Write-A-Video: Computational Video Montage from Themed Text

MIAO WANG, State Key Lab of Virtual Reality Technology and Systems, Beihang University; Tsinghua University, Beijing
GUO-WEI YANG, BNRist, Tsinghua University, Beijing
SHI-MIN HU*, BNRist, Tsinghua University, Beijing
SHING-TUNG YAU, Harvard University
ARIEL SHAMIR, The Interdisciplinary Center, Herzliya

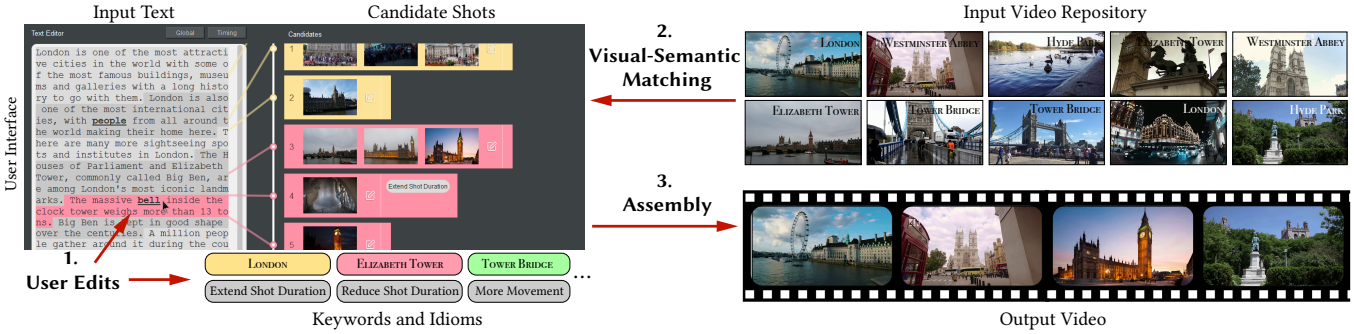

Fig. 1. The general *Write-A-Video* pipeline proceeds in three steps: (1) the user writes text and edits the attributes of text segments via a novel interface (highlighted in pink in the Text Editor), (2) candidate shots are retrieved automatically from an input video repository using visual-semantic matching, and (3) final movie shots are assembled by optimizing cinematographic rules with user-specified idioms.

We present *Write-A-Video*, a tool for the creation of video montage using mostly text-editing. Given an input themed text and a related video repository either from online websites or personal albums, the tool allows novice users to generate a video montage much more easily than current video editing tools. The resulting video illustrates the given narrative, provides diverse visual content, and follows cinematographic guidelines. The process involves three simple steps: (1) the user provides input, mostly in the form of editing the text, (2) the tool automatically searches for semantically matching candidate shots from the video repository, and (3) an optimization method assembles the video montage. Visual-semantic matching between segmented text and shots is performed by cascaded keyword matching and visual-semantic embedding, that have better accuracy than alternative solutions. The video assembly is formulated as a hybrid optimization problem over a graph of shots, considering temporal constraints, cinematography metrics such as camera movement and tone, and user-specified cinematography idioms. Using our system, users without video editing experience are able to generate appealing videos.

*Shi-Min Hu is the corresponding author (shimin@tsinghua.edu.cn). This work was mainly completed while Miao Wang was a Postdoc at Tsinghua University.

Authors' addresses: Miao Wang, State Key Lab of Virtual Reality Technology and Systems, Beihang University; Tsinghua University, Beijing, miaow@buaa.edu.cn; Guo-Wei Yang, BNRist, Tsinghua University, Beijing, ygw19@mails.tsinghua.edu.cn; Shi-Min Hu, BNRist, Tsinghua University, Beijing, shimin@tsinghua.edu.cn; Shing-Tung Yau, Harvard University, yau@physics.harvard.edu; Ariel Shamir, The Interdisciplinary Center, Herzliya, arik@idc.ac.il.

0730-0301/2019/11-ART177 $15.00
https://doi.org/10.1145/3355089.3356520

CCS Concepts: • **Information systems** → **Multimedia content creation**.

Additional Key Words and Phrases: Video Montage, Computational Cinematography, User Interaction

ACM Reference Format:
Miao Wang, Guo-Wei Yang, Shi-Min Hu, Shing-Tung Yau, and Ariel Shamir. 2019. Write-A-Video: Computational Video Montage from Themed Text. *ACM Trans. Graph.* 38, 6, Article 177 (November 2019), 13 pages. https://doi.org/10.1145/3355089.3356520

1 INTRODUCTION

Intelligent tools that assist inexperienced users in creative processes are becoming more abundant: for image editing, for drawing and even for 3D modeling and fabrication. One process that is still challenging for novices is the creation and editing of video. Professional video editors use editing tools such as *Adobe Premiere* or *Apple Final Cut Pro* to manipulate raw footage and produce a coherent video according to a narrative or storyline. However, non-professionals may find it difficult to handle and learn how to use such software, and can lack cinematographic or aesthetic knowledge for video editing.

In contrast to video editing, the ability to tell a story using text is simpler and more widespread. Therefore, in this work, we present a method that bridges the gap between editing text and editing video—we present *Write-A-Video*, a computational tool that generates video montage from themed text. The tool allows the user to create a video by editing text. We use the term "themed text" to emphasize that our method does not work on arbitrary text, but rather on inputs where the subject is focused, not too abstract, and not too specific. We assume that the user wants to create a video using a narrative text

such as a travel video (*e.g.* visiting London), or a video illustrating a given song.

A video repository related to the subject is assumed to be available, gathered from online resources or from personal albums. One of the main challenges in our setting is how to match the semantics of the text to the visual content of the video shots. We propose to divide the text into segments, and for each segment retrieve candidate shots from the repository using visual-semantic matching. Another key challenge is how to use the text to assemble a video given the set of candidate shots. We formulate shot selection and assembly as an optimization problem that considers cinematographic guidelines, temporal constraints and user preferences for cinematography idioms. Towards this end, we present a novel method to enforce cinematographic guidelines based on 2D cues in video, and build on previous idiom-based editing approaches to efficiently automate low-level editing tasks based on high-level objectives given by the user for the text segments.

Our intelligent tool is illustrated in Figures 1 and 4. The user can add text in the Text Editor, while the tool automatically finds appropriate shots for segmented text fragments. Text editing operations such as insertion or deletion, and changing the order of segments, are converted to editing of the video montage by adding, deleting or changing the order of shots respectively. The tool allows the user to explore visual styles for each segment using cinematographic idioms and other high-level directives. At any point, the user can render the movie and preview the video montage result, which is assembled by optimization from the candidate shots with an accompanying voice-over narration. Thus, we combine the abilities of an intelligent tool for video assembly and an inexperienced user using text processing to create a video montage that illustrates the given narrative, follows cinematographic guidelines, and provides diverse visual content.

The proposed approach has been tested on various pieces of themed text and video repositories, with quantitative evaluation and user studies. Users without video editing experience could produce satisfactory videos using our tool. The user study results demonstrate that all video assembly energy terms are meaningful to the video montage quality. Moreover, the tool lets even novice users produce results significantly faster than skilled video editors using commercial frame-based editing software.

Our work expands on the concept of "text-based video editing" by introducing a tool for creating video montages from themed text. Our tool lets users operate on text instead of manually editing potentially large collections of video. More broadly, this work demonstrates the potential of automatic visual-semantic matching in idiom-based computational editing, offering an intelligent way to make video creation more accessible to non-professionals.

2 RELATED WORK

Content-based Video Indexing and Retrieval. Content-based video indexing and retrieval has a long research history in computer vision [Hu et al. 2011; Smoliar and Zhang 1994]. Ramesh Naphade *et al.* [2002] proposed a probabilistic multimedia object model to represent high-level semantics of video. The VideoScapes system [Tompkin et al. 2012] builds a graph to index unstructured videos with 3D reconstruction technology. The Video Digests system [Pavel et al. 2014] provides a new format for browsing and skimming informational videos. The Event Sketch was proposed to represent dynamic properties of semantic object actions from video events [Zhang et al. 2016]. Voice has been employed to experience and navigate how-to videos [Chang et al. 2019]. In the deep learning era, visual-semantic embedding techniques [Faghri et al. 2018; Karpathy and Fei-Fei 2015; Miech et al. 2018] were proposed to match visual content to other forms of semantics using neural networks. Our method uses visual-semantic embedding [Faghri et al. 2018] as a component to retrieve shots semantically matching text segments. In our experiments, we have demonstrated the effectiveness of our visual-semantic matching method with various themed texts. Nevertheless, any new visual-semantic embedding algorithms could replace the current component.

Computational Cinematography. Film editing requires professional skills to ensure that fluent narration and artistic expression are experienced by viewers. Recently, computational cinematographic models have been investigated by computer graphics researchers, with the goal of creating better tools for both professionals and non-professionals. Several works have addressed use of cinematographic rules in 3D animation and virtual reality [Elson and Riedl 2007; Galvane et al. 2015; Serrano et al. 2017]. With the holistic sense of the scene, the content is rendered by planing an optimal camera path, driven by continuity of actors, actions and camera movements. For live videos, domain specific footage editing systems have been developed, including lecture videos [Heck et al. 2007; Machnicki and Rowe 2001; Pavel et al. 2014; Shin et al. 2015], social camera videos [Arev et al. 2014], instructional videos [Chi et al. 2013; Truong et al. 2016], portrait videos [Berthouzoz et al. 2012; Huang et al. 2017], dialogue scenes [Leake et al. 2017], first-person videos [Joshi et al. 2015; Wang et al. 2018, 2019] and aerial videos [Xie et al. 2018; Yang et al. 2018]. Live video-footage editing relies on semantic cues and computational features that are usually extracted from the video content in a pre-processing step. In our application, the input videos are diverse and may have little overlap, so are unsuitable for 3D scene reconstruction or tracking. Therefore, our method assesses computational cinematography rules using 2D-based video cues.

