker, and we want to retain the coherence of the video-maker's intentions as far as possible.

We introduce a novel method of music video generation which is better able to preserve the flow of shots in the videos because it is based on the multi-level segmentation of the video and audio tracks. A shot boundary in a video clip can be recognized as an extreme discontinuity, especially a change in background or a discontinuity in time. However, even a single shot filmed continuously with the same camera, location and actors can have breaks in its flow; for example, actor might leave the set as another appears. We can use these changes of flow to break a video into segments which can be matched more naturally with the accompanying music.

Our system analyzes the video and music and then matches them. The first process is to segment the video using flow information. Velocity and brightness

J.-C. Yoon and I.-K. Lee (✉)
Department of Computer Science, Yonsei University, Seoul, Korea
e-mail: media19@cs.yonsei.ac.kr; iklee@yonsei.ac.kr

S. Byun
Department of Digital Media, Anyang University, Anyang, Korea
e-mail: swbyun@anyang.ac.kr

B. Furht (ed.), *Handbook of Multimedia for Digital Entertainment and Arts*,
DOI 10.1007/978-0-387-89024-1_17,

features are then determined for each segment. Based on these features, a video segment is then found to match each segment of the music. If a satisfactory match cannot be found, the level of segmentation is increased and the matching process is repeated.

Related Work

There has been a lot of work on synchronizing music (or sounds) with video. In essence, there are two ways to make a video match a soundtrack: assembling video segments or changing the video timing.

Foote et al. [3] automatically rated the novelty of segment of the metric and analyzed the movements of the camera in the video. Then they generated a music video by matching an appropriate video clip to each music segment. Another segment-based matching method for home videos was introduced by Hua et al. [8]. Amateur videos are usually of low quality and include unnecessary shots. Hua et al. calculated an attention score for each video segment which they used to extract the more important shots. They analyzed these clips, searching for a beat, and then they adjusted the tempo of the background music to make it suit the video. Mulhem et al. [16] modeled the aesthetic rules used by real video editors; and used them to assess music videos. Xian et al. [9] used the temporal structures of the video and music, as well as repetitive patterns in the music, to generate music videos.

All these studies treat video segments as primitives to be matched, but they do not consider the flow of the video. Because frames are chosen to obtain the best synchronization, significant information contained in complete shots can be missed. This is why we do not extract arbitrary frames from a video segment, but use whole segments as part of a multi-level resource for assembling a music video.

Taking a different approach researches, Jehan et al. [11] suggested a method to control the time domain of a video and to synchronize the feature points of both video and music. Using timing information supplied by the user, they adjusted the speed of a dance clip by time-warping, so as to synchronize the clip to the background music. Time-warping is also a necessary component in our approach. Even the best matches between music and video segments can leave some discrepancy in segment timing, and this can be eliminated by a local change to the speed of the video.

System Overview

The input to our system is an MPEG or AVI video and a .wav file, containing the music. As shown in Fig. 1, we start by segmenting both music and video, and then analyze the features of each segment. To segment the music, we use novelty scoring

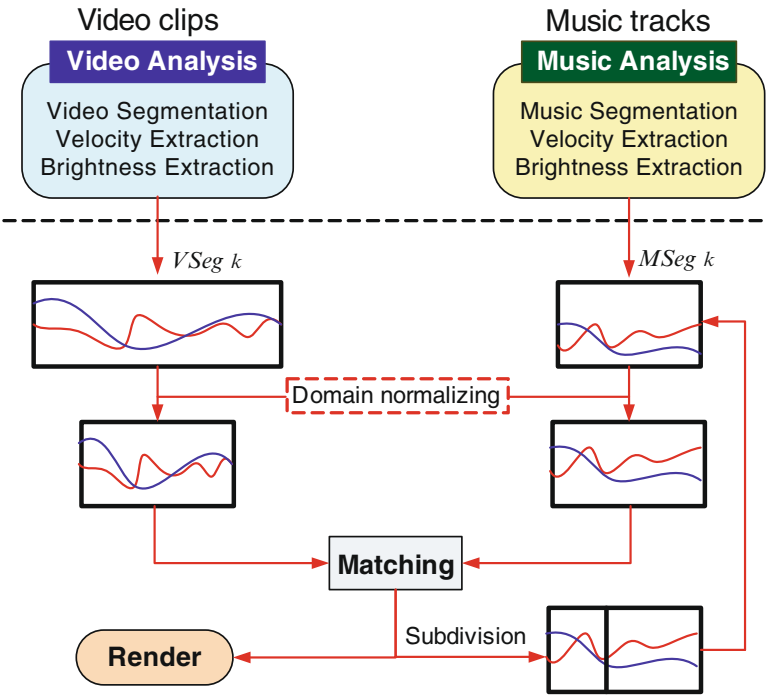

Fig. 1 Overview of our music video generation system

[3], which detects temporal variation in the wave signal in the frequency domain. To segment the video, we use contour shape matching [7], which finds extreme changes of shape features between frames. Then we analyze each segment based on velocity and brightness features.

