Virtual Camera Layout Generation using a Reference Video

Jung Eun Yoo*
KAIST, Visual Media Lab
jey920@kaist.ac.kr

Kwanggyoon Seo*
KAIST, Visual Media Lab
skg1023@kaist.ac.kr

Sanghun Park
KAIST, Visual Media Lab
reversi@kaist.ac.kr

Jaedong Kim
KAIST, Visual Media Lab
jaedong27@kaist.ac.kr

Dawon Lee
KAIST
qwer2345@kaist.ac.kr

Junyong Noh
KAIST, Visual Media Lab
junyongnoh@kaist.ac.kr

Figure 1: Given a reference video (the first row), our method generates a virtual camera layout in a 3D scene, which follows the cinematic intention of the reference video. Our method works for characters with different (the third row) as well as similar (the second row) body proportions from/to those of humans. The frames on the top left are from *Back to the Future (R. Zemeckis, 1985) ©Universal Pictures.* Characters and background assets used for bottom right images: *©Mix and Jam, ©Jeremy Vikery and Alex Mateo.*

ABSTRACT

We propose a method that generates a virtual camera layout of a 3D animation scene by following the cinematic intention of a reference video. From a reference video, cinematic features such as the start frame, end frame, framing, camera movement, and the visual features of the subjects are extracted automatically. The extracted information is used to generate the virtual camera layout, which resembles the camera layout of the reference video. Our method handles stylized as well as human characters with body proportions different from those of humans. We demonstrate the effectiveness of our approach with various reference videos and 3D animation scenes. The user evaluation results show that the generated layouts are comparable to layouts created by the artist, allowing us to assert that our method can provide effective assistance to both novice and professional users when positioning a virtual camera.

*Both authors contributed equally to the paper

CHI '21, May 8–13, 2021, Yokohama, Japan

ACM ISBN 978-1-4503-8096-6/21/05…$15.00
https://doi.org/10.1145/3411764.3445437

CCS CONCEPTS

• **Computing methodologies** → **Computer graphics**; **Graphics systems and interfaces**; *Computer vision tasks.*

KEYWORDS

Virtual Camera, Content Analysis, Cinematography

ACM Reference Format:
Jung Eun Yoo, Kwanggyoon Seo, Sanghun Park, Jaedong Kim, Dawon Lee, and Junyong Noh. 2021. Virtual Camera Layout Generation using a Reference Video. In *CHI Conference on Human Factors in Computing Systems (CHI '21), May 8–13, 2021, Yokohama, Japan.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3411764.3445437

1 INTRODUCTION

Camera layout is an important component of cinematography to deliver the emotion and suspense of a scene. The director creates a guideline, known as a shot list, to communicate with a 3D animation layout artist. The layout artist uses the shot list as a reference to position the virtual camera in order to best deliver the cinematic intention of the director. However, for most novice users, it is difficult to express intentions precisely through a virtual camera due to its high degree of freedom (DOF). For professional artists, it is time-consuming to position numerous virtual cameras repeatedly, as is common during the production of a TV series.

Automatic methods have been proposed to analyze the camera layout of a reference video and to generate a virtual camera layout for a given scene. These methods typically use computer vision

and machine learning techniques to classify types of framing [4, 7, 33] and camera movement [5, 10, 17]. However, the framing and camera movement types they can handle are limited to only a small subset of various types, meaning that their practical use is highly limited. Furthermore, previous studies mostly focus on designing a specific language model to generate the virtual camera layout [3, 11, 26, 31, 35], which thwarts its use by novice users.

To ease the virtual camera layout process, we designed our method considering the common practices performed in TV or animation studios, where a reference video or images are heavily used in the early stages of the production process or for previz purposes. Our method works with 3D scenes for which the staging and animation are similar to those of the reference video. Specifically, we analyze the intention of a reference video and create the virtual camera layout of a given scene using a shot list as a communication medium to transfer the cinematic intention. Because reference videos such as live-action movie clips have well-established cinematography, imitating the camera layout in these cases will help novice users as well as experts in this process to generate a satisfying initial virtual camera layout without much effort.

The workflow of the proposed method consists of the following three steps. First, the reference video is segmented into a sequence of shots. Second, we analyze each shot to extract information related to the camera layout, which includes the types of framing and camera movements, and the visual features of the subjects. Next, we match the subject to the character in the 3D scene. All of this information is stored in the shot list and is then used to generate the virtual camera layout. Third, based on this information, the virtual camera is positioned and animated with respect to the characters in the 3D scene. A stylized or exaggerated character with a different body proportion from that of a human is common in 3D animation. To handle these characters when applying the intended framing type, we utilize the skeletal information of the target character. Once the camera is positioned based on the framing type, we create a smooth camera path between the start and end frames to follow the camera movement type.

We demonstrate the effectiveness of our approach by visually comparing the generated virtual camera layout with that of the reference video. We also conducted a user study to evaluate the quality of our framing results on exaggerated characters by comparing them with the results from the previous method [25] and from the artist. We successfully confirmed that the results with the proposed method were preferred over those from the previous method and that the quality was comparable to that of the artist's layout. We conducted an additional user study of the effectiveness of our method. This study confirmed that both amateurs and professional layout artists spent much less time when replicating a reference video using our method as compared to relying on the conventional manual process.

In summary, our method can automatically generate a virtual camera layout for a 3D animation scene. The generated virtual camera position effectively assists the user in achieving the final layout result that reflects the cinematic intention of the reference video. We also propose a customized solution for stylized or exaggerated characters that are commonly used in 3D animation. Our method is able to reproduce the intended semantics of the camera layout from the reference video for characters with significantly different body proportions from those of a human using optimization based on skeletal parts and the framing type.

The main contributions of the paper can be summarized as follows:

(1) A system that analyzes a reference video to generate a virtual camera layout for a 3D animation and that is capable of adapting to stylized characters with non-realistic body proportions.
(2) A means to analyze and extract framing and camera movements from a reference video.
(3) A novel optimization scheme that positions a virtual camera for both stylized as well as human characters.
(4) A camera layout generation better than or comparable to the Toric space method and the artist's camera layout.

2 RELATED WORK

2.1 Camera Layout Analysis

To construct a shot list of a reference video, the following information is needed: the start frame, end frame, framing type, and the camera movement type. In this section, we focus on framing and camera movement. Regarding the extraction of the start and end frames of a shot from a video, please refer to the survey by Smeaton et al. [34].

