A Reinforcement Learning-Based Automatic Video Editing Method Using the Knowledge from Vision-Language Model

Anonymous

25 Apr 2023 (modified: 05 May 2023) ACMMM 2023 Conference Blind Submission Readers: Conference, Paper840 Authors Show Revisions

Edit Revision

Abstract: In this era of videos, automatic video editing techniques attract more and more attention from industry and academia since they can reduce workloads and lower the requirements for human editors. Existing automatic editing systems are mainly scene- or event-specific, e.g., soccer game broadcasting, yet the automatic systems for general editing, e.g., movie or vlog editing, which covers various scenes and events, is rarely studied before, and converting the event-driven editing method to a general scene is nontrivial. To this end, this paper proposes a two-stage scheme for general editing. Firstly, unlike previous works extracting scene-specific features, we leverage the pre-trained Vision Language Model (VLM) knowledge to extract the editing-relevant representations as editing context. Moreover, to close the gap between the professional-look videos and the automatic productions generated with simple guidelines, we propose a Reinforcement Learning (RL)-based editing framework to formulate the editing problem and train the virtual editor to make better sequential editing decisions. Finally, we define a new editing task based on a real movie dataset to promote the research in an automatic editing direction. Experimental results demonstrate the effectiveness and benefits of the proposed context representation and the learning ability of our RL-based editing framework.

Primary Subject Area: [Engagement] Multimedia Search and Recommendation

Secondary Subject Area: [Engagement] Summarization, Analytics, and Storytelling, [Experience] Multimedia Applications, [Content] Vision and Language

Open Discussion: Yes

Reviewer Participation: Yes

Revealed to Panwen Hu, 119010343@link.cuhk.edu.cn, wujingqi0605@gmail.com, Rui Huang

17 Apr 2023 (modified: 05 May 2023) ACMMM 2023 Conference Submission

Authors: *Panwen Hu, Nan Xiao, Hong Xiao, Jingqi Wu, Rui Huang*

Add Withdraw

A Reinforcement Learning-Based Automatic Video Editing Method Using the Knowledge from Vision-Language Model

Anonymous Author(s)*

ABSTRACT

In this era of videos, automatic video editing techniques attract more and more attention from industry and academia since they can reduce workloads and lower the requirements for human editors. Existing automatic editing systems are mainly scene- or event-specific, e.g., soccer game broadcasting, yet the automatic systems for general editing, e.g., movie or vlog editing, which covers various scenes and events, is rarely studied before, and converting the event-driven editing method to a general scene is nontrivial. To this end, this paper proposes a two-stage scheme for general editing. Firstly, unlike previous works extracting scene-specific features, we leverage the pre-trained Vision Language Model (VLM) knowledge to extract the editing-relevant representations as editing context. Moreover, to close the gap between the professional-look videos and the automatic productions generated with simple guidelines, we propose a Reinforcement Learning (RL)-based editing framework to formulate the editing problem and train the virtual editor to make better sequential editing decisions. Finally, we define a new editing task based on a real movie dataset to promote the research in an automatic editing direction. Experimental results demonstrate the effectiveness and benefits of the proposed context representation and the learning ability of our RL-based editing framework.

CCS CONCEPTS

• **Information systems** → **Multimedia content creation**; • **Computing methodologies** → *Video summarization*; Scene understanding.

KEYWORDS

video editing, video representation, reinforcement learning

ACM Reference Format:

Anonymous Author(s). 2018. A Reinforcement Learning-Based Automatic Video Editing Method Using the Knowledge from Vision-Language Model. In *Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

With the bandwidth's development, video has become a prime medium for conveying information [53]. In this situation, creating high-quality videos becomes more and more crucial. However, editing video or live multi-camera directing requires a multitude of skills and domain expertise. Even for professional editors/directors, editing and directing video streams are still challenging and labor-intensive processes [36]. Typically, video editing involves the shot selection problem and requires the editors to select and order the shots from a vast footage gallery.

A few works have proposed different automatic editing systems [2, 5, 35, 38, 48] and directing systems [4, 23, 44] for event-driven scenes, where different view selection strategies are developed. For example, Wang et al.[43] proposed a Finite State Machine (FSM)-based online lecture broadcasting system; Arev et al. [2] used a trellis graph to model the shot selection problem for editing social videos; Recently, Pan et al. [31] proposed an event-driven smart directing system for broadcasting soccer matches. The editing process is illustrated in Fig.1(a), where the virtual editor selects a view based on the scene events at each time point. They formulated the view selection as an integer programming problem, maximizing the holistic correlation and camera view diversity. However, these methods are heuristic expert systems, and the qualities of their productions heavily depend on empirical parameter tuning. Although these systems are developed to mimic human editors using various scenario-related hand-crafted features and empirical rules [14, 15], there is still an aesthetic gap between automatic productions and professional-looking videos due to the difficulties in transforming empirical practices into computable formulas. Some systems [42, 45] may allow users to tune the hyper-parameters, but trying different parameter combinations will still cost a lot of time.

A similar problem also exists in the automatic cinematography area, which focuses on automatically placing and moving the cameras to capture well-composed photos or videos [8, 17]. To mitigate the gap between professional productions and automatically generated productions, some recent works [16, 20, 22] start studying the methods that learn a camera controller from expert behaviors directly or human productions. Inspired by these works, this paper will explore a learning-based strategy to learn the editing pattern styles from professional demonstrations, e.g., movies, for general editing purposes. However, the definition of editing tasks is ambiguous. Conventional heuristic editing systems are highly scene-specific, limiting their applications to other scenes. Recently, Argaw et al. [3] propose a video editing dataset and define a task for assembling shots as selecting the next shot (Fig.1(b)), i.e., given a sequence of proceeding shots as context, this task aims to retrieve the next shot from the list of available shots. In practical application scenarios, the shot list is not always available, and the editor might need to edit the raw footage first, thus retrieving the next shot is not feasible. From this angle, we start to study the intelligent virtual editor by defining a new editing task (Fig.1(c)). Specifically, this task aims to predict the multi-dimensional attributes of subsequent shots, telling the apprentice editors what the next shot looks like rather than which shot is the next one. Thereby, the editor is able to retrieve the shot or edit the raw footage based on the predicted attributes. More details will be discussed in Sec.3.

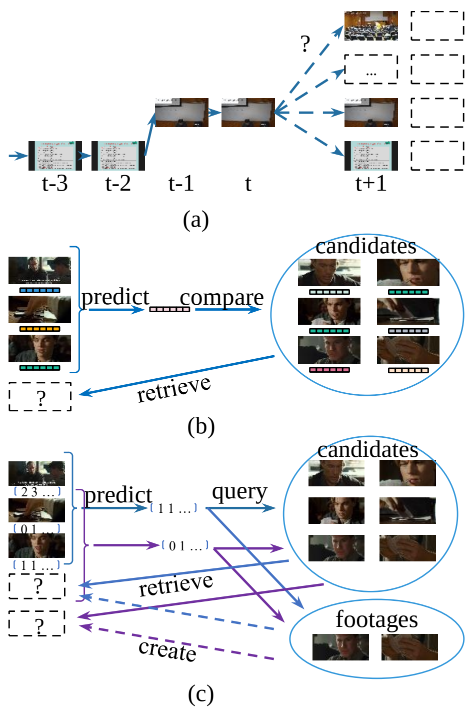

Figure 1: Definitions of different editing tasks. (a) is event-driven editing; (b) edits video by retrieval; (c) is attribute prediction-based editing.