Intelligent Video Editing. Frame-based video editing as found in commercial software can be difficult for non-professionals to use. Recently, user-friendly editing tools have been presented to simplify the tedious video editing process. Methods have been proposed to improve the performance of alignment or retrieval of video content from either film scripts or book chapters [Pavel et al. 2015]. Such retrieval provides fast jumps and navigation of video by navigating a script or a text-based story [Hu et al. 2011]. Our goal differs, aiming to edit video footage by editing text. In terms of video footage editing, the Hitchcock system [Girgensohn et al. 2000] provides a semi-automatic approach to home video editing, by analyzing input videos and rating unsuitability scores. The user can create a movie by dragging rated video shots into a storyboard and manually adjusting shots' lengths. Berthouzoz *et al.* [2012] presented an interview video editing tool based on editing an interview script: each time the user updates the script, a continuous shot is regenerated to

Table 1. Brief comparison of our method and representative video montage works. **Async. Takes** stands for synchronized input videos; **Scene Variation** stands for single or multiple scenes in a video; **Shots Per Seg.** stands for the number of shots can be included in a text segment.

Methods	Input		Editing		Optimization	Result	
	Video Source	Async. Takes	Editing Texts	Editing Idioms		Scene Variation	Shots Per Seg.
Social Cameras [Arev et al. 2014]	Personal Album	No	-	-	Frame-wise DP	Single (Event)	-
Dialogue Video [Leake et al. 2017]	Personal Album	No	No	Global	HMMs	Single (Dialogue)	Single
QuickCut [Truong et al. 2018]	Personal Album	Yes	No	-	Frame-wise DP	Multiple	Multiple-Interactive
Write-A-Video (Ours)	Personal Album or Internet	Yes	Yes	Global or Local	Hybrid DP	Multiple	Multiple-Automatic

coincide with the script. Jain *et al.* [2015] proposed a video cropping-and-warping algorithm based on human gaze analysis. Zhang *et al.* [2015; 2018] presented algorithms to detect and remove visual distractors via camera path optimization. Arev *et al.* [2014] presented an algorithm to create transitions following cinematographic rules between synchronized videos, but it relies on 3D reconstruction of the scene and a large overlap between camera views. Liao *et al.* [2015] synthesize videos driven by music, not text. Video editing algorithms [Bellini et al. 2018; Davis and Agrawala 2018] have been proposed to map the visual motion of an input video to the beat of background music. Kim et. al. [2014] considered summarizing online videos given photo streams as storyline guidance. However the output video neither follows a narrative story, nor uses cinematographic rules and timing constraints. Leake *et al.* [2017] developed an idiom-based interface for dialogue-driven footage editing. Users can globally edit film-editing idioms; the tool then generates footage editing results. Our work uses a similar approach for idiom based guidance, and further provides fine segment-level controls.

The closest tool to our work is QuickCut [Truong et al. 2016], which allows interactively editing of narrated video for a script. The main differences between QuickCut and our method are as follows. (1) In QuickCut, personally captured shots for a fixed script are collected. In contrast, our method processes shots either downloaded from the Internet or provided by the user without accurate correspondence to a pre-determined script. (2) In the QuickCut interface, the input text is fixed while manual candidate shot selections must be provided for each segmented text. In contrast, our interface allows the user to edit the text. More importantly, the assembled shot sequence can be automatically optimized. (3) QuickCut optimization supports *alternatives* mode and *ordered-grouping* mode. In *alternatives* mode, only one shot is employed to represent a sentence, while in *ordered-grouping* mode, candidate shots and their order must be determined manually. Our optimization process automatically determines the shot sequence and the best cuts following cinematographic rules and idioms. Table 1 briefly compares our method and prior art.

3 OVERVIEW

Write-A-Video aims to convert a themed text to a video montage by assembling shots from a video repository. The design objectives for our solution include the following.

- The video content should illustrate the narration.
- The video content should be diverse.
- The video montage should follow cinematographic rules, concerning visual continuity, tone, duration etc.
- The user should be able to iteratively edit the text and the video montage results should be updated accordingly.

The first three objectives regard automatic cinematography and the last one regards the usability of our system. To meet the last objective, we develop an intelligent tool supporting idiom-based editing of text segments to allow iterative video montage editing (Section 5). To meet the first objective, we break the whole text into text segments, and perform visual-semantic matching between text segments and segmented video shots (Section 4). In essence, each segment provides more accurate matching than the whole text. To ensure content diversity, we allow users to merge or split segmented text, and edit segment-wise visual styles by editing segment attributes. The third objective is met using a hybrid optimization algorithm using dynamic programming, jointly considering shot-wise, cut-wise and segment-wise cinematographic guidelines and user defined high-level idioms, to gather the best matching shots and cut positions (Section 6).

4 VISUAL-SEMANTIC MATCHING

The key to composing a video illustration of a given text is a search for candidate shots matching text segments. Towards this end we use a visual-semantic matching approach including two cascaded steps: keyword matching and visual-semantic embedding. We assume that for each themed text, a set of keywords such as {London, Tower Bridge, ...} or {zebra, giraffe, ...} are provided by the user. Such keywords are used to label the segmented texts and to index video shots in the repository.

For a given text segment with one or more labeled keywords, the *keyword matching* step retrieves all video shots indexed by the same keywords from the video repository. Next, a matching score is calculated for each retrieved shot based on a *visual-semantic embedding* technique; the top ranked ones are selected as candidates. Figure 2 illustrates the whole visual-semantic matching procedure.

4.1 Text Labeling and Shot Indexing Using Keywords

Both texts and videos are associated with a set of keywords indicated by the user. The user can either define descriptive keywords relevant to the themed text in advance, or incrementally add new keywords to the keyword set during editing.

Text Labeling. The themed text is represented by a sequence of text segments. Each segment r is a continuous interval of text, from part of a sentence up to several sentences. Initially, each sentence is a segment; during editing, segments can be merged or split. For each text segment, we use string matching to search for keywords, and use matched keywords to label the text. If no matches are found, the labels of the current segment follow those of its previous neighboring segment, but they can be modified during editing.

Shot Indexing. All videos in the repository are segmented into shots. Shots provide flexibility in semantic matching, and allow use of cinematographic guidelines at a fine level while assembling

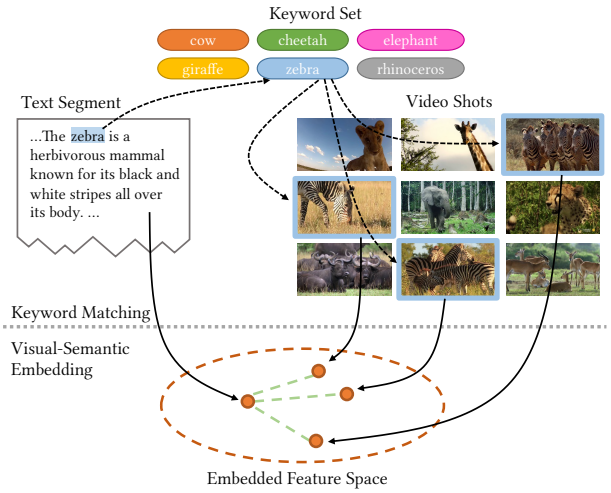

Fig. 2. Visual-semantic matching. All possible keywords from the keyword set are sought in each text segment. Next, video shots indexed using the same keywords are retrieved. Finally, the text segment and retrieved shots are embedded into a visual-semantic feature space to compute similarity scores. Top ranked shots are returned as candidates for video montage of each segment.

the video montage. We use a simple but effective histogram-based segmentation algorithm similar to the one in [Chu et al. 2015]. A shot boundary is marked between frames if their histograms in HSV color space differ by more than 80%, and more than 80% of tracked SURF keypoints [Bay et al. 2006] fail to match between the frames. We discard very short (< 2 seconds) and very long (> 30 seconds) shots, as short ones reduce visual quality, and long ones reduce efficiency and variability.

The shots are indexed using the aforementioned keywords. If the video repository is collected from the Internet using keyword searching, then these are used for indexing the shots. If a keyword is an object class label from an object detection dataset, we employ object detection on the shots and index shots according to object presence. For example, we automatically add labels for keywords such as "zebra" and "giraffe" to various shots using an object detection network [He et al. 2017] pre-trained on the MS-COCO dataset [Lin et al. 2014]. We also use face recognition to label shots containing specific people, given their portrait photos. Other object detection or action recognition methods [Caba Heilbron et al. 2015; Ma et al. 2016] could also be used to index shots. Finally, further keyword labels can be manually assigned to shots.

4.2 Visual-Semantic Embedding

To quantitatively compute matching scores between text segments and video shots with the same indexing keywords, we use the state-of-the-art visual-semantic embedding with hard negatives (VSE++) approach [Faghri et al. 2018]. This method learns to encode cross-model content such as a text r and an image i into a joint feature space using a neural network with triplet loss. In the embedded space, the feature similarity between the encoded objects reveals the similarity of cross-model content. Let $\eta(i, r)$ be the visual-semantic similarity between an image i and a text r. Given

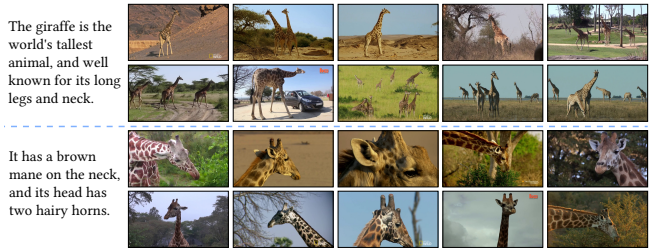

Fig. 3. Shot similarity results from two different textual segments. Each result shows a text segment and the corresponding top 10 retrieved shots.

a positive pair (i, c) (*e.g.* a text describing an image), the hardest negative samples for learning are given by $i' = \text{argmax}_{j \neq i}\, \eta(j, r)$ and $r' = \text{argmax}_{u \neq r}\, \eta(i, u)$. The loss function for optimizing the feature embedding network is defined as:

$$\mathcal{L}(i, r) = \max_{r'}[\tau + \eta(i, r') - \eta(i, r)]_+ + \max_{i'}[\tau + \eta(i', r) - \eta(i, r)]_+, \quad (1)$$

where $\tau = 0.2$ serves as the margin parameter, and $[x]_+ \equiv max(x, 0)$. Using this loss function, a ResNet152 network [He et al. 2016] is trained on the MS-COCO captioning data [Lin et al. 2014] with the output embedding dimension set to 2048. For more details, please refer to [Faghri et al. 2018]. We clarify that visual-semantic embedding is an active area of research in computer vision. Our system uses the VSE++ algorithm [Faghri et al. 2018] as a modular component, that can be replaced by newer or better alternatives as the field progresses.

