A NOVEL FRAMEWORK AND DESIGN METHODOLOGIES FOR OPTIMAL ANIMATION PRODUCTION USING DEEP LEARNING

By

Zixiao Yu

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Electrical Engineering – Doctor of Philosophy

2023

ABSTRACT

In this dissertation, we introduce an innovative automatic animation production framework and its modules aimed at overcoming the inherent challenges in traditional animation production, including the requirement for a substantial investment of time, resources, and expertise. We have structured the production continuum into four distinct but interrelated segments: *Action List Generation*, *Stage Performance*, *Auto Cinematography*, and *Video Generation*.

In the realm of *Stage Performance*, we propose an innovative Real-time Phoneme Recognition Network (RealPRNet) for real-time lip-sync animation generation. Designed to correlate incoming audio input with the corresponding viseme (visual representation of a phoneme) in real-time, RealPRNet employs a multifaceted approach incorporating spatial, discrete temporal, and cohesive temporal features of the audio data. Significantly, the architecture of RealPRNet incorporates a stacked long short-term memory (LSTM) block and exploits long short-term features to optimize phoneme recognition. RealPRNet offers substantial improvements in reducing phoneme error rate and increasing in visual performance.

Next, we delve into the text-to-animation framework (T2A), a pivotal construct in our research. While extant auto cinematography paradigms largely hinge on rudimentary cinematographic principles (aesthetics), T2A offers a more nuanced approach. It amalgamates the input script's visual fidelity with the resultant video's aesthetic compliance, by concurrently harnessing both fidelity and aesthetic models. Our evaluative studies, conducted with animators, attest to the superior quality of the resulting animations vis-à-vis methodologies solely reliant on aesthetic models, while reducing the workload of manual animation production by approximately 74%.

However, it is germane to note that *Auto Cinematography*, when solely premised on canonical rule-based optimization, often falls short of audience expectations. we introduce

RT2A (Reinforcement learning-based Text to Animation), a pioneering architecture that synergistically merges reinforcement learning techniques into the realm of automated film direction. Within the RT2A framework, each choice concerning camera orientation and positioning is systematically documented and integrated into subsequent training cycles for the reinforcement learning agent. Utilizing a carefully constructed reward function, the algorithm is guided toward the optimization of camera configurations that emulate the stylistic decisions commonly employed by human directors. Quantitative analysis reveals that RT2A can achieve 50% improvement in audience-approved camera placements and an 80% increase in fidelity over benchmark camera placements.

Furthermore, due to the recent trend of personalized animation content influenced by user-generated interactions, the unpredictability and complexity of open-world settings pose a substantial challenge for automated cinematography techniques. To tackle this, we extend our framework along with an adaptive automated cinematography approach based on Generative Adversarial Networks (AACOGAN). According to empirical analyses, AACOGAN significantly outperforms existing methods in capturing the dynamism of open-world interactions. The efficacy of AACOGAN is underscored by a 73% improvement in the correlation between user behaviors and camera trajectory adaptations, as well as a marked elevation—up to 32.9%—in the quality of multi-focus scenes. These metrics offer compelling evidence that our methodology not only refines cinematographic outputs but also considerably augments user immersion within these complex virtual environments.

ACKNOWLEDGMENTS

First and foremost, I wish to convey my profound gratitude to my advisor, Dr. Jian Ren, for his invaluable guidance and unwavering support throughout my academic journey and personal endeavors. The opportunity to have Dr. Ren as a mentor has been an unparalleled privilege. My heartfelt appreciation also extends to Dr. Haohong Wang, whose expertise and counsel have been instrumental both academically and personally. I am further indebted to TCL Research America for their generous support, without which this project would not have come to fruition.

I am also obliged to acknowledge my esteemed colleagues and cohort members—Chen Mi, Lin Sun, Duo Lv, Xinyi Wu, Chao Wang, and Kai—for their indispensable contributions to the project's development. Their assistance in feedback sessions and unwavering moral support have been invaluable to the completion of this work.

I extend my sincere appreciation to Dr. Tongtong Li, Dr. Sijia Liu, and Dr. Wen Li for graciously agreeing to serve on my committee. Their constructive feedback and insights have greatly enriched this dissertation. A special note of gratitude is reserved for Dr. Tongtong Li, whose guidance and concern for my well-being have been particularly meaningful.

Finally, it would be remiss not to acknowledge the unfaltering love and support from my family, particularly my parents. Their enduring belief in my capabilities and their emotional sustenance have been the bedrock upon which I have built my academic and personal life.

TABLE OF CONTENTS

Chapter 1

Introduction

1.1 Overview

After centuries of extensive and rapid evolution, film, often referred to as the "seventh art", has firmly established itself as an integral component of contemporary entertainment. With the advancement of computer graphics (CG) technology, the creation of film art is no longer limited to reality. The use of computers to create animated films in virtual space has become an essential part of the film industry. In recent years, with the rapid development of the game industry, most games also contain a significant amount of CG animations to introduce the characters and background of the game and promote the development of the story. These animations provide a more immersive experience to the viewers and players. However, the production of animated films is an extremely complex and time-consuming endeavor that requires a significant amount of effort, expertise, and cooperation from experts in different fields to complete the entire production process. Over the past few decades, a considerable amount of research has been dedicated to games and movies to streamline this process, either conserve resources or reduce production periods. Particularly noteworthy in these efforts are the use of game engines to build virtual scenes, quickly generate a real-time CG animation, or create an animation video that relies on the game's existing scenes and resources (i.e. Animations based on *Minecraft*). Based on these concepts, various types of

software have been developed for different stages of animated film production (i.e. *iClone* and *Wolf3D*), which significantly reduce the associated staff and resource requirements for high-quality content creation. The default users of these software programs are experts in the respective fields of animation production with sufficient knowledge about the empirical rules and established conventions. It takes a significant amount of time and effort for the user to learn and master the software. Therefore, there is still an insurmountable barrier for amateurs to create their own animations even with these tools.

However, with the rapid expansion of the internet industry in recent decades, the creation and distribution of content are no longer monopolized by a few large corporations. Among the forms of content creation, video is an easier way to capture the audience's attention than text. Anyone can publish their newly created video clips on YouTube with a few button clicks. Of course, as one of the best ways to tell a story, there is no exception for animated videos. Even though most of them are not comparable to the high-quality productions produced in the visual performance by companies such as *Pixar* and *Netflix* for various reasons, many of them demonstrate the creations and impressive plots that can compete with these famous animations. We believe there is a significant advantage in helping textual content creators by visualizing their stories by animating their works. Motivated by these insights, we have developed the 3D automatic animation production framework, and it ideally operates by automatically creating the corresponding 3D animation with just a standard script input. Users are no longer required to have extensive knowledge of each stage in the production of the animation, which can further reduce the consumption of time and resources.

We initiated our research by collaborating with animators and directors to deconstruct the entire animation production process. This collaborative effort led to the segmentation of the process into several discrete modules: *Action List Generation, Stage Performance,*

Auto Cinematography, and *Video Generation*. Each module is underpinned by distinct technologies, with outputs that modify the original script format. This modular architecture allows users to pause and manually adjust the output of any stage, should they find it unsatisfactory. Given that the evolution of plots in animation is typically character-centric, the *Action List Generation* module transforms the script into a chronological list of actions. These range from movements and dialogues to interactions. Each entry in this list carries comprehensive information, enabling characters to perform the specified action in the virtual environment. The *Stage Performance* ingests this action list and directs virtual characters to execute actions in a selected scene sequentially. The *Auto Cinematography* module has the responsibility of capturing these performances in three dimensions and translating them into two-dimensional frames through optimal virtual camera placements. Finally, the *Video Generation* module crafts the ultimate animation video using the optimized camera configurations provided by the *Auto Cinematography* module.

Two modules, in particular, significantly influence the animation's final quality: the lip-sync speech animation within the *Stage Performance* and *Auto Cinematography*. Precise lip-syncing is paramount as viewers instinctively focus on the speaker's mouth in animated conversations. Any inconsistencies here can drastically reduce immersion. Meanwhile, optimal camera placement necessitates considerable expertise in cinematography. Existing auto-cinematography solutions either produce subpar results or struggle to adapt to diverse scenarios. In modern virtual environments, especially those with open-world designs, users have expressed a growing interest in crafting and recording bespoke narratives. The unpredictability of these scenarios makes it challenging to employ conventional auto-cinematography techniques. To address this, we propose an innovative methodology based on Generative Adversarial Networks (GAN) for auto-cinematography. Alongside this, we

introduce novel evaluation criteria, seeking to enhance adaptability in line with evolving technological demands.

This dissertation delves deep into these challenges, offering insights by amalgamating techniques from various disciplines.

1.2 Summary of Contributions

This section elucidates the methodologies we propose for advancing towards ideal automated animation production. The foundational motivations and the architecture of our framework are highlighted herein. The core contributions of our research can be dissected as follows:

1.2.1 Basic Automatic Animation Production Framework

Our paramount objective was to integrate automation within the animation production workflow. To our understanding, our work stands as a pioneering attempt to forge a comprehensive 3D animation production framework that encompasses the entire journey from script conceptualization to the culminating animation video.

Through iterative discussions with industry experts, we distilled the animation production procedure into four cardinal modules: *Action List Generation, Stage Performance, Auto Cinematography*, and *Video Generation*. Each of these modules is tasked with specific duties, utilizing disparate AI technologies for automation.

Given the current technological constraints, there's a necessity for manual intervention by animators to refine module outputs. Nevertheless, we are optimistic that as associated technologies evolve, our framework could seamlessly generate animations straight from scripts.

1.2.2 RealPRNet: A Real-time Phoneme Recognized Network for "Believable" Speech Animation

The domain of real-time speech animation, particularly driven by singular modalities like audio, remains a challenging frontier. The success of such endeavors largely hinges on robust real-time phoneme recognition.

In light of this, we introduce a cutting-edge neural network scheme—RealPRNet—for real-time phoneme recognition and subsequently implement a real-time audio-driven 3D speech animation production system. Comprehensive evaluations reveal that RealPRNet surpasses contemporary algorithms in phoneme recognition accuracy. Empirical analyses denote that, in comparison to leading algorithms [1, 2], RealPRNet achieves a noteworthy 20% PER enhancement and a 4% uptick in animation quality based on subject testing.

1.2.3 Enhancing Auto Cinematography with Fidelity

To streamline the process and mitigate the intricacies of animation production, we introduce Text2Animation (T2A). This framework is a derivative of our foundational animation production framework, designed to transform input scripts into animations while concurrently establishing optimal virtual camera placements.

A novel evaluation metric, termed "fidelity distortion," is introduced to ascertain the consistency between auto cinematography-produced videos and their source content. This metric offers a unique perspective on gauging the quality of the resultant animation. A computational model, engineered to evaluate fidelity distortion efficiently and expediently, has been established through a thorough analysis of contemporary video comprehension techniques. Our pioneering text-to-animation framework, T2A, bridges the gap between

script and visual content, reporting a reduction in 3D animation production time by a staggering 74%.

Empirical evidence suggests that T2A can curtail the manual input in animation production by approximately 74%, with the novel optimization framework enhancing the perceptual video quality by up to 35%.

1.2.4 Enabling Auto Cinematography with Reinforcement Learning

Building upon existing auto cinematography algorithms [3–5], we have refined our T2A framework [4] to allow directors to capture their preferred camera configurations. These configurations subsequently serve as training data for reinforcement learning models aimed at understanding lens language paradigms.

Our framework's capability to seamlessly gather data for auto cinematography during animation production renders reinforcement learning integration feasible. To this end, we propose RT2A (Reinforcement learning-based Text to Animation), a holistic framework channeling reinforcement learning towards auto cinematography. In RT2A, every directorial decision concerning camera configurations is documented, laying the foundation for subsequent agent training iterations. A meticulously crafted reward function guides the algorithm towards the optimal policy, simulating the human director's decision-making approach in camera selections for specific scenes.

Preliminary outcomes underscore RT2A's prowess in emulating directorial lens language patterns. Bench-marked against reference algorithms, RT2A boasts an uptick of 50% in camera placement approval rate and an impressive 80% surge in mirroring the tempo of

camera transitions.

1.2.5 Automated Adaptive Cinematography For User Interaction in Open World

Addressing the intricacies of open-world character interactions demands a specialized approach. We thus introduce AACOGAN, an innovative auto cinematography framework hinged on Generative Adversarial Networks (GANs). AACOGAN is adept at orchestrating camera maneuvers in harmony with the spontaneous actions of characters in expansive virtual terrains. Customized quality assessment metrics, specifically tailored for automated cinematography, have been formulated to rigorously evaluate generated camera trajectories. To ensure congruence between character actions and camera movements, especially within scenes populated by multiple characters, AACOGAN employs a distinctive input feature coupled with a generator architecture. This strategic design choice ensures the meticulous alignment of camera trajectories with the gamut of character interactions, exhibiting pronounced efficacy in multi-character scenarios.

Empirical analyses lend credence to AACOGAN's capability in amplifying auto cinematography proficiency within open-world scenarios. Notable improvements observed include a 73% enhancement in the correlation between user activities and camera paths and a 32.9% elevation in the rendition quality of scenes demanding multi-focal attention.

1.3 Dissertation Organization

The rest of this dissertation is structured as follows. In Chapter 2, we introduce the animation production framework we have developed and used throughout the entire dissertation. All

of our efforts are intended to enhance the capabilities of this framework and achieve a truly automatic animation production without human involvement. Following that, in Chapter 3, we present our RealPRNet for the virtual character's lip-sync animation creation. Next, in Chapter 4, we present a novel auto-cinematography approach that includes the fidelity model in the optimization process. In Chapter 5, we present a reinforcement-learning-based auto cinematography approach that can learn the lens language from the human director and apply it automatically to the new animation production. Furthermore, in chapter 6, we present an auto-cinematography base on Generative Adversarial Networks. This innovative automated photography technique, when deployed in unscripted open-world settings, can synthesize camera movement trajectories in real time, taking into account factors such as emotion, aesthetics, and character actions to meet specific requirements.

Finally, in Chapter 7, we conclude this dissertation and present our ongoing and future work.

Chapter 2

Animation Production Framework for Automation

2.1 Introduction

During the initial phase, one of the most critical steps in developing an automated animation framework is to determine the number of modules within the framework and the functionality associated with each module. Because each module may involve several different steps in the traditional production process and demands various technologies to support it, the development and research of each module require collaboration with specialists in various fields.

On the other hand, because of the limitations of the current technologies, the computational analysis results are not guaranteed to be completely accurate. For example, the *Action List Generation* results which are analyzed by the Natural Language Processing (NLP) technology, especially the location information of the characters in the scene, still require manual revision to correct the mistakes. Therefore, confirming that the output of each module meets the criteria of the expert is an essential step in the framework development process. In order to facilitate understanding, comparison, and quick identification of problems, we have designed the output of each module as a variant of the original input script content. The

user can directly compare the results generated by each module to the original script, and unsatisfactory parts or errors in the results can be modified conveniently.