Framing is determined by the distance from the camera to the subject. Conventional methods [4, 7] use hand-crafted features, while a more recent method [33] uses a convolutional neural network (CNN) to classify framing into various types. While these methods have reported some success in classifying the framing of input images, they are limited to three types of framing. Camera movement can be analyzed using a sequence of frames in each shot. Most of the previous methods do not include the movement type [5, 10, 17], which expresses the direction of the motion. Although Derue et al. [10] is an exception, their method uses only two consecutive frames to predict the camera movement type. Recently, SGNet [29] used a CNN to analyze the camera framing and movement in a given shot jointly. This method classifies framing into five types (*extreme close-up*,*close-up*, *medium shot*,*full shot*, and *long shot*) and movements into four types (*static*, *motion*, *push*, and *pull*). In this work, we use six different scales for framing and 13 different types of camera movement to represent translational, rotational, or curved motions with the direction. This faithfully reflects the basic camera movement semantics, which are widely used in practice (Fig. 2).

2.2 Virtual Cinematography

Reproducing a camera layout using a shot specification has been studied in the fields of virtual cinematography. One popular approach is to provide a shot specification through a high-level camera composition language that describes established filming styles and techniques. These languages, often delivered in the form of text, describe how the shot is composed and what the setup constraints are for virtual camera placement [11, 23, 26, 31, 35]. This method is direct and clear yet requires a certain degree of cinematic knowledge. Shot specifications can be given through images as well. Bares

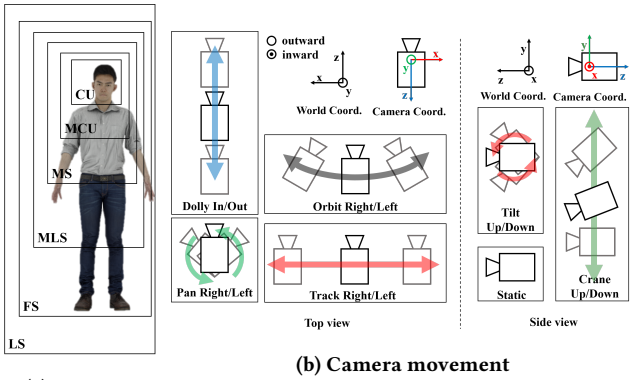
et al. [3] designed an interface for a user to visualize camera compositions through storyboard frames. This allows the user intuitively to design a shot composition, and the finished storyboard frame is later translated into an established language for virtual camera placement. In this work, we analyze the layout information and visual features of the subjects from the reference video and use them as a medium for a specification.

Various techniques have been proposed to compute the camera position and motion for both drone and virtual cinematography [9] . Studies of drone cinematography investigate drone control and path planning during aerial shots [2, 13, 15, 16, 19–21, 27]. Regarding the positioning of a virtual camera, Bares et al. [3] proposed a constraints-based approach that heuristically searches for possible camera parameter values that satisfy the constraints. Ranon and Urli [28] proposed an optimization approach that finds the viewpoint of a camera that maximizes the objective function. Lino and Christie [24, 25] proposed a novel camera viewpoint representation, known as the Toric space, which reduces the conventional 7 DOF search space to 4 DOF. Constraints such as the size and on-screen position of the character's head and the camera distance can be algebraically expressed in the Toric space, enabling intuitive placement as well as control of the camera by the user. Utilizing the Toric space, Wu et al. [35] calculated the position and orientation of a virtual camera based on the visual features delivered through their proposed film language. Galvane et al. [12] attempted to generate a smooth path and movement of the camera using a character motion and user-defined framing for the start and end frames. Burg et al. [6] proposed a method that generates real-time occlusion-aware camera motion by maximizing the visibility of a target subject throughout a motion path.

Recently, Jiang et al. [22] employed an example-driven method to control a virtual camera automatically and produce various stylistic variations of a 3D animation. Similar to Jiang et al. [22], our system takes the analyzed layout information and visual features from a reference video as input. While Jiang et al. [22] focus on generating continuous camera motions only for human characters, we focus on reproducing the framing of a scene, which works with various targets including stylized characters with non-realistic body proportions matching those common in computer animations. Furthermore, with our method shot-level editing can be achieved easily given that the virtual camera is generated for each shot.

3 A CAMERA LAYOUT PRIMER

In this paper, we focus on two main components of the camera layout (Fig. 2): framing and camera movement. For a complete list of camera layout components, please refer to the book by Arijon [1]. Framing is the artistic style of placing a subject in the shot. It can be classified into different types based on how much of the subject is included on-screen. While framing can be defined for any subject, we assume that the subject in the reference video is human, as observed in most live-action movies. Framing can be categorized into six different types based on the representative body parts of a human subject. A *close-up* (CU) includes the subject's face area only. A *medium close-up* (MCU) includes both the subject's face and shoulders. A *medium shot* (MS) includes the waist and upper parts of the subject's body. A *medium long shot* (MLS) includes the knees and upper parts of the body. A *full shot* (FS) fills the frame with the entire body of the subject, from the head to the feet. A *long shot* (LS) positions the camera farther away than a FS does, with the subject occupying only part of the frame.



(a) Framing **(b) Camera movement**

Figure 2: Camera layout components: (a) six different framing types, and (b) 13 different camera movements.

Camera movement creates dynamics in a shot. While numerous film techniques related to camera movement exist, we chose seven basic camera movements, 13 in total considering the direction. *Static* refers to a stationary camera. *Tilt* and *pan* have rotational motion about the *x-axis* and *y-axis* of the camera coordinate frame, respectively. *Crane* has rotational motion about the *x-axis* of the camera coordinate frame and translational motion along the *y-axis* of the world coordinate frame. *Orbit* has rotational motion about the horizontal axis of the world coordinate frame, resulting in an arc motion. *Track* and *dolly* have translational motion along the *x-axis* and the *z-axis* of the camera coordinate frame, respectively. With directional information included, the camera movement types are complete enough to generate virtual camera movement.

4 VIRTUAL CAMERA LAYOUT GENERATION FRAMEWORK

Our method generates a virtual camera layout by following the cinematic intention of the reference video. Given an input reference video with an arbitrary length, first we detect the shot boundaries using the method of Zhang and Wang [36]. The method is trained with the TRECVID dataset [34] for the labeling of the start and end frames of the shots in the video. It is important to segment the video into a sequence of shots because every single shot has a different camera layout, resulting in a unique visual style and emotional tone. Using the detected shot boundaries, we enter the start and end frame numbers of each shot on the shot list. Next, in the Camera Layout Analysis (CLA) stage (Section 4.1), we use a shot from the previous stage as input and utilize computer vision techniques to extract the subjects' visual features (head orientation, area, and position), the framing, and the camera movement. Additionally, by comparing the reference video and the staged 3D assets, we manually associate the subjects with the characters. All of this information above is stored as a shot list which will be used to generate a virtual camera. In the Virtual Camera Layout Generation (VCLG) stage (Section

4.2), we calculate the virtual camera position in the Toric space [25] and further optimize it to satisfy the visual constraints of the reference shot. Finally, the camera motion is generated based on the information from the shot list. The user can also fine-tune the camera position, as is done during the 3D animation work flow using 3D software. An overview of our system is shown in Fig. 3. In this work, we focus on identifying the framing and camera movement of the reference video to construct a shot list and then to generate a virtual camera layout using the shot list. Therefore, we will discuss the CLA and VCLG in detail in the following sections.