Video Segmentation and Analysis

Synchronizing arbitrary lengths of video with the music is not a good way to preserve the video-maker's intent. Instead, we divide the video at discontinuities in the flow, so as to generate segments that contain coherent information. Then we extract features from each segment, which we use to match it with the music.

Segmentation by Contour Shape Matching

The similarity between two images can be simply measured as the difference between the colors at each pixel. But that is ineffective for a video only to detect short

boundaries because the video usually contains movement and noise due to compression. Instead, we use contour shape matching [7], which is a well-known technique for measuring the similarities between two shapes, on the assumption that one is a distorted version of the other. Seven Hu-moments can be extracted by contour analysis, and these constitute a measure of the similarity between video frames which is largely independent of camera and object movement.

Let $V_i (i = 1, \ldots, N)$ be a sequence of N video frames. We convert V_i to an edge map F_i using the Canny edge detector [2]. To avoid obtaining small contours because of noise, we stabilize each frame of V_i using Gaussian filtering [4] as a preprocessing step. Then, we calculate the Hu-moments h_g^i, $(g = 1, \ldots, 7)$ from the first three central moments [7]. Using these Hu-moments, we can measure the similarity of the shapes in two video frames, V_i and V_j, as follows:

$$I_{i,j} = \sum_{g=1}^{7} \left| 1/c_g^i - 1/c_g^j \right|$$

where

$$c_g^i = \text{sign}\left(h_g^i\right) \log_{10} \left| h_g^i \right|, \tag{1}$$

and h_g^i is invariant with translation, rotation and scaling [7]. $I_{i,j}$ is independent of the movement of an object, but it changes when a new object enters the scene. We therefore use large changes to $I_{i,j}$ to create the boundaries between segments. Figure 2a is a graphic representation of the similarity matrix $I_{i,j}$.

Foote et al. [3] introduced a segmentation method that applies the radial symmetric kernel (RSK) to the similarity matrix (see Fig. 3). We apply the RSK to the diagonal direction of our similarity matrix $I_{i,j}$, which allows us to express the flow discontinuity using the following equation:

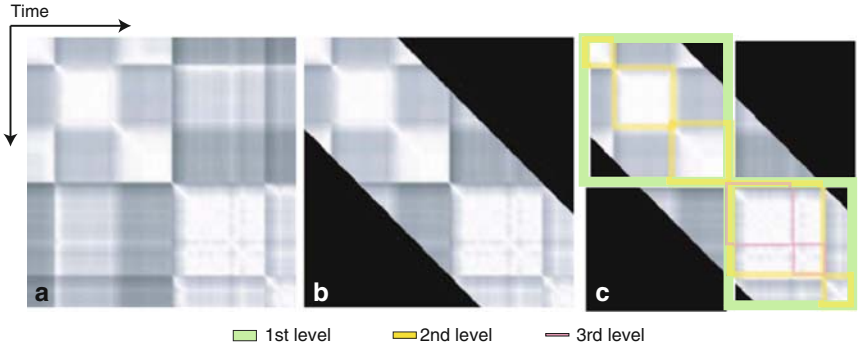

Fig. 2 Video segmentation using the similarity matrix: **a** is the full similarity matrix $I_{i,j}$, **b** is the reduced similarity matrix used to determine maximum kernel overlap region, and **c** is the result of segmentation using different sizes of radial symmetric kernel

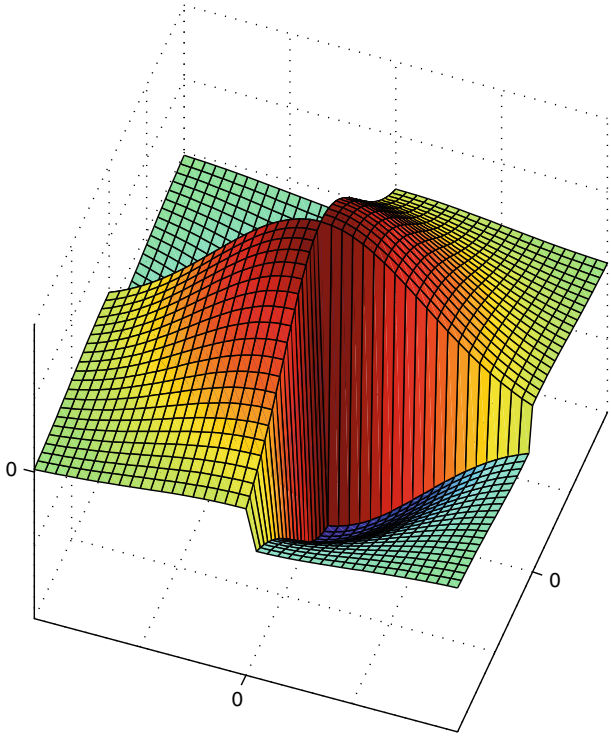

Fig. 3 The form of a radially symmetric Gaussian kernel

$$EV(i) = \sum_{u=-\delta}^{\delta} \sum_{v=-\delta}^{\delta} \mathrm{RSK}(u, v) \cdot I_{i+u,i+v}, \tag{2}$$