Learning the editing pattern from demonstration videos of the general scene is nontrivial. As the editing process of existing event-driven editing systems [2, 35, 48], the learning-based editing undergoes two stages, i.e., the representation extraction of context shots and the methodology for training virtual editor. Although event-driven editing systems have explored various features, e.g., soccer events, these features are highly scene-specific and only sometimes available when editing general scenes, e.g., movie scenes. On the other hand, the representation for general editing is rarely studied before, so it is still challenging to figure out the methods to distill the context information for editing general scenes. Recently, pre-trained Vision-Language Model (VLM) [32] has successfully assisted various methods to make great progress in different tasks [29, 50] by its great generalization and adaption abilities. Motivated by this, we study how this powerful VLM can assist general editing tasks and propose to leverage its extraordinary zero-shot recognition capability to extract the attribute distributions of context shots as the context representations. Afterward, a transformer-based neural network is developed to encode the temporal relationship of context representations for the prediction task. Relevant experiments are conducted in Sec.4.

The manner of training a virtual editor also plays a crucial role. Previous methods have attempted to train a Long Shot-Term Memory (LSTM) model [9, 47] as a virtual director with manual annotations in a supervised manner. Whereas editing video might require sequential decisions, the virtual director/editor trained in a conventional supervised way cannot produce optimal editing results. To this end, we propose a Reinforcement Learning (RL)-based editing framework to optimize the virtual editor for making better sequential editing decisions. The details are discussed in Sec.3. Furthermore, to investigate the style/pattern learning capability and the generalization of the proposed framework, we apply it to the task of learning personal preferences in an event-driven scene, i.e., the automatic multi-camera lecture broadcasting [26, 34, 43] scene, in Sec.4. Finally, we design a set of metrics to quantitatively evaluate the effectiveness of the proposed representations and the editing framework on the AVE dataset [3]. To the best of our knowledge, this is the first work to explore the knowledge of VLM for general editing purposes. It is also the first time to study the RL technique for automatic editing/directing tasks.

In summary, we have made the following contributions:

- We define a new editing task and attempt to leverage the knowledge of pre-trained VLM to represent the context shots for general editing purposes.
- We proposed a general RL-based editing framework to optimize the decisions for sequential editing.
- We develop a set of metrics and conduct experiments to validate the benefits of the proposed components. Finally, we further show an application practice of our framework in a real-world broadcasting scene to examine its learning capability.

2 RELATED WORK

2.1 Automatic Video Editing

Automatic video editing/directing has attracted much attention recently [33, 36, 39, 53]. As a core part of video editing, various camera view selection algorithms have also been proposed for different applications. For example, Wang et al.[44, 51] define a set of selection rules in their system to select the camera for broadcasting soccer games, mimicking an experienced broadcaster. Some other systems [5, 38] adopted a simple yet feasible greedy selection strategy for editing user-generated videos, in which the cameras are selected to maximize the pre-defined scores. For the scene of online lecture broadcasting, the FSM has been employed to model the view-switching process in previous works [26, 34, 43]. Moreover, Daniyal2011 et al.[12] adopted a Partially Observable Markov Decision Process (POMDP) to model the view selection in editing basketball videos. There are also some studies [21, 31, 46, 48] that formulate the camera selection as an optimization problem, which can be solved with dynamic programming techniques.

Nevertheless, The productions from these heuristic methods cannot always fulfill audiences' preferences or professional styles. These methods are developed upon a set of scene-specific features, e.g., visibility of objects-of-interest [7, 46], the size of the object [46], transition constraints [5, 7, 46], and stability [5, 44], etc. Although they are expected to represent professional editing/directing practices precisely, there is a gap in different preferences and a gap in the transformation from a library of features to professional editing elements. To this end, another line of works studies the data-driven camera selection strategies and trains a regressor [9] to rank the camera's importance. Yet these methods require large numbers of real-valued labels(visual importance). In addition, these data-driven methods, and the above heuristic methods, are limited to particular working scenes. Therefore, this paper explores a more general editing representation and proposes an RL-based editing framework for editing videos of general scenes.

2.2 Learning-Based Video Cinematography

Video cinematography is also a crucial part of the art of video production. Automated cinematography also encounters a similar problem: the productions from rule-based cinematography are far from satisfactory [52]. Therefore, research in cinematography started to explore learning-based methods. In drone cinematography, Gschwindt et al.[16] proposed a system trained under a deep RL framework to choose the shot type that maximizes a reward based on a handcrafted aesthetic metric. This work is further improved by Bonatti et al.[6] to solve the occlusion avoidance problem and optimize the trajectory. Another line of works [19, 20] for drone cinematography train a prediction network to estimate the next location and shot angle of the drone, given the past locations and the pose of the character. The goal of these works is to make the style of autonomous videos approach that of trained videos. Besides drone cinematography, some other works also have studied autonomous cinematography for filmmaking [22, 52] or photograph [1] using deep RL. The reward functions promoting the agent training are derived from the aesthetics measurements on frame content or a trained model with aesthetic analysis dataset [30]. However, it is a non-trivial task to apply the learning algorithms of autonomous cinematography to autonomous editing. The cinematography style or aesthetics reward can be measured on the frame content, but measuring the editing style of the camera sequence is super challenging.

3 THE PROPOSED METHOD

3.1 Problem Definition

Although the concept of automatic editing has been proposed for a long time, the definition of the editing task still remains ambiguous. Recently, Argaw et al. [3] publish a large-scale movie dataset AVE that covers a wide variety of scenes and defines a total of 8 shot attributes, including shot size, shot angle, shot type, shot motion, shot location, shot subject, number of people, and sound source. An editing task is defined as the next shot selection in their work, but the candidate list they assume is not always available in practice. In more general cases, the apprentice editor might prefer to obtain some guidance to create a suitable subsequent shot. To this end, we formulate a new editing task as predicting the shot attributes of subsequent shots. Concretely, given the historical context shots $\{S_0, S_1, \ldots, S_n\}$, our task aims to predict the attributes $\{A_{n+1}, \ldots, A_{n+M}\}$ of subsequent shots $\{S_{n+1}, \ldots, S_{n+M}\}$. The attributes $A_i = (a_1, \ldots, a_8)$ of each shot is an 8-dimension vector, and each element represents a class of the corresponding attribute. As a result, with the predicted attributes A_i, users can retrieve a suitable shot to assemble from the available candidate list or create a suitable shot from raw footage.