The similarity between a text r and a shot s is computed as the average cosine similarity between the embedding of r and the embedding of a set of frames sampled from s (we uniformly sample one in ten frames). The top K =10 ranked shots are returned as the candidates for each query text segment. Figure 3 shows shot retrieval results for two segments from the Animals in Africa example.

5 USER INTERFACE AND INTERACTION

Our method provides an interactive video montage interface, allowing the user to concentrate on writing the text and editing the segments without tedious editing of video frames. Figure 4 shows the user interface. It includes four main components: the Text Editor, the Candidates window, the Keywords and Idioms window and the Preview window.

The Text Editor is the primary tool in which the user performs most of the work. It is an enhanced text editor with extra features. While the user can write text as in a normal text editor, the interface additionally provides functions to manipulate text segments. Initially, each sentence is a text segment. Users can right click the mouse between two segments and select "Merge Segments" to merge two neighboring segments, or select "Split Segment" to split one. The default keyword labeled in each text segment can be changed or supplemented from the keyword set using a drag-and-drop operation; the keyword set can be expanded. Users can easily edit segment attributes, including inserting *local visual-semantic match points*, performing *split timing editing*, and assigning *cinematographic idioms* for diverse stylistic control in shot assembly.

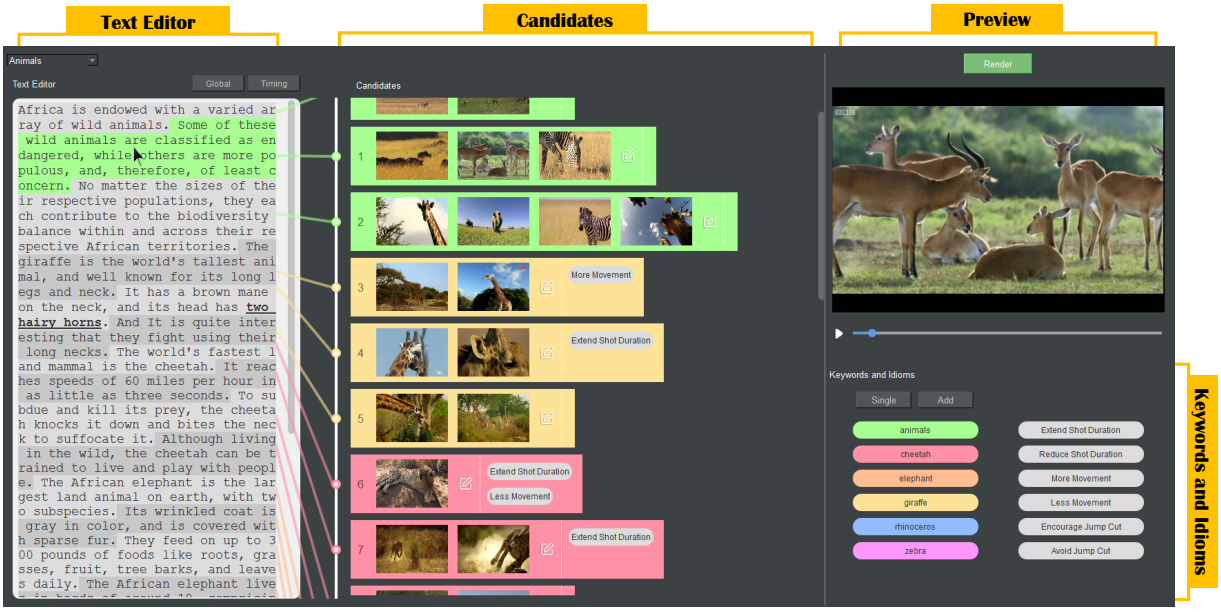

Fig. 4. The interactive video montage interface. The **Text Editor** is the main component for text and segment attribute editing. The **Candidates** window visualizes candidate shots for text segments using shot thumbnails. The **Keyword and Idioms** window provides the current keyword list and cinematography idioms that can be assigned to text segments. Any shot and the assembled video can be played in the **Preview** window.

Local visual-semantic match points allow users to require an exact visual-semantic match for a word or phrase instead of the whole segment. Local visual-semantic match points can be set by selecting some text, right-clicking the mouse and choosing "Visual-Semantic Match". The selected text will be highlighted using a bold font with an underline. Shots that better match the semantics of marked text are automatically retrieved and used in the resulting video.

Split timing editing provides a stylistic effect that intentionally delays the start time of audio or video. By default, the timings of shots match the start and end of the audio of the text segments. Users can click the "Timing" button and drag the visual cut indicator forwards or backwards from the segment boundary for cut adjustment.

Cinematographic idioms can be assigned to each text segment, to further adjust the visual style of the assembled video. Cinematographic idioms were used by Leake *et al.* [2017] for fast manipulation of style, for example, to avoid jump cuts. However, instead of setting a global idiom configuration for the whole movie, our tool allows the user to locally edit such idioms for segments. In our implementation, the user can require more or fewer shots in a text segment and encourage the shots to have more static or more dynamic camera movements by dragging these idioms onto a segment.

Once editing is finished, *Write-A-Video* automatically runs shot retrieval (Section 4) and shot assembly optimization (Section 6), and displays thumbnails of the selected shots in the Candidates window. Each row represents a shot sequence linked to a text segment. The user can expand the view of each row to explore the top candidate shots, click on a thumbnail to watch the shot in the Preview window, and select or de-select specific shots for detailed control of assembly. The whole video can be rendered with a voice-over and displayed in the Preview window by pressing the "Render" button. To synthesize voice-over audio, we use the off-the-shelf text-to-speech synthesis technique WaveNet [Oord et al. 2016]. Results without segment attribute editing provide a default cinematographic style, while the user can iteratively edit the text and its attributes to explore results in diverse styles.

6 CINEMATOGRAPHY-AWARE SHOT ASSEMBLY

Cinematographic guidelines are commonly used in professional films to enhance the flow of the story [Dmytryk 1984; Niu and Liu 2012]. We propose a cinematography-aware shot assembly algorithm that depends on 2D-based camera motion estimation and tone analysis, which can be efficiently computed. Shot assembly is formulated as an optimization problem that composes the output video by choosing and cutting a sequence of shots $\widehat{S}$ that follows cinematographic guidelines, text timing, and user-specified segment attributes (see Section 5). The novelty of our shot assembly process lies in that: (1) the cinematographic guidelines, including camera movement, tone and duration, are estimated from 2D-based cues and (2) a hybrid optimization method is employed, considering shot-wise, cut-wise and segment-wise energies together.

6.1 2D-based Cinematographic Guidelines

Computational cinematographic guidelines are often estimated and used in automatic video editing by reconstructing 3D scenes [Arev et al. 2014], or using facial landmark alignment [Leake et al. 2017]. In our setting, such techniques are not always applicable. The shots retrieved using visual-semantic matching may not contain the same scene or object for 3D scene reconstruction, and faces may not be available for alignment. Below we describe how such guidelines are formulated and adapted to our setting; more details are provided in Appendix A.

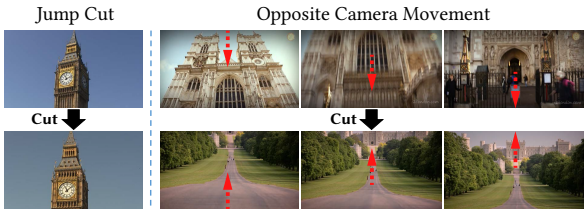

Fig. 5. Jump cuts and opposite camera movements should be avoided in consecutive shots. In a jump cut (left) the visual content of two consecutive shots is close but still different. Opposite camera movement (right) means the camera path directions are opposite before and after the cut.

Content Stability. Shots with greater stability are preferred in assembling the video. The stability of a shot s, denoted by $F_{\text{stab}}(s)$, depends on local acceleration of the video content: the greater the acceleration, the lower the content stability. The stability term is computed using 2D feature-based frame alignment between consecutive frames [Bay et al. 2006; Fischler and Bolles 1981], and represented as the negative average of frame corner offsets needed for aligning neighboring shot frames.

Saturation and Brightness. It has been shown [Niu and Liu 2012] that professional shots are typically vivid and bright. Shots with frames having high saturation and brightness are preferred. We compute the saturation and brightness of frames using color histograms. The tonal term $F_{\text{tone}}(s)$ for a shot s is measured as the ratio of high saturation and brightness regions over the shot.

Shot Duration. Varying the shot duration can affect visual smoothness. Short-duration shots with frequent cuts can be disturbing, while long-duration shots without content diversity can be boring. The shot duration term $F_{\text{dur}}(s)$ is computed as the deviation from a standard duration, set to 3.5 seconds to avoid unnecessary long-short duration changes [Niu and Liu 2012]. The standard duration can be globally adjusted to suit the user's preference.

Jump Cuts. Jump cuts should be avoided (see Figure 5). Jump cuts appear when camera positions and angles of two consecutive shots are different but similar, leading to a noticeable portion of overlap between their frames. We detect jump cuts $F_{\text{JC}}(s, s')$ between neighboring shots (s, s') by feature alignment on SURF keypoints [Bay et al. 2006; Fischler and Bolles 1981].