where δ is the size of the RSK. Local maxima of $EV(i)$ are taken to be boundaries of segments. We can control the segmentation level by changing the size of the kernel: a large δ produces a coarse segmentation that ignores short variations in flow, whereas a small δ produces a fine segmentation. Because the RSK is of size δ and only covers the diagonal direction, we only need to calculate the maximum kernel overlap region in the similarity matrix $I_{i,j}$, as shown in Fig. 2b. Figure 2c shows the result of for $\delta = 32$, 64 and 128, which are the values that we will use in multi-level matching.

Video Feature Analysis

From the many possible features of a video, we choose velocity and brightness as the basis for synchronization. We interpret velocity as a displacement over time derived

from the camera or object movement, and brightness is a measure of the visual impact of luminance in each frame. We will now show how we extract these features.

Because a video usually contains noise from the camera and the compression technique, there is little value in comparing pixel values between frames, which is what is done in the optical flow technique [17]. Instead, we use an edge map to track object movements robustly. The edge map F_i, described in the previous section can be expected to outline. And the complexity of edge map, which is determined by the number of edge points, can influence the velocity. Therefore, we can express the velocity between frames as the sum of the movements of each edge-pixel. We define a window $\phi_{x,y}(p,q)$ of size $w \times w$, on edge-pixel point (x, y) as its center, where p and q are coordinates within that window. Then, we can compute the color distance between windows in the i^{th} and $(i+1)^{\text{th}}$ frames as follows:

$$D^2 = \sum_{p,q \in \phi^i_{x,y}(p,q)} \left(\phi^i_{x,y}(p,q) - \phi^{i+1}_{(x,y)+vec^i_{x,y}}(p,q) \right)^2, \tag{3}$$

where x and y are image coordinates. By minimizing the squared color distance, we can determine the value of $vec^i_{x,y}$. We avoid considering pixels which are not on an edge, we assign a zero vector when $F_i(x, y) = 0$. After finding all the moving vectors in the edge map, we apply the local Lucas-Kanade optical flow technique [14] to track the moving objects more precisely.

By summing the values of $vec^i_{x,y}$, we can determine the velocity of the i^{th} of the video frames. However, this measure of velocity is not appropriate if a small area outside the region of visual interest makes a large movement. In the next section, we will introduce a method of video analysis based on the concept of significance.

Next, we determine the brightness of each frame of video using histogram analysis [4]. First, we convert each video frame V_i into a grayscale image. Then we construct a histogram that partitions the grayscale values into ten levels. Using this histogram, we can determine the brightness of the i^{th} frame as follows:

$$V^i_{bri} = \sum_{e=1}^{10} B(e)^2 B_{mean_e}, \tag{4}$$

where $B(e)$ is the number of pixels in the e^{th} bucket and B_{mean_e} is the representative value of the e^{th} bucket. Squaring $B(e)$ means that a contrasty image, such as a black-and-white check pattern, will be classified as brighter than a uniform tone, even if the mean brightness of all the pixels in each image is the same.

Detecting Significant Regions

The tracking technique, introduced in the previous section, is not much affected by noise. However, an edge may be located outside the region of visual interest.

This is likely to make the computed velocity deviate from a viewer's perception of the liveliness of the videos. An analysis of visual significance can extract the region of interest more accurately. We therefore construct a significance map that represents both spatial significance, which is the difference between neighboring pixels in image space, and temporal significance, which measures of differences over time.

We use the Gaussian distance introduced by Itti [10] as a measure of spatial significance. Because this metric correlates with luminance [15], we must first convert each video frame to the YUV color space. We can then calculate the Gaussian distance for each pixel, as follows:.

$$G^i_{l,\eta}(x, y) = G^i_l(x, y) - G^i_{l+\eta}(x, y), \tag{5}$$

where G_l is the l^{th} level in the Gaussian pyramid, and x and y are image coordinates. A significant point is one that has a large distance between its low-frequency and high-frequency levels. In our experiment, we used the $l = 2$ and $\eta = 5$.