To solve the general editing problem, this paper proposes a two-stage scheme that includes representation extraction and virtual editor training stages, as shown in Fig.2. In the first stage, we transfer the cross-modal knowledge of VLM to extract the editing-relevant representation without using any other manual label. In the second stage, our editing framework trains a virtual editor/actor based on the extracted features to make sequential decisions.

3.2 Context Representation

The features used to present the historical context are super crucial in video editing. The virtual editor needs to fully understand the high-level semantics of context before making a decision like a human editor, and a proper context representation helps the agent understand the context better. As previous high-level semantics e.g., the positions and numbers of players [46], the event [43], for event-driven scenes might be unavailable in general scenes, some works [3, 9] use pre-trained action recognition model, e.g., R3D [41], to extract the context features. However, these deep features are not interpretable and do not carry the semantics needed for editing, since the feature model focuses more on the information for discriminating action categories rather than editing.

The context representation should be general and editing-relevant, but no previous work has studied this field before. Fortunately, we notice that pre-trained VLM, e.g., CLIP [32], XCLIP [28], has shown its strong generalization abilities in different tasks recently, and its comprehensive knowledge is leveraged to improve traditional close-set methods to an open task domain, e.g., open-vocabulary detection [29] and segmentation [50]. Inspired by this, we attempt to transfer the general knowledge of VLM to the editing domain. In this paper, we choose the XCLIP model, which is trained with masses of paired video-text data covering a variety of domains, as the knowledge source, and perform zero-shot recognition ability to extract the attribute distributions of context shots as representation. Specifically, for each attribute a_i, we first construct a set of text prompts using all class names of a_i. For example, the set of prompts P_i for shot angle attribute is constructed as "*it is a/an aerial shot*", "*it is a/an overhead shot*", "*it is a/an low angle shot*", etc. These text prompts are fed into the text encoder of XCLIP to calculate the prompt embedding $E^t_i \in R^{C_i \times d}$ where C_i denotes the number of classes of a_i, and the visual embedding $E^v_j \in R^d$ of context shot S_j are extracted by the video encoder:

$$E^t_i = TextEncoder(P_i), \quad E^v_j = VideoEncoder(S_j)$$

The attribute distribution $p_{j,i} \in R^{C_i}$ of S_j over a_i are obtained by applying softmax function on the similarity $D_{j,i} \in R^{C_i}$ between E^t_i and E^v_j, that is

$$D_{j,i} = E^t_i E^v_j, \quad p_{j,i}[k] = \frac{e^{D_{j,i}[k]}}{\sum e^{D_{j,i}[k]}}$$

We calculate the distributions over all attributes $\{a_1, \ldots, a_8\}$ in the same way for each context shot, and the information of S_j is represented as $\bar{A}_j = [p_{j,1}|\ldots|p_{j,8}]$ where $|$ denotes the concatenation operation. As a result, the information of all context shots is defined as $\{\bar{A}_0, \ldots, \bar{A}_n\}$.

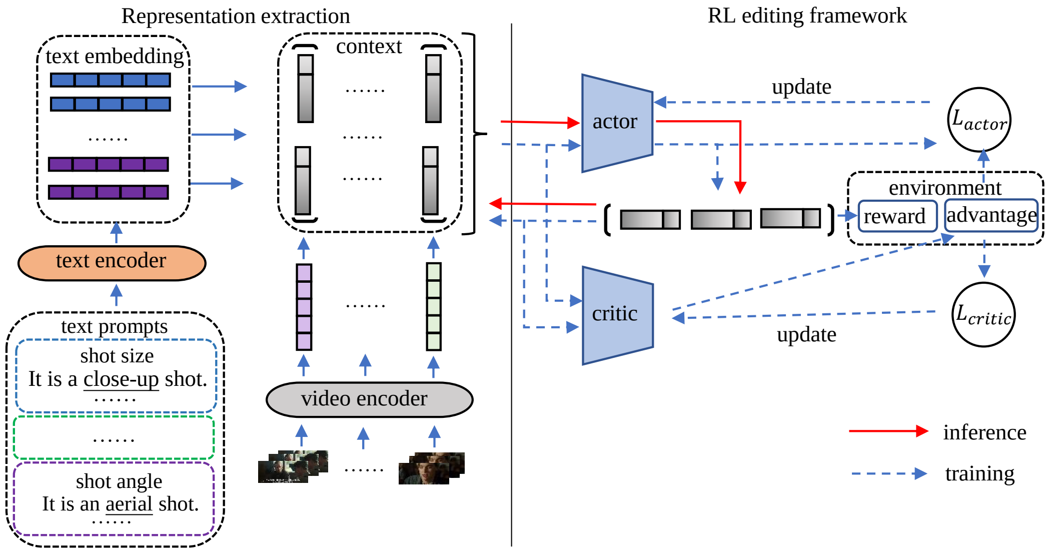

Figure 2: The architecture of the proposed method and its training and testing processes.

3.3 RL-Based Editing Framework

Another essential stage in the automatic video editing method is shot/view selection. Existing heuristic methods integrate various empirical rules summarized by professional experts in the selection process. Yet the difficulties in precisely transferring these rules into computable formulas may result in a gap between the resultant videos and the professional-look videos. In addition, the users are still required to manually fine-tune the parameters to ensure a high-quality resultant video. Recently, some studies start to explore learning-based methods to broadcast a scene automatically. A main difference between broadcasting/directing and general editing is that general editing may involve sequential decisions, while broadcasting can be treated as one-shot editing only. Therefore, traditional supervised learning methods cannot tackle sequential editing well. To overcome this problem, we propose an RL editing framework to handle sequential editing problems. There are three key elements, i.e., state, action, and reward, to be defined.

State: The state E in our framework is the information source to guide the virtual editor to make decisions. It can be arbitrary features extracted from the input streams. For example, for event-driven editing, handcrafted features [11, 27] or high-level semantic features [18, 45] can be used to define a state. In the movie editing scene, we define a state as the attribute distributions of context shots, i.e., $E_0 = [\bar{A}_0|\dots|\bar{A}_n]$. State E_0 is fed into the actor network to predict the attribute distribution $\hat{A}_{n+1}$ of S_{n+1}, then a subsequent state $E_1 = [\bar{A}_1|\dots|\bar{A}_n|\hat{A}_{n+1}]$ is used to predict the attributes $\hat{A}_{n+2}$ of S_{n+2}. To intensify the temporal relationship among the shots, we will additionally add position embedding to the shot representation, just like the word embedding in the transformer model will be assigned with positional encodings.

Action The action O is the editing decision by the virtual editor, and the definition of its space depends on the goal of the task. For example, in the directing scene, the action space could be defined as the indices of cameras to be broadcasted. In the movie editing task, we define an action as an 8-dimensional vector, $O \in \mathbb{R}^8$, with each element indicating a class of the attribute.