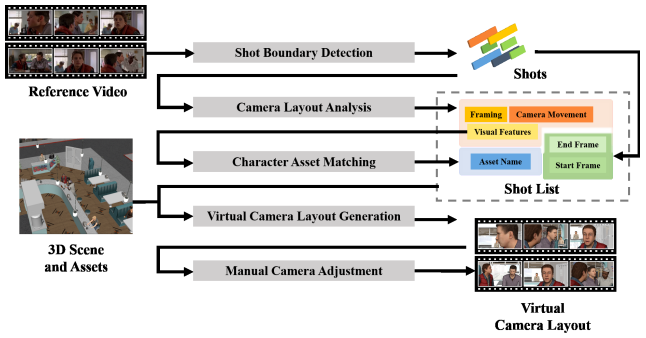


Figure 3: The proposed framework initially divides the input reference video into a sequence of shots using a shot boundary detection method. The shots are then analyzed to extract the camera layout information, whose results are stored in a shot list. Using the shot list, 3D assets, and matched characters, the initial virtual camera layout is generated in the 3D scene, which matches the semantics of the camera layout of the reference shot. The user can fine-tune the virtual camera position further if desired. The reference video on the top left are from *Back to the Future (R. Zemeckis, 1985) ©Universal Pictures.*

4.1 Camera Layout Analysis

With the detected shot boundaries for a given video, we separate the video into shots and analyze each shot to extract the camera layout.

4.1.1 Framing. When positioning the camera, there are a few important factors to consider, specifically the screen position and framing type of the subject. The framing type of the subject can be determined using the visibility of the estimated human keypoints or the skeletal parts of the subject, as described in Section 3. The screen position of the subject can be extracted using an off-the-shelf 2D human pose estimation method, LCR-Net [30]. However, in some cases, LCR-Net fails to detect or inaccurately estimate the keypoints. In addition, classifying the framing type when two or more subjects are present in a scene can lead to ambiguous results. In this case, we select as the main subject one who is front-facing for the analysis or classify the subject based on its detected region by LCR-Net [30]. We employ and fine-tune a pretrained CNN model, ResNet [18], for framing type classification, as was done in Savardi et al. [33]. Because no such dataset for the six framing types defined in Section 3 exists, we manually collected and labeled live-action movies. In total, 6180 and 697 frames were labeled for training and testing, respectively. Please refer to the supplementary document for more details. At inference time, the network classifies the framing type given an input frame.

We also estimate the head size and orientation of the subject, a process which is essential for positioning the virtual camera in the Toric space. The head size is calculated by constructing a bounding circle of the subject's head. The center coincides with the head position p^{head}, and the radius is computed as $r = ||p^{head} - p^{neck}||_2$ where p^{neck} is the position of the neck joint. The ratio between the head area and the image resolution, s_{ratio}, is then calculated using the radius r and the height and width of the image. Lastly, from the given head area, we estimate the head orientation of the subject in the camera coordinate using a method devised by Ruiz et al. [32]. The framing information, specifically the framing type, head position, head area, and the head orientation, are added to the shot list for later use.

Table 1: Correspondence for framing. Each framing type is determined based on the corresponding keypoints. The camera framing in a virtual scene is generated using the corresponding skeletal parts.

Framing	Keypoints	Skeletal Parts
CU	Eye	Head
MCU	Shoulder	Spine
MS	Hip	
MLS	Knee	Leg
FS	Ankle	Toe
LS		

4.1.2 Camera Movement. Given a shot, we analyze the camera movement by constructing a motion vector and training a feed-forward neural network as supervised learning. To construct the motion vector for a shot, we follow the method of Derue et al. [10]. A dense optical flow is initially computed for every consecutive frame of the shot. To improve the accuracy when capturing the camera motion, we mask out dynamic objects, in our case humans, using a semantic segmentation method [8]. Subsequently, we construct a N-bin histogram using the orientation of the optical flow for all frames after optical flows with low magnitudes are discarded. Here, the camera movement direction (i.e. dolly in and out) cannot be differentiated if only orientation values are used because the values do not convey spatial information. To analyze both spatial and temporal information in the entire shot, we divide the dense optical flows into nine equal sections using the rule of thirds. For each region, a histogram is computed, and all nine sections are concatenated to form a single motion vector M. M is further normalized with respect to the length of the shot, as every single shot differs in terms of the total number of frames. Using M as the input, a three-layer fully connected neural network is trained for camera movement classification. The model is trained on a synthetic dataset built using the Unreal Engine. Please refer to the supplementary document for more details. At the inference time, the network inputs M and classifies the camera movement type of the shot. The

inferred camera movement for each shot is stored in the previously generated shot list.

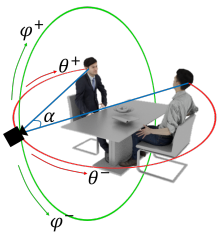
4.2 Virtual Camera Layout Generation

With a shot list consisting of the camera layout and visual features of the reference video, we generate the framing and camera movement of the virtual camera for the 3D animation scene. First, we manually match the subjects from the reference video with the names of the target virtual characters. When there are two subjects in the video, the one with better visual features (i.e., head facing the camera direction) is selected as the main subject. The corresponding matched character is considered as the main target. Using this association, the virtual camera is positioned and keyed. The following explains how the framing and camera movement of the virtual camera are determined based on the information contained in the shot list.

4.2.1 Framing. Given a shot list and 3D characters, we calculate the camera position and orientation in the 3D scene. Finding the camera position for a single character is straightforward. With the on-screen ratio s_{ratio}, the distance d from the camera to the character is computed as

$$d = \frac{W \cdot R}{2\tan(\phi/2) \cdot r}, \tag{1}$$

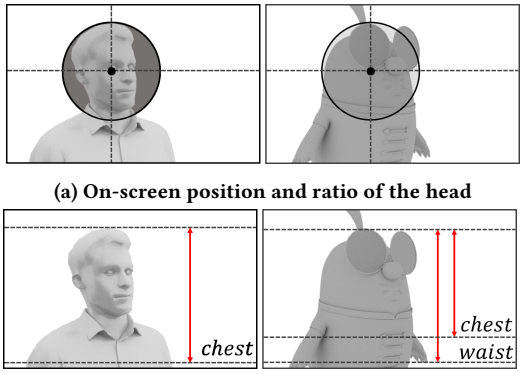
where $r = \sqrt{(W \cdot H \cdot s_{ratio})/\pi}$ is the on-screen radius, R is the actual radius of the character's head, and ϕ is the horizontal field of view of the camera. W and H represent the width and the height of the screen resolution of the camera, respectively. This distance determines a spherical manifold of potential camera positions. With a given head orientation estimated by the CLA method and the head rotation of the virtual character, a point on this manifold is determined as the final camera position.