The temporal significance of a pixel (x, y) can be expressed as the difference in its velocity between the i^{th} and the $(i + 1)^{\text{th}}$ frames, which we call its acceleration. We can calculate the acceleration of a pixel from $vec^i_{x,y}$, which is already required for edge-map, as follows:

$$T^i(x, y) = N\left(\left\| vec^i_{x,y} - vec^{i+1}_{x,y} \right\|\right), \tag{6}$$

where N is a normalizing function which normalizes the acceleration so that it never exceeds 1. We assume that a large acceleration brings a pixel to the attention of the viewer. However, we have to consider the camera motion: if the camera is static, the most important object in the scene is likely to be the one making the largest movement; but if the camera is moving, it is likely to be chasing the most important object, and then a static region is significant. We use the ITM method introduced by Lan et al. [12] to extract the camera movement, with a 4-pixel threshold to estimate camera shake. This threshold should relate to the size of the frame, which is 640×480 in this case. If the camera moves beyond that threshold, we use $1 - T^i(x, y)$ rather than $T^i(x, y)$ as the measure of temporal significance.

Inspired by the focusing method introduced by Ma et al. [15], we then combine the spatial and temporal significance maps to determine a center of attention that should be in the center of the region of interest, as follows:

$$\begin{aligned} x^i_f &= \frac{1}{CM} \sum_{x=1}^{n} \sum_{y=1}^{m} G^i(x, y) T^i(x, y) x. \\ y^i_f &= \frac{1}{CM} \sum_{x=1}^{n} \sum_{y=1}^{m} G^i(x, y) T^i(x, y) y, \end{aligned} \tag{7}$$

where

$$CM = \sum_{x=1}^{n} \sum_{y=1}^{m} G^i(x, y) T^i(x, y) \tag{8}$$

and where x_f^i and y_f^i are the coordinates of the center of attention in the i^{th} frame.

The true size of the significant region will be affected by motion and color distribution in each video segment. But the noise in a home video prevents the calculation of an accurate region boundary. So we fix the size of the region of interest at 1/4 of the total image size. We denote the velocity vectors in the region of interest by $\overline{vec}_{x,y}^i$ (see Fig. 4d), which those outside the region of interest are set to 0. We can then calculate a representative velocity V_{vel}^i, for the region of interest by summing the pixel velocities as follows:

$$V_{vel}^i = \sum_{x=1}^{n} \sum_{y=1}^{m} \left\| \overline{vec}_{x,y}^i \right\|, \tag{9}$$

where $n \times m$ is the resolution of the video.

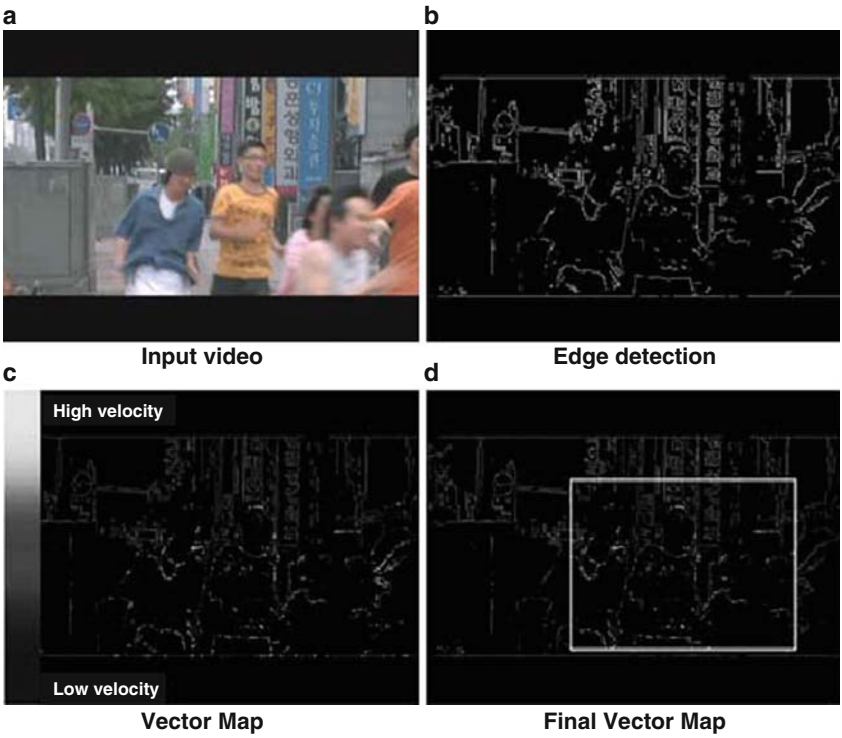

Fig. 4 Velocity analysis based on edge: **a** is a video segment; **b** is the result of edge detection; **c** shows the magnitude of tracked vectors; and **d** shows the elimination of vectors located outside the region of visual interest

Home video usually contains some low-quality shots of static scenes or discontinuous movements. We could filter out these passages automatically before starting the segmentation process [8], but we actually use the whole video, because the discontinuous nature of these low-quality passages means that they are likely to be ignored during the matching step.