Reward Reward r is an essential part that encourages the virtual editor to make decisions, and it measures how good the action taken is. Technically, a larger reward means that the action token is better. In our task setting, an action is an 8-dimensional vector, so we set the reward at each step is also an 8-dimensional vector to measure the predicted attributes independently. Let $\hat{A}_i \in \mathbb{R}^8$ and $A_i \in \mathbb{R}^8$ denote the action predicting the attributes of S_i and the ground-truth attributes, respectively, we define the reward vector as:

$$
\begin{aligned}
r &= R(\hat{A}_i, A_i) \\
&= [\mathbb{I}(\hat{A}_i[1], A_i[1]), \dots, \mathbb{I}(\hat{A}_i[8], A_i[8])], \text{where,}
\end{aligned}
$$

$$
\mathbb{I}(\hat{a}, a) = \begin{cases} 1, & \text{if } \hat{a} = a \\ -1, & \text{otherwise.} \end{cases}
$$

Training We develop our RL framework with an actor-critic scheme [25, 37, 40] to make framework more effective and general. The goal of training is to find the optimal policy/actor network π maximizing the total discounted reward $\mathcal{R}$ in an editing episode. Let r_t and γ denote the immediate reward at the t-th step and the discount factor, respectively, the total discounted reward $\mathcal{R}$ and the optimal policy π^* is defined as:

$$
\mathcal{R} = \sum_{t=0}^{T} \gamma^{t-1} r_t, t = 0, 1, \dots, T, \quad \pi^* = \arg\max \mathbb{E}(\mathcal{R}|\pi)
$$

To achieve this goal, two networks, a critic network V and a policy/actor network π will be trained. The critic network aims to measure how good the action taken is, and its outputs will be used to indicate the updated direction of the policy network. Specifically, we build a Multi-Layer Perception (MLP) model as the critic network and it takes as inputs the context representation $E_i = [\bar{A}_i|\dots|\bar{A}_{n+i}]$ and the action $\hat{A}_{n+i+1}$ sampled from $\pi(E_i)$, and returns an 8-dimensional vector indicating the quality score of sequence $\{A_i, \ldots, A_{n+i}, \hat{A}_{n+i+1}\}$ and the expected future cumulative discounted reward from state $E_{i+1} = [\bar{A}_{i+1}| \ldots |\bar{A}_{n+i}|\hat{A}_{n+i+1}]$. Training critic network requires defining the advantage function, which indicates how much reward is gained by taking current action compared to the average decisions. The advantage function $\mathcal{A} \in \mathbb{R}^8$ at time t is defined as:

$$\begin{aligned} \mathcal{A} &= \delta_t + \gamma\delta_{t+1} + \cdots + \gamma^{T-t+1}\delta_{T-1} \\ \delta_t &= r_t + \gamma V(E_{t+1}, \hat{A}_{n+t+1}) - V(E_t, \hat{A}_{n+t}) \end{aligned} \tag{1}$$

The objective function L_{critic} for training V is to minimize the advantage function, $L_{critic} = \frac{1}{2}||\mathcal{A}||^2$.

We use a transformer-based architecture as the backbone of the actor network due to its excellent ability to explore the temporal relationships among the tokens. The actor network takes as input the editing context $E_i = \{\bar{A}_i, \ldots, \bar{A}_{n+i}\}$ and treats each shot representation as a token embedding, and the final global embedding is passed to eight MLP heads followed by a softmax function to generate the distributions $\hat{A}_{n+i+1}$ of 8 attributes. The update direction of policy network π will depend on the advantage function $\mathcal{A}$ as it measures the profits of the actions sampled from π, thus the objective for training policy network is defined as follows:

$$L_{actor} = \sum -\log \pi_\theta(\hat{A}_{n+t}|E_t)\mathcal{A} \tag{2}$$

Intuitively, when the sampled action $\hat{A}_{n+t}$ leads to a large positive $\mathcal{A}$, the probability of $\hat{A}_{n+t}$ will be increased, and vice verse.

The overall training process at a step is shown in Fig.2. At state E_i, an action $\hat{A}_{n+i+1}$ is first sampled from $\pi(E_i)$ and passed to the environment to acquire the reward r_i, then a new state E_{i+1} is constructed and a new action $\hat{A}_{n+i+2}$ is sampled. This process proceeds until the maximum length is reached, and a sequence of $(E_i, \hat{A}_{n+i+1}, r_i)_{i=1:T}$ is obtained. Afterward, $\hat{A}_{n+i+1}$ and E_i are fed into V to compute a score, which will be compared with cumulative discounted reward to calculate the advantage of $\hat{A}_{n+i+1}$. This advantage will be used to update V and π.

4 EXPERIMENTS

In this section, we will first introduce the dataset and the evaluation metrics of the proposed new task. Afterward, we will conduct various experiments to validate the superiority of the proposed context representation and the RL-based editing framework. Finally, we will apply our editing framework on an online lecture broadcasting scene, which is an event-driven scene, to examine its generalization ability.

4.1 Dataset and Metric

Dataset We conduct the experiments on a public dataset, AVE, which collects the videos from real movie scenes that cover a wide range of genres. A movie scene consists of 35.09 shots on average, and the shot boundaries in seconds have been annotated. Thereby a total number of 196,176 shots are segmented out from all the movie scenes, and the attributes of each shot are manually labeled which helps to evaluate algorithms objectively. We follow the same protocol as AVE, where 70 percent of scenes are used for training, 10 percent for validation, and the left 20 percent are used as the test set. In line with the work, we sample 9 consecutive shots from each scene at a time, the first 4 shots, i.e., n=4, in this sequence are encoded as the initial editing context to predict the attributes of the next shot. For a fair comparison with previous work which formulates the editing process as a retrieval problem, we also perform a retrieve task with the predicted attributes, and the candidate list is composed of the 5 remaining shots.

Evaluation Metrics For the retrieval task, we evaluate the method with the $rank1$ metric which is defined as the percentage of the correct retrievals where the ground-truth shot is the first shot of the original candidate sequence. For the attribute prediction task, we report average per-class accuracy Acc_i and the overall accuracy Acc. Let $\hat{a}_i$ and a_i denote the predicted and ground-truth class of the i-th attribute, the metrics is defined as:

$$Acc_i = \frac{1}{N}\sum^{N} \hat{a}_i == a_i$$

$$Acc = \frac{1}{N}\sum^{N}(\hat{a}_1 == a_1)\&\& \ldots \&\&(\hat{a}_8 == a_8) = \frac{1}{N}\sum^{N} \hat{A} == A$$

where N is the number of samples. The above metrics are designed for one-shot editing, yet editing is a sequential process. To validate the ability to make sequential decisions, we further develop the metrics, two-shot retrieve accuracy $2 - rank1$ and two-shot overall attribute accuracy $2 - Acc$, to evaluate the two-shot editing ability [49]. The two-shot editing metrics are based on the assumption that the decisions for the first shot are completely correct. Let $\hat{S}_{i+n+1}$ and $\hat{S}_{i+n+2}$ denote two consecutive retrieved shots for a sample $\{S_i, \ldots, S_{i+n}\}$, and the corresponding ground-truth shots are S_{i+n+1} and S_{i+n+2}, then $2 - rank1$ is defined as:

$$2 - rank1 = \frac{1}{N}\sum^{N}(\hat{S}_{i+n+1} == S_{i+n+1})\&\&(\hat{S}_{i+n+2} == S_{i+n+2})$$