In this chapter, we introduce a basic automatic animation production framework and the functions of each module. Awareness of each module in this framework would help the reader to better understand our motivation and the importance of our subsequent efforts. In Section 2.2, we will introduce the workflow of the entire animation production framework, the functionality of each basic module, and demonstrate the output of each module. In Section 2.3, a conclusion is drawn to summarize the work done in this part.

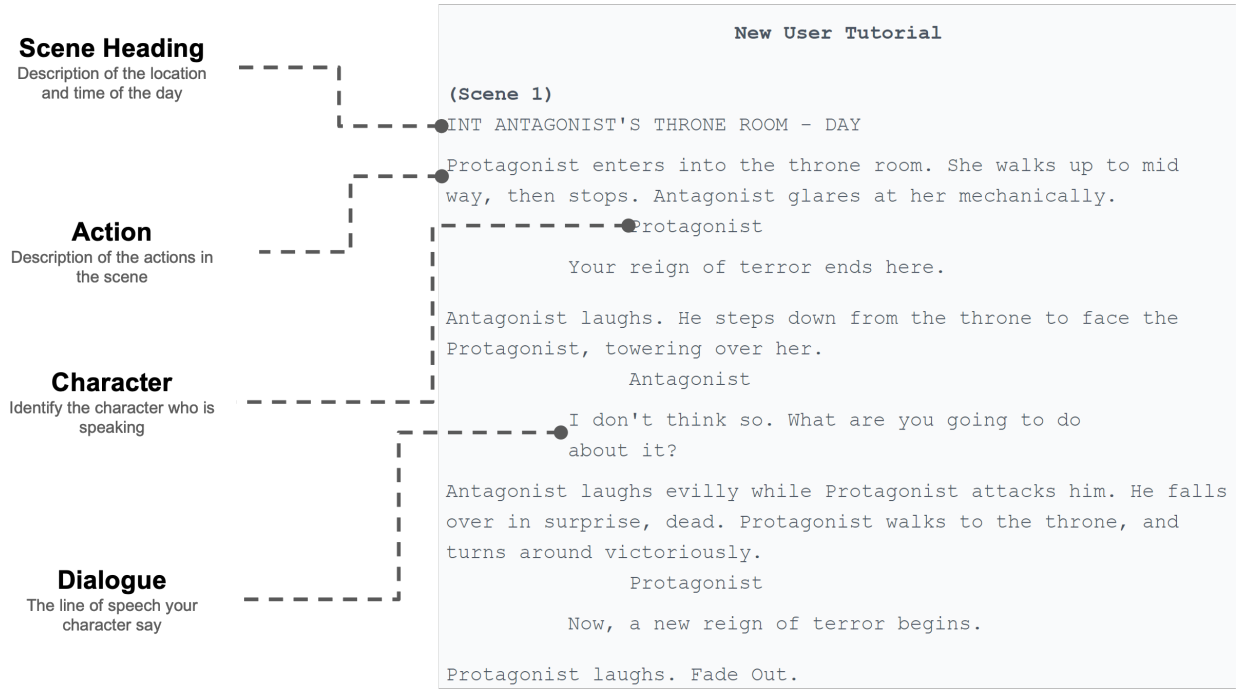


Figure 2.1: Animation Script Example

2.2 Framework

In this section, we illustrate the proposed automatic animation production framework in detail which includes the flow of using the four primary modules and the related technologies

involved.

2.2.1 Action List Generation

The *Action List Generation* module is designed to analyze the original input script, and then generate the corresponding action list and related scene information through the related Natural Language Processing (NLP) technologies.

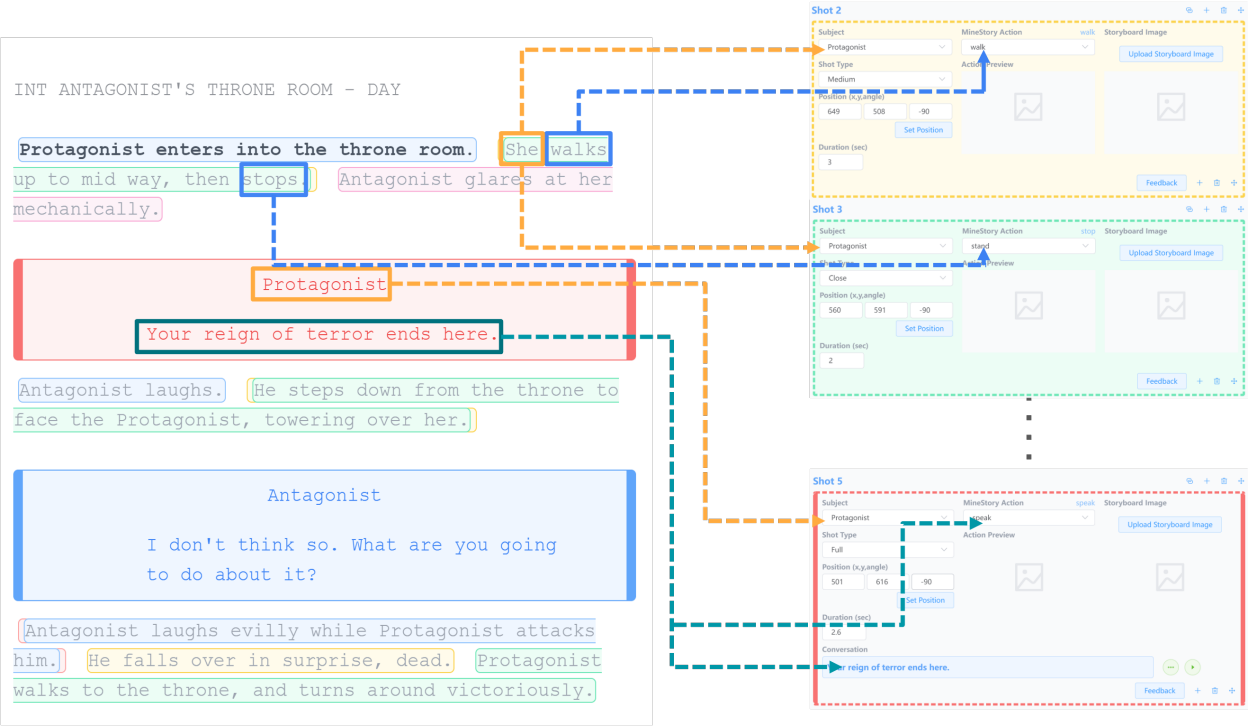


Figure 2.2: Analyzing the script and generating the corresponding action list by NLP technologies

Using the script illustrated in Figure. 2.1 as a case study, it becomes apparent that the script adheres to a unique format. As delineated in Figure. 2.2, the *Action List Generation* module initiates its procedure by analyzing the Scene Heading to discern the settings in which the narrative unfolds. It then references our database to select the most fitting scene resources. Subsequent to this, the module collates all script-embedded actions in a chronological fashion, thereby constructing a comprehensive action list. Each entry, or ac-

tion object, within this list, is mandated to encompass key attributes: action type, initiation and termination timestamps, duration, initiating agent, receiving entity, and the spatial coordinates of characters at both the commencement and conclusion of the action. Notably, the act of speaking warrants special attention due to its real-time generation requirements, as opposed to pre-stored database entries.

The operational intricacies of *Action List Generation* extend to the pairing of audio resources and the subsequent generation of lip-sync animation files. In instances where requisite audio files are absent from the database, users are prompted to contribute by manually dubbing the action. A thorough exposition of the methodology employed for generating lip-sync animations will be articulated in Chapter 3. Upon the action list's completion, users are afforded the opportunity to scrutinize its integrity, amending any inaccuracies or inconsistencies based on their interpretative understanding of the script.

2.2.2 Stage Performance

We have developed the Stage Performance environment (in Figure. 2.3) by using the game engine, Unity [].

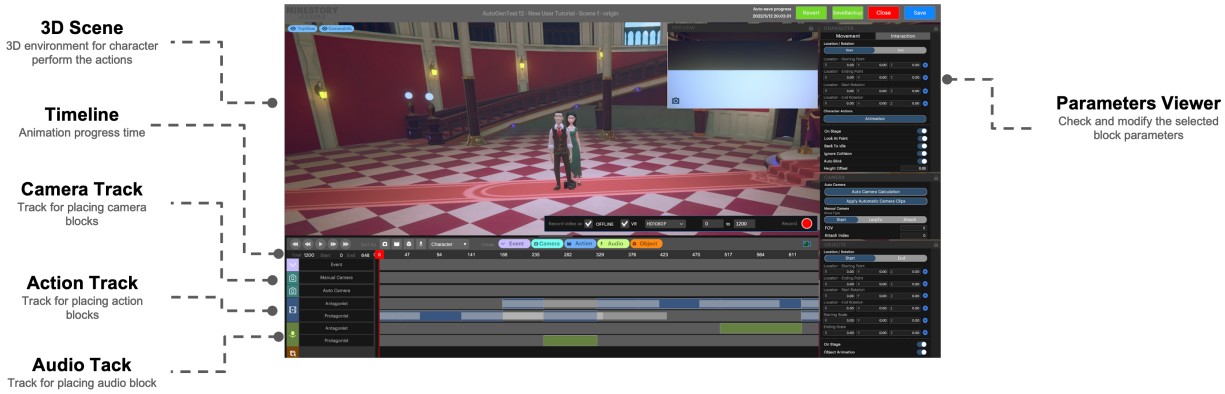


Figure 2.3: Our *Stage Performance* virtual environment developed by Unity software

As shown in Figure. 2.4, *Stage Performance* automatically matches the scene model,

character model, action model, and other resources, then generates all actions according to the input action list. Each character that appears in the story has a separate action track, as different characters may perform various actions in the same period. Each block in the action track represents an action object and contains all its relevant properties, which can be adjusted at the user's discretion. In this software, the user can observe the entire performance generated according to the input action list from any viewpoint in the virtual scene and decide whether further modifications are needed. All data corresponding to these action performances of the virtual characters in the 3D scene in the *Stage Performance* environment are collected and used as input for the *Auto Cinematography*.

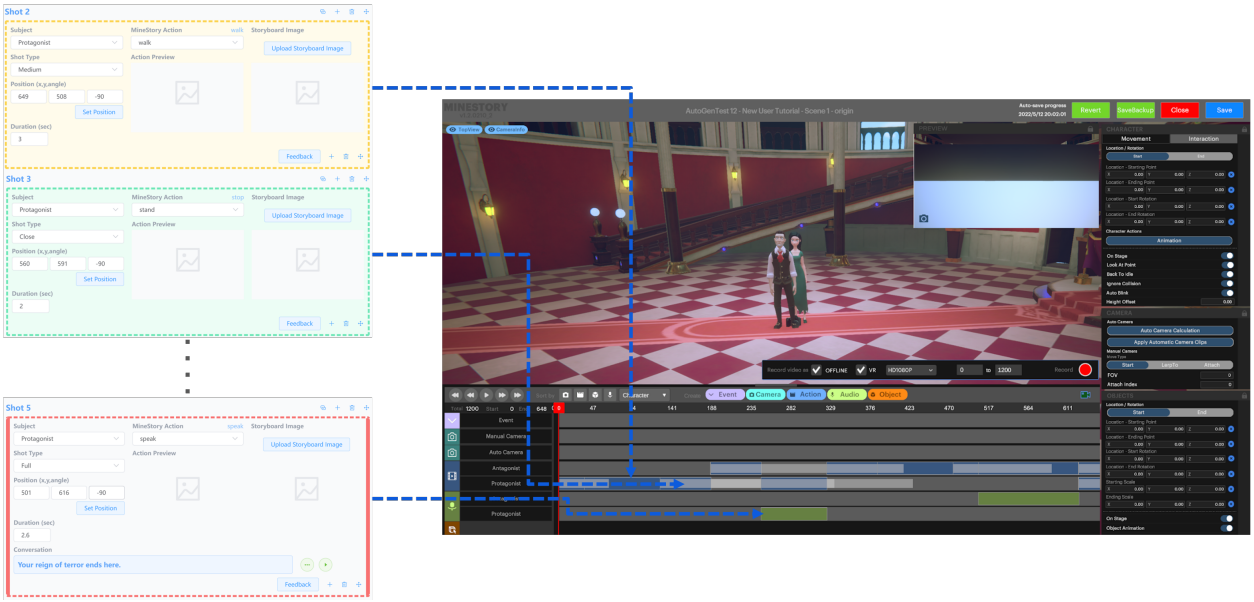


Figure 2.4: The corresponding action blocks and audio blocks are automatically generated in the appropriate tracks according to the action list

2.2.3 Auto Cinematography

The function of the *Auto Cinematography* is to calculate the optimal lens language usage for the entire performance in the 3D scene based on the performance data and relevant

environmental information obtained by the *Stage Performance*.

This module can significantly reduce the user's knowledge requirement of cinematography and time consumption compared to the traditional workflow. Thus, we have concentrated our efforts on improving the capabilities of this module. In Chapter 4 and Chapter 5, We have described the two different approaches that we proposed in detail, a rule-based dynamic optimization approach and the other based on reinforcement learning which learning the lens language usage from the human directors.

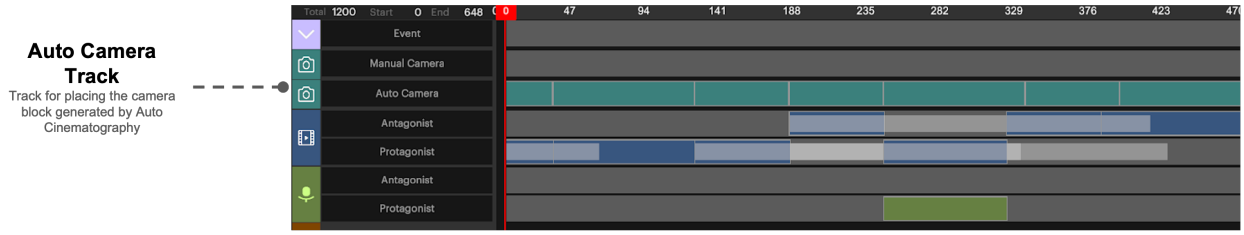


Figure 2.5: The corresponding camera blocks audio blocks are automatically generated in the auto camera track according to the result from *Auto Cinematography*

As shown in Figure. 2.5, after the automatic cameras have completed the calculation, the related results are displayed in the interface of our *Stage Performance* software. If the user is not satisfied with the result, the user can select any camera block in the camera track to adjust its corresponding parameters.

2.2.4 Video Generation

After the user confirms the output of *Auto Cinematography*, the corresponding video file can be generated directly by the *Video Generation*.

This module corresponds to the editing work in the traditional animation production workflow. The movie edit is a complex and tedious endeavor that requires sufficient relevant knowledge. The automated implementation of this module will be part of our future work

but is outside the scope of this dissertation.

Thus, this module does not edit the results in our current production process but directly outputs video according to the original camera track shown in *Stage Performance* software.

2.3 Conclusion

In this chapter, we have introduced the fundamental component of the automatic animation production framework that we have proposed and developed, which will be used as the foundation for our subsequent works to make the animation production process more convenient, more accurate, and more efficient.

