ACM Transactions on Graphics (special issue of SIGGRAPH 2015), 34(4).

audeosynth: Music-Driven Video Montage

Zicheng Liao
Zhejiang University

Yizhou Yu
The University of Hong Kong

Bingchen Gong
Zhejiang University

Lechao Cheng
Zhejiang University

Abstract

We introduce music-driven video montage, a media format that offers a pleasant way to browse or summarize video clips collected from various occasions, including gatherings and adventures. In music-driven video montage, the music drives the composition of the video content. According to musical movement and beats, video clips are organized to form a montage that visually reflects the experiential properties of the music. Nonetheless, it takes enormous manual work and artistic expertise to create it. In this paper, we develop a framework for automatically generating music-driven video montages. The input is a set of video clips and a piece of background music. By analyzing the music and video content, our system extracts carefully designed temporal features from the input, and casts the synthesis problem as an optimization and solves the parameters through Markov Chain Monte Carlo sampling. The output is a video montage whose visual activities are cut and synchronized with the rhythm of the music, rendering a symphony of audio-visual resonance.

CR Categories: I.3.0 [Computer Graphics]: General.

Keywords: audio-visual synchresis

The word and its sound, form and its color
are vessels of a transcendental essence that we dimly surmise
As sound lends sparkling color to the spoken word
so color lends psychically resolved tone to form

JOHANNES ITTEN, 1888 - 1967

1 Introduction

We have seen a thrilling series of work on visual media synthesis in the past decades, including Video Textures [Schödl et al. 2000], stochastic motion textures [Chuang et al. 2005], visual mosaic [Agarwala et al. 2004; Irani et al. 1996; Kopf et al. 2010], video synopsis [Pritch et al. 2008], cinemagraphs [Beck and Burg 2012] and its extensions [Bai et al. 2012; Joshi et al. 2012; Liao et al. 2013]. Perhaps at a higher level is the concept of moment images envisioned by Cohen and Szeliski [2006]. By their definition, a moment image captures a universal yet subjective state by combining visual snapshots from multiple shutter instants.

While such work successfully creates lively visual representations of the world, other dimensions of human sensation are absent, e.g.

Figure 1: *Given a set of video clips and a piece of music, our system analyzes the video and music content, and produces a video montage that "dances" to the beat of the music.*

hearing, touch, smell or taste. To break the status-quo, we advocate the concept of *audio-visual synchresis* by French theorist Michel Chion [1994], and introduce Music-Driven Video Montage: video montages that "dance" to a piece of background music. Music-driven video montage belongs to a type of audio-visual media called *music-driven imagery* [Goodwin 1992; Wesseling 2004], where the music drives the composition of the video content. According to musical movements and beats, video clips are reorganized to form a montage that "visually reflects the experiential properties of the music" [Wesseling 2004], e.g., body parts moving back and forth to the rhythm of music beats, footsteps on music notes, portamento when a bird glides over lake Tahoe, and on a music transition the scene cuts to the deep valley of Yosemite.

Music-driven video montage suggests a pleasant way to browse or summarize video clips collected from gatherings, adventures, sport events, or time-lapse photography. A niche of application is on the widespread mobile devices. With these devices, people nowadays shoot moments of daily life and share them online very easily. Music-driven video montage would be a nice way to organize such video clips and play them to the beat of a selected background music. The result is of higher aesthetics value, and provides a new experience for personal viewing and online sharing. However, it takes enormous manual work and artistic expertise to create such result. Our goal is to develop a computer-aided solution to creating such audio-visual composition (Figure 1).

Creating a high-quality music-driven video montage with a machine algorithm is a challenging new task. First of all, given a piece of background music and a set of video clips, it is not obvious how should the problem be formulated, because producing an audio-visual composition has a large degree of freedom. The algorithm does not have to use all the video clips or all the frames in a chosen video. It also needs to determine the order of the chosen videos on the music timeline, the cut between the videos, as well as the play speed of every chosen video. To make the problem tractable, we adopt two important thumb-of-rules when artists create music-driven imagery: *cut-to-the-beat* and *synchronization* [Wesseling 2004; Dmytryk 2010]. Cut-to-the-beat means video sequences should be cut at significant note onsets in the mu-

sic, and synchronization means visual motion in the video should match the rhythm and pace of the music. The second challenge is, music and videos are two different types of media. The waveform of a music piece is a one-dimensional time series while videos have two spatial dimensions in addition to one temporal dimension. It is unclear what kind of features and properties should be used to define synchronization between them, let alone an automatic method for achieving such synchronization. Third, choosing a subset of video clips and deciding their order is an expensive combinatorial problem that has a very large search space. How to obtain a reasonable solution in an efficient manner is another challenge.

In this paper, we develop a framework for automatically generating music-driven video montage. The input is a set of video clips and a selected piece of background music with fit melody or theme. By analyzing the music and video content, our system first performs music segmentation and extracts carefully designed temporal features from both. Next, it casts the synchronization between the music segments and video clips as an optimization problem and solves the synthesis parameters in the optimization through Markov Chain Monte Carlo sampling. The output is a video montage whose visual activities are cut and synchronized with the rhythm of the background music, rendering a symphony of audio-visual resonance. Such effect is highly desirable in music-driven video editing. With our algorithms, artists perform the high-level work of shooting video clips and selecting the music, leaving the labor-intensive and precision-demanding editing work (e.g. cut and synchronization) to the machine.

In summary, we make the following contributions in this paper:

- We propose a new research problem on computer-aided generation of music-driven video montage. We further quantitatively formulate this problem as an optimization. Two of the energy terms in our formulation address two important composition principles.

- To achieve perceptual synchronization between music and video, we propose carefully designed and experimented temporal features for both music and video. To make the features of the two different media comparable, temporal video features are aggregated features derived from pixelwise optical flows. Temporal music features are interpolated from saliency scores at note onsets using the Gaussian kernel.

- We apply Markov Chain Monte-Carlo sampling to solve the above optimization problem. To improve efficiency, we factor the optimization into two stages, a precomputation stage and an MCMC based stochastic sampling stage.

2 Background

Sound as a primitive medium form has been intervened with the visual medium for a long time in human history. Ancient Greek were experts on music and dance in stage performance back to the 6th century BC. The first collection of Chinese poetry, *Shijing*, was performed with bronze bells and wooden drums during the same period of time in the east. In the history of filmmaking, after a very short period of silent film (1890s to 1920s), images and sound were inevitably combined together in the cinema and opened up an era of a flourishing industry. Later in the 1980s, the storyline-based audio-visual synthesis approach in filmmaking was challenged by the music video industry (MTV), where fragments of videos were sequenced to coordinate with and emphasize the music instead of the other way around. Our work takes the music-driven imagery approach. Its distinction from music video is that the two elements, music and video, are of equivalent importance in the composition to mutually intensify one another.

The practice of music-video composition, although being very subjective and following no strict rules, has a few guidelines. The first one is called *cut-to-the-beat*. It is related to video cuts, best known as the montage of scenes from different time and space to create a visual impression. A critical question of video montage is where to cut. According to American film editor and sound designer Walter Murch, when we watch a film, the scenes on the screen invoke a flow of ideas (emotions and thoughts) in the head. A cut is to "separate and punctuate an idea from what follows, like a blink of eye to the thought process" [Murch 2001]. In music-video composition, the bars in a music piece provide natural transition points where cuts should be exactly placed – not a frame sooner or later. This is the origin of the cut-to-the-beat principle.