In the calculation of 2-shot accuracy $2-Acc$, the predicted attributes $\hat{A}_{i+n+1}$ and $\hat{A}_{i+n+2}$ are compared with the ground-truth attributes A_{i+n+1} and A_{i+n+2}, that is:

$$2 - Acc = \frac{1}{N}\sum^{N}(\hat{A}_{i+n+1} == A_{i+n+1})\&\&(\hat{A}_{i+n+2} == A_{i+n+2})$$

4.2 Implementation Details

During the representation process in Sec.3.2, we adopt the text and video encoders of the pre-trained XCLIP model to extract multimodal representations and 32 frames are uniformly sampled from each shot as the input to the video encoder. The editing context length n is set to 4, the total dimension of attribute distributions is $7 + 6 + 3 + 6 + 6 + 9 + 8 + 6 = 51$, thus the dimension of the editing context is $4 * 51 = 204$. In our RL-based editing framework, we build the critic network with a Multi-Layer Perception (MLP) network, which takes the input of the concatenation of the editing context and the sampled action. As for the policy network, we adopt the ViT [13] architecture without the patch embedding module, thus the editing context is reshaped as a token sequence with a length of 4 and fed directly to the first transformer block. The critic and policy networks are trained using two Adam optimizers under the Pytorch framework with a learning rate of 1e-4. To speed up the training under the RL framework, we first pre-train a model which has the same architecture as the policy network in a supervised manner, then its parameters are used to initialize the policy network to conduct reinforcement learning.

Methods	Acc_{np}	Acc_{sa}	Acc_{sl}	Acc_{sm}	Acc_{ss}	Acc_{sub}	Acc_{st}	Acc_{sou}	$1-Acc$	$2-Acc$	reward	$rank1$	$2-rank1$
Random	12.9	18.2	33.8	16.1	17.1	11.5	12.4	16.9	0.0	0.0	2.7	18.7	4.1
MLP	39.1	85.2	81.4	54.2	73.7	82.3	46.4	83.2	7.7	2.5	10.9	42.4	25.3
LSTM	39.2	85.2	81.6	54.9	73.7	82.6	46.5	83.3	7.9	2.3	11.0	42.5	25.6
Trans	40.8	85.3	81.4	55.5	**75.0**	83.7	46.1	83.8	8.4	2.4	11.0	43.7	24.7
RL(Ours)	**52.3**	**86.3**	**89.1**	**59.9**	74.9	**85.3**	**51.2**	**84.0**	**11.0**	**3.2**	**11.5**	**46.7**	**28.4**

Table 1: The performance comparisons among different methods. The best result is highlighted with bold type.

Methods	Feat.	Acc_{np}	Acc_{sa}	Acc_{sl}	Acc_{sm}	Acc_{ss}	Acc_{sub}	Acc_{st}	Acc_{sou}	$1-Acc$	$2-Acc$	$rank1$	$2-rank1$
RL	**R3D**	44.8	84.2	82.7	55.3	73.0	82.1	46.4	83.2	8.4	2.3	42.4	22.5
	CARL	37.6	82.4	73.1	45.2	74.9	83.1	46.1	81.0	6.9	1.8	42.6	24.0
	CLIP	38.3	83.6	74.8	51.1	75.0	83.8	47.6	83.4	7.2	1.9	42.7	23.9
	XCLIP-vis	35.2	86.4	72.0	47.1	74.3	84.4	44.3	84.0	8.0	2.2	43.2	22.8
	GT-ad	**59.2**	**86.5**	**94.7**	**73.8**	**80.1**	**90.3**	**59.0**	**86.4**	**23.8**	**10.4**	**50.5**	**30.0**
	XCLIP-ad	52.3	86.3	89.1	59.9	74.9	85.3	51.2	84.0	11.0	3.2	46.7	28.4

Table 2: The performance comparisons among the methods using different representations as editing context.

4.3 Comparisons

In this section, we will compare our method with serval baselines in terms of the attribute prediction task. We conduct this experiment with 5 methods: 1) Random, randomly predict the shot attributes; 2) MLP, an MLP network with 8 classification heads is trained with the cross entropy as the objective function, using the features extracted from R3D model; 3)LSTM [9], to model the temporal relationship, we implement an LSTM-based network and train it with the R3D features. 4) Trans., the actor network trained in the same way as MLP using the proposed representations. 5) RL, the proposed method. The results are reported in Table.1, it can be observed that the proposed method outperforms all the baseline methods in most of the metrics except for the accuracy of the shot subject attribute Acc_{ss}. Especially, for the metrics of $\{1-Acc, 2-Acc\}$, our method improves traditional **MLP** and **LSTM** by $\{3.3\%, 0.7\%\}$ and $\{3.1\%, 0.9\%\}$, respectively. After applying our RL editing method, the reward gained also improves from 11.0 to 11.5. These improvements benefit from the proposed representations and RL-based editing framework. Besides attribute prediction, we also compare these methods on the retrieval task. The predicted attributes are used as a query to retrieve the shot with the closest attributes from the candidate list, and the one-shot and two-shot retrieval results, $rank1$ and $2-rank1$, are also listed in Table.1. Even if evaluated on the retrieval task, our method of predicting attributes is still effective and outperforms baseline methods in both metrics consistently.

4.4 Ablation Study

Representation. In this section, we will inspect the effectiveness of the proposed context representations. Specifically, we compare the performances of methods using other five kinds of representations, including 1).**R3D** [41], this feature has been used by previous automatic editing systems; 2). **CARL** [10], a self-supervised learning-based video representation model; 3). **CLIP** [32], a cross-modal model trained with image-text pair data. We use the image encoder of CLIP to extract frame-wise features, and the average frame feature is used as the shot representation; 4). **XCLIP-vis**, XCLIP [28] is also a cross-modal model trained with video-text pair data. Unlike the proposed representation which calculates the similarities between vision features and textual features, we use the embedding from the video encoder directly as the representation in this experiment; 5). **GT-ad**, To prove the rationality of the assumption that the attribute information of context shots helps to make correct editing decisions, we encode the ground-truth attributes of context shots as one-hot vectors and cascade these vectors as representations.

The results are listed in Table.2, where **XCLIP-ad** represents the proposed attribute distribution representation. **R3D**, **CARL**, **CLIP**, and **XCLIP-vis** are all visual features, so it is not surprising that the methods using them achieve similar performances. However, since they are not designed particularly for the editing task, the performances using them are relatively low when compared with the method using our representation, **XCLIP-ad**. Specifically, the best feature among them, **R3D**, attains 2.6% and 0.9% lower $1-Acc$ and $2-Acc$ than **XCLIP-ad** dose. Moreover, we can observe that the ground-truth attribute encoding **GT-ad** outperforms other methods by a large margin and achieves the best result in all metrics. This observation also justifies that the direction of exploring context attributes is promising, even though there is still a gap.