Music Segmentation and Analysis

To match the segmented video, the music must also be divided into segments. We can use conventional signal analysis method to analyze and segment the music track.

Novelty Scoring

We use a similarity matrix to segment the music, which is analogous to our method of video segmentation combined with novelty scoring, which is introduced by Foote et al. [3] to detect temporal changes in the frequency domain of a signal. First, we divide the music signal into windows of 1/30 second duration, which matches that a video frame. Then we apply a fast Fourier transform to convert the signal in each window into the frequency domain.

Let i index the windows in sequential order and let A_i be a one-dimensional vector that contains the amplitude of the signal in the i^{th} window in the frequency domain. Then the similarity of the i^{th} and j^{th} windows can be expressed as follows:

$$SM_{i,j} = \frac{A_i \cdot A_j}{\|A_i\|\|A_j\|}. \tag{10}$$

The similarity matrix $SM_{i,j}$ can be used for novelty scoring by applying the same radial symmetric kernel that we used for video segmentation as follows:

$$EA(i) = \sum_{u=-\delta}^{\delta} \sum_{v=-\delta}^{\delta} \text{RSK}(u, v) \cdot SM_{i+u,j+v}, \tag{11}$$

where $\delta = 128$. The extreme values of the novelty scoring $EA(i)$ form the boundaries of the segmentation [3]. Figure 5 shows the similarity matrix and the corresponding novelty score. As in the video segmentation, the size of the RSK kernel determines the level of segmentation (see Fig. 5b). We will use this feature in the multi-level matching that will follow in Section on “Matching Music and Video”.

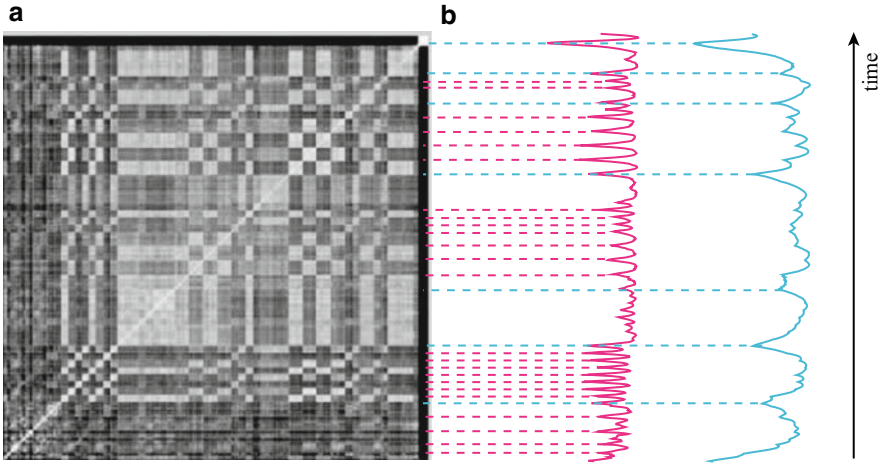

Fig. 5 Novelty scoring using the similarity matrix in the frequency domain: **a** is the similarity matrix in the frequency domain; **b** the novelty scores obtained with different size of RSK

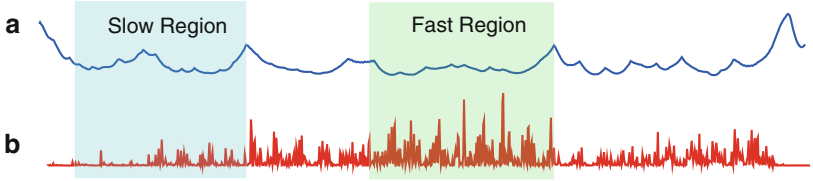

Fig. 6 **a** novelty scoring and **b** variability of RMS amplitude

Music Feature Analysis

The idea of novelty represents the variability of music (see Fig. 6a). We can also introduce a concept of velocity for music, which is related to its beat. Many previous authors have tried to extract the beat from a wave signal [5, 18], but we avoid confronting this problem. Instead we determine the velocity of each music segment from the amplitude of the signal in the time domain.

We can sample the amplitude $S_i(u)$ of a window i in the time domain, where u is a sampling index. Then we can calculate a root mean square amplitude for that window:

$$RMS_i = \frac{1}{U}\sum_{u=1}^{U}(S_i(u))^2, \tag{12}$$

where U is the total number of samples in the window. Because the beat is usually set by the percussion instruments, which dominate the amplitude of the signal, we can estimate the velocity of the music from the RMS of the amplitude. If a music segment has a slow beat, then the variability of the amplitude is likely to be relatively

low; but if it has a fast beat then the amplitude is likely to be more variable. Using this assumption, we extract the velocity as follows:

$$M^i_{vel} = |RMS_i - RMS_{i-1}|. \tag{13}$$

Figure 6a shows the result of novelty scoring and Fig. 6b shows the variability of the RMS amplitude. We see that variability of the amplitude changes as the music speeds up, but the novelty scoring remains roughly constant.