The second guideline is related to audio-visual synchresis: the linkage between the visual and the auditory. In music videos, the pace and timing of visual activities typically flow with the pace and energy of the song. For example, footsteps are synchronized with the beat, popping with the drum. This association between action and sound is critical for narration and has a deep root in mankind's hereditary perception – that every sound rises from a source, a pure instinct developed from the ancient to learn from the association of sound and visual about threats for survival in the wild (explained in depth in Chion's book on synchresis [Chion 1994]). So the brain is more comfortable to accept visual and auditory signals that *co-occur* in time. This is the origin of the synchronization principle.

In addition, there are other dimensions where music and video can match in quality, for example, pace and velocity, loudness and dynamism, timbre and color, melody and mood. Our work primarily implements the cut-to-the-beat and synchronization principles. We also define a set of corresponding audio and visual features that can accommodate the matching between audio and visual in these other dimensions.

MIDI Format We rely on the Musical Instrument Digital Interface (MIDI) [Huber 1991; Swift 2014] format to parse the music input. A MIDI file is a semantic encoding of musical signals (in contrast with waveforms or the MP3 format) by specifying note onset parameters (time, pitch, volume and duration), instrument type, sound effect controller (e.g. vibrato and percussion), track information, as well as music meta data (clef, meter and tempo). In other words, it is a digital music sheet. Since 1983, MIDI has become an industrial protocol standard used by a wide variety of electronic musical instruments, computers and other sound devices to connect and communicate with one another. MIDI is now a mainstream language for digital music. All music pieces performed on electronic musical instruments are sequenced into the MIDI format. A large body of music pieces with public awareness has its MIDI format available in online databases such as 8notes (`http://8notes.com`) or free-midi (`http://free-midi.org`).

The MIDI data format lists a sequence of musical note events. For example, on track 0 second 2.5 a piano key of pitch 75 is pressed with volume 80; on track 1 second 2.75 a flute note of pitch 50 is on with volume 95; on track 0 second 3.0 the piano key of pitch 75 is released. The MIDI music meta data also defines the start and end times of bars using meter and tempo. Thus musical notes are first separated into tracks, and then grouped into bars. Notes from different tracks belong to the same bar if their onsets fall into a bar's time interval. Bar is the most basic unit of a music piece, like cells to a body. On top of bars, rhythm is developed, repeated and progressed to form the whole musical movement.

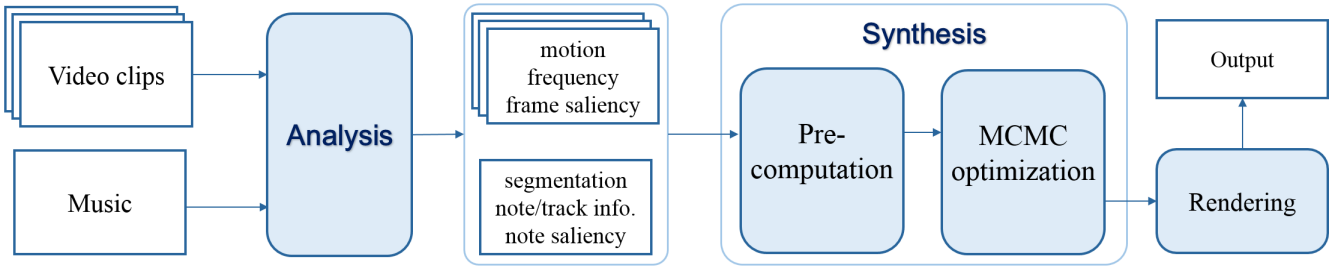


Figure 2: *System overview. Given a set of video clips and a piece of music, our system first performs feature analysis from both inputs, and then cast the synthesis problem as an optimization problem, solved by a two-stage algorithm. Lastly, the rendering component renders the final video montage that are cut and synchronized to the beat of the music.*

3 System Overview

Given a set of video clips and a music sequence, how do we fuse them into one piece? According to the music-driven imagery approach, the music provides the structural layout and remains unchanged. It is the video clips that are rearranged and edited on the music timeline. We first extract a set of carefully designed features from both the musical piece and the video clips, and then cast the problem of synthesizing a music-driven video montage as an energy minimization problem. Figure 2 shows an overview of the framework.

Problem Formulation Let $\mathbf{V} = \{v_1, v_2, ..., v_p\}$ be the set of input video clips and $\mathbf{M} = \{m_1, m_2, ..., m_q\}$ be the set of sequential segments of an input music. A music segment consists of a group of consecutive music bars. Music segments are obtained by analyzing the hierarchical musical structure. The task is to choose a video subsequence to match each music segment (Fig. 3), such that the chosen video subsequences minimize the following energy function:

$$E(\boldsymbol{\theta}, \mathbf{M}) = E_{match}(\boldsymbol{\theta}, \mathbf{M}) + E_{transit}(\boldsymbol{\theta}, \mathbf{M}) + E_{global}(\boldsymbol{\theta}, \mathbf{M}), \quad (1)$$

where the first term is the matching cost between corresponding music and video segments; the second term measures transition compatibility between consecutive segment pairs; and the third term accommodates higher-order constraints. Solution to the energy minimization is a mapping function, $a : i \rightarrow j$, that maps each music segment i $(= 1, \ldots, q)$ to a subsequence of a video clip j $(\in \{1, \ldots, p\})$, and parameters of the video subsequences. A video subsequence is defined by three parameters: start frame sf, end frame ef, and a temporal scaling factor $scale$ that resamples the video frames in the temporal domain, thus controls the video playback speed. We denote all the unknowns by $\boldsymbol{\theta} = \{\theta_1, \ldots, \theta_q\}$, where $\theta_i = (v_{a_i}; sf_i, ef_i, scale_i)$. θ_i defines the video subsequence matching music segment i. $\boldsymbol{\theta}$ and M together define all the components needed to synthesize a music-driven video montage.

Analysis Stage The analysis stage of our framework performs signal analysis on the video clips and the music piece in preparation for synthesis. Video analysis extracts features from video frames by performing such tasks as motion analysis, saliency estimation, and frequency detection. Music analysis heavily relies on the semantic information collected from MIDI data; the associated waveform data is also used. From the MIDI data, we extract note onsets, pitch, duration. We further group notes into bars according to tempo and meter. Then we cluster the bars into music segments, on top of which segment-level features, such as pace, volume, pitch variation, number of tracks, are computed.

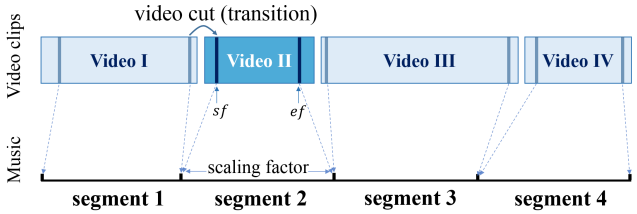


Figure 3: *Illustration of problem formulation. Each music segment is mapped to a video subsequence defined by a start frame sf, an end frame ef, and a scaling factor such that the chosen video subsequence can be rescaled to cover the entire music segment.*

Synthesis Stage The synthesis stage takes the video and music features from the analysis stage, and solves the optimization problem defined in Eq 1 using Markov Chain Monte Carlo sampling. The first two energy terms determine when video clips transition from one to another on the music timeline (cut-to-the-beat principle), and between two consecutive transitions, which video clip should be used and how it is aligned with salient musical notes (synchronization principle). In addition to such unary and binary constraints, we also consider pairwise and higher-order matching constraints across cuts according to music and video features. The optimization is factorized into a two-stage process for improved efficiency: a precomputation stage that computes locally optimal video subsequences for individual music segments, and an optimization stage that samples the parameter space using the Metroplis-Hasting algorithm.