Toric space

When two characters are present, we find the camera position in the Toric space [24, 25], where the position of a point in the space is represented by three camera-character-related parameters as shown in the left inset figure: the angle α between the two vectors from the camera to the characters and the horizontal and vertical camera angles around the characters, θ and φ, respectively. α can be calculated using the on-screen positions of the characters and the angle of view of the camera. θ and φ are estimated by matching the head orientation of the selected main target character to that of its corresponding reference subject.

Framing stylized characters. While the method described above can handle virtual human characters with normal proportions, characters appearing in a virtual scene can have different body proportions from that of humans. As can be seen in the inset figure on the right, the same on-screen position and ratio can result in different on-screen visibility outcomes due to the characters' different body proportions.



(a) On-screen position and ratio of the head

(b) On-screen visibility of body parts

Figure 4: The on-screen visibility of body parts can differ depending on the character's unique body proportions, even if the on-screen position and the on-screen ratio of the head are identical. A Character on the right column ©KYOWON.

To address this issue, we interviewed professional layout artists with more than five years of experience. The interview helped us to understand how the artist deals with mismatches between the body proportions of the subjects from the storyboard and those of the actual target characters. Three factors that concerned the artists were *visibility*, the degree to which the subject's body part is visible; *headroom*, the vertical space between the subject's on-screen head-top position and the upper boundary of the screen; and *composition*, the on-screen placement of the subject which focuses more on the horizontal arrangement.

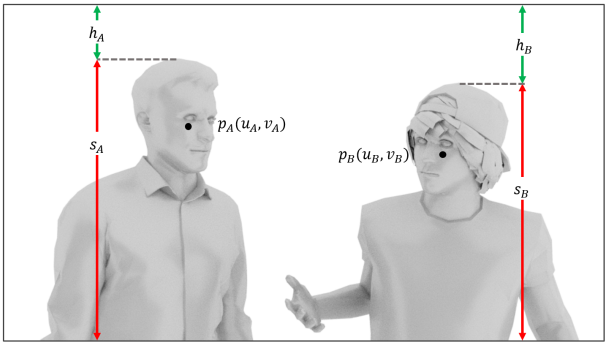


Figure 5: On-screen visibility refers to the vertical space occupied by the character on the screen, whereas headroom refers to the vertical space between the top of the character's head and the upper boundary of the screen. The horizontal on-screen midpoint is the u coordinate of the center of the two characters' on-screen positions, $p_A(u_A, v_A)$ and $p_B(u_B, v_B)$.

We adopt the factors mentioned by the artists as constraints to layout the stylized characters. Similar to how different keypoints are used to classify different framing types, different skeletal parts of the target are used to calculate the camera-character distance that

ensures the desired level of visibility. We optimize the normalized on-screen position $p_A(u_A, v_A) \in [0,1]^2$ and $p_B(u_B, v_B) \in [0,1]^2$ of target characters A and B to determine α in the Toric representation, such that the resulting layout satisfies the constraints given by the reference subjects A' and B' specified on the shot list. First, we modify the definitions of some of the notations used in Equation 1. The vertical length from the character's head to a specific part serves as the actual size R, and the ratio between the vertical pixel length from the on-screen head position to the lower boundary of the screen and the height of the screen serves as the desired on-screen ratio s_{ratio}. In the *LS* case, which has the same corresponding skeletal parts as a *FS* but covers a greater area of the scene, we subtract an offset of 0.3 from the desired on-screen ratio. Thus, the target character is captured in a smaller area of the screen, leaving more room for the background. Given that the on-screen ratio is now a length ratio instead of an area ratio, the equation for the on-screen pixel length r is modified as $r = H \cdot s_{ratio}$. Using these modified definitions and equations, the distance from the camera to the body feature of the character is recalculated.

In order to follow the framing types specified on the shot list, we add a visibility term, as follows:

$$E_V = |s_A - s_{A'}| + |s_B - s_{B'}|, \tag{2}$$

where s_A and s_B denote the on-screen visibility of the target characters in the current layout (Fig. 5), which can be calculated by the function derived from Equation 1. $s_{A'}$ and $s_{B'}$ are the desired visibility of the target characters according to the framing type.

To ensure that the resulting layout has a reasonable headroom size similar to that of the reference, we add a headroom term.

$$\begin{aligned} h &= 1 - v - r_{head}, \\ E_H &= |\min(h_{A'}, h_{B'}) - \min(h_A, h_B)|. \end{aligned} \tag{3}$$

Here, $h_A, h_B \in [0,1]$ correspondingly denote the headroom areas of characters A and B, respectively, and $h_{A'}, h_{B'} \in [0,1]$ are likewise the headroom areas of reference subjects A' and B', respectively. $v \in [0,1]$ is the normalized vertical coordinate of the on-screen position of a target character whose radius of the on-screen head size is r_{head}. Note that there may be a case where the height difference between the target characters is not similar to or even opposite from that between the reference subjects. Therefore, instead of optimizing the headroom sizes of both characters, we select a smaller headroom area among each of the target characters and the reference subjects for the calculation. This prevents a taller character's head from being unwantedly cut-off by the screen boundary.

Finally, we add a horizontal arrangement term to ensure that the on-screen horizontal arrangement of the target characters is similar to that of the subjects in the reference. For instance, if the subjects from the reference are projected on the left half of the screen, this term prevents new on-screen positions from moving towards the right half.

$$E_M = |(u_A + u_B)/2 - (u_{A'} + u_{B'})/2|. \tag{4}$$

Here, u_A and u_B are the normalized horizontal coordinates of the on-screen positions of characters A and B, respectively, and $u_{A'}$ and $u_{B'}$ are likewise the normalized horizontal coordinates of the on-screen positions of the reference subjects A' and B', respectively.

The final optimization can be expressed as a linear combination of Equations 2 to 4 and the camera roll term E_{roll}:

$$\arg\min_{p_A, p_B} \omega_1 E_V + \omega_2 E_H + \omega_3 E_M + \omega_4 E_{roll}. \tag{5}$$

Here, $\omega_1, \omega_2, \omega_3$, and ω_4 are the weights for each term and are set to 0.7, 1.5, 0.7, and 1.5, respectively. E_{roll} penalizes the camera's right axis tilt to prevent camera roll. We used the SLSQP algorithm for optimization.

4.2.2 Camera Movement. Once the framing is determined, the camera movement is generated by interpolating the camera placements at the start and end keyframes of the shot. However, the initial placement often does not satisfy the constraints of the camera movement rules (i.e., fixed position for panning). Furthermore, a linear interpolation would not be applicable because our camera movements include complex types, such as an orbit that requires arc motion. Thus, we employ a rule-based approach to modify the camera movement. It recalculates the position at the end frame or adds keyframes in the middle so that the camera can correctly follow the conventional rules defined in Fig. 2. The magnitude of the camera movement is estimated according to the change of the camera-character relationship (i.e., the relative position and orientation) at the start and end frames of the shot. Please refer to the supplementary document for more details.