Popular music is often structured into a pattern, which might typically consist of an intro, verse, and chorus, with distinct variations in amplitude and velocity. This characteristic favors our approach.

Next, we extract the brightness feature using the well-known spectral centroid [6]. The brightness of music is related to its timbre. A violin has a high spectral centroid, but a tuba has a low spectral centroid. If $A_i(p)$ is the amplitude of the signal in the i^{th} window in the frequency domain, and p is the frequency index, then the spectral centroid can be calculated as follows:

$$M^i_{bri} = \frac{\Sigma p\,(A_i(p))^2}{\Sigma\,(A_i(p))^2}. \tag{14}$$

Matching Music and Video

In previous sections, we explained how to segment video and music and to extract features. We can now assemble a synchronized music video by matching segments based on three terms derived from the video and music features, and two terms obtained from velocity histograms and segment lengths.

Because each segment of music and video has a different length, we need to normalize the time domain. We first interpolate the features of each segment, especially velocity, brightness, and flow discontinuity, using a Hermite curve and then normalize the magnitude of the video and music feature curves separately. The flow discontinuity was calculated for segmentation and velocity and brightness features were extracted both videos and music in previous sections. Using Hermite interpolation, we can represent the k^{th} video segment as a curve in a three-dimensional feature space, $V^k(t) = (cv^k_{ext}(t),\ cv^k_{vel}(t),\ cv^k_{bri}(t))$, ever the time interval [0, 1]. The features of a music segment can be represented by a similar multidimensional curve, $M^k(t) = (cm^k_{ext}(t),\ cm^k_{vel}(t),\ cm^k_{bri}(t))$. We then compare the curves by sampling them at the same parameters, using these matching terms:

– **Extreme boundary matching** $Fc_1(V^y(t), M^z(t))$.
The changes in Hu-moments $EV(i)$ in Eq. 2 determine discontinuities in the video, which can then be matched with the discontinuities in the music found by novelty scoring $EA(i)$ in Eq. 11. We interpolate these two features to create the continuous functions $cv^k_{ext}(t)$ and $cm^k_{ext}(t)$, and then calculate the difference by sampling them at the same value of the parameter t.

– **Velocity matching** $Fc_2(V^y(t), M^z(t))$.
  The velocity feature curves for video, $cv^k_{vel}(t)$, and the music, $cm^k_{vel}(t)$ can be interpolated by V^i_{vel} and M^i_{vel}. These two curves can be matched to synchronize the motion in the video with the beat of the music.
– **Brightness matching** $Fc_3(V^y(t), M^z(t))$.
  The brightness feature curves for the video, $cv^k_{bri}(t)$, and the music, $cm^k_{bri}(t)$ can be interpolated by V^i_{bri} and M^i_{bri}. These two curves can be matched to synchronize the timbre of the music to the visual impact of the video.

Additionally, we used match the distribution of the velocity vector. We can generate a histogram $VH^k(b)$ with K bins for the k^{th} video segment using the video velocity vector $\overline{vec}_{x,y}$. We also construct a histogram $MH^k(b)$ of the amplitude of the music in each segment, in the frequency domain A_k. This expresses the timbre of the music, which determines its mood. We define the cost of matching each pair of histogram as follows:

$$Hc(y,z) = \sum_{b=1}^{K} \left(\frac{VH^y(b)}{N_y} - \frac{MH^z(b)}{N_z} \right)^2, \tag{15}$$

where y and z are the indexes of a segment, and N_y and N_z are the sum of the cardinality of the video and music histograms. This associates low-timbre music with near-static video, and high-timbre music with video that contains bold movements. Finally, the durations of video and music should be compared to avoid the need for excessive time-warping. We therefore use the difference of duration between the music and video segments as the final matching term, $Dc(y,z)$. Because the range of $Fc_i(V^y(t), M^z(t))$ and $Hc(y,z)$ are [0,1], we normalize the $Dc(y,z)$ by using the maximum difference of duration.

We can now combine the five matching terms into the following cost function:

$$Cost_{y,z} = \sum_{i=1}^{3} w_i \; Fc_i(V^y(t), M^z(t)) + w_4 \; Hc(y,z) + w_5 \; Dc(y,z), \tag{16}$$

where y and z are the indexes of a segment, and w_i is the weight applied to each matching terms. The weights control the importance given to each matching term. In particular, w_5, which is the weight applied to segment length matching, can be used to control the dynamics of the music video. A low value of w_5 allows more time-warping.