Rendering To render the final composition, we first map corresponding video subsequences to music segments with downsampling and upsampling according to the synthesis parameters. Concatenation of the resampled video subsequences form the whole video montage. The video montage is guaranteed to have the same length of the underlying music and can be played with it to render our audio-visual synthresis.

4 Analysis

4.1 Video Analysis

Video analysis estimates generic visual activities in a video using optical flows. Optical flow algorithms are effective in detecting motion caused by object movement (e.g. swaying trees, running dogs) and camera movement. We adopt a standard optical flow algorithm in [Liu et al. 2005]. Let us denote frame f of video j as

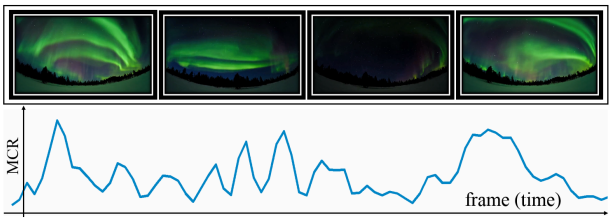


Figure 4: *Motion change rate (bottom curve) for a video sequence.*

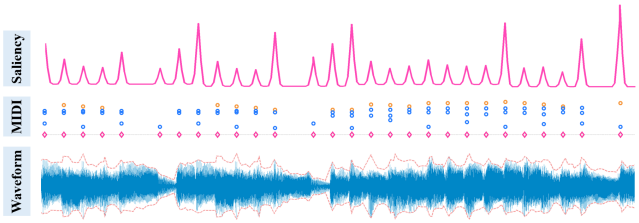


Figure 5: *Music note saliency computed from MIDI and the associated waveform data.*

$v_j(f)$, and the optical flow field between two consecutive frames as $\phi(v_j, f) = OpticalFlow(v_j(f-1), v_j(f))$ (pixel indices are omitted; same in the following notations when it is clear in context). Saliency estimation is carried out on each frame using the method in [Cheng et al. 2014]. We denote the saliency map for frame f of video j as $\alpha(v_j, f)$. This saliency map is used to reweight pixels flow field in every frame.

Motion Change Rate (MCR) We wish to detect frames with salient motion from the optical flow $\phi(v_j, f)$. Imagine the scene of a baseball player hitting a ball, the most significant moment is not when the ball is moving fast but the instant when the bat hits the ball, i.e. the moment when the ball has its maximum magnitude of acceleration. Thus, instead of directly using the optical flow, we compute the pixelwise temporal difference of the optical flow for each frame.

$$\begin{aligned} \nabla\phi(v_j, f; x) &= \phi(v_j, f; x) - \phi(v_j, f-1; x'), \\ \text{where} \quad x &= x' + \phi(v_j, f-1; x'). \end{aligned} \tag{2}$$

where x and x' are pixel indices. The key is to find, for each pixel x, its advection source x' in the previous frame given the forward flow field. Inspired by [Yang et al. 2011], we use an iterative back propagation solver to find the advection source. We then compute the mean saliency weighted motion change rate by $\Phi(v_j, f) = \frac{1}{N}\sum_{x,y}\alpha(v_j, f; x, y)\|\nabla\phi(i, f; x, y)\|/max_f\Phi(v_j, f)$ as the MCR at frame $v_j(f)$. N is the number of pixels. Figure 4 shows a MCR series of a video sequence.

Flow Peak In addition to average motion change, we would also like a feature that characterizes the most interesting bits of the flow field at a frame. To this end, we take the flow peak φ for this purpose: $\varphi(v_j, f) = prctile(\alpha(v_j, f)\|\phi(v_j, f)\|, 99.9)/max_f\varphi(v_j, f)$. The flow peak profile is also useful in general cases where a small fraction of pixels with large motion (e.g. the baseball example) can stand out without being marginalized.

Dynamism We also compute the level of dynamism as a feature that can characterize a video frame. Intuitively, this feature should reflect the percentage of pixels that have large motion. It is calculated as the ratio of pixels whose flow magnitude exceeds a threshold:

$$\delta(v_j, f) = \frac{1}{N}\sum_{x,y} 1(\|\phi(v_j, f; x, y)\| > \delta) \tag{3}$$

where δ is empirically set to 2.

Peak Frequency We use the peak frequency of visual motion at every frame as a feature to match musical pace in later optimization. This peak frequency is computed using a time window centered at the frame. For the following time window of size K centered on frame f in video j, we compute the power spectral density ρ of the following motion profile vector $(\Phi(v_j, f - K/2), ..., \Phi(v_j, f), ..., \Phi(v_j, f + K/2))$. The frequency with the maximum power spectral density (if greater than a threshold) is taken as the peak frequency at frame f:

$$\psi(v_j, f) = \begin{cases} k, & \rho(k) > -10 \text{ and } \rho(k) \geq \rho(k')\ \forall k' > 0; \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

4.2 Music Analysis

During music analysis, the following tasks are performed: (1) divide the musical sequence into a series of musical segments, each of which is a single cut; (2) compute a saliency weight for each note onset, and these weights are used for music-video matching; and (3) compute segment features for defining the transition cost between consecutive musical segments.

Music Segmentation According to the "cut-to-the-beat" principle in music video editing, and especially the insight, that cuts are used to "separate and punctuate on transition of ideas", made by Walter Murch [2001], we use the bars in the music as the atomic units for music segmentation. Every musical segment must start with a bar. Agglomerative clustering is performed bottom-up to form a hierarchical clustering tree; the bars form the initial set of segments. During each step of clustering, the pair of consecutive segments with the minimum segment distance is chosen to merge. This process is repeated until the number of notes in all segments has reached a desired segmentation granularity. Segment distance is defined as follows,

$$\begin{aligned} \chi(m_i, m_{i+1}) = w_0 \frac{|\text{pace}(m_i) - \text{pace}(m_{i+1})|}{\text{mode}(pace)} + \\ w_1 \frac{|\text{median}(\text{pitch}(m_i)) - \text{median}(\text{pitch}(m_{i+1}))|}{\sigma_{pitch}} + \\ w_2 \frac{|\sigma(\text{pitch}(m_i)) - \sigma(\text{pitch}(m_{i+1}))|}{\sigma_{pitch}}, \end{aligned} \tag{5}$$

where pace$(\cdot)$ is defined as the number of unique note onsets divided by the time length of a music sequence (bar, segment, etc.), mode$(pace)$ is the most common pace across all music bars, median(pitch(m_i)) and σ(pitch(m_i)) represent the median and standard deviation of the note pitches in segment m_i, and $\sigma(pitch)$ is the standard deviation of the note pitches in the entire musical piece. In practice, $w_0 = 10, w_1 = 1, w_2 = 1$.