Opposite Movements. Neighboring shots with opposite camera movement should be avoided (see Figure 5). We detect opposite movement $F_{\text{OM}}(s, s')$ before and after a cut between shots (s, s') by comparing the estimated 2D movements from shot frame corners.

Tonal Continuity. Continuity of saturation and brightness are important to the smoothness of visual perception. The tonal continuity term $F_{\text{TC}(s,s')}$ for a shot pair (s, s') is measured by histogram differences of saturation and brightness. See Appendix A.

6.2 Hybrid Optimization of Video Assembly

The input text $\mathcal{R} = \langle r_1, r_2, \cdots, r_L \rangle$ is composed of L segments with associated segment attributes $\mathcal{A} = \langle A_1, A_2, \cdots, A_L \rangle$. Each text segment r_l starting at time t_l has an associated attribute set A_l, and a set of candidate shots $S_l = \{s_l^i | 1 \le i \le K\}$. The goal of shot assembly

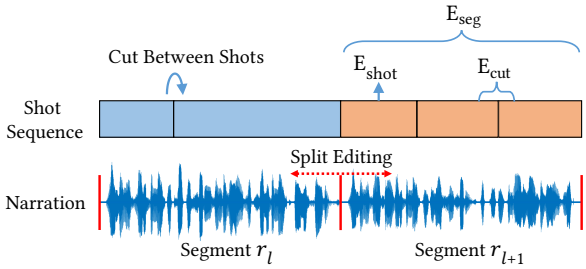

Fig. 6. Shot assembly stage. Hybrid energy optimization is based on E_{shot} for individual shots, E_{cut} for transition between shots and E_{seg} for matching between shots and text segments attributes.

is to generate a sequence of shots $\widehat{\mathcal{S}} = \langle \widehat{S}_1, \widehat{S}_2, \cdots, \widehat{S}_L \rangle$ by selecting and ordering a subset of candidate shots and cutting between them. We present a novel hybrid, cinematography-aware optimization objective, that considers shot-wise, cut-wise and segment-wise energies:

$$\text{E}(\widehat{\mathcal{S}}, \mathcal{A}) = \sum_{l=1}^{L} (\text{E}_{\text{shot}}(\widehat{S}_l) + \text{E}_{\text{cut}}(\widehat{S}_l) + \text{E}_{\text{seg}}(\widehat{S}_l, A_l)), \tag{2}$$

where *shot energy* $\text{E}_{\text{shot}}(\widehat{S}_l)$ evaluates the quality of individual selected shots in the l-th segment, *cut energy* $\text{E}_{\text{cut}}(\widehat{S}_l)$ measures the cuts between consecutive selected shots in the l-th segment, as well as the boundary cuts, and *segment energy* $\text{E}_{\text{seg}}(\widehat{S}_l, A_l)$ measures the assembled shots in terms of segment attributes. Figure 6 briefly illustrates the above energy terms. Compared to existing video editing methods (see Table 1), this hybrid optimization approach not only ensures the diversity of visual styles, but also preserves the style consistency of each segment.

Shot Energy. When considering a candidate shot sequence $\widehat{S}_l$ for text segment r_l, the shot energy term is considered for each shot s independently from other shots, including its visual-semantic matching score $F_{\text{vsm}}(s)$, camera stability $F_{\text{stab}}(s)$, and tonal term $F_{\text{tone}}(s)$. The shot energy term E_{shot} is defined as:

$$\text{E}_{\text{shot}}(\widehat{S}_l) = \frac{1}{|\widehat{S}_l|} \sum_{s \in \widehat{S}_l} \alpha_1 F_{\text{vsm}}(s) + \alpha_2 F_{\text{stab}}(s) + \alpha_3 F_{\text{tone}}(s),$$

where $\widehat{S}_l$ is a possible shot sequence segment corresponding to r_l in the assembled video, and $\alpha_1 = 1.0, \alpha_2 = 0.3, \alpha_3 = 0.3$ are the weights for the shot energy terms.

Cut Energy. We measure the visual compatibility of two consecutive shots $\langle s, s' \rangle$ in the output video by measuring whether opposite camera motions and jump cuts are avoided ($F_{\text{OM}}(s, s')$ and $F_{\text{JC}}(s, s')$), and whether tonal continuity is preserved ($F_{\text{TC}}(s, s')$). The overall *cut energy* E_{cut} for all neighboring shots related to text segment r_l is formulated as:

$$\text{E}_{\text{cut}}(\widehat{S}_l) = \frac{1}{|C|} \sum_{s \in \widehat{S}_l} \beta_1 F_{\text{OM}}(s, s') + \beta_2 F_{\text{JC}}(s, s') + \beta_3 F_{\text{TC}}(s, s'),$$

where $\widehat{S}_l$ is a possible shot sequence segment corresponding to r_l, s' is the shot following shot s, $|C|$ is the number of cuts, and $\beta_{\{1,2,3\}} = -1.0$ are the uniform weights of the cut energy terms.

Fig. 7. Representative frames from the 7 video repositories for video montage. Image courtesy of YouTube users Kendra Cus, ENG FRED, Gary Pilarchik, Billy Wolffe, Howl Of A Dog, Min Kids TV.

Segment Energy. Let the sub-sequence of shots in the video assembly for text segment r_l be $\widehat{S}_l$, and A_l be the segment attributes assigned by the user. A_l contains local visual-semantic match points, split timing editing, and duration and content movement idioms. Based on these attributes, let $F_{\text{match}}(\widehat{S}_l, A_l)$ be the local visual-semantic match energy, $F_{\text{split}}(\widehat{S}_l, A_l)$ be the split timing editing energy, and $F_{\text{idiom}}(\widehat{S}_l, A_l)$ be the cinematographic idiom energy. The segment energy for text segment r_l is given by:

$$\mathrm{E}_{\text{seg}}(\widehat{S}_l, A_l) = \gamma_1 F_{\text{idiom}}(\widehat{S}_l, A_l) + \gamma_2 F_{\text{match}}(\widehat{S}_l, A_l) + \gamma_3 F_{\text{split}}(\widehat{S}_l, A_l),$$

where $\gamma_1 = -1.0, \gamma_2 = 3.0, \gamma_3 = -3.0$ are the segment term weights. All the term weights were fixed to provide the best visual quality based on our experiments.

6.3 A Dynamic Programming Solver

To efficiently optimize this objective function, we present a dynamic programming method and also use some acceleration strategies. Let $\langle l, \widehat{S}_l, t\rangle$ denote the state for the current selected shot sequence, with cuts for the first l segments ending at (global) time t, where $\widehat{S}_l$ is the partial shot sequence already chosen for current segment r_l. Let $D(l, \widehat{S}_l, t)$ denote the optimal energy for this state. When choosing the next shot s for shot assembly from the current state $\langle l, \widehat{S}_l, t\rangle$, the current state can transition to a state still in the current segment r_l or in the next segment r_{l+1}. In the former case, the next state would be $\langle l, \widehat{S}'_l, t'\rangle$ with a new partial shot sequence $\widehat{S}'_l$ including the chosen shot s and ending at time t'. In the latter case, the next state would be $\langle l+1, \widehat{S}'_{l+1}, t'\rangle$ where $\widehat{S}'_{l+1}$ is the new partial shot sequence in segment r_{l+1} with only one chosen shot s ending at t'. State transition details are provided in Appendix B.

The optimal solution can be obtained by solving:

$$\widehat{S}^*_L, t^* = \arg\max_{\langle \widehat{S}_L, t\rangle} D(L, \widehat{S}_L, t), \tag{3}$$

using dynamic programming and back-tracking the shot sequence with cuts from $\widehat{S}^*_L$ with a global ending time of t^*. The worst case optimization complexity is $O(L\bar{T}^2|\bar{S}|!)$, where L is the number of segments, $|\bar{S}|$ is the average number of candidate shots retrieved by visual-semantic matching for a textual segment, and $\bar{T}$ is the average temporal length of segments. This can become infeasible to compute, and we rely on two strategies to speed up the optimization. The first one is to search for possible solutions only among sampled frames. In our implementation, we sample one frame in every ten frames and only perform cuts at sampled frames. The second one is to set an upper bound on the number of shots actually selected in each segment in the final video. We found that four shots are typically enough for the different stylistic effects. These strategies make the problem tractable in practice. The user can relax these strategies to achieve higher quality outputs at the cost of time. To avoid using repetitive shots in the montage, we iteratively run the optimization and remove repeated shots until shot uniqueness is met.

7 RESULTS AND EVALUATION

We have used *Write-A-Video* to generate 20 video montage results, covering various topics such as tour guiding, an introduction to animals, gardening and driving tutorials, and personal stories. The texts were collected from public websites, online video voice-overs, lyrics of nursery rhymes and personal creations. In total, 7 video repositories (720p at 30 fps) were used (see Figure 7), either downloaded from YouTube (London Tour, Animals in Africa, Gardening Tips, Autumn Driving, I Have a Pet, Play in the Playground) using user specified keywords, or collected from an album (Brothers). The supplementary material provides interaction demos and full details, including texts, video montage, keywords, editing operations and time. We encourage readers to watch the videos.

For our YouTube video repositories we collected about 250 videos for each user-provided keyword, segmented them into shots and labeled the shots automatically with corresponding query keywords in a pre-processing stage. Object detection was employed to label shots for the Animals in Africa and I Have a Pet repositories. For the video repository Brothers, portrait photos of the main characters and their names were used to automatically index shots by character name using face recognition. Two additional keywords were interactively assigned to shots in this repository. Next, the cinematography-aware energy terms were pre-computed to support interactive video montage editing. The pre-processing was executed for each video repository once, allowing it to be used to create multiple video montages.