5 IMPLEMENTATION

Our system was implemented in *Python* and *QT*, and the VCLG application was built on top of Autodesk Maya 2018. This application allows the artist to adjust the shot list and the characters' properties further if necessary. For more information on the application, please refer to the supplementary document. The computation time for the CLA method is 0.368 seconds per frame for a video of resolution 576×420 on a machine with an Intel Core i7-5820 processor running at 3.3GHz, 32GB of memory, and a NVIDIA GeForce GTX 980 Ti graphics card. A typical shot lasts around five seconds. In the VCLG, the computation time is 6.85 seconds per optimization.

6 EVALUATION

To validate our method, we present visual results from our method along with the findings from an extensive user evaluation. The user study utilized various stylized characters to compare the following: layouts generated using our method, layouts generated using the Toric space method [25], and layouts created by a professional artist. In addition, we conducted a user study of the effectiveness of the entire system with novice users and artists.

6.1 Qualitative Evaluation for the Virtual Camera Layout

Using our system, we automatically generated virtual camera layouts for two datasets, *Back to the Future* [12] and *Counseling*, which consist of a reference video and a 3D scene with characters staged in advance. The *Back to the Future* video clip is 48 seconds long and has 12 mostly static shots. The length of the *Counseling* video is 34 seconds and it contains seven shots. The shots include static, track, dolly, orbit, and tilting camera movements. The results are shown in Fig. 6 and Fig. 7 for *Back to the Future* and *Counseling*,

respectively. For more results, please refer to the supplementary video.

As shown in Fig. 6a, the framing results resemble the reference shots. With the visual features identified only at the start and end frame along with the classified camera movement type, the camera movement of the reference shot is successfully reproduced for the 3D scene, as shown in Fig. 6b. We also compared the results with the artist's layout provided by previous work [14], as shown in Fig. 8. We find from the layout by the artist that they focused on the framing and headroom of the character. The Toric space method, which is the initial position of the virtual camera before optimization by our method, fails to deliver the cinematic intention of the reference shot in some cases despite the close resemblance of the target characters to the reference subjects, as shown in the bottom row of Fig. 8. This may stem from a slight difference in the characters' heights, animation, and/or staging. The method proposed in Jiang et al. [22] is also affected by a similar limitation given that the method is learned from human characters with Toric representation. In contrast, our method was able to handle slight discrepancies between the reference and target scenes through optimization and generate a layout similar to that from the artist.

For the *Counseling* dataset, we show the results with two different types of characters. The first type is human characters (Fig. 7b), and the second type is robot characters with body proportions different from those of humans (Fig. 7c). Note that our method works well in both cases, as shown in Fig. 7b and Fig. 7c. In contrast, when only using the Toric space method to position the camera, the virtual camera layout is vastly different from the reference video, as shown in Fig. 7d, when the character does not have human-like body proportions.

Additionally, we investigated how each term in Equation 5 affects the virtual camera layout of exaggerated characters, as shown in Fig. 9. Without E_V, the framing type does not match the framing of the reference shot. While the composition and the framing type match without E_H, the headroom is vastly different from the reference video. E_M has an effect on the horizontal composition of the character. Hence, without E_M, the positions of the characters can be shifted to the right or left.

6.2 User Study: Comparison of Virtual Camera Layout Results

To evaluate the effectiveness of our virtual camera layout, we conducted a user study in which the participants were asked to evaluate and compare the results from the following methods: our method, the Toric space method, and the virtual camera layout created by the artist. We prepared 56 static shots, which were generated from 14 reference shots using 4 different pairs of target characters (Fig. 10b). The reference subjects had average human body proportions, whereas the target characters had various exaggerated body proportions, as shown in Fig. 10a.

Fifteen participants (6 males and 9 females; ages: 24 to 32; average age: 28.1; three artists with 3 to 5 years of 3D animation experience) filled out a questionnaire consisting of two parts: a subjective rating for user satisfaction and a 2-alternative forced choice (2AFC) test after examining the provided shots. For the subjective rating, the participants were presented with a pair of shots consisting of a reference shot and the corresponding shot; these were randomly selected among the three methods. They were asked to score on a 5-Likert Scale how satisfied they were with the framing result. For the 2AFC test, framing results produced by the two different methods were presented in random order, along with the corresponding reference shots. The participants were asked to choose the preferred result.

Table 2: User study results from subjective rating of framed layouts generated by three different methods. For each method, 56 shots were rated by 15 participants on scale of 1 (low) to 5 (high). Total of 840 ratings were made per method. *% positive* refers to percentage of layouts that were rated above 3 (neutral).

	1	*2*	*3*	*4*	*5*	*% positive*
Artist	27	64	104	292	353	77%
Ours	37	57	134	295	317	73%
Toric	110	164	169	227	170	47%

The results in Table 2 verify that the virtual camera layouts generated by our method (73%) were more faithful reproductions of the reference camera layouts than were the virtual camera layouts generated by the Toric space method (47%), and similar in quality to the virtual camera layouts created by the artist (77%). For statistical analysis, we applied the Kruskal-Wallis rank sum test. The results showed that there exist significant differences among the ratings of the layout methods ($F(2, 2517) = 126.59, p < 0.05$). A post-hoc analysis using the Dunn test with Bonferroni correction revealed that, as shown in Fig. 11, ratings on layouts by the artist and our method were significantly different from those by the Toric space method ($p = 0.00$), whereas ratings between these first two methods were not significantly different ($p = 0.19$). Fig. 12 shows average ratings for the layouts created by each method with respect to each character pair. We applied the Kruskal-Wallis test and the Dunn test to all of the sub-data. These tests showed that the Toric space method scored significantly lower ($p<0.05$) than the other two methods for all pairs ($p<0.001$ for Pair B and D).

The pairwise comparisons reported in Table 3 show that our method was preferred over the Toric space method by 74%. Compared to the layouts created by the artist, those created by our method were preferred by 47%, showing a similar preference. These results verify that our method can generate layouts for a wide range of stylized characters with different body proportions while faithfully reproducing the original camera layout of the reference shot. In addition, the layouts generated by our method are comparable to the virtual camera layouts created by the artist.