Saliency We define the following eight types of binary saliency scores for note onsets: *pitch-peak, before-a-long-interval, after-a-long-interval, start-of-a-bar, start-of-a-new-bar, start-of-a-different-bar, pitch-shift, deviated-pitch*. All these saliency scores are initially set to zero at all note onsets.

- *Pitch-peak* is set to 1 if the highest pitch (multiple notes could be hit at the same time) at a note onset is higher than twice the

highest pitch at the preceding note onset and also higher than twice the highest pitch at the following note onset.

- The *Before-a-long-interval* (*after-a-long-interval*) score at a note onset is set to 1 if the following (or preceding) note onset is at least one beat away (1 beat = 60/tempo).

- *Start-of-a-bar* is set to 1 at the first note onset within each bar.

- *Start-of-a-new-bar* is set to 1 at the first note onset of new bars. A new bar is a bar with the preceding bar empty.

- *Start-of-a-different-bar* is set to 1 at the first note onset of a bar that possesses a different pattern from the preceding bar. A bar is "different" from the preceding one if match$(bar_i, bar_{i-1}) = 0$, where match(bar_i, bar_{i-1}) is set to 1 if more than 80% of their note onsets are aligned and more than 50% of their notes have the same relative pitches and match(bar_i, bar_{i-1}) is set to 0 otherwise.

- When two consecutive bars match and more than 90% of their notes maintain the same relative position, *pitch-shift* is set to the mode of pitch differences at the first note onset of the second bar.

- When two consecutive bars match and the pitch difference between notes in the second bar and their corresponding notes in the first bar is larger than two standard deviations, *deviated-pitch* is set to 1 at the first note onset of the second bar.

The final saliency score ω at a note onset t_i is accumulated as follows,

$$\omega(t_i) = (1 + \text{vol}(t_i) \sum_{i=1}^{8} \text{score}_i)/max(\omega(t_i)), \tag{6}$$

where vol$(\cdot)$ means the volume of a note and it is computed as the mean squared magnitude of the waveform samples in the first 20% of the note duration. This time window is defined according to the volume envelope of a note strike in musical performance. That is, upon the strike of a note, the first 20% of the note duration is called the attack stage, during which the sound reaches the peak volume. The attack stage is followed by the decay, sustain, and finally release stages.

We further define the following continuous temporal saliency function using the above saliency score for discrete note onsets,

$$\Omega(m_i; t) = \sum_{t_i=1}^{K} \omega(t_i) G(t - t_i; \sigma_{t_i}), \tag{7}$$

where t is the continuous timeline coordinate (t_i for discrete note onsets), $G(\cdot)$ refers to the Gaussian kernel, and σ_{t_i} is the standard deviation of the Gaussian centered at time t_i. σ_{t_i} is set such that the Gaussian at t_i diminishes to 0.01 at the boundary of a time window with its center at t_i and its width equal to $\min(0.1, 0.25(t_{i+1} - t_{i-1}))$. Figure 5 visualizes the continuous temporal saliency computed from its MIDI source and the associating waveform data.

5 Synthesis

Given the problem formulation in Section 3, in this section, we elaborate the definition of the energy terms using features extracted from the analysis stage, and our MCMC optimization algorithm for solving the synthesis parameters $\boldsymbol{\theta}$.

5.1 Energy Terms

Matching Cost To follow the *synchronization* principle, we would like see the "ups and downs" of a video sequence strongly correlate with those of the corresponding music segment. However, music and videos are very different types of media, and are not directly comparable. Therefore, we need to rely on the derived features in the previous section as well as metaphors and intuitive connections between such features to achieve this goal. For example, in the following, we use the temporal sequence of motion change rates computed for a video to represent its "ups and downs".

The matching cost term in Eq 1 is a summation of all individual matching costs between each music segment and a subsequence of its corresponding video clip,

$$E_{match}(\boldsymbol{\theta}, \mathbf{M}) = \sum_{i=1}^{q} \text{Match}(m_i, \theta_i), \tag{8}$$

where the individual matching cost can be further split into the following two terms,

$$\text{Match}(m_i, \theta_i) = \Psi(m_i, \theta_i) + \Gamma(m_i, \theta_i). \tag{9}$$

The first term in Eq (9), $\Psi(\cdot)$, is the co-occurrence cost. It penalizes temporal mismatch between the motion change rate $\Phi(a_i, t)$ and the music saliency function $\Omega(m_i, t)$. The co-occurrence cost is defined as follows,

$$\Psi(m_i, \theta_i) = \begin{cases} G(x; \sigma_{co}), & x \geq 0; \\ 2 - G(x; \sigma_{co}), & x < 0; \end{cases} \tag{10}$$

where the dot product $x = \Omega(m_i, t)^T \Phi(a_i, t)/N$ is evaluated at discrete time samples on the music timeline at the video frame rate and normalized by the number of samples N, and σ_{co} is set to 0.05. $\Omega(\cdot)$ and $\Phi(\cdot)$ are subtracted by the their mean before the dot product. Although musical pieces and video clips are not directly comparable, this dot product computes the correlation between two derived features, music saliency score and video motion change rate.

The second term in Eq (9), $\Gamma(\cdot)$, penalizes mismatch between musical pace and the detected peak frequency of the video (Section 4.1). That is, videos with a high peak frequency are preferred to match music segments with a fast pace, and vice versa. The cost function for this term is expressed as

$$\Gamma(m_i, \theta_i) = \begin{cases} 1, & pace(m_i) > 1, \psi(\theta_i) < 0.5; \\ 1, & pace(m_i) < -1, \psi(\theta_i) > 2; \\ 0, & \text{otherwise}, \end{cases} \tag{11}$$

where $pace(m_i)$ is the standardized value of pace with mean subtracted and standard deviation normalized to one, $\psi(\theta_i)$ is the peak frequency averaged over video subsequence specified in θ_i.

Transition Cost The transition cost term in Eq 1 encourages video transitions across cuts to match characteristics of musical transitions across segments. For example, if the music goes from a slow pace to a faster one, we would like to see the accompanying video also switch from a scene of slow motion to another one with faster motion; or when extra sound tracks are on, the visual scene becomes more dynamic. We define two terms in the transition cost,

$$\begin{aligned} \text{Transit}(i, i+1) \quad = \quad & \Delta(m_i, m_{i+1}, \theta_i, \theta_{i+1}) + \\ & \Lambda(m_i, m_{i+1}, \theta_i, \theta_{i+1}). \end{aligned} \tag{12}$$

The $\Delta(\cdot)$ cost evaluates the matching quality between musical pace transition and velocity transition in visual motion,

$$\Delta(m_i, m_{i+1}, \theta_i, \theta_{i+1}) = \begin{cases} 1, & \kappa_p < 0.5, \kappa_v > 0.75; \\ 1, & \kappa_p > 2, \kappa_v < 1.5; \\ 0, & \text{otherwise}, \end{cases} \quad (13)$$

where $\kappa_p = \text{pace}(m_{i+1})/\text{pace}(m_i)$ and $\kappa_v = \text{vel}(p_{i+1})/\text{vel}(p_i)$, where $\text{vel}(\cdot)$ is the mean flow magnitude in a video frame.