We tested our tool on a 2.6 GHz PC with 16GB memory. The pre-processing time varies with example, depending on the input video length and the number of segmented shots. As reported in Table 2, shot segmentation and feature embedding speeds were 111 fps and 470 fps respectively on a single core; this could be significantly accelerated with parallelization. Shot retrieval took less than 0.10 seconds. Considering efficiency, diversity and accuracy, we set $K = 10$ as the number of candidate shots in all of our video montage results. Video assembly time varied from 0.06 seconds to 8.56 seconds, typically determined by text length. The rendering procedure to generate output video performs at 255 fps.

7.1 Video Montage Results

We briefly introduce video montage results generated for various applications. Full results are provided in supplementary files.

Table 2. Pre-processing statistics for video repositories. For each repository, the number of videos, their total duration, shot segmentation time (Segmentation), feature embedding time (Embedding), shot energy computation time (Energy) and manual labeling time (Labeling) are given. We also show statistics of the pre-processed results including the number of shots (Shots), the average shot duration (Avg. Duration), and number of keywords (Keyword).

Repository	Raw Input		Pre-Processing Time				Pre-Processed Shots		
	Video	Total Duration	Segmentation	Embedding	Energy	Labeling	Shots	Avg. Duration	Keywords
London Tour	469	31:32:39s	7:33:28s	04:01s	1:03:22s	0s	6817	12.98s	8
Animals in Africa	451	22:00:02s	5:42:05s	03:06s	0:30:05s	0s	4210	12.15s	6
Gardening Tips	449	35:34:19s	8:31:59s	03:44s	0:28:16s	0s	2716	17.43s	6
Autumn Driving	403	23:40:25s	6:57:03s	03:26s	0:43:08s	0s	3945	13.07s	6
Brothers	4	03:22:47s	2:06:19s	06:08s	1:40:33s	32:33s	1206	9.04s	4
I Have a Pet	126	10:05:46s	3:39:31s	03:13s	45:43s	0s	2348	10.85s	6
Play in the Playground	100	7:29:53s	2:33:40s	03:03s	42:36s	0s	1961	11.44s	4

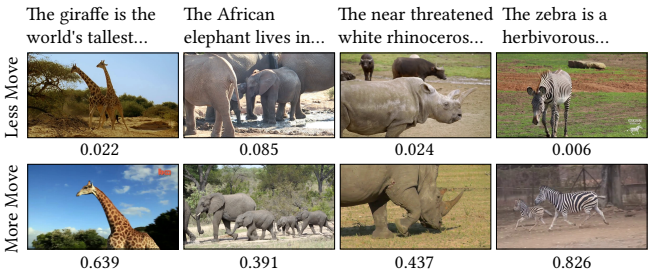

Fig. 8. Two video montage results with different cinematographic styles, using the repository Animals in Africa. Each row from left to right shows four shot thumbnails from corresponding text segments. Top row: video montage with less camera movement; bottom row: video montage with more camera movement. Movement term values are listed below the thumbnails.

Video Montage for Online Text. Several London Tour videos were created based on a text from a website introducing scenic locations in *London*: *Elizabeth Tower*, *Westminster Abbey*, *Hyde Park*, *Buckingham Palace*, *Tower Bridge* and the *National Gallery*. The texts were iteratively edited with various visual styles. Animals in Africa videos were created to introduce several *animals* that live in Africa, including *giraffe*, *elephant*, *cheetah*, *zebra*, *rhinoceros* and *cow*. Figure 8 shows two representative results with different stylistic movement controls. The Autumn Driving results provide tutorials giving advice on driving in *autumn*, including key subjects *check tires*, *fallen leaves*, *dark road*, *winter tires* and *icy road*. Gardening Tips videos provide tips on *gardening*, including *container gardening*, *gardening soil*, *gardening location*, *watering plants* and *trimming leaves*. Five local visual-semantic matches were applied to better match the action timing.

Video Montage for Personal Stories. We created two videos from an original story named Brothers; the video repository comes from an album. It introduces a short story about a chemistry teacher John, with different visual styles.

Video Montage for Human Voice-overs. Instead of using the text-to-speech synthesis technique to generate voice-overs for texts, we used the inverse speech recognition technique [Chiu et al. 2018] to recognize words and get the timing from voice-overs. We tested this on three different human voice-overs extracted from existing YouTube videos, and generated new videos for them using London Tour, Animals in Africa and Gardening Tips repositories respectively.

Table 3. Precision of visual-semantic matching methods.

Method	Top-1	Top-5	Top-10	Top-20
Random	7.34%	10.01%	10.71%	10.74%
MEE	13.59%	14.99%	16.12%	16.37%
VSE++	57.43%	47.28%	41.58%	38.30%
Random&Keyword Matching	48.47%	56.18%	54.99%	54.83%
MEE&Keyword Matching	51.09%	58.19%	57.65%	58.53%
VSE++&Keyword Matching (Ours)	**76.85%**	**71.28%**	**69.95%**	**64.43%**

Video Montage for Nursery Rhymes. We generated music videos for nursery rhymes I Have a Pet and Play in the Playground using their lyrics and timing.

A Minimum Interaction Mode. We developed a simple interface, that only allows users to write and edit text. The rest of the processing is fully automatic and uses default cinematographic guidelines. After updating the text, a video montage result is immediately generated. A demo video is provided in the supplementary material.

7.2 Evaluation of Visual-Semantic Matching

We evaluated our visual-semantic matching method by measuring the precision of top-ranked shots for query texts. The evaluation was performed on video repositories London Tour, Animals in Africa, Autumn Driving and Gardening Tips with corresponding segmented texts. The ground truth visual-semantic matched shots for text segments were labeled by human workers in advance. Note that in our video montage problem, because top-ranked shots are returned as candidates, the precision value for the returned candidates is more important. We compared our method against ones without either keyword matching, without visual-semantic embedding, or both. We also compared the employed visual-semantic embedding algorithm VSE++ to an alternative text-video embedding algorithm, MEE [Miech et al. 2018]. Overall statistics shown in Table 3 demonstrate that our visual-semantic matching performance is better than that of alternatives and each cascaded stage plays an important role.

7.3 Evaluation of Cinematography-Aware Shot Assembly

Cinematography-aware shot assembly is a key component of our method. We evaluated results using all terms against results without certain individual terms. In our formulation, some terms are used by default, such as visual-semantic matching and camera stability, while some can be employed by users via cinematography idioms, such as adjusting shot duration. To be consistent across all evaluations, to

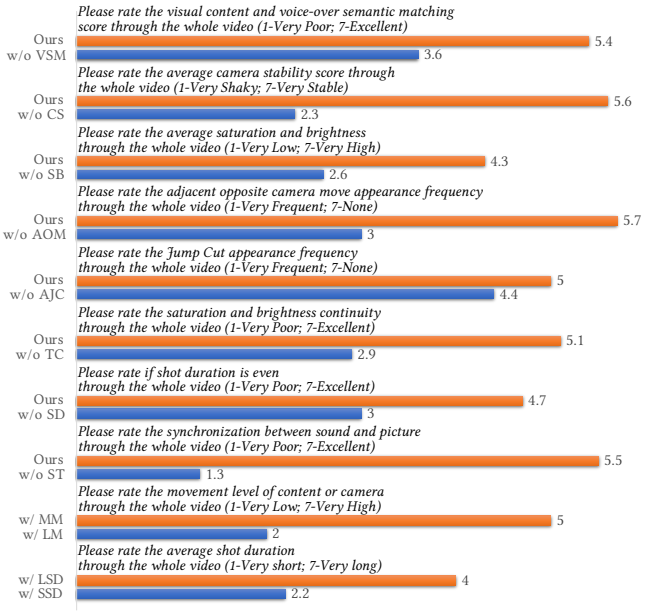

Fig. 9. User study questions and results for individual energy terms, see text for details.

generate a video montage, the same London Tour repository was used with a specific reference text composed of 5 text segments. In addition to our standard result (Ours), we generated various results without certain terms: visual-semantic matching (w/o VSM), camera stability (w/o CS), saturation and brightness (w/o SB), avoiding opposite motions (w/o AOM), avoiding jump cuts (w/o AJC), keeping tonal continuity (w/o TC), shot duration constraints (w/o SD) and segment timing constraints (w/o ST). We also generated videos with a cinematography idiom assigned globally to the text, including ones with more movement (w/ MM), less movement (w/ LM), long shot duration (w/ LSD) and short shot duration (w/ SSD).

We performed pairwise evaluation tests to rate the video results. Paid tasks were created via a crowd-sourcing platform; each worker was asked to watch and rate a pair of videos. The workers were asked to rate each video using a 7-point Likert scale according to the term being evaluated. In total, 10 scores were collected for each video. During the tests, videos were displayed in full-screen mode; within each pair, the video to be played first was randomly chosen. Workers could watch each video multiple times before making a decision. Figure 9 shows the evaluation results. They demonstrate that each individual term plays a positive role in its corresponding visual style.

7.4 Comparison to QuickCut

We compared our hybrid optimization to QuickCut's optimization procedure. QuickCut supports two main modes: *alternatives* mode and *ordered-grouping* mode. The *alternatives* mode only assembles one shot for a segmented script; this can limit diversity in the result. The *ordered-grouping* mode optimizes cut positions for a manually determined shot sequence. In contrast, our optimization method automatically decides the shot sequence order and cut positions for shot assembly. Furthermore, our hybrid optimization method runs faster than QuickCut's plain dynamic programming. QuickCut employs sharpness, stability and jump cut terms as cinematographic guidelines, while our optimization further considers saturation and brightness, opposite movements and shot duration. We used the same reference text from the London Tour example to generate the following results for evaluation: *Our Optimization+Our Energy, Our Optimization+QuickCut Energy, QuickCut Alternatives, QuickCut Ordered-grouping*. Because *QuickCut Ordered-grouping* needs a predetermined order of shots as input, we simply feed their method the sequence order determined by *Our Optimization+Our Energy*. We invited 10 subjects to rate all videos using a 7-point Likert scale, considering cinematographic aesthetics, content diversity and visual-semantic matching of narration and visual content (1 = very poor, 7 = excellent). During testing, videos were displayed in full-screen mode with videos played in random order for each subject. Subjects rated each video after watching it. Table 4 shows the run time and the evaluation result. Our method was more highly preferred by users than all other optimization methods, and ran faster than *QuickCut Ordered-grouping*.