RL-based framework. Another core component in our method is the RL-based editing framework. In this section, we will investigate the benefits of learning to make sequential editing decisions. We apply our RL framework to three baseline methods, i.e., **MLP**, **LSTM**, and **Trans.**, and the results with and without RL are reported in Table.3. By comparing the results in two-shot metrics, i.e., $2-Acc$ and $2-rank1$, where our editing framework consistently improves the baseline methods, it can be concluded that the proposed framework does help to make sequential decisions. Surprisingly, after applying our framework, the performances of baseline methods in one-shot metrics, i.e., $1-Acc$ and $rank1$, are increased as well. The reason is that the first shot decision influences the decision of the second shot, so the framework will adjust the first shot decision to make sequential decisions to achieve higher rewards during the training process.

Figure 3: The retrieved sequences generated by our method and a feature-based method.

Methods	$1-Acc$	$2-Acc$	$rank1$	$2-rank1$
MLP	8.4	1.5	44.1	24.4
MLP+RL	9.1	2.0	45.8	25.5
LSTM	9.0	2.3	45.7	25.1
LSTM+RL	9.9	2.5	46.0	26.0
Trans.	8.4	2.4	43.7	24.7
Trans.+RL	**11.0**	**3.2**	**46.7**	**28.4**

Table 3: The performance comparisons between the methods with (+RL) and without RL.

4.5 Qualitative Results

Previous method [3] has attempted to use the LSTM network to estimate the visual feature to retrieve the next shot, and the features used for training are extracted with a pre-trained R3D model. To intuitively illustrate the benefits of our method, we implement the above method by ourselves and visualize three retrieved sequences from both our attribute-based method and the above feature-based method in Fig.3. From this figure, it can be seen that the feature-based method tends to retrieve visually similar shots e.g., the 1-st row, the 3-rd row, and the 1-st retrieved shot of the fifth row, and these resultant sequences do not follow the empirical editing rules in dialogue scenes [24]. One reason might be that the shot retrieval is based on the feature similarities between the estimated visual feature and the features of candidate shots, so visually similar shots are preferred by the feature-based method. In contrast, our attribute-based retrieval results in more diverse sequences, which basically follow the shot order in the dialogue scenes, e.g., show the speaking characters alternatively. From this angle, the proposed presentation is more suitable for the editing task.

4.6 Evaluation in Event-Driven Scene

Application background To verify that our editing framework is also feasible in event-driven directing/editing scenes, we apply our RL-based editing framework to learn the editing/directing style of online lecture broadcasting. The task of automatic online lecture broadcasting is to select a view from multiple cameras to broadcast at a time. A few automatic editing systems [26, 34, 43] for this task have been studied, yet these heuristic methods only generate a mechanical broadcasting stream and cannot meet the preferences of different students. For example, some students prefer to watch slide view when there is no particular event happening, while some students might focus more on the teacher's view. Therefore, we will conduct experiments to validate that the proposed editing framework can also tackle this problem. In other words, we will show that our method can learn individual preferred styles from some given watching records.

Data preparation The experiment is conducted in a classroom where 7 cameras are deployed to shoot the class from different angles, including the slide close-up shot, left and right blackboard close-up shots, left and right medium shots, overall long shot, and student shot. To train and evaluating the policy network, it demands stylistic watching records as the ground-truth selections. As it is challenging to collect the stylistic records, we employ a simulated scheme to generate them. Concretely, we collect 7 synchronized videos with a length of T time units (frame or second) from a real class for training, and 7 videos from another class are used as a testing scene. Next, the representations $F = \{f_t\}_{t=1:T}, f_t = [f_t^e|f_t^v|f_t^s]$ which counts the event information f^e, view transition constraints f^v, and switch penalty f^s, are extracted and fed to a parameterized heuristic editing method $H()$ [18] to generate a view sequence as watching records. Therefore, the ground-truth view selections, $Y^{tr} = \{y_t^{tr}\}_{t=1:T}$ and $Y^{test} = \{y_t^{test}\}_{t=1:T}$ for training and testing are obtained as :

$$Y^{tr} = H(F^{tr}, \omega), \quad Y^{test} = H(F^{test}, \omega)$$

where ω is the parameter vector controlling the styles of view selections. The view sequences generated with different ω are considered sequences of different styles.

RL problem Supposing Y^{tr} and Y^{test} are generated with the same ω on two different classes, the goal of our framework is to train a policy network with data F^{tr} and Y^{tr}, which then takes as input F^{test} to generate view sequence approaching Y^{test}. The purpose of this experiment is to prove whether the policy network learns the style of Y^{tr} with our framework. As we mainly focus on investigating the learning ability and the feasibility of learning the style from

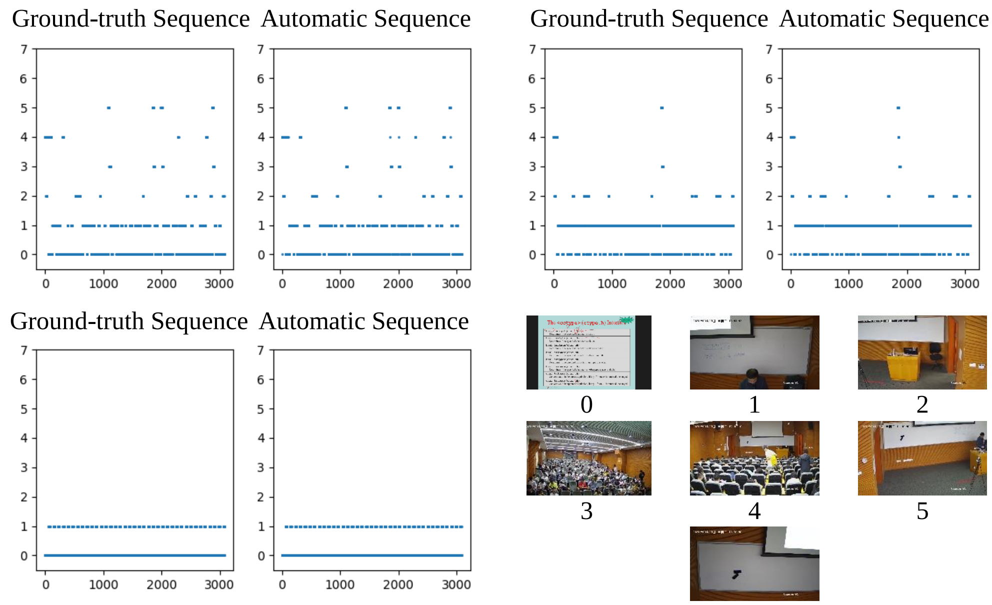

Figure 4: The sequences generated by the trained actors of three styles and the corresponding ground-truth sequences.

the labeled sequence, the state representation at time t is simply defined as f_t, and the action space is defined as the camera indices, $\hat{y}_t \in \{1, \ldots, 7\}$. The reward function $R()$ is:

$$R(\hat{y}, y) = \begin{cases} 1, & \text{if } \hat{y} == y \\ -1, & \text{otherwise.} \end{cases}$$

when the sampled action $\hat{y}$ is the same as the ground-truth action y, the function returns a positive reward, and vice versa. The objective functions for training critic and actor networks remain the same as Eq.1 and Eq.2. For simplicity, two MLP models are adopted as the architectures of critic and actor networks.