The $\Lambda(\cdot)$ cost evaluates the matching quality between the change in the number of musical tracks and the change in visual dynamism,

$$\Lambda(m_i, m_{i+1}, \theta_i, \theta_{i+1}) = \begin{cases} 1, & \nabla t < 0, \nabla d > -0.3; \\ 1, & \nabla t > 0, \nabla d < 0.3; \\ 0, & \text{otherwise}, \end{cases} \quad (14)$$

where $\nabla t = \text{numtrack}(m_{i+1}) - \text{numtrack}(m_i)$, and $\nabla d = \delta(v_{a_{i+1}}, sf_{i+1}) - \delta(v_{a_i}, ef_i)$. Note that the change in dynamism is calculated as the difference between the dynamism of the last frame of a video clip and that of the first frame of the following video clip, instead of using whole sequence averages.

The transition cost in Eq 1 sums up the transition costs over all musical cuts,

$$E_{transit}(\boldsymbol{\theta}, \mathbf{M}) = \sum_{i=1}^{q-1} \text{Transit}(i, i+1). \quad (15)$$

Global Constraints The global cost term in our energy function focuses on constraints on the global statistics of the music-video composition. We define a duplication cost to prevent a single video clip from appearing multiple times in the synthesis result. The duplication cost is defined as $\sum_{i=1,..,p} \left(2^{\text{count}(v_i)-1} - 1\right)$, which imposes an exponentially growing penalty as a video clip is repeatedly used in the output.

5.2 Optimization

Minimizing the energy function in Equation 1 is an intractable combinatorial problem. We factor the problem into a two-stage optimization: a precomputation that computes locally optimal video subsequences for music segments, and an MCMC Metroplis-Hasting algorithm [Andrieu 2003; Chib and Greenberg 1995] to sample the parameter space. The precomputation stage further consists of a global alignment of video subsequences to music segments, and a temporal snapping procedure that aligns local keyframes to salient music notes.

The parameter space for each music segment, $\theta_i = (v_{a_i}, sf_i, ef_i, scale_i)$, has three degrees of freedom because the position of the last frame of the video subsequence, ef_i, can be determined jointly by the length of the music segment, the position of the first frame and the temporal scaling factor for this subsequence. The entire parameter space has $3q$ degrees of freedom where q is the number of music segments. This paramter space is too large for the Metroplis-Hasting algorithm to traverse. To make the optimization computationally feasible, we factor the computation into two stages. In a pre-computation stage, we compute for each music-video candidate pair (m_i, v_j), the optimal starting position of the subsequence, sf_i^j, and the optimal temporal scaling factor for this subsequence, $scale_i^j$. Thus the 4-tuple, $(v_j, sf_i^j, ef_i^j, scale_i^j), j = 1, .., p$, is precomputed and will become the candidate set for θ_i in the second stage. In the second stage, when a music segment m_i chooses a matching video subsequence, the precomputed matching parameters for every video clip can be used immediately. In this way, the optimal matching parameters between any music-video pair need to be computed only once. With such precomputation, for a problem with 30 music segments, 60 video clips, and a maximum 1000000 sampling steps, computing its solution takes about 10-15 seconds.

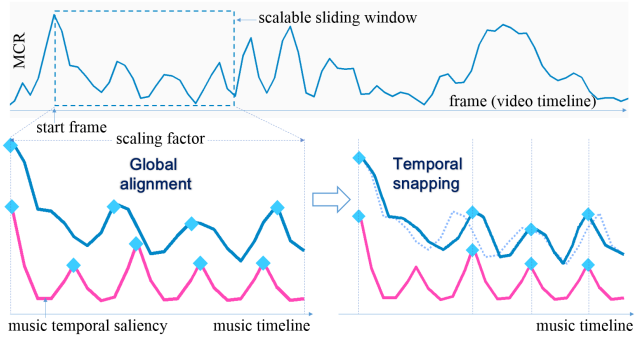


Figure 6: *Audio-visual synchronization by global alignment and non-linear snapping. The global alignment is performed in the pre-computation stage, which uses a scalable sliding window to search for video subsequences that aligns with music segments. The temporal snapping "snap" local MCR peaks to the exact music note onsets by dynamic programming.*

Global Alignment The global alignment optimizes the position of the first frame of the video subsequence and its associated temporal scaling factor for every possible combination of music segment and video clip. The matching cost for a single music-video candidate pair is adopted as the energy function. Specifically, for every possible position of the starting frame of the video subsequence, we perform a continuous optimization on the temporal scaling factor using gradient descent. Figure 6 (global alignment) illustrates this process. Since there may still exist many local minima in this simple one-dimensional optimization, in practice, we run gradient descent multiple times with regularly spaced initializations. At the end, we return the location of the starting frame and the scaling factor that together achieve the lowest matching cost.

Temporal Snapping In our original problem formulation, a constant temporal scaling factor is used for each music-video pair. However, synchronization achieved with a single scaling factor per segment cannot be perfect because visual motions usually do not follow the rhythm of music beats by a constant scaling factor (except for artificial motions such as animation and dance). We further add a temporal snapping feature to the global alignment result. It makes salient visual frames precisely "snap" to salient musical notes. Temporal snapping allows a temporally varying scaling factor within a single video clip. It identifies a set of keyframes in the video subsequence and a series of note onsets in the music segment, and then snaps a subset of video keyframes to salient musical notes by solving a matching problem using dynamic programming. Between every pair of consecutive snapping points, the video is temporally scaled to match the music length.

The following video frames are chosen as keyframes, (a1) the starting frame, (a2) the last frame, and (a3) any intermediate frame whose motion change rate is a local peak, i.e. large than that of the preceding and following frames and above the 90 percentile. Likewise, the following note onsets are labeled as salient, (b1) the first one of a music segment, (b2) the last one of a music segment, and (b3) any note onset with a saliency score 0.5 or above. Let

$\{(t_1, w_1), ..., (t_k, w_k)\}$ be the set of salient note onsets in the music along with their saliency scores, and $\{(t'_1, w'_1), ..., (t'_l, w'_l)\}$ be the set of time stamps of the keyframes in the video along with their motion change rates. Temporal snapping searches for a matching between these two sets of temporal points, $\{(t_{m_i} \rightarrow t'_{n_i})\}^{(o)}$, where $m_i \in \{1..., k\}, n_i \in \{1..., l\}, o \leq \min(k, l)$, by minimizing the following cost function,

$$\sum_{i=1}^{o} \text{match}(i) + \sum_{i=2}^{m-1} \text{smooth}(i-1, i, i+1) + \sum_{m_i \in \Upsilon} w_{m_i} + \sum_{n_i \in \Pi} w'_{n_i}, \quad (16)$$

where the first term is the matching cost between note onset (t_{m_i}, w_{m_i}) and video frame (t'_{n_i}, w'_{n_i}); the second term is the local smoothness cost; the third and fourth terms together penalize temporal points missing from the matching. The matching cost is empirically defined as

$$\text{match}(i) = (t_{m_i} - t'_{n_i})^2 - \min(w_{m_i} w'_{n_i}, 0.25),$$

and the smoothness cost is defined as

$$\text{smooth}(i-1, i, i+1) = (t_{m_i} - t_{m_{i-1}}) \max\left(\frac{\alpha}{\beta} - 1, \frac{\beta}{\alpha} - 1\right) + (t_{m_{i+1}} - t_{m_i}) \max\left(\frac{1-\alpha}{1-\beta} - 1, \frac{1-\beta}{1-\alpha} - 1\right), \quad (17)$$

where $\alpha = (t_{m_i} - t_{m_{i-1}})/(t_{m_{i+1}} - t_{m_{i-1}})$ and $\beta = (t'_{n_i} - t'_{n_{i-1}})/(t'_{n_{i+1}} - t'_{n_{i-1}})$. We solve this matching problem with the classic dynamic programming algorithm in $O(k^2 l^2)$ time. Figure 6 bottom right illustrates our non-linear temporal snapping operation.