Chapter 3

RealPRNet: A Real-time Phoneme Recognized Network for "Believable" Speech Animation

3.1 Introduction

In the *Action List Generation*, when a dialogue scene occurs in the input script, the virtual character will have a corresponding particular animation for its mouth movement. These mouth movements cannot be pre-generated in the *Stage Performance* action library. Thus, *Action List Generation* needs to generate correlative mouth animations for the virtual character's dialogues in real time during the production process. Human beings are very sensitive to any facial artifacts, or uncoordinated or unsynchronized performance of virtual characters, which makes facial animation, in particular speech animation production, very challenging since animation simultaneously involves voice and mouth movement.

Realistic speech animation [1,6] has been always very compelling since it can provide the most immersive experiences with high-fidelity human-like virtual characters. However, the high cost involved in the production process and the substantial data requirements, including audio and video, are likely to create privacy issues.

For applications that accept lower realism effects but require good privacy preservation, such as avatar-based online conferences, anonymous alerts, and customer service, audio data could become the only media available during the process. In such scenarios, the virtual characters are required to mimic mouth movements matching the voice input seamlessly, and this is what is called believable speech animation. The "believable" speech animation requires that the algorithm works, under practical resource constraints, for all possible virtual faces with various 3D models and produces synthesized video with sufficient realism for users on the remote end to feel comfortable.

The widespread use of NLP technologies, such as speech recognition [7, 8] and speaker identification [9], demonstrates that such a "believable" speech animation is achievable by using only audio input [10–12]. In general, the audio input is fragmented into small pieces called frames from where features are extracted. Phoneme, which is the distinct unit of sound, is predicted by using the extracted features and then mapped into the corresponding static mouth shape called viseme (its counterpart in the visual domain [13]). In such an audio-driven speech animation framework, the accuracy of the phoneme recognition can directly affect the quality of the speech animation.

With the latest advances in deep learning, the phoneme recognition topic has been revisited using completely new methodologies [14]. The advances in phoneme recognition accuracy can lead to better speech animation. Compared with the traditional models, such as Hidden Markov models (HMM), deep-learning neural network (DNN) based approaches generally decrease the phoneme recognition error rate by 10%.

For real-time applications, finding the balance between latency and accuracy is critical, as indicated in the design philosophy of RNN and LSTM. The temporal correlation between neighbor audio frames can play a very significant role in recognition accuracy improvement.

However, the phoneme recognition accuracy improvement achieved by adding more neighbor frames in the sliding window is at the cost of latency. Therefore, a crucial problem that needs to be addressed to improve the real-time speech animation quality is to find a phoneme recognition solution that can achieve the best accuracy with reasonably low latency for real-time animation. In this work, we propose a novel deep neural network scheme, called RealPRNet. With a carefully designed network architecture that considers both temporal and spatial correlations, RealPRNet predicts phoneme stream for a sliding window of audio frame input. A novel concept called LSTM Stack Block (LSB) is introduced to maximize the learning efficiency of the temporal-spatial patterns(more details are covered in the network design section).

To build an end-to-end audio-driven real-time believable speech animation system, the RealPRNet is inserted into the typical speech animation process that converts the audio input into a recognized phoneme sequence and drives a facial animation module. Inspired by the JALI model [12] and properties of the blend shape facial model, the animation is achieved by mapping the recognized phoneme label to a set of parameters to control four basic blend shapes that have hidden physical correlation.

The major contributions of this work can be summarized as follows:

1. We propose a novel neural network-based real-time phoneme recognition scheme.
2. We develop a real-time audio-driven 3D speech animation production system.
3. We conduct a comprehensive evaluation to show that the proposed RealPRNet scheme can achieve great improvement over the state-of-the-art algorithms.

The remaining part of the chapter is organized as follows: In Section II, we describe the details of our animation production system components and introduce the deep-learning-based Realtime Phoneme Recognition Network (RealPRNet) with our insight in Section III.

We present an evaluation and experimental results in Section IV. Finally, in Section V, a conclusion is drawn to summarize the work done in this research.

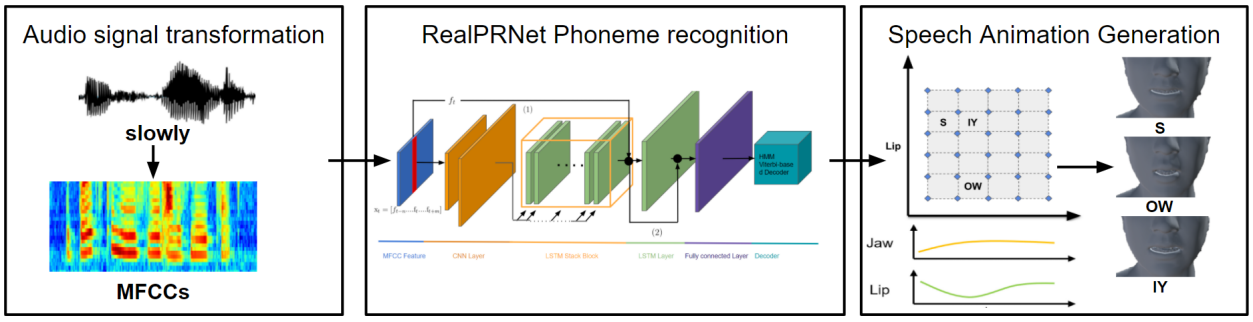


Figure 3.1: System overview of the proposed real-time audio-only speech animation system corresponds to the phoneme stream {S, OW, IY} of vocabulary /slowly/

3.2 Related Work

A typical audio-driven speech animation uses viseme as an intermediary. It can be seen as a combination of the phoneme recognition and the phoneme-driven speech animation.

3.2.1 Phoneme Recognition

Phoneme recognition using audio's fundamental frequency [10] and HMM [15–19] has been an active research topic for decades. However, the accuracy of these traditional schemes is not sufficient for speech recognition and speech animation production. The advances in deep learning and neural networks have greatly enhanced the accuracy of phoneme recognition [14, 20–22]. In [20], a feed-forward deep neural network model was proposed and achieved an error rate of 23.71%. In [14], it was reported that a feed-forward deep neural network architecture lowered the error rate to 16.49%. In [22], a network architecture called CLDNN that combines Convolutional Neural Network (CNN) LSTM and fully connected DNN was

proposed. It can further improve the performance for 4-6% compared to the LSTM. In particular, around 39 different phonemes can be recognized compared to the fundamental schemes [10], which can recognize less than 10 different phonemes. However, these methods are all designed for offline situations without latency constraints, which enables any feature extraction schemes to be used. In this work, we develop the RealPRNet to achieve accurate real-time phoneme recognition.

3.2.2 Speech Animation

In [10], the audio input is fragmented into small pieces called frames, where fundamental frequency features are extracted from. Phonemes are predicted by recognizing the vowels and the basic fricative consonants from the features and then mapped into the corresponding static animation called viseme (its counterpart in the visual domain [13]). In this frame-based processing mechanism, the latency is negligible as it almost equals the processing time of a single frame. However, a lack of consideration on neighborhood context information during the process may significantly limit the recognition accuracy and quality of the generated animation as the system can only recognize basic phonemes. In [11] and [12], a word-based processing mechanism is adopted to achieve much higher animation quality. By utilizing force alignment [23], phoneme transcription can be extracted from an audio chunk that contains multiple words with reasonably high accuracy. In [24], the neighboring phoneme pronunciation transition, named co-articulation, was considered. However, the latency of word-level duration becomes unacceptable for real-time applications. In this work, we use a sliding window of frames to obtain the corresponding phoneme at each timestamp. Our designed facial model addresses the co-articulation problem by considering the duration of the current phoneme's articulation and the corresponding associated phonemes before and

after it.

3.3 System Components Design

In a typical real-time audio-driven speech animation system, the phoneme stream first is recognized from the audio input and then mapped to the corresponding parameter stream to derive the 3D facial models.

Viseme is the visual mouth shape representation that has been widely used in speech recognition and animation, as it can be mapped directly from the phoneme. However, vice versa is not true since multiple phonemes may be mapped to the same viseme if they have similar mouth shapes during the pronunciation, such as /b/ and /p/). It is important to realize that so far there is no common standard to regulate the viseme classes [25], for example, [2] used 20 different visemes, [26] used 26 different visemes, and [27] used 16 visemes, etc.

Figure. 3.1 gives a system overview of the proposed real-time audio speech animation systems corresponding to the phoneme stream {S, OW, LY} of vocabulary /slowly/. When the system receives an audio signal, it is transformed into the corresponding MFCC features, which is the input of the RealPRNet. The RealPRNet predicts the phoneme stream and then maps it into the corresponding points (or blocks) in the 2D viseme field. The animation curve shown in Figure. 3.1 connects the points on the 2D viseme field and generates a parameter stream that can drive the 3D facial model smoothly and seamlessly. To support real-time interactivities, the latency between receiving audio input and outputting animation is required to be controlled below certain thresholds(e.g.,200ms [28, 29]) to ensure a believable and audience-comfortable result. A buffer mechanism is adopted in the system to

dynamically determine the size of the sliding window and ensure that the latency is within the required threshold. The four buffers are input feature buffer B_1, HMM tied-state output buffer B_2, output phoneme buffer B_3, and phoneme selected buffer B_4.

3.3.1 Input Features Extraction Component

We employ the general audio process pipeline to extract the input audio features. When the raw audio signals are received in the system, they are transformed into frequency-domain signals called MFCC, which is a data format that has been widely applied in automatic speech recognition (ASR). It is observed that the human voice is a combination of sound waves at different frequencies. The MFCCs can balance the variation of the sound change at different frequency levels. Generally, there are 100 frames per second in audio, and each audio frame is 25ms with a 15ms overlap. For each audio frame, the first and the second derivative components of the MFCCs are aggregated to a vector that represents the single audio frame, f. The input feature x_t at time t is transformed to the network feature f_t at time t surrounded by its forward and backward contextual vectors, denoted as $[f_{t-n}, \ldots, f_t, \ldots, f_{t+m}]$. The value of n represents the number of audio frames before time t, and the value of m represents the number of future audio frames. The integrated x_t is used to predict the phoneme label at time t. Thus, the value selection of m directly impacts the latency, which is $10m$ ms. The larger the value of m, the better potential recognition accuracy but longer latency. When $m = 0$, no future frames are buffered to introduce additional latency. However, the potential advantages of context information have been taken into consideration to improve phoneme recognition performance.

3.3.2 Real-Time Phoneme Recognition Component Design

Real-time application systems need to be able to extract the audio features with high accuracy and low latency. Our proposed phoneme recognition scheme RealPRNet (described in Section III) can ensure a high-accuracy real-time recognition. The output of the RealPRNet is used as input to the HMM tied-state decoder H to calculate the output phoneme P. For the system to produce a smooth animation, the phoneme recognition system needs to be able to output the predicted phoneme with every A ms. Thus, a buffer mechanism is introduced in the system. An overview of the phoneme recognition system with buffers is shown in Figure. 3.2. All buffer in the figure is fixed size first in first out (FIFO).

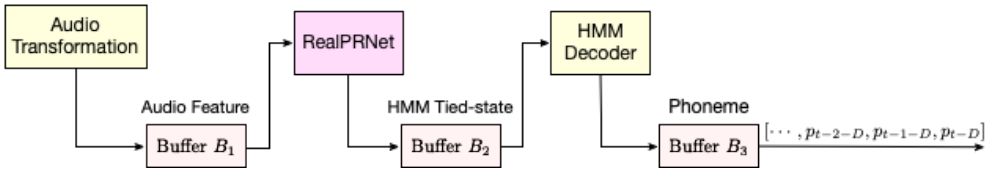


Figure 3.2: Phoneme recognition System with Buffers

In the first step, the input raw audio signal is transformed into the network input feature f_t every A ms time interval (equal to the sampling interval), and the f_t is stored in the input feature buffer B_1. The transformation of the raw audio signal to audio features causes the first delay $d_1 + e_1$, where d_1 is the median calculation time and e_1 is the corresponding fluctuation time. In our experiment, time consumption by this process is very stable and 100% less than A ms (10ms in our experiment) as shown in Figure. 3.3 (blue).

All the audio frame features in B_1 are then used to construct x_t. The size of B_1 depends on two things: the selected values m and n, and the RealPRNet time consumption $d_2 + e_2$ for each prediction. To guarantee a smooth output, the following inequality needs to be

satisfied:

$$d_2 + e_2 \leq br \cdot A, \quad br = 1, 2, 3, \cdots \tag{3.1}$$

br is the batch size used in the prediction progress. The neural network can parallelly predict br outputs in one run with a minimal increase in the computational overhead if br is a small value (i.e. $br = 10$). This is because when br is small, the input features data sizes are relatively small compared with the RealPRNet's parameters. There is no sensible difference to today's computational power (e.g. in our experiment, one RTX 2080 Ti graphics card has been used) when processing single or br input features. The major time consumption is in data parsing and transmission. The RealPRNet takes the input features from B_1 every $br \cdot A$ ms and predicts br outputs $[h_t, h_{t-1}, \cdots]$. These predicted outputs are stored in HMM tied-state buffer B_2. There are br sub-buffers in B_1 with size $m + n + 1$ each and contents $f_{t-n-i}, \cdots, f_{t-i}, \cdots, f_{t+m-i}$. Because m forward audio frames are used to construct the x, the system latency is increased to $m \cdot A + br \cdot A$ ms. In our experiment, the value of br is 4 which can ensure that equation (3.1) is satisfied in 99% of the cases. The network time consumption distribution is shown in Figure. 3.3 (orange).

The neural network in our phoneme recognition system does not directly predict phoneme labels but predicts HMM tied-states instead which is because the combination of neural network and HMM decoder can further improve the system performance. Such kind of recognition system structure can be found in many related works [30, 31]. The predicted HMM tied-states in B_2 are used as input of the HMM tri-phone decoder. For each time interval $br \cdot A$, the decoder takes all the values in B_2, calculates the corresponding phonemes $[P_t, \cdots, P_{t-br+1}]$ and stores it in B_3. The calculation time is $d_3 + e_3$ and the size of B_3 depends on the number of previous states used to calculate the $[P_t, \cdots, P_{t-br+1}]$. The HMM

decoder improves the results by using the previously predicted tied-state together with the h_t to calculate the corresponding phoneme P_t. However, the decoder should only use the tied-state as a reference rather than relying on it as in the speech recognition system because the phoneme-to-phoneme does not have a strong inner logic relation as word-to-word. Thus the decoder is constructed with a bigram phoneme model which focuses on the acoustic part (signal to phoneme) of the system rather than the language part (phoneme-to-phoneme) [14]. In our experiment, the calculation time consumed in decoding is stable and 100% less than $br \cdot A$. Thus, the overall latency from the raw input audio signal to the corresponding output phoneme is $(m + br) \cdot A + D + e_t$, where $D = d_1 + d_2 + d_3$ and $e_t = e_1 + e_2 + e_3$.