6.3 User Study: Replicating a Reference Video

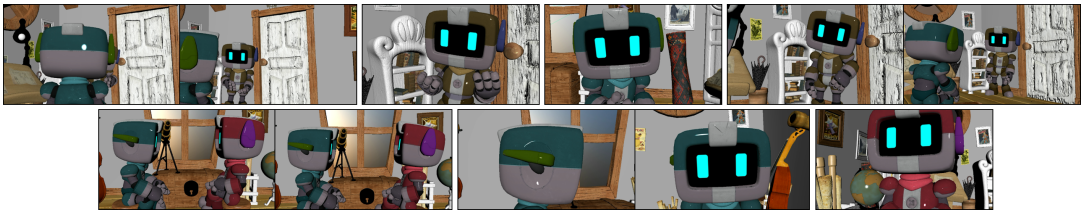
We conducted an additional user study to validate the effectiveness of the entire system. Fifteen participants were recruited (10 males and 5 females; ages: 25 to 46; average age: 32.9) and given the task of positioning and moving the virtual camera according to the reference video. Nine of the participants were novice Maya users with limited experience in 3D animation. The remaining participants were professional artists in a related industry, three

(a) Static shots (b) A shot with the camera panning left

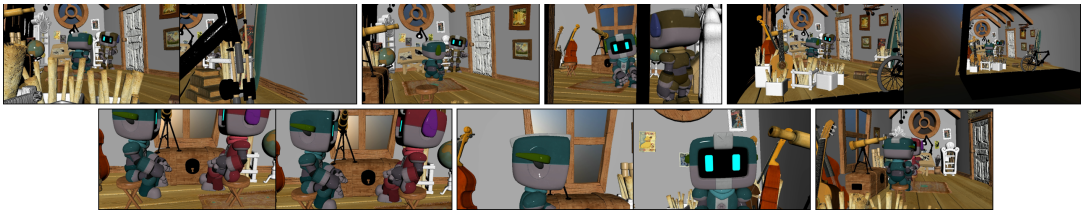
Figure 6: Results from our method for shots from *Back to the Future* using 3D scenes from Galvane et al. [12]. Here, (a) shows the first frames of the shots from the reference video, and (b) shows the panning movement following *Goldie*. The frames in the first rows of (a) and (b) are from *Back to the Future (R. Zemeckis, 1985) ©Universal Pictures*.

(a) Reference video

(b) Our method with human characters



(c) Our method with stylized characters



(d) Toric space method with stylized characters

Figure 7: Generated virtual camera layouts from the *Counseling* dataset. Here, (a) shows reference shots, (b) and (c) show generated camera layouts using our method with similar as well as different body proportions to/from those of humans, and (d) shows results from the Toric space method. Characters and background assets, respectively, in (c) and (d): *©Mix and Jam, ©Jeremy Vikery and Alex Mateo*.

of whom had more than 10 years experience. Three of the artists were from an animation studio (ASA1, ASA2, ASA3); the remaining three were from a VFX studio (VA1, VA2, VA3). Four out of the six artists reported that they used Maya on a regular basis. The user study, on average, took approximately 50 minutes.

The participants were asked to replicate the camera layout of a reference video into a Maya scene (i) without initialization of

Figure 8: Comparison with the artist's layout [14] of two shots from *Back to the Future*. The reference frames are from *Back to the Future (R. Zemeckis, 1985) ©Universal Pictures.*

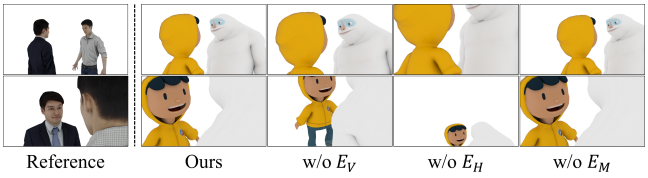
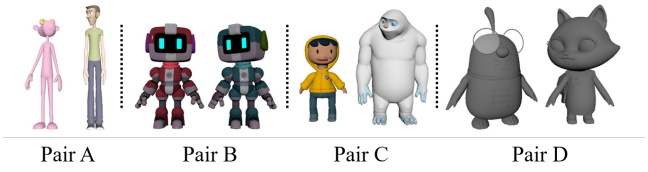
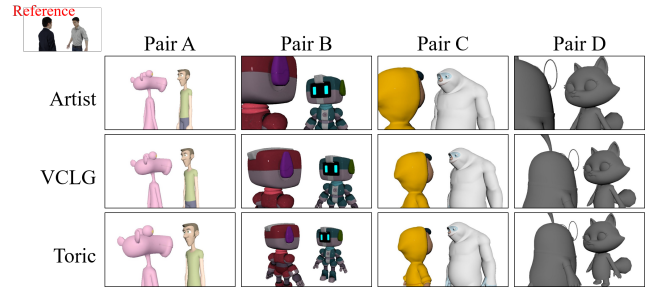


Figure 9: Ablation study for optimization terms. E_V, E_H, and E_M are the visibility, the headroom, and the horizontal arrangement term, respectively. Yellow hoodie character by ©Mario Nagamura.



(a) Four different pairs of target characters used in user study.



(b) Virtual camera layouts produced by different methods.

Figure 10: Using virtual character pairs from (a), we compare virtual camera layout created by artist, our method, and the Toric space method reflecting the reference shot (b). Both reference subjects are in *MS*. Characters in Pair A and the left character in Pair C: *©Karim Kashefi, ©AnimSchool, ©Mario Nagamura*. Character pairs in Pair B and Pair D: *©Mix and Jam, ©KYOWON*

the virtual camera, as in the conventional layout process and (ii) with the initialized virtual camera using our method. For this task,

Figure 11: Average rating scores for layouts generated by each method.

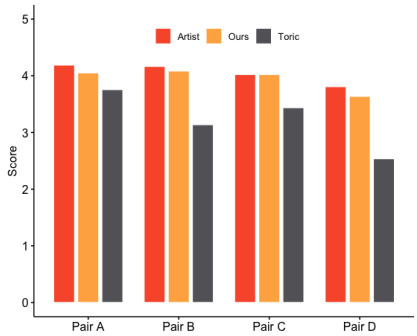


Figure 12: Average rating scores for layouts generated for each character pair.

Table 3: User preference results for layouts generated by three different methods: Artist, our method, and the Toric space method. Each cell of table shows % of row preferred over column.

row > column	*Artist*	*Ours*	*Toric*
Artist	-	53%	74%
Ours	47%	-	74%
Toric	26%	26%	-

we used the *Counseling* dataset and measured the time the participant spent to replicate the reference video. Before starting the experiment, the participants were asked to fill out a demographic questionnaire, followed by a brief explanation of our system. We asked the participants to use the virtual camera to replicate the layout observed in the reference video as closely as possible. The order of replication with or without using the initialized virtual camera layout was randomly selected. During the task, the participants actively referred to the reference video and were allowed to seek help from the proctor if needed. Once the task is completed, the participants were asked to fill out a survey about the usability of the system.