MCMC Sampling The standard Metroplis-Hasting algorithm is implemented to sample the label space for an optimal solution. Because of the precomputation, the new label space is simply the sequence of video indices corresponding to the sequence of music segments. Two types of mutations are designed: with probability 0.7, the video index for a music segment is updated to a random index between 1 and n, where n is the total number of video clips, and with probability 0.3, two music segments' corresponding video indices are swapped. We adopt a uniform distribution over the random video index in the first type of mutations and also a uniform distribution over the pair of music segments in the second type of mutations to enforce the reversibility condition for the M-H algorithm [Chib and Greenberg 1995]. The Boltzmann distribution $P(x) = \exp(-x/T)$ is used to convert energy levels to probabilities in the algorithm. Algorithm 1 summarizes the overall MCMC based optimization. In the algorithm, $\tilde{\boldsymbol{\theta}} = \{\tilde{\theta}_i^j, i = 1, ...q, j = 1, ...p\}$ is the result from the local optimization.

Note that this two-stage factorization can only produce an approximate solution to the original problem (Eq 1). The trade off is orders of magnitude computational cost reduction.

Algorithm 1

procedure METROPLIS-HASTING($\mathbf{M}, \tilde{\boldsymbol{\theta}}, numIter$)
  Init: $\boldsymbol{\theta} = \{\theta_i | \theta_i = \arg\min \tilde{\theta}_i^{(\cdot)}, i = 1, ...q\}$
  $iter = 1$
  while $iter < numIter$ **do**
    Probability $P0 = P(E(\boldsymbol{\theta}, M))$
    Mutate if $(u(0,1) < 0.7)$ $\theta_i = \tilde{\theta}_i^j, j = rand()\%p$
      else swap(θ_i, θ_j)
    Probability $P1 = P(E(\boldsymbol{\theta}', M))$
    Accept/Reject if $p > \min(P1/P0, 1)$, undo mutation
    iter += 1
  end while
  return P
end procedure

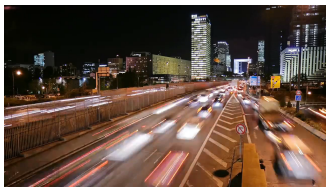
6 Rendering

It is straightforward to render the video subsequence matching each music segment m_i given θ_i and the temporal snapping parameters. A constant scaling factor is used between every pair of consecutive snapping points. Upsampling and downsampling are applied according to the temporal scaling factor in θ_i and the relative position of snapping. These scaled video subsequences are then concatenated to form the entire video montage. Finally, the music is added to the video, forming the final composition, where visual activities are synchronized with musical notes within every music segment, and transitions between music segments are echoed with suitable visual transitions, rendering an audio-visual synchresis.

7 Results and Evaluation

We have successfully applied our algorithm to a variety of video clips and background musical pieces. Here we show six examples. We have further conducted a user study on these six examples. All results have been included in the supplemental material and need to be watched with background music.

Aurora This example comprises a set of aurora and lightening scenes with the background music excerpted from *Easy Going* by Bjorn Lynne. The video collection has 36 candidate clips, each of which is 3-5 seconds long. The background music is the segment from second 10 to 30 of the original piece. It is divided into 10 segments with even length. Thus 10 out of the 36 candidate video clips were chosen to generate the montage. This result demonstrate the synchronization of motion change peaks and music beats. For example, in second 8-10, two strong optical flashes co-occur with the punctuation of two music strikes, rendering a pleasant audio-visual resonance.



City Timelapse This example features breathtaking timelapse videos shot by professional photographers. The shots were taken from Paris, Venice, and Besançon. Most of them have dynamic and rhythmic timelapse motions. We took 55 separate shots of these videos, each of 2-7 seconds long, and chose a music segment excerpted from *The Chaos Warrior* as the background music. This example demonstrates striking effects by cutting and synchronizing timelapse videos with the beat of a piece of music. This is because timelapse videos are in general more rhythmic than regular shots, and it is easier for the algorithm to find matching points between such videos and music, such as the pulsing waves in the videos and the drum beats in the music.

Happy Birthday This example demonstrates the application of our algorithm in organizing video collections of gatherings and events to a specific song. The *Happy Birthday To You* song in this example has 6 bars. We segmented the song to the finest granularity–each bar is an individual segment (thus a cut). The 18 input video clips were taken from a birthday party, each 1-7 seconds long. Our algorithm was able to find short moments in the videos to match the beat of the music from a pool of random human/artificial activities that were not intended for the rhythm of the song at all, such as the clown show, kids play, and flashing lights.

Adventure This example demonstrates the application of our algorithm in organizing video collections of outdoor activities, such as sky diving, water skiing, and canoeing. 52 short video clips were used as the visual input. The background music was excerpted from *Lottyr, Lady of the Hells* (second 40 - 85). Aside from audio-visual synchronization inbetween cuts, this example also demonstrates our algorithm's capability of automatically varying videos play speed to match the pace of the background music. In the first part of the composition, a switch to a scene of extremely slow motion was made to match a sudden slow-down in the music. The music then transitions to a faster pace and later to a highly dynamic scale with a mixture of multiple sound tracks. The associated video segments also deliver a visual experience that resonates with the music.

Wild This example covers animal, flower, desert and forest scenes in the wild. 35 video clips were used as the visual input. The background music was excerpted from *Exploration* (second 0-27). The movements of the animals (the baboon, elephant and penguins), being irregular at first, were successfully temporally scaled and snapped to the musical beats as well. Variations in pace and motion have also been demonstrated (second 15-22).

7.1 User Study

A set of user studies were designed and conducted to evaluate the qualitative performance of our algorithm comparing to 4 control groups. The first two groups are results of our algorithm with the *cut-to-the-beat* feature turned off (Group A) and the *synchronization* feature turn off (Group B), respectively. The other two groups are results created by human users (Group C and Group D). The user study reveals that our full result is significantly better than the results with either of the two features turned off. Our result is also rated higher (in most cases) than the two groups created by human users.

The user study was performed on six examples (for space limits, only five of the six are described above). For group A, we took our result, and shifted the video cutting points by a Gaussian kernel (up to 2 seconds): the length of one video was shrunk and the length of its neighbor was extended by the same amount. The shrunk part was replaced with extra frames taken from the extended video. All other audio-visual correspondence remains the same. This preserves existing synchronization in most of the frames. For group B, we perturbed the temporal scaling factor and the position of the start frame for half of the videos, and chose a random new video (with a random temporal scaling factor and start frame) for the other half, while keeping the cutting points unchanged. Group C and D are manually created results on the same music and video input set. The users were instructed to apply the cut-to-the-beat and synchronization principles during their editing. The users were also encouraged to apply any high level video editing heuristics. But the editing operations were restricted to be the same as what our algorithm does (choosing a start frame and varying the temporal scaling factor). The user who created Group C is a television and broadcast journalism major with 4 years of experience in video editing (Expert User); the user who created Group D only had a few video editing practices in the past (Average User). Both of them used the Adobe Premiere software to complete this task. Average editing time per example is about 15 minutes. The 4 control groups, plus our result, form a 6×5 video set in our subsequent user study.