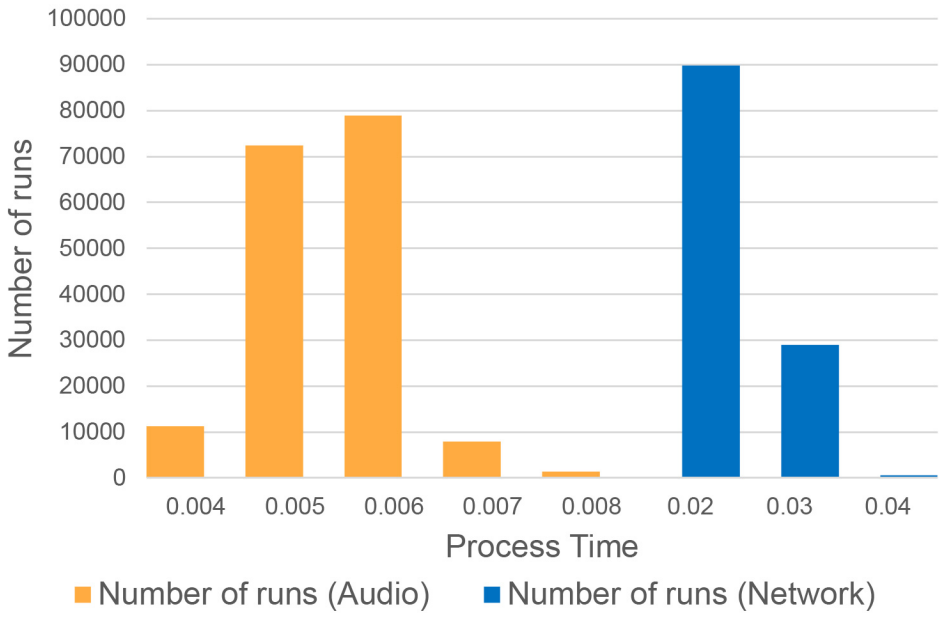


Figure 3.3: Audio feature transformation time consumption distribution (blue), Network Prediction time consumption distribution (orange)

B_3 is the buffer that is used to control the final output of P of the phoneme recognition system. This output phoneme with a timestamp is stored into B_3. The system continuously takes the first phoneme from B_3 and uses it as the input to the animation system every A

ms time interval. If a predicted phoneme with a timestamp has negative e_t, the system waits for A ms before the de-queued output from B_3. If a predicted phoneme P_{e_t} with a time stamp has positive e_t and the buffer contains no more phoneme, the system outputs the last phoneme and stores P_{e_t} as the last phoneme in the buffer. P_{e_t} can be used as an output if the next predicted phoneme also has a positive e_t. If not, the system drops the P_{e_t} and then outputs the next predicted phoneme after a time interval of A ms. With this mechanism, the phoneme recognition system can have a stable output stream with a time interval A ms, and the overall latency from raw input audio signal to the corresponding output phoneme stream is $(m + br) \cdot A + D$, where $D = d_1 + d_2 + d_3$. This stable output phoneme stream is stored in B_4. The following pseudo-code represents this process. The buffer is B_3, and output d_P is the output phoneme which is going to be stored in B_4.

```
while true:

    RealPRNet_Thread:
    if cur_time == pre_time1 + a_t:
        P = RealPRNet(x_t)
        p_last = P
        buffer.enqueue(P)

    Output_Thread:
    p_last = None
    if cur_time == pre_time2 + d_t:
        if buffer is empty:
            pre_time2 = cur_time
            return p_last
        else:
            pre_time2 = cur_time
            d_P = buffer.dequeue()
            p_last = d_p
            return d_p
```

Figure 3.4: Python code for RealPRNet processing

In this pseudo-code, the buffer is a queue structure, the cur_time is the worldwide system

time and the d_P is the beginning element in the buffer. The animation subsystem takes data in B_4 and uses it to produce the corresponding speech animation.

For the final speech animation generation, the output phoneme sequence of each time interval A ms from the previous component needs to be further processed. B_4 is used to select the appropriate next phoneme frame for the animation curve generation. As shown in Figure. 3.6, the same phoneme can occur in different frames. The size of B_4 is the average single phoneme pronunciation time in the audio frame. The output phoneme of the recognition system is stored in this buffer first. The phoneme pronunciation is dynamic progress which means that the viseme phoneme at the beginning of the pronunciation is not the same as the corresponding phoneme viseme as shown in Figure. 3.5.

Figure 3.5: A phoneme corresponding viseme at different frames during the pronunciation of phoneme /o/

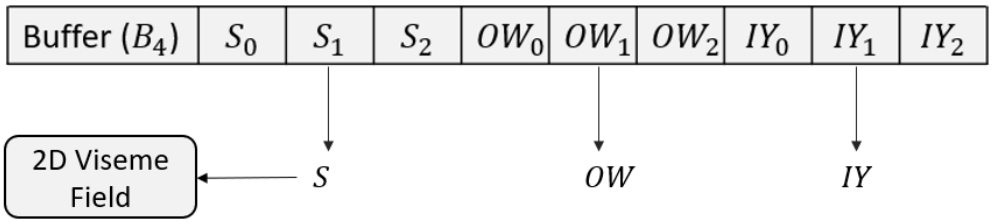


Figure 3.6: Phoneme selection for 2D viseme field animation curve generation

Thus, an appropriate phoneme frame (for example, the phoneme frame can represent the rightmost one in Figure. 3.5 in a sequence of phoneme /o/ frames) should be selected from

certain phoneme pronunciation frames to calculate the complete viseme's transformation time using the animation curve generation. If the minimum recognizable phoneme pronunciation time in the data set is $pr_{\min}$ audio frames and the length of the continuously predicted phoneme in the buffer is less than $pr_{\min}$, then the corresponding phoneme will be replaced by the previous phoneme. The upcoming phoneme section rules can be described as follows:

1. The same phonemes are continuously appended to the buffer for at least $pr_{\min}$ units.
2. If the number of continuous phonemes is more than $pr_{\min}$ units and less than the size of B_4. The appropriate frame that represents the phoneme will be selected from that part of the buffer.
3. If the number of the continuous phonemes is more than the size of B_4, all phonemes in the buffer will be used to select the appropriate frame, and no new frame will be selected until the next different phoneme is appended to the buffer.

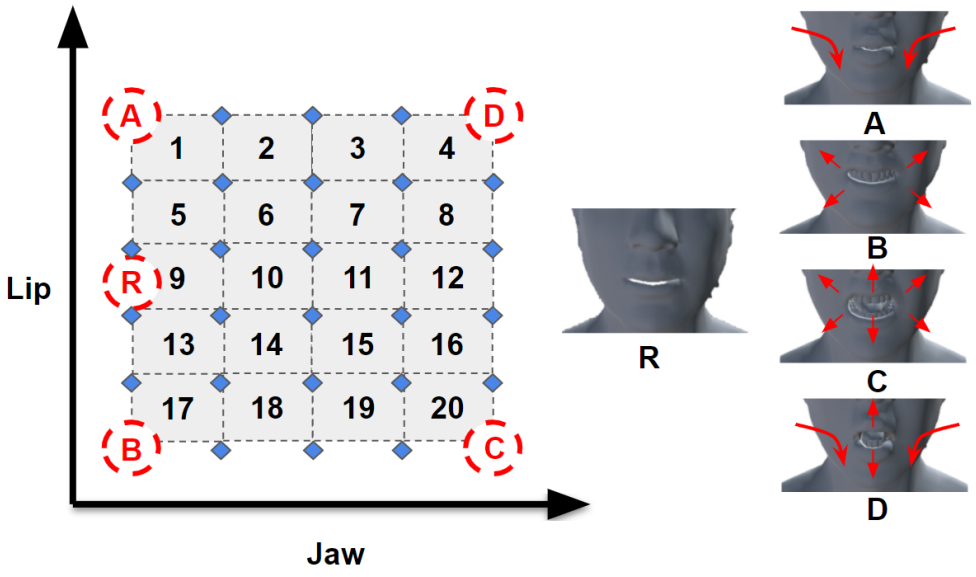


Figure 3.7: 2D viseme field with block index (left). The combination of the jaw and lip parameters in the extreme (A, B, C, D) and regular cases (R) is visualized by the 3D model viseme (right)

3.3.3 Speech Animation Component Design

The speech animation component has been designed following the JALI viseme field and the state-of-the-art procedural lip-synchronization system [12]. The implementation in our system is slightly different from the original JALI viseme field. Most of the procedural systems, such as [12, 32, 33], that use the keyframe viseme to produce the final animation have a similar problem, that is the phoneme is mapped to one fixed static viseme without any variation. Through our observation of human speech behaviors, visemes corresponding to a phoneme may be slightly different in different situations. This is true even when the current phoneme has a sufficiently long pronunciation time to erase the co-articulation of the sound caused by the previous phoneme pronunciation. For example, the visemes that represent the phoneme $/s/$ are pronounced differently in the $'things'$ and $'false'$ under the same speaking style. Thus, the 2D viseme field has been divided into different blocks (20 blocks in our implementation), as shown in Figure. 3.7. Each phoneme corresponds to a block region rather than a static point in the 2D field.

3.4 Neural Network Design for Real-Time Phoneme Recognition

In recent years, neural network methods [34, 35] have dominated the phoneme recognition field and have dramatically improved recognition performance. We design a novel neural network architecture for our proposed real-time phoneme recognition task. The overall network architecture is shown in Figure. 3.8. The CNN layer takes x_t as input and applies frequency modeling on the input features. The n-dimensional vector output of CNN is passed into

an n-layer stack of LSTMs for parallel processing and temporal modeling. The output is combined with the f_t to form an input to additional LSTM layers for temporal modeling and then goes through a fully connected layer. In the end, the HMM tri-phone decoder is adopted to predict the phoneme label.

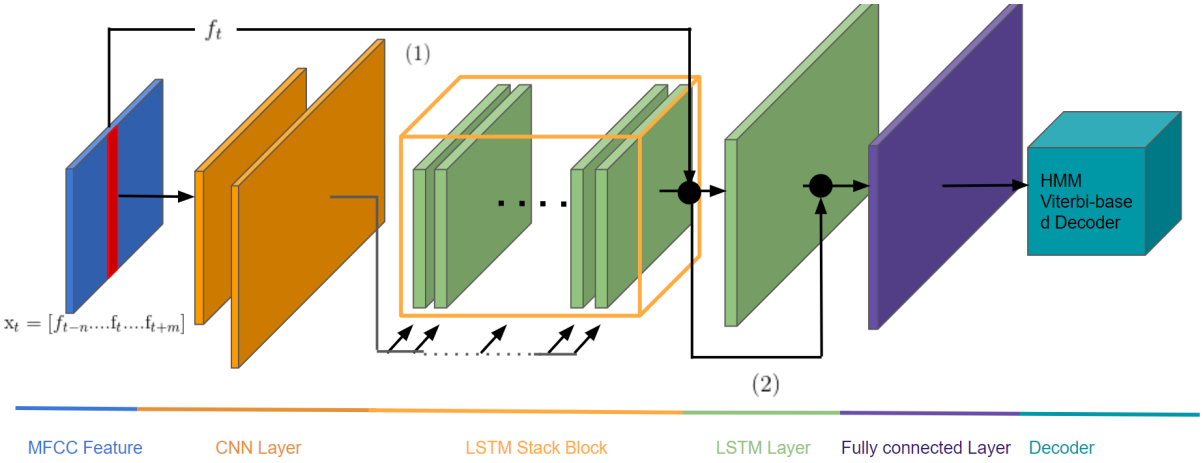


Figure 3.8: Network Architecture Overview

3.4.1 CNN Layer

CNN has demonstrated outstanding performance in audio-related fields [35, 36]. The CNN layer is mainly used as a frequency modeling layer. An important contribution of the CNN layers is to reduce frequency variation. Based on the observation that voice of different people contains different ranges of frequencies even when they are speaking the same utterance, the traditional GMM/HMM-based speech recognition systems use techniques to reduce the frequency variation in the input feature, such as vocal tract length normalization (VTLN) [37] and feature space maximum likelihood linear regression (fMLLR) [38]. It has been reported recently [22] that CNN can provide the same feature improvement to the audio feature.

CNN layers also play an important role in the temporal-spatial domain [22], here spatial refers to frequency-domain audio feature pattern detection and learning. The input feature

contains frequency information since each coefficient in Mel-frequency cepstral is generated by passing through different frequency filter banks. Each CNN filter can learn the different frequency patterns from the input features during the training. Our network architecture emphasizes these frequency patterns in the input features by separately connecting the CNN output features ($\mathsf{CNN}_{\mathsf{fout}}$) of different CNN filters to different LSTM in the LSTM stack Block module. We use a 9×9 frequency (spatial)-temporal filter in the first CNN and a 3×3 filter in the second layer. The first layer has 256 output channels and the second layer has 16 output channels. We set no pooling in both CNN layers after observing that neither max nor average pooling helps to improve the result.

3.4.2 LSTM Stack Block

After frequency modeling is applied to CNN layers and the CNN filters have learned the acoustic features from the input feature, each CNN output channel produces the intermediate features as shown in Figure. 3.8. These features are applied to a parallel process of the temporal modeling in LSTM stack block (LSB).

The $\mathsf{CNN}_{\mathsf{fout}}$'s from different CNN channels pass (Equation (3.2)) to different LSTM modules, called LSTM Tube (LT) which is inside the LSB as shown in Figure. 3.8. The number of LT inside the LSB depends on the number of the last CNN layer output channel, one LT for each CNN output channel. An LT is a relatively small and independent LSTM network. The LSTM has been proven to be advantageous in temporal modeling because it has memory cells to remember the information on previous features [21, 39]. Each output from different CNN channels can be viewed as a derivative feature of the original audio feature with the context within a sliding window. Thus, separate temporal modeling is applied to each different $\mathsf{CNN}_{\mathsf{fout}}$. The $\mathsf{CNN}_{\mathsf{fout}}$ is passed to LT0, where the "0" represents the CNN

filter with index 0, inside the LSB for independent temporal modeling (Equation (3.3)) and the output feature is Lt_0. These separate output features are aggregated together as the input feature for the next LSTM layer (Equation (3.4)).

$$\mathsf{CNN}_{\mathsf{outputs}} = \left\{C_{\mathsf{fout},1}, C_{\mathsf{fout},2}, \ldots, C_{\mathsf{fout},n}\right\} \tag{3.2}$$

$$\mathsf{Lt}_n = \mathsf{FCL}(\mathsf{LSTM}(\mathsf{CNN}_{\mathsf{fout},n})) \tag{3.3}$$

$$\mathsf{LSB}(\mathsf{CNN}_{\mathsf{outputs}}) = \{\mathsf{Lt}_1, \mathsf{Lt}_2, \cdots, \mathsf{Lt}_n\} \tag{3.4}$$

In our network architecture, the LSB contains 16 LTs inside and each LT includes 2 LSTM layers and 1 fully connected layer. Each LSTM layer has 512 hidden units and a 0.3 dropout rate, and each fully connected layer has 128 output units.

3.4.3 LSTM Layer

In [14], it is demonstrated that the LSTM layer has its advantage in extracting temporal patterns in input feature space as we mentioned in the LSB. The gate units in LSTM are used to control the information flows inside the LSTM. The input gate decides what information can be added to the cell state. The forget gate decides what information needs to be removed from the cell state, and the output gate decides what information can be used in the output. Thus the LSTM can selectively "remember" the temporal features. After the LSB performs the separate temporal modeling, we pass its output to the LSTM layers for the unified temporal modeling with the same logic. There are 4 LSTM layers, and each has 1024 hidden units, 512 output units, and a 0.2 dropout rate.