Overall, the time performance was greatly improved, as evidenced by the 54.9% decrease in average time spent composing the virtual camera layout (without initialization using our system: 12 min 36 sec; with initialization using our system: 6 min 56 sec). For the novice users, the average time to replicate the reference

video decreased by 52.4% (without initialization using our system: 14 min 39 sec; with initialization using our system: 6 min 58 sec). The time for the professional artists to replicate the reference video decreased by 27.9% (without initialization using our system: 9 min 31 sec; with initialization using our system: 6 min 52 sec).

6.3.1 Usability Survey and User Observation. The survey consists of six statements, and the participants rated how much they agreed with each statement on a five-point Likert scale. This survey checks whether the overall system and our GUI help to alleviate the burden of and lower the barrier of entry when positioning the virtual camera. The average scores of the survey were 3.47 and 4.74 for professional artists and novice users, respectively. These results show that the novice users found our system helpful when generating the camera layout, whereas the professional artists were more neutral regarding its usability.

We attribute this to familiarity differences with regard to animation tools between artists and novice users. As we observed throughout the experiment, the novice users experienced some difficulty when navigating the native interface of Maya, especially when searching for corresponding target characters. Our GUI provides a shot-list-like interface that allows users quickly to locate target characters and navigate shots. Furthermore, the novice users had difficulty controlling the camera and were uncertain about their work when replicating the layout. As one novice user commented after replicating the layout manually, *"even if I kept spending more time revising the camera by myself, the results did not improve that much."* Our system provides the initial layout that best follows the cinematic intention of the reference. Thus, compared to manual work from scratch, the initial layout effectively mitigates the effort needed when replicating the reference.

6.3.2 Interview of Professional Artists. The general consensus we found from the artists interview was that the system was useful because it utilizes a shot list to generate an initial virtual camera layout for animation. ASA2 said he would *"definitely use this system"* if he had to replicate a reference video. Additionally, ASA3 commented that the system would be very useful when an artist has to work on hundreds of shots, which is common during the production of a TV series. The ASA group further unanimously commented that if a story reel can be similarly analyzed to generate a virtual camera layout, it would be very helpful in the animation studio. In addition, VA3 mentioned that our system would be very useful for directors, who have little or no knowledge of 3D animation software, allowing them to previsualize the layout using a reference video. On the other hand, both the VA group and the ASA group felt that generating a camera layout based on classified framing and camera movements may limit the artistic style of a virtual camera layout. Thus, they would always prefer to retouch the virtual camera's position and orientation. Nevertheless, the artists were generally optimistic about the direction of our approach.

7 DISCUSSION

7.1 Use Cases of the System

Our system can be used by both novice and professional users. From the user study and interview, we found that many novice users had difficulty placing the virtual camera in a conventional environment. In contrast, with the initial position already generated by the system, the users comfortably began to manipulate the virtual camera and move it to a desired position with a considerable time improvement. This implies that any person with little knowledge about 3D animation tools can rapidly test out different virtual camera layouts in the previsualization stage. For professional use, the proposed system can be highly instrumental for TV cartoon animations, during which artists must work on many shots regularly as the series continues. For example, scenes with dialogues consist of multiple shots with short-term intervals. These shots are often similar in terms of their layouts, but as the target character changes after each shot, continuing requires animators manually to locate the frame and place the camera based on the target character one by one. With our system, using the prepared shot list containing the shot information, such repeated work can be expedited, allowing more time for animators to focus on the details of the layout.

7.2 Limitations and Future Work

While our method can successfully generate a virtual camera layout using a reference video, it has certain limitations. Since the CLA method relies on the performance of a previous computer vision algorithm [10, 30], the classification often fails for videos with over and under exposed frames. In most misclassified framing task, the framing type differs within a single scale. With regard to camera movement classification, most misclassified results stem from similar camera motions with the same direction. To remedy these instances, we provided a GUI for the user, allowing them easily to correct misclassified information. We expect the classification performance of the camera layouts to improve in the future, since various computer vision algorithms are being actively studied. In addition, our method considers only the relationship between the camera and the characters. Thus, unexpected occlusion or cut-off of background assets by the screen boundary may occur. Consideration of occlusion and staging of assets [6, 26] can be an interesting future work. Lastly, we defined the camera layout in the simplest way and assumed a single type of camera movement. Many live-action movies convey a unique style of camera layout that is sometimes hard to define as a simple type. While our method focuses more on the framing of stylized characters, incorporating an example-based camera control as in Jiang et al. [22] would allow the analysis and reproduction of a greater variety of camera motions.

An interesting direction for future work will be to directly analyze a storyboard or a story reel. Hand drawn story reels are often used in animation studios. Therefore, analyzing such a story-reel to extract camera information can be instrumental in the placement of virtual cameras. We envision that layout artists will greatly benefit from such a system because the process can be directly integrated with the current animation pipeline. We believe that our method can further foster new research ideas for interactive and intuitive camera manipulation techniques for stylized characters, an area that has largely been ignored in the community.

8 CONCLUSION

This paper introduces a method to automatically generate a virtual camera layout in a 3D scene using a reference video. To achieve

this task, the method classifies the camera layout and estimates visual features of subjects in the reference video. The resulting information is then stored in the form of a shot list, with which we replicate the camera layout of the shots in the 3D scene with consideration of the virtual character's framing type. This enables our approach to be applicable to 3D scenes that contain characters with exaggerated as well as human-like proportions. From user studies, we confirmed that the results of our method are comparable to those of virtual camera layouts composed by an artist. In addition, using our system, both professional artists and novice users required less time to replicate the camera layout of a reference video in a 3D scene than was needed to position the virtual camera from scratch. The artists reported that the automatically generated shot list can be instrumental in creating an initial virtual camera layout. Moreover, the system can be useful for novice users or directors who do not have extensive knowledge of 3D animation software.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their valuable comments; Junghee Kim and his colleagues at MOTIF for the support and helpful discussion about the animation layout; and Haemin Kim, Nicolas Nghiem, and Allen Kim for participating as actors for the reference video. This research is supported by Ministry of Culture, Sports and Tourism and Korea Creative Content Agency (Project Number: R2020040180).

REFERENCES

[1] Daniel Arijon. 1991. *Grammar of the film language.* Silman-James Press.

[2] Amirsaman Ashtari, Stefan Stevšić, Tobias Nägeli, Jean-Charles Bazin, and Otmar Hilliges. 2020. Capturing Subjective First-Person View Shots with Drones for Automated Cinematography. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH)* 39, 5, Article 159 (Aug. 2020), 14 pages. https://doi.org/10.1145/3378673

[3] William Bares, Scott McDermott, Christina Boudreaux, and Somying Thainimit. 2000. Virtual 3D camera composition from frame constraints. In *Proceedings of the eighth ACM international conference on Multimedia.* ACM, 177–186.

[4] Sergio Benini, Luca Canini, and Riccardo Leonardi. 2010. Estimating cinematographic scene depth in movie shots. In *2010 IEEE International Conference on Multimedia and Expo.* IEEE, 855–860.