Table 4. Comparison to QuickCut optimization. **Our Opt.&Nrg.:** our hybrid optimization and our energy terms; **Our Opt.+QC Nrg.:** our hybrid optimization and QuickCut energy terms; **QC Alter.:** QuickCut Alternatives mode; **QC Ordered.:** QuickCut Ordered-grouping mode.

Metric	Our Opt.&Nrg.	Our Opt.+QC Nrg.	QC Alter.	QC Ordered.
Time	5.82s	8.05s	0.10s	25.4s
Score	5.0	3.5	2.0	2.7

7.5 Comparison to Professional Manual Editing.

We compared video editing time using the same reference text, for a video editing novice using *Write-A-Video*, and a professional video editor using commercial frame-based editing software. We further conducted a user study to evaluate the quality of the videos produced. The professional editor had 8 years of video editing experience. She was asked to edit a video for a text with corresponding voice-over using the given London Tour video repository, and to ensure visual-semantic consistency and maximum visual quality. We also asked her to use only simple cuts between shots, without fades, speed ups or other special effects. She was free to determine the editing style.

In the editing task, we asked the professional editor to generate videos both from the original video repository (Task I) and from the segmented shots indexed by keywords (Task II). The repository in Task I consisted of all downloaded raw videos. Editing video from this repository corresponded to all processing stages of our tool. The repository in Task II was segmented into shots and indexed by the keyword set. The professional editor was free to use such keywords for semantic assistance or not, as desired.

The editor chose Adobe Premiere as the frame-based editing tool. For a fair comparison, we only counted the human active time during editing. As shown in Table 5, Task I took the editor more than 7 hours while Task II took her more than 4.5 hours. For both editing tasks, the human active time using our tool was significantly lower than these times. After finishing the tasks, we invited the editor to watch and comment on the results created by our tool. Generally, she rated

Table 5. Comparison between the professional editor and a novice using *Write-A-Video*.

Method	Time		Score	
	Task I	Task II	Task I	Task II
Professional Editor	7:20:20s	4:40:12s	5.3	5.8
Novice using *Write-A-Video*	0:13:16s		5.0	

our videos as *reasonable and plausible*. She also commented that her personal preference would be to use more precisely matched visual content to the voice-over, even if this meant using more frequent cuts between shots.

To further evaluate the generated videos, 10 non-expert subjects were invited to watch the editor's results and our results. They were asked to rate for each video with a 7-point Likert scale (1 = poor, 7 = excellent), taking into account the general visual quality and visual consistency with the narration. During the tests, videos were displayed in full-screen mode using a random order within each pair. Subjects could watch each video multiple times before making a decision. The user study results are reported in Table 5. It can be seen that although the visual quality of our results are slightly inferior to the professional editor's, our results rated 5.0 points on average. More importantly, *Write-A-Video* drastically reduced the editing time compared to frame-based manipulation.

7.6 Inexperienced User Study

Because the primary goal of *Write-A-Video* is to help inexperienced users to create videos, its effectiveness is key. We invited 5 users aged 18 to 30 without any video editing skills to try the tool. Their goal was to create a video based on a reference text and a video repository. When editing videos, the user could iteratively edit the reference text by adding, removing, re-ordering the text or changing it, and setting text segment attributes, until the video montage met his or her preference. We started the study by showing the participants a video tutorial and explaining the editing operations provided. We gave each participant the pre-processed London Tour repository and corresponding reference text, and allowed him/her to explore the tool for 15 minutes. After a 5-minute rest, they started the formal task with the repository and reference text of Animals in Africa.

Table 6 reports the editing statistics. Full editing operations are provided in the supplementary material. On analysis of the editing process, we found that in addition to editing the text, most frequently used operations were cinematographic idioms and local visual matches to explore different visual styles. We interviewed the users after the tests, and asked them to rate the tool and the generated video, using the following questions:

- Q1: How would you rate the usability of our tool (1-7)?
- Q2: How would you rate the result of the generated video (1-7)?
- Q3: Would you be willing to post the output video to Facebook, Twitter or Instagram?

For Q1, we got an average response of 6.0; for Q2 the average response was 6.6 (see Table 6). All participants were willing to share their video with friends via a social network. They also confirmed that without this tool, it would have been impossible for them to produce such a nice video in such a short time.

Table 6. User experience statistics. **Seg.** is the number of segments; **Length** is the length of output video; **Editing Time** stands for the video editing time; **Sys. Rating** stands for user rating on our tool; **Res. Rating** stands for the visual quality rating of generated video.

	Seg.	**Length**	**Editing Time**	**Sys. Rating**	**Res. Rating**
User 1	20	1:50s	35:47s	6.0	6.0
User 2	18	1:45s	27:01s	6.0	7.0
User 3	15	1:25s	8:19s	5.0	6.0
User 4	14	1:09s	9:48s	7.0	7.0
User 5	14	1:19s	10:11s	6.0	7.0

8 DISCUSSION AND CONCLUSION

Non-professional users find it difficult to learn and use professional frame-based video editing software, and lack cinematography knowledge for film editing. Therefore, we have presented *Write-A-Video*, a method to create video montages to illustrate a themed text using mostly text editing. Given an input repository of relevant shots, *Write-A-Video* makes it possible for any user to create visually pleasing videos by simply writing and editing the text and using high level cinematography idioms. The tool provides fast feedback, as the output video is rendered within a few seconds after the user finishes an iteration of editing. With this tool and the novel text-based video assembly concept, a user can create a video montage in minutes. As video has become a main storytelling form in modern media, we believe the proposed approach will impact computational video editing field. While our method enables users to generate video montages easily, it has limitations which suggest future research directions.

Visual-Semantic Matching. Our method focuses on themed text and relies on visual-semantic matching. If the input text is too specific, involving specific people or subjects or actions, then it is possible that good video matches may not be found. If the text is too abstract or combines several themes, it could also be difficult to build a good text-to-video embedding space. In future, incorporating more advanced visual-semantic matching algorithms may lead to better performance and support a wider scope for texts.

Artistic Expression and Fine Control. On one hand, our tool can help inexperienced users create plausible videos easily. On the other hand, as the professional editor pointed out, sometimes cinematography guidelines are intentionally broken to express artistic style. For example, intentional jump cuts with small shot lengths are sometimes favored in selfie videos. Although certain special visual effects can be obtained by use of cinematographic idioms and parameters in our tool, more fine control could be developed in future. In addition, the outputs of our video montage tool could be exported as an Edit Decision List (EDL) file for further professional frame-based adjustment using commercial software.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their valuable comments and Professor Ralph Martin for helpful discussions. Miao Wang was supported by the NSFC (Project Number: 61832016, 61902012). Shi-Min Hu was supported by the NSFC (Project Number: 61521002). Ariel Shamir was supported by the ISF as part of the ISF-NSFC Joint Program (2216/15).

REFERENCES

Ido Arev, Hyun Soo Park, Yaser Sheikh, Jessica Hodgins, and Ariel Shamir. 2014. Automatic editing of footage from multiple social cameras. *ACM Trans. Graph.* 33, 4 (2014), 81.

Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. 2006. *SURF: Speeded Up Robust Features.* 404–417.

Rachele Bellini, Yanir Kleiman, and Daniel Cohen-Or. 2018. Dance to the beat: Synchronizing motion to audio. *Computational Visual Media* 4, 3 (2018), 197–208.

Floraine Berthouzoz, Wilmot Li, and Maneesh Agrawala. 2012. Tools for Placing Cuts and Transitions in Interview Video. *ACM Trans. Graph.* 31, 4 (2012), 67:1–67:8.

Fabian Caba Heilbron, Victor Escorcia, Bernard Ghanem, and Juan Carlos Niebles. 2015. ActivityNet: A Large-Scale Video Benchmark for Human Activity Understanding. In *IEEE CVPR*.

Minsuk Chang, Anh Truong, Oliver Wang, Maneesh Agrawala, and Juho Kim. 2019. How to Design Voice Based Navigation for How-To Videos. In *ACM CHI*. Article 701, 701:1–701:11 pages.

Pei-Yu Chi, Joyce Liu, Jason Linder, Mira Dontcheva, Wilmot Li, and Bjoern Hartmann. 2013. Democut: generating concise instructional videos for physical demonstrations. In *ACM UIST*. 141–150.

Chung-Cheng Chiu, Tara N Sainath, Yonghui Wu, Rohit Prabhavalkar, Patrick Nguyen, Zhifeng Chen, Anjuli Kannan, Ron J Weiss, Kanishka Rao, Ekaterina Gonina, et al. 2018. State-of-the-art speech recognition with sequence-to-sequence models. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 4774–4778.

W. S. Chu, Yale Song, and A. Jaimes. 2015. Video co-summarization: Video summarization by visual co-occurrence. In *IEEE CVPR*. 3584–3592.

Abe Davis and Maneesh Agrawala. 2018. Visual Rhythm and Beat. *ACM Trans. Graph.* 37, 4, Article 122 (2018).

Edward Dmytryk. 1984. *On Film Editing: An Introduction to the Art of Film Construction.* Focal Press.

David K Elson and Mark O Riedl. 2007. A Lightweight Intelligent Virtual Cinematography System for Machinima Production. (2007).

Fartash Faghri, David J Fleet, Jamie Ryan Kiros, and Sanja Fidler. 2018. VSE++: Improving Visual-Semantic Embeddings with Hard Negatives. In *BMVC*.

Martin A. Fischler and Robert C. Bolles. 1981. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Commun. ACM* 24, 6 (1981), 381–395.

Quentin Galvane, Rémi Ronfard, Christophe Lino, and Marc Christie. 2015. Continuity Editing for 3D Animation. In *AAAI*. 753–762.