3.4.4 Fully Connected Layer

Following the state-of-the-art design of a typical deep learning network, we use the fully connected layer with softmax activation as the last layer so that a fully connected layer can provide the model with the ability to mix output signals from all neurons in the previous layer and the softmax can shape the output probabilities of each class so that the target class has a higher probability. The fully connected layer has 1024 hidden units. The output is an 1896-dimension vector which represents the possibility of 1896 HMM-GMM tied-states. The HMM tri-phone decoder takes this as an input to predict the final phoneme label.

3.4.5 Multi-Scale Features Addition

The idea of the multi-scale feature addition was originally explored in computer vision and also used in ASR-related problems [36]. In a neural network, each layer focuses on different input features and varies from general to specific concepts. In ASR tasks, the lower layers (e.g., CNN layers in RealPRNet) focus more on speaker adaptation, and higher layers (e.g., LSTM layers in RealPRNet) focus more on discrimination [40]. Thus, the input features of the different layers are complementarity, and the network's performance improvement has been observed by using these techniques [22]. In our implementation, we have explored two feature addition strategies, which are illustrated in Figure. 3.8 through line (1) and line (2), where line (1) represents the original frame feature, f_t, in x_t combines with the LSB output and line (2) represents the LSB output combines with the LSTM output.

The first feature addition explores the complementary information in short-term feature f_t and long-term feature output feature from LSB (high order representation of x_t). x_t is aggregated by using f_t and it's context features $[f_{t-n}, \cdots, f_{t-1}]$ and $[f_{t+1}, \cdots, f_m]$. However

the original LSTM does not consider their different values in prediction but takes all f's in x_t as consecutive features [41] and equally considers all the f's in x_t. This feature addition emphasizes the importance of f_t in x_t when predicting p_t.

The second feature addition checks the LSB and the LSTM outputs, the separated and unified complementary temporal feature information. These features are the high-order feature representations of x_t with different pattern complementarity since network layers focus on different information in x_t with these patterns.

In the evaluation part, the performance improvement by multi-scale feature addition has been explored, and the results show a positive effect on the network performance.

Table 3.1: Network Parameters

	hidden units	**output units**	**drop out rate**	**ksize**
conv0	N/A	256	N/A	9
conv1	N/A	16	N/A	3
lsb0	1024	128	0.3	N/A
lstm0	1024	512	0.2	N/A
lstm1	1024	512	0.2	N/A
lstm2	1024	512	0.2	N/A
lstm3	1024	512	0.2	N/A
fcl0	1024	1896	N/A	N/A

3.5 Experimental Results

In this section, we evaluate the performance of the system and show that our proposed system can generate competitive speech animation results in real-time using only audio input, compared with a speech animation system using multimedia (video and audio) or offline methods. The performance of the proposed system is evaluated in three areas: (1) the RealPRNet phoneme recognition accuracy, (2) the buffer occupancy dynamics to enable

the real-time application, and (3) the subjective and objective speech animation quality.

3.5.1 Experiment Setup

Our experimental results are conducted using the TIMIT data set, which is widely used for phoneme recognition evaluation. It contains 6300 sentences, consisting of 10 sentences spoken by 630 speakers each from 8 major dialect regions of the United States. By following the standard TIMIT setup, we use the standard TIMIT training set, 3696 utterances from 462 speakers, to train our network and evaluate it on the TIMIT core test set, which consists of 192 utterances.

We use the TIMIT s5 recipe in Kaldi [42] to calculate phoneme duration in each utterance through force alignment technique and generate an HMM tied-state tri-phone model, the corresponding 1896 tied-states and their properties (i.e., state probability, transfer probability, corresponding phoneme, etc.). We then use a tri-phone decoder with a bi-gram phone model in our system at the end of the neural network architecture, which takes the tied-states stream as input and outputs the predicted phoneme stream. The output ground truth y for the corresponding input feature x in the training data set is the index of the HMM-tied states. Kaldi enabled audio to HMM tied-states force alignment. For the network training, 10 epochs have been set as the minimum training epoch and to enable early stop (if the validated loss change in epochs is less than 0.001) during the training. The Adam optimizer was used in the first epoch and the momentum stochastic gradient descent (MSGD) optimizer for the rest epochs. The batch size is 256 for the first epoch and 128 for the rest. The learning rates are 0.01, 0.001, 0.0005, 0.0001 for the first four epochs and 0.0001 for the rest of them. The weight initialization has been used for the CNN layer's parameters, and the 0.3 dropout rate has been used for all the LSTM layers. The ReLU activation was applied

to most of the layers except the final fully connected layer, which used softmax.

The performance of the network with a different m in the input feature was evaluated in this section. In our experiment, value n in x_t has been set to $m + 1$.

3.5.2 Error Metrics

3.5.2.1 Phoneme error rate

We first evaluate our proposed RealPRNet with a standard metric. During the evaluation, we first map the 60 phonemes to 39 and then calculate the "Levenshtein distance" between the recognized phoneme sequence and the ground truth phoneme sequence. Levenshtein distance is a method to measure the difference between the two sequences. It calculates the minimum number of single-character edits (including insertions, deletions, and substitutions) required to change one sequence into another. The number of edits required to change the recognized phoneme sequence into the ground truth phoneme sequence is first calculated as the ratio between this minimum number of edits and the whole phoneme sequence length. This ratio is also known as the phoneme error ratio (PER). Under the real-time situation, the phoneme is predicted for each audio frame. Thus, the PER is calculated based on the audio frame-level phoneme sequence in our evaluation.

3.5.2.2 Block distance error

The block distance error (BDE) measures the average distance between the trajectory curve in the viseme field produced by the recognized phoneme sequence and the curve produced by the ground truth phoneme sequence. The basic unit used here is the short edge of the 2D viseme field block. For each audio frame, the corresponding points on the two curves

and the absolute distance between these two points are calculated, also known as Euclidean distance. Then the average distance between these two curves at each time t was calculated.

$$\mathsf{BDE} = \frac{1}{T}\sum_{0}^{T} P_{\mathsf{predict}} - P_{\mathsf{groundtruth}}, \tag{3.5}$$

where P_{predict} is the predict phoneme position and $P_{\mathsf{groundtruth}}$ is the ground truth position in viseme filed. T is the total number of time intervals.

3.5.3 Phoneme Recognition Performance

In the experiment, RealPRNet was compared with two other phoneme recognition systems: the 4 layers LSTM architecture presented in [14], and the CLDNN [22].

Since most of the existing related work uses offline recognition schemes, we first evaluate the offline phoneme recognition capability of the baseline models. The best performance of the offline phoneme recognition of the above models in the evaluation using the TIMIT data set are 4 layer LSTM 18.63%, CLDNN 18.30%, and RealPRNet 17.20%. The RealPRNet outperformed other methods by 7.7% PER in the best case.

In Figure. 3.9, the real-time performance of these networks was compared. The x-axis in the figure represents the number of frames used in a single x_t, and the y-axis represents the phoneme error rate. It is interesting to observe that LSTM has low performance when the temporal context information is insufficient (i.e., when x_t is aggregated by using less than 10 fs), but its performance improvement continuously with the increasing of the x_t value until a sweat spot (20 audio frame features in x_t) is achieved. In the figure, RealPRNet outperformed LSTM for a minimum of 20% and 10%, respectively. The combination of frequency and temporal modeling enabled the RealPRNet a smooth performance across

various selections of the number of frames for x_t.

To understand the advantage of RealPRNet further, we considered two additional variations: RealPRNet without the LSTM layer, and CLDNN and RealPRNet without long short-term feature addition.

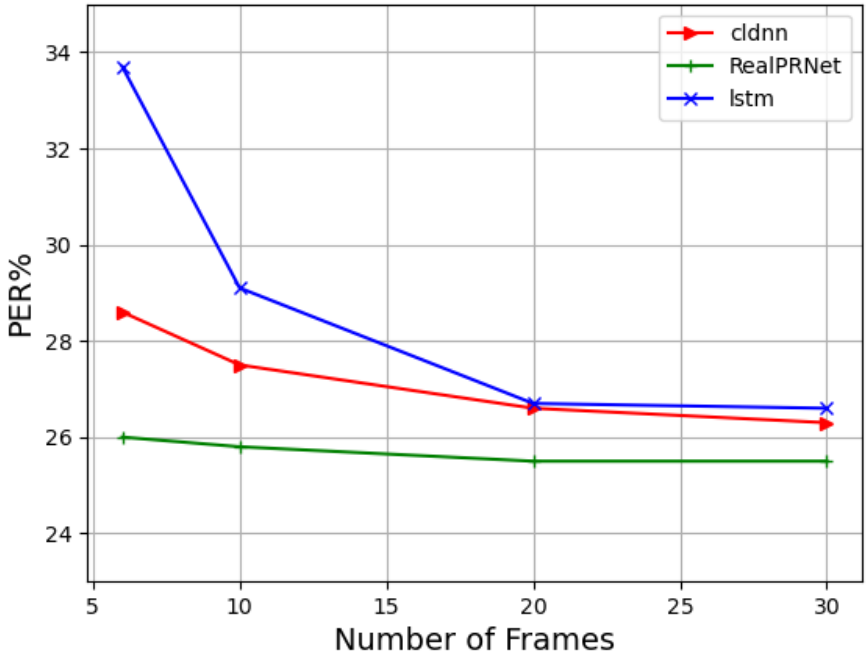


Figure 3.9: Phoneme Error Rate per Frames

The RealPRNet applies two different types of temporal modeling to x_t, the separate temporal modeling performed by LSB and unified temporal modeling performed by LSTM. Firstly, we explore the difference between these two temporal modelings that may affect the performance of the neural networks. The LSTM has been removed from the RealPRNet, and the new network is named 'RealPRNet-separate'. By comparing RealPRNet-separately and CLDNN, the result in Figure. 3.10 shows that the separate temporal modeling alone cannot outperform the unified temporal modeling. Thus, the performance of RealPRNet is benefited when the two temporal modelings are used together. The PER of RealPRNet

under different latency scenarios (different values m and n), shown in Figure. 3.10, illustrates that it outperforms both temporal modeling network architectures when used individually.

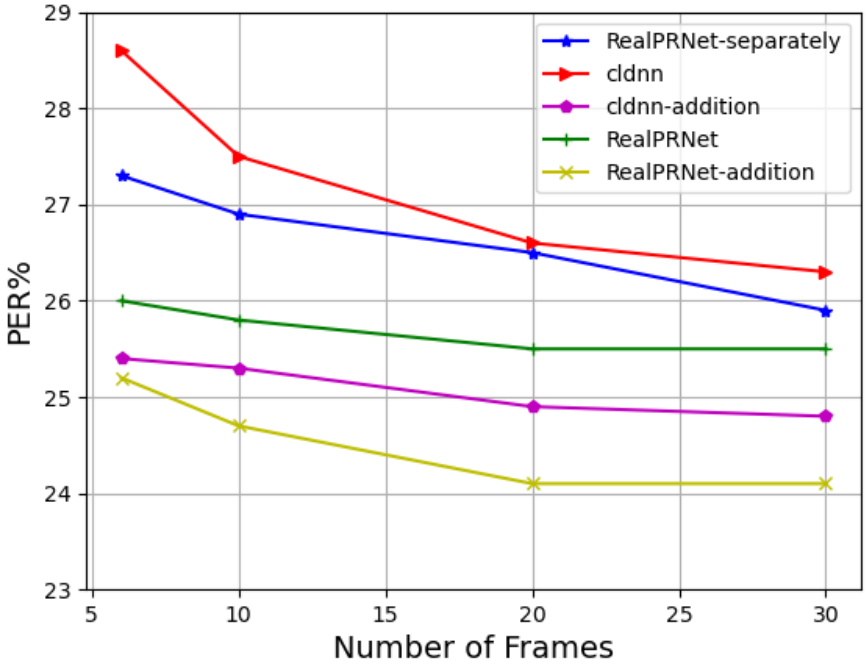


Figure 3.10: Varies Networks Performance in Phoneme Error Rate

As illustrated in the previous section, the performance of the network can be further improved by multi-scale feature addition techniques. We explore this technique with our RealPRNet and compare its performance with the strongest baseline CLDNN model with feature addition structure and also the previous networks without feature addition. As shown in Figure. 3.10, the performance of both RealPRNet and CLDNN with multi-scale feature addition has been improved. The additional short-term feature gave complementary information to the intermediate input features, which forced the networks to focus on the current frame f_t. Thus, RealPRNet outperforms other networks in most of the latency scenarios. In particular, it can achieve an additional 4% relative improvement in the worst scenario in real-time PER evaluation.

3.5.4 The Buffer Occupancy in Real-time Application

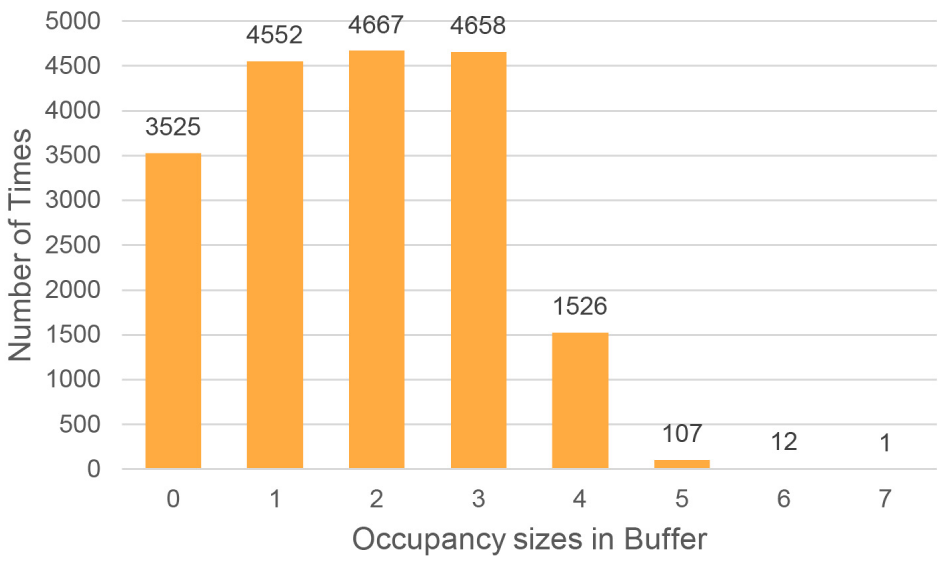


Figure 3.11: Output Phoneme Buffer (B_3) Occupancy

The buffer mechanism is used to stabilize system output and ensure that the delay of the output is within the tolerance range. The occupancy status of B_3 of our experiment in run-time is shown in Figure. 3.11. In ideal case, $[P_t, \cdots, P_{t-br+1}]$ is queued to the B_3 for every $br \times A$ ms, the B_3 dequeues the first values for every A ms.