We are now able to generate a music video by calculating $Cost_{y,z}$ for all pairs of video and music segments, and then selecting the video segment which matches each music segment at minimum cost. We then apply time-warping to each video segment so that its length is exactly the same as that of the corresponding music segment. A degree of interactivity is provided by allowing the user to remove any displeasing pair of music and video segments, and then regenerate the video. This facility can be used to eliminate repeated video segments. It can also be extended, so that the user is presented with a list of acceptable matches form which to choose a pair.

We also set a cost threshold to avoid low-quality matches. If a music segment cannot be matched with a cost lower than the threshold, then we subdivide that segment by reducing the value of δ in the RSK. Then we look for a new match to each of the subdivided music segments. Matching and subdivision can be recursively applied to increase the synchronization of the music video; but we limit this process to three levels to avoid the possibility of runaway subdivision.

Experimental Results

By trial and error, we selected (1, 1, 0.5, 0.5, 0.7) for the weight vector in Eq. 16, set $K = 32$ in Eq. 15, and set the subdivision threshold to $0.3 \times mean(Cost_{y,z})$.

For an initial test, we made a 39-min video (Video 1) containing sequences with different amount of movement and levels of luminance (see Fig. 7). We also composed 1 min and 40 s of music (Music 1), with varying timbre and beat. In the initial segmentation step, the music was divided into 11 segments. In the subsequent matching step, the music was subdivided into 19 segments to improve synchronization.

We then went on to perform more realistic experiments with three short films and one home video (Videos 2, 3, 4 and 5: see Fig. 8), and three more pieces of music which we composed. From this material we created three sets of five music videos. The first set was made using Pinnacle Studio 11 [1]; the second set was made using

Fig. 7 Video filmed by the authors

Fig. 8 Example videos: **a** Short film "Someday", directed by Dae-Hyun Kim, 2006; **b** Short film "Cloud", directed by Dong-Chan Kim; **c** Short film "Father", directed by Hyun-Wook Moon; **d** Amateurs home video "Wedding"

Foote's method [3]; and the third was produced by our system. The resulting videos can all be downloaded from URL.[1]

We showed the original videos to 21 adults who had no prior knowledge of this research. Then we showed the three sets of music videos to the same audience, and asked them to score each video, giving marks for synchronization (velocity, brightness, boundary and mood), dynamics (dynamics), and the similarity to the original video (similarity), meaning the extent to which the original story-line is presented. The 'mood' term is related to the distribution of the velocity vector and 'dynamics' term is related to the extent to which the lengths of video segments are changed by time-warping. Each of the six terms was given a score out of ten. Figure 9 shows that our system obtained better than Pinnacle Studio as Foote's methods on five out of the six terms. Since our method currently makes no attempt to preserve the original orders of the video segments, it is not surprising that the result for 'similarity' were more ambiguous.

Table 1 shows the computation time required to analyze the video and music. We naturally expect the video to take much longer to process than the music, because of its higher dimensionality.

[1] http://visualcomputing.yonsei.ac.kr/personal/yoon/music.htm

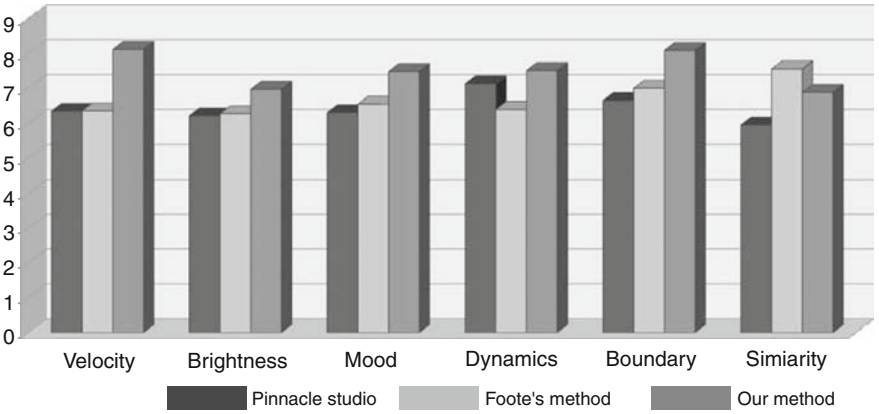

Fig. 9 User evaluation results

Table 1 Computation times for segmentation and analysis of music and video

Media	Length	Segmentation	Velocity	Brightness
Video 1	30 min	50 min	84 min	3 min
Video 2	23 min	44 min	75 min	2.5 min
Video 3	17 min	39 min	69 min	2.1 min
Video 4	25 min	46 min	79 min	2.6 min
Video 5	45 min	77 min	109 min	4.1 min
Music 1	100 s	7.2 s	4.2 s	2.1 s

Table 2 A visual count of the numbers of shots in each videos, and the number of segments generated using different values of δ in the RSK

Media	Visually counted num. of shots	$\delta = 128$	$\delta = 64$	$\delta = 32$
Video 1	44	49	63	98
Video 2	34	41	67	142
Video 3	36	50	57	92
Video 4	43	59	71	102
Video 5	79	92	132	179

Table 2 shows how the number of video segments was affected by the value of δ in the RSK. We also assessed the number of distinct shots in each video by inspection. This visual count tallies quite well with the results when $\delta = 128$. By reducing δ to 32, the number of distinct shots are approximately doubled.