Our study collected 29 participants' ratings. Each participant was asked to watch the above six sets of videos. The 5 results of each set (our result plus 4 control groups) were displayed on the same screen side by side in random order. The participant could watch several of them simultaneously, or one at a time. He/she was then asked to rate the results in a scale between 1 and 5, 1 for the worst, 5 for the best. The same score could be given to two different results if they were regarded as of indiscernible quality. Table 1 shows the average score of every group. Our result was rated the highest (3.92), followed with a margin of 0.3 by the result from Expert User (Group C). The result without the cut-to-the-beat feature and the result from Average User (Group D) fell in the next range and were close to each other. The result without synchronization had the lowest score.

Table 1 also shows average ratings by different subpopulations according to information collected from questionnaire. The first subpopulation division is expert/non-expert. In the expert subpopulation, the advantage of our result is more significant, and the margin between our result and that without synchronization is even larger. The second subpopulation division is male/female. Significance test shows the difference of the two sub-groups' ratings is not statistically significant. The last division separates the participants' ratings of the first 3 sets of videos from those of the next 3 sets in case the subjects had learned the patterns in the first half and became biased in the second half. Data from these two subpopulations did not pass the significance test neither. Notably, in all the subpopulations, our result holds an advantage of two standard deviations (computed by bootstrap sampling) away from the control groups.

We then summarize the fraction that each method was rated the best or the worst (Fig. 7 left), and head-to-head comparison between our method and the control groups in terms of the fraction that one method was rated higher than the other (Fig. 7 right). The first plot confirms that our result and that of Expert User have the highest rating, and the result without synchronization is the worst. Notably, the result from Expert User has almost the same fraction of best ratings as our result, while its average score is lower (Table 1). In the second plot, our result again, holds a consistent advantage over the control groups. Note that the sum of the fractions is not equal to 1 because if two methods were both rated 5, they are both counted as best rating.

Subpopulation	#rating	Ours	- cut-to-beat	- synchronization	Expert User 1	Average User
all	29 × 6	**3.92±0.08**	3.37±0.09	2.86±0.10	3.58±0.10	3.20±0.09
expert	8 × 6	**4.27±0.19**	3.31±0.14	2.69±0.16	3.50±0.19	2.97±0.17
non-expert	21 × 6	**3.79±0.09**	3.40±0.11	2.93±0.12	3.61±0.11	3.29±0.11
male	18 × 6	**3.94±0.10**	3.30±0.10	2.80±0.12	3.68±0.12	3.28±0.12
female	11 × 6	**3.88±0.13**	3.56±0.14	2.97±0.16	3.42±0.16	3.08±0.15
1st half	29 × 3	**3.86±0.12**	3.17±0.12	3.31±0.14	3.22±0.13	3.25±0.14
2nd half	29 × 3	**3.98±0.11**	3.57±0.11	2.41±0.12	3.94±0.13	3.15±0.12

Table 1: *Average rating (from 1 - 5, higher is better) of results by different methods and subpopulations.*

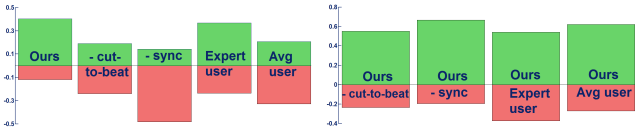


Figure 7: *Left: Fraction that one method is rated the best (green) or the worst (red). Right: Fraction that one method is rated higher than another*

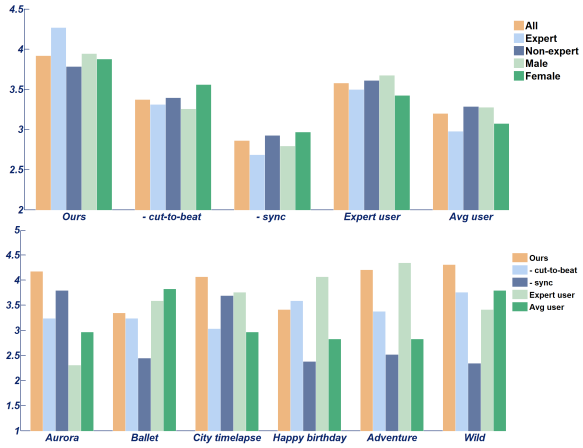


Figure 8: *Top: Average rating for different methods by different groups. Bottom: Average rating of each example among different methods.*

In Figure 8 (bottom), we compare the average user rating among the 5 groups on *each* of the six examples. The plot shows our result was rated the highest on 3 of the 6 examples (Aurora, City timelapse, and Wild). The result from Expert User was rated the highest on 2 examples (Happy birthday and Adventure). The result from Average User was rated the highest on "Ballet". Figure 8 top is a bar chart visualization of Table 1.

8 Related Work and Future Extensions

Related Work Sound computation has been on an independent track in parallel of video editing in computer graphics. The most relevant topic to our work is sound-based animation [Langlois and James 2014; Shiratori et al. 2005; Lee and Lee 2005; Cardle et al. 2002; Doel et al. 2001] and music-driven photograph slideshow [Chen et al. 2006]. In particular, our work bears some resemblance to dance-to-music character animation in [Shiratori et al. 2005], which synthesizes character animation sequences by analyzing the rhythm, structure and intensity of both the music and the motion. And coincidentally, the concept of audio-visual synchresis was also advocated recently in [Langlois and James 2014]. The key difference between our work and previous sound- or music-driven animation is that our visual component is not an animated object but existing video clips. Therefore, the degree of freedom in the manipulation of the visual component is different. Specifically, we manipulate video sequences in a *passive* mode while animation synthesis is a more *active* mode.

On the other hand, recent work on video editing suggests a trend of visual computing in a multi-modal setting. For example, [Berthouzoz et al. 2012] edit interview videos according to text transcripts, [Davis et al. 2014] use invisible visual signals to detect sound from source objects. [Arev et al. 2014] edit video footages from multiple social cameras to produce video sequences with coherent cuts. [Wang et al. 2014] compute nonlinear video alignment from an arbitrary number of videos. Our work follows this trend, and extends the application of sound-based animation to music-driven video composition.

Limitations and Future Work One limitation of the proposed algorithm is the use of MIDI as the music representation in our pipeline. Currently the waveform and MP3 are more versatile music formats. Our use of MIDI limits the applicability of the proposed approach. Therefore, it is desirable to explore the waveform or MP3 audio format in music analysis. This brings up additional technical challenges, such as music segmentation [Chai 2006] and note onset detection [Bello et al. 2004], although alternative music formats and feature engineering could easily replace the current counterparts in our computational framework. On the other hand, MIDI is gaining increasing popularity with the digital music trend. Compared to the waveform and MP3, MIDI is an organically editable audio representation. This representation not only allows more accurate semantic analysis of audio signals, but also sheds light on new research methodologies on such topics as audio encoding, audio synthesis and audio editing.