In our experiment, the B_3 occupancy status is recorded after every dequeue, and the value of br is 4 in our experiment. Thus B_3 occupancy sizes should be evenly distributed in $[0, 1, 2, 3]$. 90% of our test cases are in this range which shows that our system can work in real-time situations. Most of the errors in B_3 occur in 0 and 4 (number of occupied units) because the computer program is unable to use the exact A in each step during the run time (e.g., the A fluctuated in a range from 10.3ms to 11ms). Other cases are caused by computational fluctuations of other parts of the phoneme recognition subsystems. This buffering mechanism ensures our proposed scheme is much more efficient to the extent that

it can be implemented in real time.

3.5.5 Animation Quality Assessment

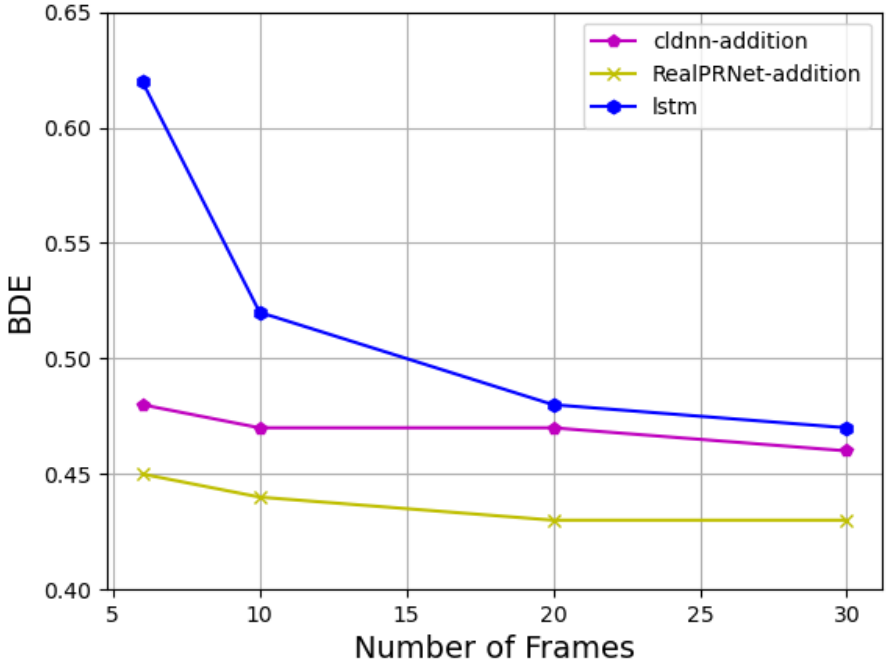


Figure 3.12: Block distance error comparison

In this section, we evaluate the quality of the output speech animation of our system.

First, we compare the trajectory curve in the viseme field produced by the recognized phoneme streams predicted by various reference networks and RealPRNet with the trajectory curve produced by the ground truth model. The block short edge length is used as a distance unit to calculate the BDE between the recognized phoneme-produced curve and the ground truth curve at each time instance. The result is shown in Figure. 3.12, where the x-axis in the figure represents the number of frames used in input features, and the y-axis represents the block distance error in the basic unit. The RealPRNet also achieves the minimum error under different latency scenarios, which is 27% or less than other networks.

In addition to the numerical evaluation, we also organized a group consisting of 30 participants in total to watch our system-produced speech animation and ask them whether they felt the animation was believable or not (is the audio able to synchronize with the 3D model lip movement animation). 29 out of 30 participants believe the audio has synchronized with the lip movement.

Figure 3.13: Speech Animation Visual Comparison with Human Performance and Current State-of-the-Art

We also compare our result facial model's viseme with the current state-of-the-art and real human performance. We take the viseme screenshots at the peak of each phoneme pronunciation in an utterance to compare the results visually. As we can see in Figure. 3.13, all results are comparable.

3.6 Conclusions

In this chapter, we proposed an audio-driven believable speech animation production system with a new phoneme recognition neural network called RealPRNet. The system can produce

a competitive real-time speech animation with only raw audio input.

The RealPRNet has performed better than the strong baseline model in phoneme recognition, and the speech animation system is easy to implement in most of the existing virtual avatars. The real-human-like speech animation needs a lot of effort on the pre-train model with both video and audio data, even post-edit from the artist. Our framework focuses on producing believable real-time speech animation with simple implementation, which also gives the user high freedom for online post-effect editing.

Our proposed animation production framework has implemented this method to generate the viseme animation for all the dialogue in the script.

Chapter 4

Auto Cinematography with Fidelity

4.1 Introduction

In this chapter, we demonstrate our proposed method for rule-based auto cinematography. In contrast to related works, we introduce a new optimization metric, fidelity, to determine the similarity between the visual content captured by the cameras in the animation and the respective textual content in the input script.

Many animation production-related software and tools have been developed to help animators produce better animations in less time and with fewer resources. With the latest developments in AI technology, many jobs in the animation production process can be achieved by a computer. Ref. [3] shows that simple animation can be generated automatically by using an animation movie script with a small amount of human effort in the adjustment. Thus, ordinary people also have the opportunity to create films on their own with the help of modern technologies by writing their stories into a specific format script [43]. The bar for animation movie making has been lowered with the application of these AI advances to replace part of the human labor in the whole filmmaking process. In this process, transforming the cinematography stage into an automatic process, which is called "Auto Cinematography", is one of the attractive areas because it can require substantial knowledge of the relevant reserves [44] and a considerable amount of time for the manual placement of these cameras

in the virtual 3D environment.

From the perspective of video editing, using the writing process to establish the video-making process has been investigated in [45–47], which assumes that online video resources can be utilized to put together video montages that can match with the text input, and then the selected shots are assembled by optimizing cinematographic rules to generate the final video output. However, the above approaches cannot be applied directly in the filmmaking process for a screenplay as the combined online shots can only vaguely visually present the screenplay but are hard to meet the standard of an animation. Ref. [3] targeted for 2D animation, binds script-writing with performance capture and post-processing edits, to achieve greater efficiency and flexibility for the production process. However, the resultant video was simply shot with the characters placed in a given 2D plane, without sufficient optimization of what and how to shoot and select the most appropriate shots and visual content for the script. Ref. [48, 49] present the automatically generated camera sequences that follow cinematic rules or conventions that can be utilized in this problem. Ref. [50] shows a more diverse computational cinematographic approach, including not only the information in the frames captured by the cameras but also the parameters of the camera placement and angle. Furthermore, Ref. [5] adds the director hints into the auto cinematography optimization process, and Ref. [51] learns the camera usage and behaviors from the existing films. The above optimizations for camera selection are aimed at the aesthetic aspects of cinematography. However, simply considering the rules of cinematography is not sufficient to ensure that the scenes captured express the content of the script.

If achievable, bridging script and scene, visual, and text brings benefits to improve the capacity of auto cinematography. Therefore, in addition to the existing aesthetic model described in the previous section, we introduce an innovative new assessment model, the fidelity

model, into the cinematography optimization process to determine whether the content expressed by the two different media, video and script, is consistent. Apparently, the proposed auto cinematography needs to satisfy the following two conditions: the output animation (1) maintains reasonable fidelity of the script and (2) follows cinematic rules with cinematographic aesthetics. We also designed and developed an animation production framework to better demonstrate our proposed optimization method.

Under such a background, we propose T2A, a framework to help speed up the script-to-animation video process and minimize manual adjustments. We introduced a new dimension, fidelity, in the auto cinematography optimization process to evaluate the quality of animated videos instead of using only aesthetic models. By embedding the fidelity model generated with the latest video understanding advances [52] in T2A, our cinematography optimization algorithm determines the comprehensiveness of the output video and its fidelity to the script. T2A combines the aesthetics with the fidelity requirements into a unified computational cinematography framework and maps the original problem into an optimization problem that seeks to select the best options of camera settings to achieve the quality expectations for the user requirements. The optimization problem is carefully designed so that it can be solved by dynamic programming most efficiently computationally. While in the extraction of script information, We use the current state-of-the-art NLP technology [53], including subject, predicate, object, adjective, emotion, dialogue, etc. However, since the current NLP technology still has some limitations, We still need to manually fix the errors in the process of extracting information.

The contribution of our work can be summarized in the following aspects:

1. We introduce a new evaluation metric, fidelity distortion, that allows auto cinematography to produce videos that are more consistent with the original content. This metric

provides a novel aspect to assess the quality of the resulting animation.

2. We propose a numerical model to calculate fidelity distortion while keeping the delay for the optimization process within reasonable limits. Such a model is developed by analyzing the behavior of state-of-the-art video understanding techniques.
3. We develop a novel text-to-animation framework, T2A. It not only builds a link between the script and video content, but also reduces the production time in 3D animation by 74%.

To the best of our knowledge, T2A is the first framework that introduces the fidelity distortion factor as a new perspective in the optimization process of auto cinematography. The rest of the chapter is organized as follows: Section 2, overviews the related work to this study, such as computational cinematography, video editing, and video understanding; Section 3, demonstrates the overall framework, the problem formulation, and the dynamic programming solution; Section 4, discusses the impact of the possible error in video understanding to the whole framework and the experimental results; Section 5 concludes the work and the last section discusses the limitation of our work and future works.

4.2 Related Work

Automatic 3D animation filmmaking can be seen as the combination of two main research topics, 3D avatar animation from the script and computational cinematography in the virtual environment.

From more than a decade ago, the former topic was achieved by transforming prepared stories into unique scripting languages and allowing the 3D avatar to automatically generate the corresponding animated performances in the virtual environment based on these

scripts [54–56]. The advantage of the script language [3, 57] is that it provides accurate character action, state, position, time, and other information to generate the corresponding animations automatically. The animator can modify the detail setting easily by using such a technique. However, the prep work for this approach requires a knowledgeable animator who spends a considerable amount of time converting the original animation script to such special scripts. The T2A takes advantage of NLP [58] to generate the action list from the original animation script directly without manual labor. Since the textual information in the script does not contain all the necessary information to generate the corresponding animation (e.g., distance traveled by the character) and the current NLP is not perfect, the animator can manually adjust the action list after it has been generated, which only requires a few minutes to complete.

The camera configuration and placement are crucial in the shooting phase of the movie production process. The views captured by the camera directly affect the quality of the production. The computational cinematography [49,51,59] integrates the automatic camera placement and video editing techniques to complete the final video. The automatic camera placement process follows roughly two steps. Firstly, using the cinematography guidelines as constraints the selection of views is found. Different cinematography rules can be applied to the film-editing under various scenarios, such as dialogue scenes [59], first-person [60], and cooking [45]. The advantage of these methods is that the generated video is more closely aligned with the aesthetic standards of the desirable cinematography guidelines. Secondly, directly learning from existing movies these camera's behaviors [51, 61–63] are mimicked. The advantage of these methods is that the optimal camera solution can be matched to the existing successful camera settings based on the state of the characters and the scene. The problem with this approach is that similar content can be produced under totally different

scenes, and it cannot guarantee that obstacles in the new scenes will not block the camera path.

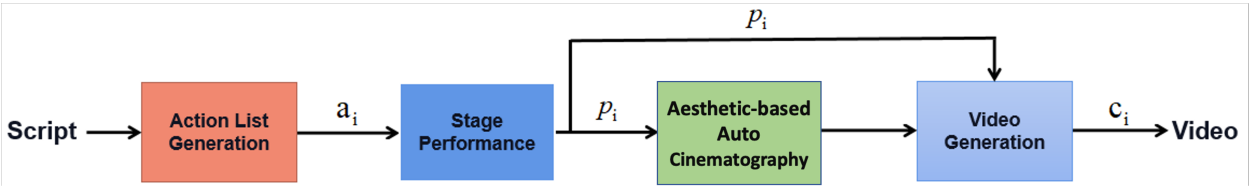


Figure 4.1: Existing Framework for Auto Cinematography

As with the issues and challenges mentioned in [64], almost all of the computational cinematography works according to our knowledge optimize the camera path from the perspective of the cinematography guideline. These can also be viewed as a subjective perspective because most of the great films break these rules. After all, the directors are not following them. We, therefore, propose an evaluation perspective, fidelity distortion, that can objectively assess the quality of the resulting video. The fidelity stands for the consistency of the video and original script content, which has been ignored in both optimizations mentioned above. It is inspired by the recent works [3, 47, 56, 59], which achieve this consistency by adding notes to all pre-shot video clips and comparing the script with the notes at the editing stage. Thus, we introduce the fidelity model to achieve this. The animation is generated by the action list derived from the original script, and the animation video aims to visually optimize the actions of the characters in the virtual environment. Thus, the fidelity model helps the framework maintain this consistency by using the video caption and action recognition model [65–68] as an objective viewer, evaluating the video content whether it has successfully visualized the corresponding actions in the action list. One of the popular areas of research in recent years, video caption [69–71] can describe more and more detail in the video, and the accuracy and recognition range of the action recognition [72, 73] have

been improved. Although none of these technologies are perfect at the moment, we believe that as they continue to improve, our proposed fidelity model will be better able to achieve its goals.

4.3 Framework Design

In a typical auto-cinematography system (as shown in Figure. 4.1), the process from the input script to the output video contains the following steps, namely, action list generation, stage performance, camera optimization, and video generation. The action list generation module analyzes the original animation script to obtain the corresponding characters' chronological action list $\{a_i|i = 1, 2, ..., N\}$, where a_i is the ith action object in the scene, and N is the total number of action objects. It is important to realize that multiple characters might perform simultaneously (e.g., two persons are fighting with each other, or a mom is hugging her daughter). Thus an action object may contain multiple characters in the same scene. In the stage performance step, the input $\{a_i\}$ is transformed into the corresponding stage performance data $\{p_t|t = 0, 1, .., T\}$, where p_t is the character stage performance at time t and T is the total performance time determined by the action list. Specifically, for each a_i, the corresponding performance can be denoted as $\{p_{t_{a_i}}, ..p_{t+1_{a_i}}, ..p_{t+l_{a_i}}\}$, where l_{a_i} is the action duration of a_i. It is important to realize that the different action objects can overlap with each other. For example, when two events occur simultaneously, both of them need to be presented to the audience. This requires that there are multiple virtual cameras available in the 3D scene that can record all the views for every character from various angles. In the camera optimization step, all the available views are considered to calculate the optimized camera $\{c_t\}$ for each time t. The video generation step assembles all the video

frames captured by camera $\{c_t\}$ at time t and outputs the final video.

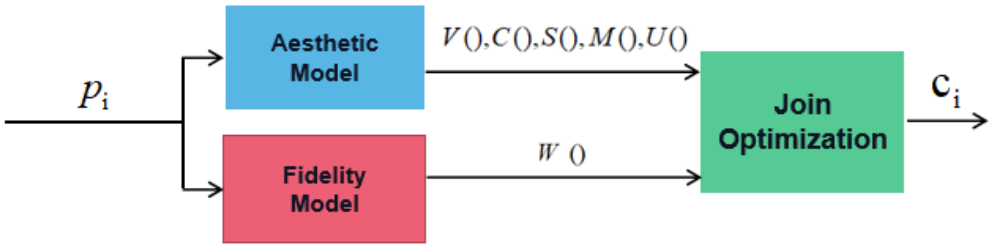


Figure 4.2: The updated camera optimization model in T2A (only reflects the camera optimization module shown in Figure. 1)

In almost all the literature that we can find so far, only aesthetic factors are considered during the camera optimization process. Various types of aesthetic distortion are defined according to the grammar of cinematography or editing, and the purpose of optimization is to find the camera path with minimum aesthetic distortion. Apparently, this solution only meets the second condition (as mentioned in section 1) in order to convert a script to an animation, while it does not satisfy the first condition that requires the output animation to maintain reasonable fidelity of the script. In order to measure the fidelity level of a video compared to the script, a mathematical model is required to be built into the camera optimization framework.