Evaluation During the experiment, we use three different $\{\omega_i\}_{i=1:3}$ to generate ground-truth training and testing sequences, $\{Y_i^{tr}\}_{i=1:3}$ and $\{Y_i^{test}\}_{i=1:3}$, and these three paired dataset $\{(Y_i^{tr}, Y_i^{test})\}_{i=1:3}$ are used to train and evaluate three policy networks separately, representing three virtual editors of different styles. For each style, we report the comparisons between the predicted sequence $\hat{Y}_i^{test}$ and the ground-truth sequence Y_i^{test} in terms of the overlap ratio *Ratio* and three sequence properties, i.e., the average shot length L_{avg}, the maximum shot length L_{max}, and the number of switches N_{sw}. The results are listed in Table.4, the trained actor can generate sequences that are close to the corresponding ground-truth sequence in testing scenes. For example, the sequence $\hat{Y}_1^{test}$ generated by the actor trained with Y_1^{tr} in the testing class achieves an overlap ratio of 99% with Y_1^{test}, and the properties such as average shot length, the maximum shot length are also close. The same conclusion is also obtained in the other two styles. These comparisons prove that our editing framework is able to learn the editing styles from the stylistic sequences.

In Fig.4, we visualize the generated sequences of three stylistic actors and the corresponding ground-truth sequences in a testing class. It is obvious that the editing patterns or styles of the three ground-truth sequences are truly different, e.g., the top-left sequence displays the slide view (0) more frequently, the top-right sequence contains more the close-up view (1), and the bottom-left sequence regularly displays the slide view and the close-up view. Thus, the goal of the stylistic actor networks is to generate the corresponding stylistic sequences. The figures of automatic sequences (generated by actor networks) show similar patterns as their ground truth. This result confirms that our RL-based editing framework can generalize to event-driven scenes and learn different editing styles.

style	*Ratio*	L_{avg}	L^*_{avg}	L_{max}	L^*_{max}	N_{sw}	N^*_{sw}
ω_1	0.99	25.3	26.6	71	71	121	115
ω_2	0.98	23.3	23.7	75	76	131	129
ω_3	0.99	40.0	40.0	61	61	76	76

Table 4: The editing results on three stylistic test sets. $*$ indicates the properties of ground-truth sequences.

5 CONCLUSION

In this paper, we formulate a new editing task as next shot attribute prediction, which is more helpful in practice compared with retrieval-based editing. Next, a new shot attribute-based editing representation is proposed, experimental results show that this representation benefits general-scene editing and is superior to other visual features. Furthermore, to bridge the gap between the videos generated by heuristic rules and the professional-look videos, we propose an RL-based editing framework to train the virtual editor with professional videos. Extensive experiments are carried out on a real movie dataset to demonstrate that our framework can directly learn the editing patterns from well-edited movies and make sequential editing decisions. Finally, we conduct experiments in an online lecture broadcasting scene, which prove that the RL editing framework can generalize to event-driven editing.

REFERENCES

[1] Hadi AlZayer, Hubert Lin, and Kavita Bala. 2021. AutoPhoto: Aesthetic Photo Capture using Reinforcement Learning. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 944–951.

[2] Ido Arev, Hyun Soo Park, Yaser Sheikh, Jessica Hodgins, and Ariel Shamir. 2014. Automatic editing of footage from multiple social cameras. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 1–11.

[3] Dawit Mureja Argaw, Fabian Caba Heilbron, Joon-Young Lee, Markus Woodson, and In So Kweon. 2022. The anatomy of video editing: A dataset and benchmark suite for ai-assisted video editing. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part VIII*. Springer, 201–218.

[4] Yasuo Ariki, Shintaro Kubota, and Masahito Kumano. 2006. Automatic production system of soccer sports video by digital camera work based on situation recognition. In *Eighth IEEE International Symposium on Multimedia (ISM'06)*. IEEE, 851–860.

[5] Sophia Bano and Andrea Cavallaro. 2016. ViComp: composition of user-generated videos. *Multimedia tools and applications* 75, 12 (2016), 7187–7210.

[6] Rogerio Bonatti, Wenshan Wang, Cherie Ho, Aayush Ahuja, Mirko Gschwindt, Efe Camci, Erdal Kayacan, Sanjiban Choudhury, and Sebastian Scherer. 2020. Autonomous aerial cinematography in unstructured environments with learned artistic decision-making. *Journal of Field Robotics* 37, 4 (2020), 606–641.

[7] Christine Chen, Oliver Wang, Simon Heinzle, Peter Carr, Aljoscha Smolic, and Markus Gross. 2013. Computational sports broadcasting: Automated director assistance for live sports. In *2013 IEEE International Conference on Multimedia and Expo (ICME)*. 1–6. https://doi.org/10.1109/ICME.2013.6607445

[8] Jianhui Chen, Hoang M Le, Peter Carr, Yisong Yue, and James J Little. 2016. Learning online smooth predictors for realtime camera planning using recurrent decision trees. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 4688–4696.

[9] Jianhui Chen, Lili Meng, and James J Little. 2018. Camera selection for broadcasting soccer games. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 427–435.

[10] Minghao Chen, Renbo Tu, Chenxi Huang, Yuqi Lin, Boxi Wu, and Deng Cai. 2022. Self-supervised and Weakly Supervised Contrastive Learning for Frame-wise Action Representations. *arXiv preprint arXiv:2212.03125* (2022).

[11] Navneet Dalal and Bill Triggs. 2005. Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, Vol. 1. Ieee, 886–893.

[12] Fahad Daniyal and Andrea Cavallaro. 2011. Multi-camera scheduling for video production. In *2011 Conference for Visual Media Production*. IEEE, 11–20.

[13] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929* (2020).

[14] Filip Germeys and Géry d'Ydewalle. 2007. The psychology of film: Perceiving beyond the cut. *Psychological research* 71, 4 (2007), 458–466.

[15] Sidney Gottlieb. 2013. Hitchcock on Truffaut. *Film Quarterly* 66, 4 (2013), 10–22.

[16] Mirko Gschwindt, Efe Camci, Rogerio Bonatti, Wenshan Wang, Erdal Kayacan, and Sebastian Scherer. 2019. Can a robot become a movie director? learning artistic principles for aerial cinematography. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 1107–1114.

[17] Rachel Heck, Michael Wallick, and Michael Gleicher. 2007. Virtual videography. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 3, 1 (2007), 4–es.

[18] Panwen Hu, Jiazhen Liu, Tianyu Cao, and Rui Huang. 2021. Reinforcement Learning Based Automatic Personal Mashup Generation. In *2021 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 1–6.