Our system has shown a great promise in automated creation of video montage without requiring a coherent storyline. To support semantics-aware editing, however, it would be necessary to have users in the loop, where humans are allowed to specify constraints in the optimization, and help the algorithm produce videos with desired semantics in an iterative manner. Another way to extend the algorithm is to develop music-video matching criteria in other dimensions, for example, timbre of sound vs color of shape, melody of music vs mood of the visual event. More sophisticated music video editing effects, such as VJing software features, could also be added. Last but not the least, given the explosive amount of visual and audio data available online, it would be of great power to have automatic algorithms that search for compatible music given video clips and vice versa.

Acknowledgements

We thank the anonymous reviewers. We also thank Yatong An and students at Zhejiang University for data collection and user

study. This work was supported in part by China 973 program under grant No. 2012CB316400, China 863 program under grant No. SS2015AA010504, NSFC grant No. 61272302, and ZJNSF Grant No. Q15F020006.

References

AGARWALA, A., DONTCHEVA, M., AGRAWALA, M., DRUCKER, S., COLBURN, A., CURLESS, B., SALESIN, D., AND COHEN, M. 2004. Interactive digital photomontage. *ACM Trans. Graph. 23*, 3, 294–302.

ANDRIEU, C., 2003. An introduction to mcmc for machine learning.

AREV, I., PARK, H. S., SHEIKH, Y., HODGINS, J. K., AND SHAMIR, A. 2014. Automatic editing of footage from multiple social cameras. *ACM Trans. Graph. (SIGGRAPH)* (August).

BAI, J., AGARWALA, A., AGRAWALA, M., AND RAMAMOORTHI, R. 2012. Selectively de-animating video. *ACM Trans. Graph. 31*, 4.

BECK, J., AND BURG, K., 2012. Cinemagraphs. `http://cinemagraphs.com/`.

BELLO, J. P., DAUDET, L., ABDALLAH, S., DUXBURY, C., DAVIES, M., S, M. B., AND MEMBER, S. 2004. A tutorial on onset detection in music signals. In *IEEE Transactions in Speech and Audio Processing*.

BERTHOUZOZ, F., LI, W., AND AGRAWALA, M. 2012. Tools for placing cuts and transitions in interview video. *ACM Trans. Graph. 31*, 4 (July), 67:1–67:8.

CARDLE, M., BARTHE, L., BROOKS, S., AND ROBINSON, P. 2002. Music-driven motion editing: local motion transformations guided by music analysis. In *Eurographics UK Conference, 2002. Proceedings. The 20th*, 38–44.

CHAI, W. 2006. Semantic segmentation and summarization of music. In *IEEE Signal Processing Magazine*, 124–132.

CHEN, J.-C., CHU, W.-T., KUO, J.-H., WENG, C.-Y., AND WU, J.-L. 2006. Tiling slideshow. In *Proceedings of the 14th Annual ACM International Conference on Multimedia*, ACM, New York, NY, USA, MULTIMEDIA '06, 25–34.

CHENG, M.-M., MITRA, N. J., HUANG, X., TORR, P. H. S., AND HU, S.-M. 2014. Global contrast based salient region detection. *IEEE TPAMI*.

CHIB, S., AND GREENBERG, E., 1995. Understanding the metropolis-hastings algorithm.

CHION, M. 1994. *Audio-Vision: Sound on the Screen*. New York: Columbia University Press.

CHUANG, Y.-Y., GOLDMAN, D. B., ZHENG, K. C., CURLESS, B., SALESIN, D. H., AND SZELISKI, R. 2005. Animating pictures with stochastic motion textures. *ACM Trans. Graph.*.

COHEN, M., AND SZELISKI, R. 2006. The moment camera. *IEEE Computer 39*, 8 (Aug).

DAVIS, A., RUBINSTEIN, M., WADHWA, N., MYSORE, G., DURAND, F., AND FREEMAN, W. T. 2014. The visual microphone: Passive recovery of sound from video. *ACM Trans. Graph. (Proc. SIGGRAPH) 33*, 4, 79:1–79:10.

DMYTRYK, E. 2010. *On Film Editing: An Introduction to the Art of Film Construction*. London: Focal Press.

DOEL, K. V. D., KRY, P. G., AND PAI, D. K. 2001. Foleyautomatic: Physically-based sound effects for interactive simulation and animation. In *Computer Graphics (ACM SIGGRAPH 01 Conference Proceedings)*, ACM Press, 537–544.

GOODWIN, A. 1992. *Dancing in the distraction factory: Music, television and popular culture*. Minneapolis: University of Minnesota Press.

HUBER, D. M. 1991. *The MIDI Manual*. Indiana: SAMS.

IRANI, M., ANANDAN, P., BERGEN, J., KUMAR, R., AND HSU, S. 1996. Efficient representations of video sequences and their applications. In *Signal Processing: Image Communication*, 327–351.

JOSHI, N., MEHTA, S., DRUCKER, S., STOLLNITZ, E., HOPPE, H., UYTTENDAELE, M., AND COHEN, M. 2012. Cliplets: Juxtaposing still and dynamic imagery. *Proceedings of UIST*.

KOPF, J., CHEN, B., SZELISKI, R., AND COHEN, M. 2010. Street slide: Browsing street level imagery. *ACM Trans. Graph. 29*, 4 (July), 96:1–96:8.

LANGLOIS, T. R., AND JAMES, D. L. 2014. Inverse-foley animation: Synchronizing rigid-body motions to sound. *ACM Trans. Graph. (SIGGRAPH) 33*, 4.

LEE, H.-C., AND LEE, I.-K. 2005. Automatic synchronization of background music and motion. In *Computer Graphics Forum, 24 (3).*, 353–362.

LIAO, Z., JOSHI, N., AND HOPPE, H. 2013. Automated video looping with progressive dynamism. *ACM Trans. Graph. 32*, 4 (July), 77:1–77:10.

LIU, C., TORRALBA, A., FREEMAN, W. T., DURAND, F., AND ADELSON, E. H. 2005. Motion magnification. *ACM Trans. Graph. 24*, 3 (July), 519–526.

MURCH, W. 2001. *In the blink of an eye*. Los Angeles: Silman-James Press.

PRITCH, Y., RAV-ACHA, A., AND PELEG, S. 2008. Nonchronological video synopsis and indexing. *IEEE Trans. on Pattern Anal. Mach. Intell. 30*, 11.

SCHÖDL, A., SZELISKI, R., SALESIN, D. H., AND ESSA, I. 2000. Video textures. In *Proceedings of SIGGRAPH 2000, Computer Graphics Proceedings, Annual Conference Series*, 489–498.

SHIRATORI, T., NAKAZAWA, A., AND IKEUCHI, K. 2005. Dancing-to-music character animation. 353–362.

SWIFT, A., 2014. An introduction to midi. `http://www.doc.ic.ac.uk/~nd/surprise_97/journal/vol1/aps2/`.

WANG, O., SCHROERS, C., ZIMMER, H., GROSS, M., AND SORKINE-HORNUNG, A. 2014. Videosnapping: Interactive synchronization of multiple videos. *ACM Trans. Graph. 33*, 4 (July), 77:1–77:10.

WESSELING, C. 2004. *Experiencing music video: Aesthetics and cultural context*. New York: Columbia University Press.

YANG, L., TSE, Y.-C., SANDER, P. V., LAWRENCE, J., NEHAB, D., HOPPE, H., AND WILKINS, C. L. 2011. Image-based bidirectional scene reprojection. In *Proceedings of the 2011 SIGGRAPH Asia Conference*, SA '11, 150:1–150:10.