In this work, we make a bold assumption that a mathematical model can be found to approximate the fidelity relationship between a video and its associated script (i.e., a_i). In other words, with any action a_i and any selected camera $a_{t_{a_i}}$, the fidelity between the action and the video generated from this camera at time t can be obtained from this approximation model. In Figure. 4.2, the camera optimization module has been updated in our proposed framework with three new modules, namely, aesthetic model, fidelity model, and optimization. The task of the aesthetic model is to provide a quality evaluation from an aesthetic

point of view for each admissible virtual camera at time t to the optimization engine, the task of the fidelity model is to provide fidelity evaluation for each admissible virtual camera at time t to the optimization engine, and the optimization engine considers all inputs and makes the optimal choice for the camera selection. In the following, these three modules will be demonstrated in detail. It is essential to realize that the purpose of this work is to showcase a framework that combines aesthetic and fidelity modeling in the camera optimization process, which is the first trial in the field. It is not our intention to claim that the proposed aesthetic and fidelity models are the best. Instead, we encourage the readers to develop a better aesthetic or fidelity model to make the framework perform better, as the framework is generic to adapt new models into the optimization process.

4.3.1 Aesthetic Model

In this section, we discuss the aesthetic evaluation model, that is, to emphasize the distortion caused by camera planning and measured by the cinematography guidelines. Aesthetic distortion is widely used in existing state-of-the-art automatic camera algorithms [45, 49, 59]. Although there is no single standard in cinematography to evaluate the quality of camera setting and shooting results, we consider several essential factors such as character visibility, character motion, camera setting configuration, screen and motion continuity, and shot duration, to form an approximation of the perceptual aesthetic error following the traditional cinematography guidelines. Again, the model proposed here is just an example as the formation methodology is extensible to include more factors that need to be considered in the cinematography process.

4.3.1.1 Camera Placement

Positioning cameras in 3D space to shoot frames that meet 2D constraints is a 7-degree of freedom(dof) problem consisting of: the position of the camera(3 dofs), orientation(3 dofs), and focal length (shot size). In Practice, the optimization of the seven dimensions is very challenging due to computational power limitations. To simplify the problem but without

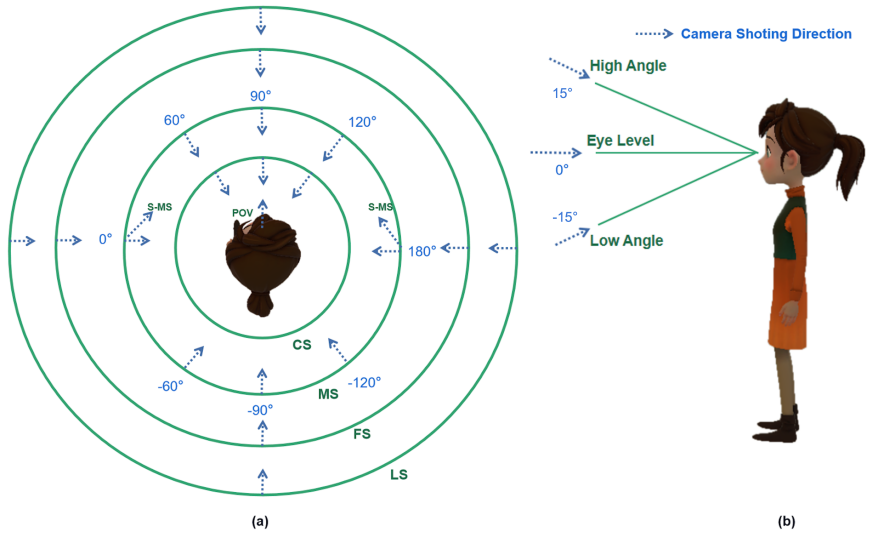


Figure 4.3: Default Camera Placement. (a) camera placement for different shot sizes and profile angles. (b) camera placement for different camera heights

the loss of generality, we narrow down the 7-of infinite search space to countable discrete camera configurations according to the camera placement of classic movies. Only cameras used with up to two people are considered because shots with more than two people in the field of view can often be replaced by single-person shots. Ref. [74] proposed a two-people camera placement mechanism. We adopt their model of two-people shots. Figure. 4.3 shows the placement of the cameras in our implementation. Each camera maintains the relative position of the character associated with it during the stage performance process except for the point of view (POV) camera. The POV camera follows the head movement of the character.

4.3.1.2 Single Frame Distortion Cost

In this section, we describe the distortion in the single frame capture by camera c_t (selected camera at time t) that follows cinematography guidelines.

Character Visibility This cost evaluates the character visibility in the c_t and it is determined by two factors: (1) the ratio, r_k, of the size of the character k ($k = 0, 1, ..., K-1$ for the total of K characters in the story) in the frame to total the frame size. r_k represents how easily the audience perceives the character in the frame. When multiple characters appear in the c_t view, the c_t always considers its bounded character as the more significant one (low value of $I(c_t, k)$, where $I()$ is the character priority). (2) $I(c_t, k)$, which depends on camera c_t and character k, represents this correlation by giving different weights to different characters and camera combinations during the calculation. Thus the cost function for character visibility ($V(c_t)$) can be represented as:

$$V(c_t) = \sum_{\text{character } k=0 \rightarrow K-1} I(c_t, k) \cdot r_k \tag{4.1}$$

Character Action is value describes whether the character has a_t (action at t) or not. In general, the audience is more likely to notice the characters in motion. Thus, if the character k is acting at t, there is a greater probability that the k bounded cameras have been selected. Hence the cost function can be represented as:

$$A(c_t) = \begin{cases} 0 & c_t \text{ bounded character k has action at time t} \\ 1 & \text{otherwise} \end{cases} \tag{4.2}$$

Camera Configuration Different camera configurations serve different purposes in

movie shooting, and the distribution of use is varied. For example, the MS (median shots) are used most often while the character performs the general action. However, when the character performs an action, such as seeing around, the LS (long shot), S-MS (surround environment camera), and person-of-a-view camera are often the better option. On the other hand, different actions prefer the camera shooting in different directions. For example, walking and running action can shoot from both the front and back of the character without too much distortion. However, speaking action can cause higher distortion when shooting from the back of the character than the front and side. Thus, the camera configuration distortion depends on the action type that can be derived from the action object (i.e., a_i) of time t, the camera position p and shooting direction d can be derived from c_t. We use the $\phi_C()$ function to describe this distortion calculation process [48], and use $\tilde{a}_t$ to represent the action object in time t. Thus the cost function of camera configuration can be represented as:

$$C(c_t) = \phi_C(p_{c_i}, d_{c_i}, \tilde{a}_t). \tag{4.3}$$

4.3.1.3 Frame to Frame Distortion

Ref. [48] has described multiple cut quality measurement metrics developed from the cinematographic guidelines. We have selected the most useful ones, which are modified and implemented here to calculate the quality between the captured views at times t and $t+1$.

Screen Continuity The Visual-spatial continuity in the video is essential to prevent disorientation for the viewer. Many classic cinematography rules describe the importance of the aspect ratio, such as the 180-degree rule [75]. The screen continuity is the summary of each single character's position change in the frame.

The cost function is defined as follows:

$$S(c_t, c_{t-1}) = \sum_{char\ k=0}^{K} v(k, c_t) \cdot \phi_S(p(k, c_t) - p(k, c_{t-1})) \tag{4.4}$$

where $p(k, t_i)$ and $p(k, t_{i+1})$ represent the kth position in the frame captured by c_t and c_{t+1}. while $v(k, c_t)$ determining whether character k is visible in the view of c_t or not, that is,

$$v(k, c_t) = \begin{cases} 1 & \text{character k is visible in camera } c_t \\ 0 & \text{otherwise.} \end{cases} \tag{4.5}$$

The minimum penalty of position change is 0 and increases when the distance between $p(k, t_i)$ and $p(k, t_{i+1}$ increases. If k only appears in one of the frames, the maximum penalty of 1 will be used. The non-linear function $\phi_S()$ represents this property [48].

Moving Continuity If an ongoing action changes the direction of a character before or after the view change, this can create disorientation in the audience. The moving continuity cost quantifies the penalty in this aspect as follows:

$$M(c_t, c_{t-1}) = \sum_{char\ k=0}^{K} v(k, c_t) \cdot \phi_M(m(k, c_t) - m(k, c_{t-1})) \tag{4.6}$$

where $m(k, c_t)$ and $m(k, c_{t-1})$ is the motion direction vector of the character in frame t and $t-1$, respectively, captured by the associated camera. The non-linear function $\phi_M()$ takes $m(k, c_t)$, $m(k, c_{t-1})$ as inputs and returns the penalty [48]. This penalty also increases as these diverge from each other. If k only appears in one of the frames, the maximum penalty of 1 will be applied.

4.3.1.4 Shot Duration

Shot duration is closely related to the concentration of the audience's attention, and it is an important element of film editing style. In general, the shorter the shot duration, the more intense the content on the screen, and the easier to grab the attention of the audience. In [76], the researcher found that the shot duration distribution is associated with the style of the movie and can be described as a log-normal distribution. To simplify the problem, we allow the average shot duration ($\overline{u}$) to be set for each scene to control the shot duration distribution or use the default value (the general $\overline{u}$ learning from existing movies [76]). The $\phi_U()$ non-linear function has the shape of the standard inverted log-normal distribution [48] with minimum value (penalty $y = 0$) at $x = \overline{u}$. Let us denote by q the longest allowable shot duration; the shot distortion is then defined to penalize the frames in the range $[t - q, ...t]$ that change cameras, as:

$$U(\overline{u}, c_t, c_{t-1}, .., c_{t-q}) = \phi_U(\overline{u}, c_t, c_{t-1}, ..., c_{t-q}). \tag{4.7}$$

By adding all the factors mentioned above together, the total aesthetic distortion D_a can be calculated by the following equation:

$$\begin{aligned} D_a = & \sum_{t=0}^{T} [\omega_0 \cdot V(c_t) \\ & + \omega_1 \cdot C(c_t, \tilde{a_t}) + \omega_2 \cdot A(c_t) \\ & + \omega_3 \cdot S(c_t, c_{t-1}) + \omega_4 \cdot M(c_t, c_{t-1})] \\ & + \sum_{t=q}^{T} (1 - \omega_0 - \omega_1 - \omega_2 - \omega_3 - \omega_4) U(\overline{u}, c_t, c_{t-1}, .., c_{t-q}) \end{aligned} \tag{4.8}$$

where ω_0, ω_1, ω_2, ω_3,$\omega_4$are the weights for each distortion (range of 0 to 1).

4.3.2 Fidelity Model

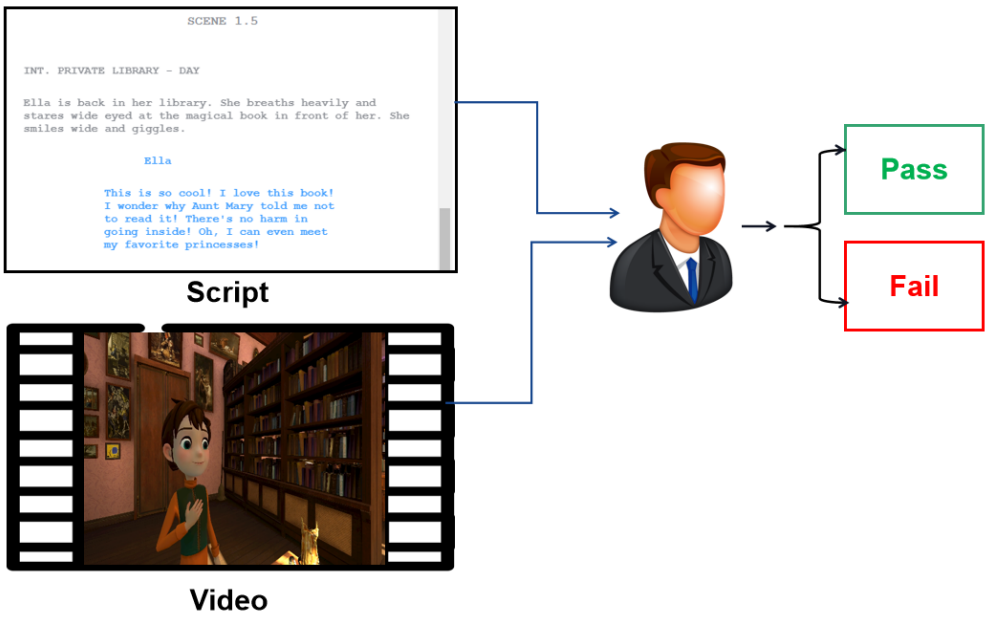


Figure 4.4: Comparsion of a script and a video by a human or human-like agent to see whether they match

The fidelity model is the essential element of our proposed auto-cinematography approach, which assures that the generated matches the video input script, although this element has been missing in the past efforts of auto-cinematography. In an ideal case, as shown in Figure. 4.4, there is a human-like agent that has comprehension intelligence similar to humans. Thus, it is able to determine whether a generated video is good enough to reflect the comprehension of the script. Clearly, a lot of research progress reviewed in section 2 has been made in this direction. However, the current state-of-the-art is still far from being widely utilized due to its low performance and accuracy. To address this challenge, an approximation method as shown in Figure. 4.5 is considered to compare each action generated

from the action list generation module (shown in Figure. 1) with a reconstructed action recognized by the corresponding output video-related to the input action using the accumulated error to represent the overall fidelity level considered in the ideal scenario. Therefore, the original text-video matching problem is approximated and converted into a text-text matching problem with the assumption that the video action recognition engine can achieve a reasonable quality.