Andreas Girgensohn, John Boreczky, Patrick Chiu, John Doherty, Jonathan Foote, Gene Golovchinsky, Shingo Uchihashi, and Lynn Wilcox. 2000. A Semi-automatic Approach to Home Video Editing. In *ACM UIST*. 81–89.

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. 2017. Mask r-cnn. In *IEEE ICCV*. 2980–2988.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *IEEE CVPR*. 770–778.

Rachel Heck, Michael Wallick, and Michael Gleicher. 2007. Virtual videography. *ACM Trans. Multimedia Comput. Commun. Appl.* 3, 1 (2007), 4.

Weiming Hu, Nianhua Xie, Li Li, Xianglin Zeng, and Stephen Maybank. 2011. A survey on visual content-based video indexing and retrieval. *IEEE Trans. Syst., Man, Cybern. C* 41, 6 (2011), 797–819.

Haozhi Huang, Xiaonan Fang, Yufei Ye, Songhai Zhang, and Paul L. Rosin. 2017. Practical automatic background substitution for live video. *Computational Visual Media* 3, 3 (2017), 273–284.

Eakta Jain, Yaser Sheikh, Ariel Shamir, and Jessica Hodgins. 2015. Gaze-Driven Video Re-Editing. *ACM Trans. Graph.* 34, 2, Article 21 (2015), 12 pages.

Neel Joshi, Wolf Kienzle, Mike Toelle, Matt Uyttendaele, and Michael F. Cohen. 2015. Real-time Hyperlapse Creation via Optimal Frame Selection. *ACM Trans. Graph.* 34, 4 (2015), 63:1–63:9.

Andrej Karpathy and Li Fei-Fei. 2015. Deep visual-semantic alignments for generating image descriptions. In *IEEE CVPR*. 3128–3137.

G. Kim, L. Sigal, and E. P. Xing. 2014. Joint Summarization of Large-Scale Collections of Web Images and Videos for Storyline Reconstruction. In *IEEE CVPR*. 4225–4232.

Mackenzie Leake, Abe Davis, Anh Truong, and Maneesh Agrawala. 2017. Computational Video Editing for Dialogue-driven Scenes. *ACM Trans. Graph.* 36, 4 (2017), 130:1–130:14.

Zicheng Liao, Yizhou Yu, Bingchen Gong, and Lechao Cheng. 2015. Audeosynth: Music-driven Video Montage. *ACM Trans. Graph.* 34, 4, Article 68 (2015).

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *IEEE ECCV*. 740–755.

Shugao Ma, Leonid Sigal, and Stan Sclaroff. 2016. Learning Activity Progression in LSTMs for Activity Detection and Early Detection. In *IEEE CVPR*.

Erik Machnicki and Lawrence A Rowe. 2001. Virtual director: Automating a webcast. In *Multimedia Computing and Networking 2002*, Vol. 4673. 208–226.

Antoine Miech, Ivan Laptev, and Josef Sivic. 2018. Learning a text-video embedding from incomplete and heterogeneous data. *arXiv preprint arXiv:1804.02516* (2018).

M. Ramesh Naphade, I. V. Kozintsev, and T. S. Huang. 2002. Factor graph framework for semantic video indexing. *IEEE Trans. Circuits Syst. Video Technol.* 12, 1 (2002), 40–52.

Yuzhen Niu and Feng Liu. 2012. What makes a professional video? a computational aesthetics approach. *IEEE Trans. Circuits Syst. Video Technol.* 22, 7 (2012), 1037–1049.

Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. 2016. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499* (2016).

Amy Pavel, Dan B Goldman, Björn Hartmann, and Maneesh Agrawala. 2015. Sceneskim: Searching and browsing movies using synchronized captions, scripts and plot summaries. In *ACM UIST*. 181–190.

Amy Pavel, Colorado Reed, Björn Hartmann, and Maneesh Agrawala. 2014. Video Digests: A Browsable, Skimmable Format for Informational Lecture Videos. In *ACM UIST (UIST '14)*. 573–582.

Ana Serrano, Vincent Sitzmann, Jaime Ruiz-Borau, Gordon Wetzstein, Diego Gutierrez, and Belen Masia. 2017. Movie Editing and Cognitive Event Segmentation in Virtual Reality Video. *ACM Trans. Graph.* 36, 4, Article 47 (2017), 12 pages.

Hijung Valentina Shin, Floraine Berthouzoz, Wilmot Li, and Frédo Durand. 2015. Visual transcripts: lecture notes from blackboard-style lecture videos. *ACM Trans. Graph.* 34, 6 (2015), 240.

S. W. Smoliar and HongJiang Zhang. 1994. Content based video indexing and retrieval. *IEEE MultiMedia* 1, 2 (1994), 62–72.

James Tompkin, Kwang In Kim, Jan Kautz, and Christian Theobalt. 2012. Videoscapes: Exploring Sparse, Unstructured Video Collections. *ACM Trans. Graph.* 31, 4, Article 68 (2012), 12 pages.

Anh Truong, Floraine Berthouzoz, Wilmot Li, and Maneesh Agrawala. 2016. QuickCut: An Interactive Tool for Editing Narrated Video. In *ACM UIST*. 497–507.

M. Wang, J. Liang, S. Zhang, S. Lu, A. Shamir, and S. Hu. 2018. Hyper-Lapse From Multiple Spatially-Overlapping Videos. *IEEE Transactions on Image Processing* 27, 4 (2018), 1735–1747.

M. Wang, G. Yang, J. Lin, S. Zhang, A. Shamir, S. Lu, and S. Hu. 2019. Deep Online Video Stabilization With Multi-Grid Warping Transformation Learning. *IEEE Transactions on Image Processing* 28, 5 (2019), 2283–2292.

Ke Xie, Hao Yang, Shengqiu Huang, Dani Lischinski, Marc Christie, Kai Xu, Minglun Gong, Daniel Cohen-Or, and Hui Huang. 2018. Creating and Chaining Camera Moves for Qadrotor Videography. *ACM Trans. Graph.* 37, 4 (2018), 88:1–88:13.

Hao Yang, Ke Xie, Shengqiu Huang, and Hui Huang. 2018. Uncut Aerial Video via a Single Sketch. *Computer Graphics Forum* 37, 7 (2018), 191–199.

F. Zhang, J. Wang, H. Zhao, R. R. Martin, and S. Hu. 2015. Simultaneous Camera Path Optimization and Distraction Removal for Improving Amateur Video. *IEEE Transactions on Image Processing* 24, 12 (2015), 5982–5994.

F. Zhang, X. Wu, R. Li, J. Wang, Z. Zheng, and S. Hu. 2018. Detecting and Removing Visual Distractors for Video Aesthetic Enhancement. *IEEE Transactions on Multimedia* 20, 8 (2018), 1987–1999.

Yu Zhang, Xiaowu Chen, Liang Lin, Changqun Xia, and Dongqing Zou. 2016. High-level representation sketch for video event retrieval. *Science China Information Sciences* 59, 7 (2016), 072103.

A APPENDIX: ENERGY TERM DETAILS

In this appendix we define all energy terms used in our optimization method for shot assembly. Values of each term are normalized to the range [0, 1] before combination.

A.1 Shot Energy Terms

Visual-Semantic Matching. $F_{\text{vsm}}(s)$ measures how semantically consistent the visual content of a shot s is with the query text segment r. We define $F_{\text{vsm}}(s)$ for a shot s as the inverse of its rank index $ind_r(s)$ when retrieved with corresponding query segment r: $F_{\text{semantic}}(s) = 1/ind_r(s)$.

Camera Stability. . $F_{\text{stab}}(s)$ evaluates average acceleration of video content as a proxy for camera stability: the smaller the acceleration, the greater the camera stability. Let $H(f, f')$ be the homography transformation matrix linking consecutive frames f and f' in a shot, computed using SURF features [Bay et al. 2006], and RANSAC regression [Fischler and Bolles 1981]. We use the estimated homography matrices $H(f, f')$ and $H(f', f'')$ from three consecutive

sampled frames f, f', f'' in shot s, to compute the local camera shake measure $lcs(f)$ for frame f:

$$lcs(f) = \frac{1}{4}\sum_{i=1}^{4} ||H(f', f'')p_{f'}(i) - p_{f'}(i) - (H(f, f')p_f(i) - p_f(i))||_2,$$

where $p_f(i), i = 1, \ldots, 4$ are the 2D positions of the four frame corners measured in pixels. The camera stability term $F_{\text{stab}}(s)$ is computed as the negative average of local shake values over time:

$$F_{\text{stab}}(s) = -\frac{1}{|s|}\sum_{f\in s} lcs(f), \tag{4}$$

where $|s|$ is the number of sampled frames in shot s, and the sampling step is 10 frames.

Saturation and Brightness. $F_{\text{tone}}(s)$ measures the saturation and brightness of frames over the whole shot. We first represent frame f in HSL color space, then compute the ratio $\xi(f)$ of pixels whose S and L channel intensities are both above 70% of the maximum value (i.e. 255). $F_{\text{tone}}(s)$ is then defined as:

$$F_{\text{tone}}(s) = \frac{1}{|s|}\sum_{f\in s} \xi(f). \tag{5}$$

A.2 Cut Energy Terms

Avoid Opposite Camera Movements. We determine whether shots move in opposite directions before and after the cut by comparing the estimated 2D movements of frame corners before and after:

$$\begin{aligned} F_{\text{OM}}(s, s') &= \begin{cases} 1 & \text{if } Q(e(s), b(s')) < \eta \\ 0 & \text{otherwise,} \end{cases} \\ Q(f_1, f_2) &= \sum_{i=1}^{4} \frac{v(f_1, i)\cdot v(f_2, i)}{|v(f_1, i)||v(f_2, i)|}, \\ v(f, i) &= p(i) - H(f)p(i), \end{aligned} \tag{6}$$

where s and s' are consecutive shots, $e(s)$ is the last frame of s and $b(s')$ is the start frame of s', $Q(f_1, f_2)$ is the cosine distance of the 2D frame corner movement vectors of f_1 and f_2, $\eta = -0.01$ is a threshold determining whether the movements are opposite, $v(f, i)$ is the function estimating the 2D movement of the i-th frame corner $p(i)$ of f, and $H(f)$ is the estimated homography transformation linking shot frame f and the next sampled frame.