[5] Subhabrata Bhattacharya, Ramin Mehran, Rahul Sukthankar, and Mubarak Shah. 2014. Classification of cinematographic shots using lie algebra and its application to complex event recognition. *IEEE Transactions on Multimedia* 16, 3 (2014), 686–696.

[6] Ludovic Burg, Christophe Lino, and Marc Christie. 2020. Real-time Anticipation of Occlusion for Automated Camera Control in Toric Space. In *Computer Graphics Forum.* Wiley Online Library.

[7] Luca Canini, Sergio Benini, and Riccardo Leonardi. 2013. Classifying cinematographic shot types. *Multimedia tools and applications* 62, 1 (2013), 51–73.

[8] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. 2018. Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation. In *ECCV.*

[9] Marc Christie, Patrick Olivier, and Jean-Marie Normand. 2008. Camera control in computer graphics. In *Computer Graphics Forum*, Vol. 27. Wiley Online Library, 2197–2218.

[10] François-Xavier Derue, Mohamed Dahmane, Marc Lalonde, and Samuel Foucher. 2017. Exploiting Semantic Segmentation for Robust Camera Motion Classification. In *International Conference Image Analysis and Recognition.* Springer, 173–181.

[11] David K Elson and Mark Riedl. 2007. A lightweight intelligent virtual cinematography system for machinima production. (2007).

[12] Quentin Galvane, Marc Christie, Chrsitophe Lino, and Rémi Ronfard. 2015. Camera-on-rails: automated computation of constrained camera paths. In *Proceedings of the 8th ACM SIGGRAPH Conference on Motion in Games.* ACM, 151–157.

[13] Quentin Galvane, Christophe Lino, Marc Christie, Julien Fleureau, Fabien Servant, Fran¸ois-louis Tariolle, and Philippe Guillotel. 2018. Directing cinematographic drones. *ACM Transactions on Graphics (TOG)* 37, 3 (2018), 1–18.

[14] Quentin Galvane, Rémi Ronfard, Christophe Lino, and Marc Christie. 2015. Continuity editing for 3D animation. In *Twenty-Ninth AAAI Conference on Artificial Intelligence.*

[15] Christoph Gebhardt, Benjamin Hepp, Tobias Naegeli, Stefan Stevsic, and Otmar Hilliges. 2016. Airways: Optimization-based Planning of Quadrotor Trajectories according to High-Level User Goals. In *SIGCHI Conference on Human Factors in Computing Systems* (San Jose, CA) *(CHI '16).* ACM, New York, NY, USA.

[16] Christoph Gebhardt, Stefan Stevsic, and Otmar Hilliges. 2018. Optimizing for Aesthetically Pleasing Quadrotor Camera Motion. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH)* 37, 4, Article 90 (2018), 11 pages.

[17] Muhammad Abul Hasan, Min Xu, Xiangjian He, and Changsheng Xu. 2014. CAMHID: Camera motion histogram descriptor and its application to cinematographic shot classification. *IEEE Transactions on Circuits and Systems for Video Technology* 24, 10 (2014), 1682–1695.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition.* 770–778.

[19] Chong Huang, Fei Gao, Jie Pan, Zhenyu Yang, Weihao Qiu, Peng Chen, Xin Yang, Shaojie Shen, and Kwang-Ting Tim Cheng. 2018. Act: An autonomous drone cinematography system for action scenes. In *2018 IEEE International Conference on Robotics and Automation (ICRA).* IEEE, 7039–7046.

[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 Conference on Computer Vision and Pattern Recognition.* 4244–4253.

[21] Chong Huang, Zhenyu Yang, Yan Kong, Peng Chen, Xin Yang, and Kwang-Ting Tim Cheng. 2019. Learning to Capture a Film-Look Video with a Camera Drone. In *2019 International Conference on Robotics and Automation (ICRA).* IEEE, 1871–1877.

[22] Hongda Jiang, Bin Wang, Xi Wang, Marc Christie, and Baoquan Chen. 2020. Example-Driven Virtual Cinematography by Learning Camera Behaviors. *ACM Trans. Graph.* 39, 4, Article 45 (July 2020), 14 pages. https://doi.org/10.1145/3386569.3392427

[23] 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.

[24] Christophe Lino and Marc Christie. 2012. Efficient composition for virtual camera control. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation.* Eurographics Association, 65–70.

[25] Christophe Lino and Marc Christie. 2015. Intuitive and efficient camera control with the toric space. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 82.

[26] Amaury Louarn, Marc Christie, and Fabrice Lamarche. 2018. Automated staging for virtual cinematography. In *Proceedings of the 11th Annual International Conference on Motion, Interaction, and Games.* ACM, 4.

[27] Tobias Nägeli, Lukas Meier, Alexander Domahidi, Javier Alonso-Mora, and Otmar Hilliges. 2017. Real-time Planning for Automated Multi-View Drone Cinematography. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH).*

[28] Roberto Ranon and Tommaso Urli. 2014. Improving the efficiency of viewpoint composition. *IEEE Transactions on Visualization and Computer Graphics* 20, 5 (2014), 795–807.

[29] Anyi Rao, Jiaze Wang, Linning Xu, Xuekun Jiang, Qingqiu Huang, Bolei Zhou, and Dahua Lin. 2020. A Unified Framework for Shot Type Classification Based on Subject Centric Lens. In *The European Conference on Computer Vision (ECCV).*

[30] Grégory Rogez, Philippe Weinzaepfel, and Cordelia Schmid. 2019. LCR-Net++: Multi-person 2D and 3D Pose Detection in Natural Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2019).

[31] Remi Ronfard, Vineet Gandhi, and Laurent Boiron. 2015. The prose storyboard language: A tool for annotating and directing movies. (2015). arXiv:1508.07593

[32] Nataniel Ruiz, Eunji Chong, and James M. Rehg. 2018. Fine-Grained Head Pose Estimation Without Keypoints. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops.*

[33] Mattia Savardi, Alberto Signoroni, Pierangelo Migliorati, and Sergio Benini. 2018. Shot Scale Analysis in Movies by Convolutional Neural Networks. In *2018 25th IEEE International Conference on Image Processing (ICIP).* IEEE, 2620–2624.

[34] Alan F Smeaton, Paul Over, and Wessel Kraaij. 2006. Evaluation campaigns and TRECVid. In *Proceedings of the 8th ACM international workshop on Multimedia information retrieval.* ACM, 321–330.

[35] Hui-Yin Wu, Francesca Palù, Roberto Ranon, and Marc Christie. 2018. Thinking Like a Director: Film Editing Patterns for Virtual Cinematographic Storytelling. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 14, 4 (2018), 81.

[36] Chi Zhang and Weiqiang Wang. 2012. A robust and efficient shot boundary detection approach based on fisher criterion. In *Proceedings of the 20th ACM international conference on Multimedia.* ACM, 701–704.