Table 3 shows computation times for matching the music and video. Because the matching is takes much less time than analysis, we stored the all the feature curves for both video and music to speed up the calculations.

Table 3 Computation times for matching Music 1 to the five videos

Media	Time (s)
Video 1	6.41
Video 2	8.72
Video 3	6.22
Video 4	7.48
Video 5	10.01

Conclusion

We have produced an automatic method for generating music videos that preserves the flow of video segments more effectively than previous approaches. Instead of trying to synchronize arbitrary regions of video with the music, we use multi-level segment matching to preserve more of the coherence of the original videos. Provided with an appropriate interface to simplify selection of the weight terms, this system would allow music video to be made with very little skill.

The synchronization of segments by our system could be improved. Although each segment is matched in terms of features, it is possible for points of discordancy between music and video to remain within a segment. We could increase the level of synchronization by applying time-warping within each segment using features [13]. Another area of concern is the time required to analyze the video, and we are working on a more efficient algorithm.

Many home videos have a weak story-line, so that the original sequence of the video may not be important. But this will not be true for all videos, and so we need to look for ways of preserving the story-line, which might involve a degree of user annotation.

Acknowledgements This research is accomplished as the result of the promotion project for culture contents technology research center supported by Korea Culture & Content Agency (KOCCA).

References

1. Avid Technology Inc (2007) User guide for pinnacle studio 11. Avid Technology Inc, Tewksbury
2. Canny J (1986) A computational approach to edge detection. IEEE Trans Pattern Anal Mach Intell 8(6):679–698
3. Foote J, Cooper M, Girgensohn A (2002) Creating music videos using automatic media analysis. In: Proceedings of ACM multimedia. ACM, New York, pp 553–560
4. Gose E, Johnsonbaugh R, Jost S (1996) Pattern recognition and image analysis. Prentice Hall, Englewood Cliffs
5. Goto M (2001) An audio-based real-time beat tracking system for music with or without drum-sounds. J New Music Res 30(2):159–171
6. Helmholtz HL (1954) On the sensation of tone as a physiological basis for the theory of music. Dover (translation of original text 1877)

7. Hu M (1963) Visual pattern recognition by moment invariants. IRE Trans Inf Theo 8(2): 179–187
8. Hua XS, Lu L, Zhang HJ (2003) Ave—automated home video editing. In: Proceedings of ACM multimedia. ACM, New York, pp 490–497
9. Hua XS, Lu L, Zhang HJ (2004) Automatic music video generation based on temporal pattern analysis. In: 12th ACM international conference on multimedia. ACM, New York, pp 472–475
10. Itti L, Koch C, Niebur E (1998) A model of saliency-based visual attention for rapid scene analysis. IEEE Trans Pattern Anal Mach Intell 20(11):1254–1259
11. Jehan T, Lew M, Vaucelle C (2003) Cati dance: self-edited, self-synchronized music video. In: SIGGRAPH conference abstracts and applications. SIGGRAPH, Sydney, pp 27–31
12. Lan DJ, Ma YF, Zhang HJ (2003) A novel motion-based representation for video mining. In: Proceedings of the IEEE international conference on multimedia and expo. IEEE, Piscataway, pp 6–9
13. Lee HC, Lee IK (2005) Automatic synchronization of background music and motion in computer animation. In: Proceedings of eurographics 2005, Dublin, 29 August–2 September 2005, pp 353–362
14. Lucas B, Kanade T (1981) An iterative image registration technique with an application to stereo vision. In: Proceedings of 7th international joint conference on artificial intelligence (IJCAI), Vancouver, August 1981, pp 674–679
15. Ma YF, Zhang HJ (2003) Contrast-based image attention analysis by using fuzzy growing. In: Proceedings of the 11th ACM international conference on multimedia. ACM, New York, pp 374–381
16. Mulhem P, Kankanhalli M, Hasan H, Ji Y (2003) Pivot vector space approach for audio-video mixing. IEEE Multimed 10:28–40
17. Murat Tekalp A (1995) Digital video processing. Prentice Hall, Englewood Cliffs
18. Scheirer ED (1998) Tempo and beat analysis of acoustic musical signals. J Acoust Soc Am 103(1):588–601