[19] Chong Huang, Yuanjie Dang, Peng Chen, Xin Yang, and Kwang-Ting Cheng. 2021. One-Shot Imitation Drone Filming of Human Motion Videos. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44, 9 (2021), 5335–5348.

[20] Chong Huang, Chuan-En Lin, Zhenyu Yang, Yan Kong, Peng Chen, Xin Yang, and Kwang-Ting Cheng. 2019. Learning to film from professional human motion videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 4244–4253.

[21] Hao Jiang, Sidney Fels, and James J Little. 2008. Optimizing multiple object tracking and best view video synthesis. *IEEE Transactions on Multimedia* 10, 6 (2008), 997–1012.

[22] Hongda Jiang, Bin Wang, Xi Wang, Marc Christie, and Baoquan Chen. 2020. Example-driven virtual cinematography by learning camera behaviors. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 45–1.

[23] Rene Kaiser, Wolfgang Weiss, Malte Borsum, Axel Kochale, Marco Masetti, and Valentina Zampichelli. 2012. virtual director for live event broadcast. In *Proceedings of the 20th ACM international conference on Multimedia*. 1281–1282.

[24] 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.

[25] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971* (2015).

[26] Qiong Liu, Yong Rui, Anoop Gupta, and Jonathan J Cadiz. 2001. Automating camera management for lecture room environments. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. 442–449.

[27] David G Lowe. 2004. Distinctive image features from scale-invariant keypoints. *International journal of computer vision* 60 (2004), 91–110.

[28] Yiwei Ma, Guohai Xu, Xiaoshuai Sun, Ming Yan, Ji Zhang, and Rongrong Ji. 2022. X-CLIP: End-to-End Multi-grained Contrastive Learning for Video-Text Retrieval. In *Proceedings of the 30th ACM International Conference on Multimedia*. 638–647.

[29] Matthias Minderer, Alexey Gritsenko, Austin Stone, Maxim Neumann, Dirk Weissenborn, Alexey Dosovitskiy, Aravindh Mahendran, Anurag Arnab, Mostafa Dehghani, Zhuoran Shen, et al. 2022. Simple open-vocabulary object detection with vision transformers. *arXiv preprint arXiv:2205.06230* (2022).

[30] Naila Murray, Luca Marchesotti, and Florent Perronnin. 2012. AVA: A large-scale database for aesthetic visual analysis. In *2012 IEEE conference on computer vision and pattern recognition*. IEEE, 2408–2415.

[31] Yingwei Pan, Yue Chen, Qian Bao, Ning Zhang, Ting Yao, Jingen Liu, and Tao Mei. 2021. Smart director: an event-driven directing system for live broadcasting. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 17, 4 (2021), 1–18.

[32] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*. PMLR, 8748–8763.

[33] Rémi Ronfard. 2021. Film directing for computer games and animation. In *Computer Graphics Forum*, Vol. 40. Wiley Online Library, 713–730.

[34] Yong Rui, Anoop Gupta, Jonathan Grudin, and Liwei He. 2004. Automating lecture capture and broadcast: technology and videography. *Multimedia Systems* 10 (2004), 3–15.

[35] Mukesh Kumar Saini, Raghudeep Gadde, Shuicheng Yan, and Wei Tsang Ooi. 2012. Movimash: online mobile video mashup. In *Proceedings of the 20th ACM international conference on Multimedia*. 139–148.

[36] Mukesh Kumar Saini and Wei Tsang Ooi. 2018. Automated Video Mashups: Research and Challenges. *MediaSync: Handbook on Multimedia Synchronization* (2018), 167–190.

[37] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

[38] Prarthana Shrestha, Peter HN de With, Hans Weda, Mauro Barbieri, and Emile HL Aarts. 2010. Automatic mashup generation from multiple-camera concert recordings. In *Proceedings of the 18th ACM international conference on Multimedia*. 541–550.

[39] Than Htut Soe. 2021. AI video editing tools. What editors want and how far is AI from delivering? *arXiv preprint arXiv:2109.07809* (2021).

[40] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. 1999. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems* 12 (1999).

[41] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. 2018. A closer look at spatiotemporal convolutions for action recognition. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 6450–6459.

[42] Shuhei Tsuchida, Satoru Fukayama, and Masataka Goto. 2018. Automatic system for editing dance videos recorded using multiple cameras. In *Advances in Computer Entertainment Technology: 14th International Conference, ACE 2017, London, UK, December 14-16, 2017, Proceedings 14*. Springer, 671–688.

[43] Feng Wang, Chong-Wah Ngo, and Ting-Chuen Pong. 2007. Lecture video enhancement and editing by integrating posture, gesture, and text. *IEEE transactions on multimedia* 9, 2 (2007), 397–409.

[44] Jinjun Wang, Changsheng Xu, Engsiong Chng, Hanqing Lu, and Qi Tian. 2008. Automatic composition of broadcast sports video. *Multimedia Systems* 14 (2008), 179–193.

[45] Xueting Wang, Takatsugu Hirayama, and Kenji Mase. 2015. Viewpoint sequence recommendation based on contextual information for multiview video. *IEEE MultiMedia* 22, 4 (2015), 40–50.

[46] Xueting Wang, Yuki Muramatu, Takatsugu Hirayama, and Kenji Mase. 2014. Context-dependent viewpoint sequence recommendation system for multi-view video. In *2014 IEEE International Symposium on Multimedia*. IEEE, 195–202.

[47] Hui-Yin Wu and Arnav Jhala. 2018. A Joint Attention Model for Automated Editing.. In *INT/WICED@ AIIDE*.

[48] Yue Wu, Tao Mei, Ying-Qing Xu, Nenghai Yu, and Shipeng Li. 2015. MoVieUp: Automatic mobile video mashup. *IEEE Transactions on Circuits and Systems for Video Technology* 25, 12 (2015), 1941–1954.

[49] Yu Xiong, Fabian Caba Heilbron, and Dahua Lin. 2022. Transcript to video: Efficient clip sequencing from texts. In *Proceedings of the 30th ACM International Conference on Multimedia*. 5407–5416.

[50] Mengde Xu, Zheng Zhang, Fangyun Wei, Yutong Lin, Yue Cao, Han Hu, and Xiang Bai. 2022. A Simple Baseline for Open-Vocabulary Semantic Segmentation with Pre-trained Vision-Language Model. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXIX*. Springer, 736–753.

[51] Danqing Yang, Longfei Zhang, Yufeng Wu, Shugang Li, Dong Liang, and Gangyi Ding. 2019. Computable Framework For Live Sport Broadcast Directing. In *2019 IEEE International Symposium on Multimedia (ISM)*. IEEE, 239–2391.

[52] Zixiao Yu, Chenyu Yu, Haohong Wang, and Jian Ren. 2022. Enabling Automatic Cinematography with Reinforcement Learning. In *2022 IEEE 5th International Conference on Multimedia Information Processing and Retrieval (MIPR)*. IEEE, 103–108.

[53] Xinrong Zhang, Yanghao Li, Yuxing Han, and Jiangtao Wen. 2022. AI Video Editing: a Survey. (2022).