Avoid Jump Cuts. A cut between consecutive shots s' and s is determined as a jump cut if the last frame $e(s')$ in s' and the first frame $b(s)$ in s can be registered and aligned with a small 2D offset:

$$F_{\text{JC}}(s, s') = \begin{cases} 1 & \text{if } \frac{1}{4}\sum_{i=1}^{4} ||H(e(s), b(s'))p_i^{e(s)} - p_i^{e(s)}||_2 < \epsilon \\ 0 & \text{otherwise,} \end{cases}$$

where $H(e(s), b(s'))$ is the homography matrix between $e(s)$ and $b(s')$, $p_i^{e(s)}, i = 1, \ldots, 4$ are the positions of the four corners of the frame $e(s)$, ϵ is a threshold determining whether the offset magnitude suffices for a jump cut. In our implementation, we set $\epsilon = 150$ for 720p video.

Tonal Continuity. We compute tonal continuity between neighboring shots $\langle s, s'\rangle$ based on saturation histogram difference and brightness histogram difference:

$$F_{\text{TC}}(s, s') = \frac{1}{2}(\Gamma(\Psi_S(s), \Psi_S(s')) + \Gamma(\Psi_L(s), \Psi_L(s'))), \tag{7}$$

where $\Psi_S(s)$ is the S-channel histogram of frame s, and $\Psi_L(s)$ is the L-channel histogram of frame s, both quantified to 256 bins. $\Gamma(\cdot, \cdot)$ is the chi-square distance of two color histograms.

A.3 Segment Energy Terms

The cinematography idioms energy term $F_{\text{idiom}}(\widehat{S}_l, A_l)$ is the weighted sum of two terms: the movement term $F_{\text{move}}(\widehat{S}_l, A_l)$ and the shot duration term $F_{\text{dur}}(\widehat{S}_l, A_l)$:

$$F_{\text{idiom}}(\widehat{S}_l, A_l) = F_{\text{move}}(\widehat{S}_l, A_l) - 0.03F_{\text{dur}}(\widehat{S}_l, A_l). \tag{8}$$

Movement Preference. $F_{\text{move}}(\widehat{S}_l, A_l)$ encourages motions of shots $\widehat{S}_l$ to follow the user preference A_l^{move}, where A_l^{move} is either 1.0 for dynamic shots or -1.0 for static shots. We first compute the local movement $lm(f)$ of each frame in a shot s: $lm(f)$ is determined using the magnitude of the 2D translations of frame corners needed to spatially align the current frame f to the next frame f':

$$lm(f) = \frac{1}{4}\sum_{i=1}^{4} ||H(f, f')p_f(i) - p_f(i)||_2, \tag{9}$$

where $p_f(i), i = 1, 2, 3, 4$ are the positions of the four corners of the frame f. $H(f, f')$ is the homography matrix linking f to f'. The movement term $F_{\text{move}}(\widehat{S}_l, A_l)$ is defined as:

$$F_{\text{move}}(\widehat{S}_l, A_l) = \frac{A_l^{\text{move}}}{\sum_i^{n_l} |\widehat{s}_l^i|}\cdot \sum_k^{n_l}\sum_{f\in\widehat{s}_l^k} lm(f). \tag{10}$$

Shot Duration Preference. $F_{\text{dur}}(\widehat{S}_l, A_l)$ encourages the shots in $\widehat{S}_l$ to follow a user specified shot duration A_l^{dur} for segment r_l:

$$F_{\text{dur}}(\widehat{S}_l, A_l) = \frac{1}{|\widehat{S}_l|}\sum_k^{n_l}(\frac{|s_l^k|}{\gamma} - A_l^{\text{len}})^2, \tag{11}$$

where $|\widehat{S}_l|$ is the number of shots in $\widehat{S}_l$, $|s_l^k|$ is the number of frames in the k-th shot of $\widehat{S}_l$, γ is the video frame rate, set to 30 fps. A_l^{dur} is the reference shot length for segment r_l in seconds, with default value 1.0 for cinematography idiom 'Reduce Shot Duration' and 10.0 for cinematography idiom 'Extend Shot Duration'; the user can further set shot length values via a slide bar. If no such idioms are chosen, the default shot length is set as 3.5 seconds to avoid unnecessary long-short duration changes [Niu and Liu 2012]. The default length can be globally changed during tool initialization according to user preference.

Split Time Editing Preference. $F_{\text{split}}(\widehat{S}_l, A_l)$ allows the user to adjust cutting points for video and audio separately. Let A_l^{off} be a user-specified starting timing offset for the first shot $\widehat{s}_l^1$ for segment r_l, with default value 0, and let $t(\widehat{s}_l^1)$ be the start time of $\widehat{s}_l^1$ in the global time-line. $F_{\text{split}}(\widehat{S}_l, A_l)$ is defined as:

$$F_{\text{split}}(\widehat{S}_l, A_l) = (t(\widehat{s}_l^1) - (t_l + A_l^{\text{off}}))^2, \tag{12}$$

where t_l is the global start time of the voice-over of segment r_l.

Local Visual-Semantic Match. $F_{\text{match}}(\widehat{S}_l, A_l)$ allows local visual match points at which the visual content appears in the movie when a specified word, phrase or character appears in the narration. Assume the times of m_l user specified keywords or character names $\langle w_l^1, w_l^2, \cdots, w_l^{m_l}\rangle$ in segment r_l are $A_l^{\text{match}} = \langle a_l^1, a_l^2, \cdots a_l^{m_l}\rangle$ in the time-line, and σ_l^k is the best shot interval that matches w_l^k in candidate shots S_l; $[\alpha_l^k, \beta_l^k]$ is the global time of σ_l^k in the assembled video. The local visual-semantic match term $F_{\text{match}}(\widehat{S}_l, A_l)$ is given by:

$$F_{\text{match}}(\widehat{S}_l, A_l) = \sum_k^{m_l} \delta(a_l^k, \alpha_l^k, \beta_l^k). \tag{13}$$

$$\delta(a, \alpha, \beta) = \begin{cases} 1 & \text{if } \alpha \le a \le \beta \\ 0 & \text{otherwise.} \end{cases} \tag{14}$$

The weights to combine shot energy, cut energy and segment energy terms are empirically set to: $\alpha_1 = 1.0$, $\alpha_2 = 0.3$, $\alpha_3 = 0.3$, $\beta_1 = -1.0$, $\beta_2 = -1.0$, $\beta_3 = -1.0$, $\gamma_1 = -1.0$, $\gamma_2 = 3.0$, $\gamma_3 = -3.0$.

B APPENDIX: DYNAMIC PROGRAMMING DETAILS

We present the details of the transition equation for the dynamic programming method in Section 6. As explained there, $\langle l, \widehat{S}_l, t\rangle$ denotes the state for the current selected shot sequence with cuts for the first l segments ending at global time t, where $\widehat{S}_l$ is the partial shot sequence already chosen for the current segment r_l. $D(l, \widehat{S}_l, t)$ denotes the optimal energy for this state.

Initially, the energy for the first globally selected shot s with cut length t is defined as:

$$D(1, \widehat{S}_1, t) = E_{\text{shot}}(\widehat{S}_1) + E_{\text{seg}}(\widehat{S}_1, A_1), \tag{15}$$

where $\widehat{S}_1 = \langle s\rangle$ only contains shot s, and A_1 is the attribute set for the first segment.

When we already have $D(l, \widehat{S}_l, t)$ for the first l segments and wish to determine the next shot s for shot assembly, the new shot can be chosen from the same segment r_l with an ending cut point t', using the following transition equation:

$$\begin{aligned} D(l, \widehat{S}'_l, t') =& D(l, \widehat{S}_l, t) + E_{\text{shot}}(\widehat{S}'_l) - E_{\text{shot}}(\widehat{S}_l) \\ &+ E_{\text{cut}}(\widehat{S}'_l) - E_{\text{cut}}(\widehat{S}_l) \\ &+ E_{\text{seg}}(\widehat{S}'_l, A_l) - E_{\text{seg}}(\widehat{S}_l, A_l), \end{aligned} \tag{16}$$

by concatenating $\widehat{S}_l$ and the newly chosen shot s $\widehat{S}'_l$ is the new partial sequence for segment r_l; E_{shot}, E_{cut}, E_{seg} are the *shot energy*, *cut energy* and *segment energy* terms respectively (see Section 6 and Appendix A). A_l is the attribute set for the l-th segment.

The other case is that the next shot s is chosen from the candidate shots of segment r_{l+1} with ending cut point t', using the following transition equation:

$$\begin{aligned} D(l+1, \widehat{S}'_{l+1}, t') =& D(l, \widehat{S}_l, t) + E_{\text{shot}}(\widehat{S}'_{l+1}) \\ &+ E_{\text{cut}}(\langle \widehat{S}_l, s\rangle) - E_{\text{cut}}(\widehat{S}_l) \\ &+ E_{\text{seg}}(\widehat{S}'_{l+1}, A_{l+1}), \end{aligned} \tag{17}$$

where $\widehat{S}'_{l+1}$ is the newly chosen partial sequence for segment r_{l+1} with only one shot s, and $\langle \widehat{S}_l, s\rangle$ denotes the concatenated shot sequence of selected shots for segment r_l and the newly chosen shot s.

With the above transition equations, we can recursively infer optimal states and get the optimal value for state $D(L, \widehat{S}^*_L, t^*)$, where $\widehat{S}^*_L$ is the selected shot sequence for the last segment r_L in the optimal solution and t^* is the optimal ending time. Backtracking from $\widehat{S}^*_L$, we are able to obtain the whole shot sequence $\widehat{\mathcal{S}}$ with cuts.