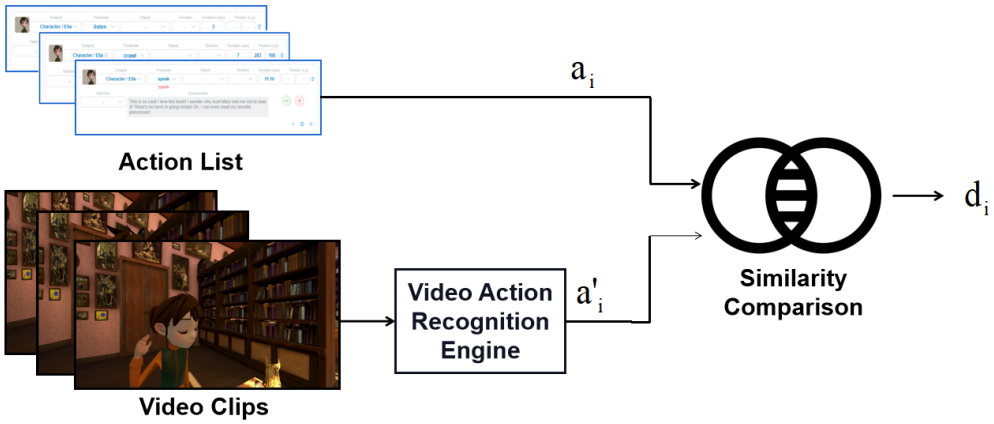


Figure 4.5: Using a video action recognition engine to convert a video clip into a text, and then compare the similarity between two actions in text format

Let us denote by m_j the jth camera of the admissible cameras set, a'_i the word or phrase that describes the action obtained from the video action recognition engine (which is able to recognize the video generated from input action a_i), d_i the similarity between the textual description of a_i and a'_i, then d_i can be measured by:

$$d_i = \begin{cases} 0 & \frac{G(a_i)\cdot G(a'_i)}{\|G(a_i)\| \times \left\|G(a'_i)\right\|} \leq \mathsf{Th}_G \\ 1 & \text{otherwise} \end{cases} \tag{4.9}$$

where Th_G is the threshold to admit that a_i and a'_i refer to the same action, and function

G is the Glove word embedding model of [77]. This way, the fidelity level of a generated video compared to the input script can be approximated by the average of all the action similarities obtained as:

$$F_j = \frac{1}{N} \sum_{i=1}^{N} d_{i,j}. \tag{4.10}$$

It is important to emphasize that the equation above is just an approximation of the real fidelity level, as there is noise introduced during the whole process by the video action recognition engine and the textual similarity comparison mechanism. The human-involved simulation observed accuracy is 87.3% in our experiment by comparing the outcome calculated by Eq. 4.10 and human subjective judgment with a pre-defined threshold for acceptance. We asked participants to watch 1000 video clips (randomly selected from the data set) and asked if they could visually understand the actions of the characters in the video clips. Even with human subjective judgment, in some cases, it is difficult to accurately distinguish the actions of the characters in the video clips (127 in our experiment), which are due to the action type and camera angle,(Figure. 4.6 shows some examples). The current action recognition engine has not yet reached the same level of subjective judgment as a human. However, given the current development speed of AI technology, the that accuracy of video action recognition has been significantly improving these years. It is reasonable to assume that the accuracy of action recognition will be further improved in the near future. Hence, from the modeling point of view, it is feasible to adopt this approximation mechanism to quantify the fidelity level.

On the other hand, another obstacle appears during practical real-time applications. Due to computational constraints, it is not practical to embed the video action recognition engine inside the camera optimization process to calculate F_j for all the admissible camera

parameters. Thus a further approximation is required to make the fidelity model practical.

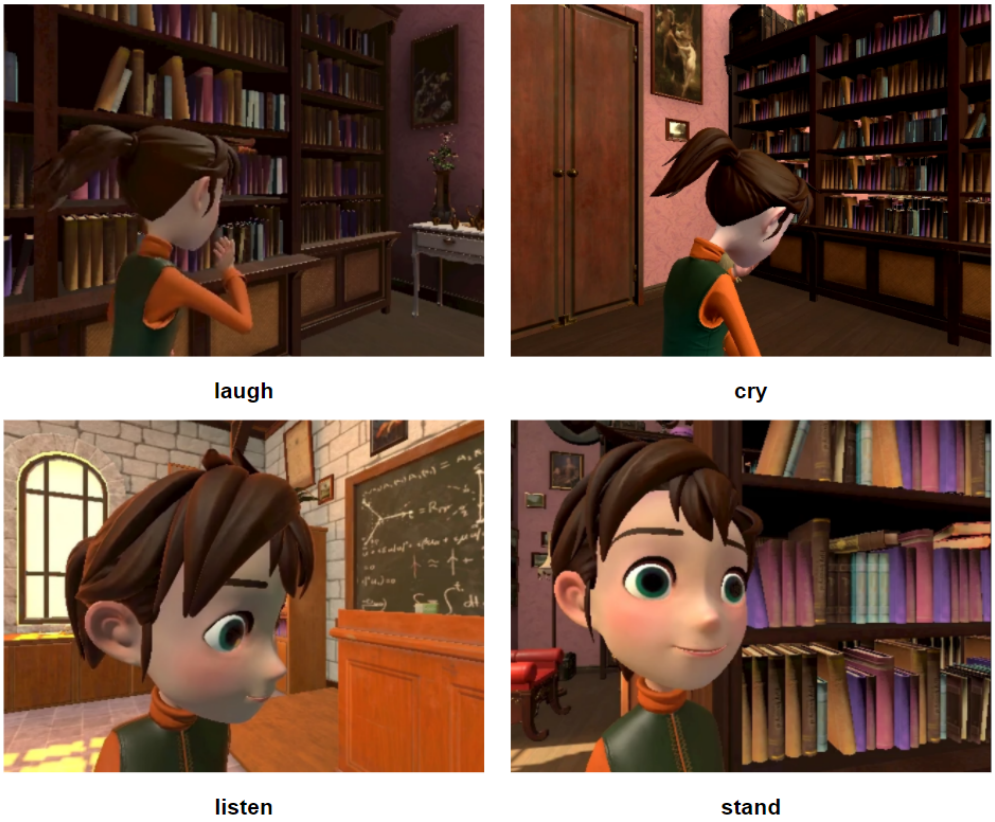


Figure 4.6: The action 'laugh' and 'cry' captured from the back of the characters (top). The actions 'listen' and 'stand' were captured from a front-side close-up shot (CU) camera (bottom)

When looking closely at the performance of the video action recognition engine, one phenomenon of failure is quite dominant, that is, the full (or partial) occlusion of some characters in action captured from various camera angles may cause the engine not to be able to recognize the supposed activity successfully (see examples of occlusion in Figure. 4.7). Hence it is intuitive to investigate whether the degree of occlusion of characters according to all admissible camera parameters is correlated to the $d_{i,j}$ obtained in equation 4.9.

Let us denote by $o_{i,j}$ the occlusion percentage of all characters involved in a_i shot by camera m_j, where J is the total number of admissible cameras. Then the average occlusion

O_j can be calculated by:

$$O_j = \frac{1}{N} \sum_{i=1}^{N} o_{i,j}. \tag{4.11}$$

According to simulations as shown in Figure. 4.8, there is a high correlation between F_0

Figure 4.7: Example of different occlusion degrees of characters

and O_0 (i.e., the fidelity and occlusion level measured by the 0th camera of the admissible cameras), and their relationship can be approximated by a linear function. Figure. 4.9 demonstrates the fitting errors (σ) of all admissible camera placements and settings in the simulation, and they are all in the range [0.04, 0.11], which is small enough to demonstrate that the high correlation between occlusion and fidelity level holds for all camera settings in the simulation. More details of the fidelity model simulation can be found in the section 4.3.4.

Let us denote by D_f the fidelity distortion; then D_f can be modeled and represented by a function of the selected camera at each time t, as the object occlusion level is determined

once the camera is specified at a certain timestamp. Therefore, D_f can be calculated by:

$$D_f = \frac{1}{T}\sum_{t=0}^{T}[\alpha O(c_t) + \beta], \tag{4.12}$$

where $O(c_t)$ is the occlusion measurement function for all subjects and objects for the selected camera c_t at time t, α and β are the parameters that can be derived by fitting the (O_j, F_j) pairs by a linear function.

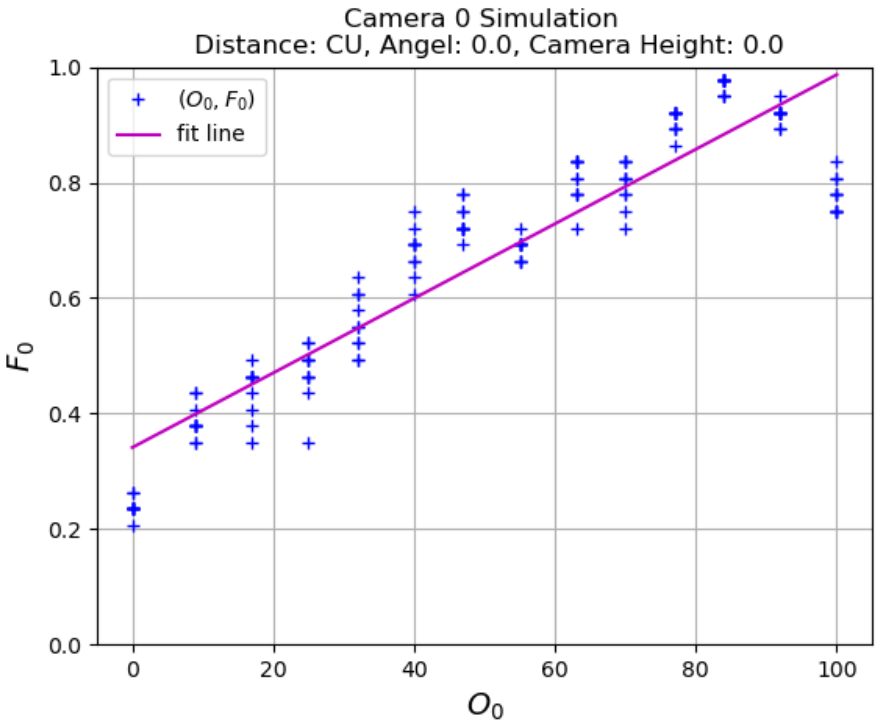


Figure 4.8: The plot of pairs of (O_0, F_0) in dots for the 0th default camera and the linear function $F_0 = \alpha O_0 + \beta$ to fit the dots

So far, the fidelity factor has been approximated into a mathematical model that is only determined by the dynamic selection of the camera. Thus, it is possible to consider this factor in the camera optimization process. It is important to emphasize that there could be other ways to approximate the fidelity between a script and a generated video. In this work,

the objective is to showcase that such a model is possible, and the effect of having such a model will be demonstrated in the next section with evaluation results.

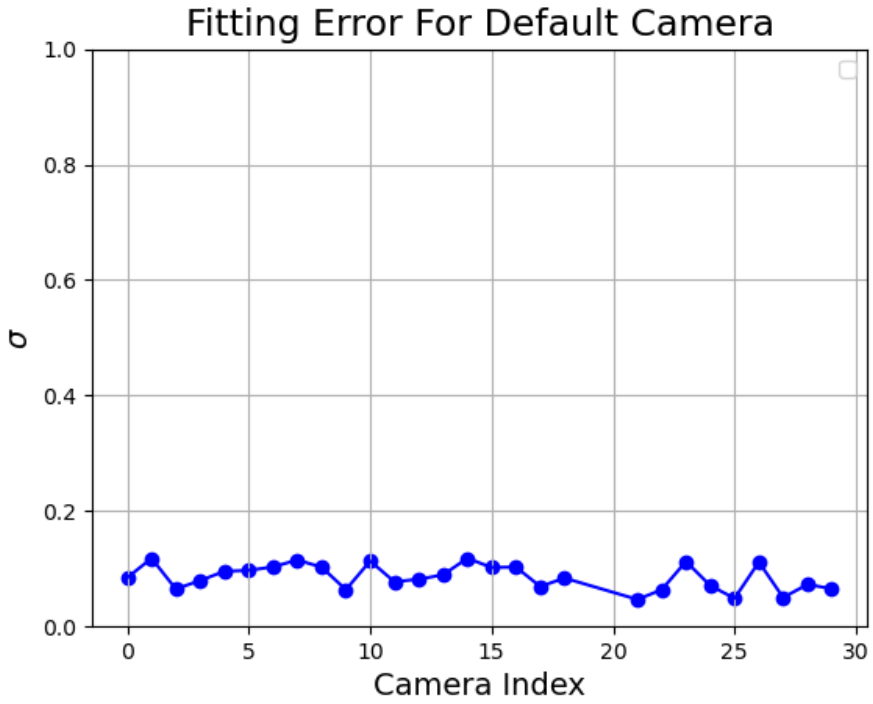


Figure 4.9: Fitting error for all the default cameras used in our experiment, except the median shot (MS) side cameras (19, 20) and the POV camera (30)

4.3.3 Joint Optimization

As mentioned above, both the aesthetic and fidelity models are utilized. Although the models can be refined and extended by considering more factors, as long as the factors can be represented by a function of camera parameters, we can optimize both aspects in a joint optimization framework. By using a weighting factor λ between [0, 1] to bridge both models, the total distortion can be represented as follows:

$$D = (1 - \lambda)D_a + \lambda \cdot D_f. \tag{4.13}$$

When λ is set to a value close to 1.0, the fidelity distortion is more important, otherwise for λ close to 0, the aesthetic distortion becomes more important. We leave the determination of λ to users so that they can decide according to the real scenarios and applications. The problem of minimizing the total distortion can be written as follows:

$$\begin{aligned}
\min\ & (1-\lambda)\cdot\sum_{t=0}^{T}[\omega_0\cdot V(c_t) \\
& +\omega_1\cdot C(c_t)+\omega_2\cdot A(c_t) \\
& +\omega_3\cdot S(c_t,c_{t-1})+\omega_4\cdot M(c_t,c_{t-1}) \\
& +(1-\omega_0-\omega_1-\omega_2-\omega_3-\omega_4)\cdot \\
& \sum_{t=q}^{T}U(\overline{u},c_t,c_{t-1},..,c_{t-q})] \\
& +\lambda\cdot\frac{1}{T}\sum_{t=0}^{T}[\alpha O(c_t)+\beta].
\end{aligned} \tag{4.14}$$

Our goal is to find the optimal solution $\{c_t^*\}$ such that $\{c_t^*\} = \text{argmin}_{c_t} D^*$. To implement the algorithm for solving the optimization problem, we define $z_k = c_k$ and a cost function $D_k(z_{k-q}, ...z_k)$, which represents the minimum total distortion up to and including the kth frame, given that $z_{k-q}, ..., z_k$ are decision vectors for the $(k-q)$th to kth frame. Therefore $D_T(z_{T-q}, ..., z_T)$ represents the minimum total distortion for all frames, and thus

$$\min_z D(z) = \min_{z_{T-q},\cdots,z_T} D_T(z_{T-q},\cdots,z_T) \tag{4.15}$$

The key observation for deriving an efficient algorithm is the fact that given $q+1$ decision vectors $z_{k-q-1},\cdots,z_{k-1}$ for the $(k-q-1)$st to $(k-1)$st frames, and the cost function $D_{k-1}(z_{k-q-1},\cdots,z_{k-1})$, the selection of the next decision vector z_k is independent of the se-

lection of the previous decision vectors $z_1, z_2, \cdots, z_{k-q-2}$. This means that the cost function can be expressed recursively as

$$
\begin{aligned}
& D_k(z_{k-q}, ..., z_k) \\
& = \min_{z_{k-q-1}, \cdots, z_{k-1}} \{D_{k-1}(z_{k-q-1}, \cdots, z_{k-1}) \\
& \quad + \lambda \cdot \frac{1}{T}[\alpha O(c_k) + \beta] \\
& \quad + (1-\lambda) \cdot \{[\omega_0 \cdot V(c_k) \\
& \quad + \omega_1 \cdot C(c_k) + \omega_2 \cdot A(c_k) \\
& \quad + \omega_3 \cdot S(c_k, c_{k-1}) + \omega_4 \cdot M(c_k, c_{k-1})] \\
& \quad + (1-\lambda) \cdot (1 - \omega_0 - \omega_1 - \omega_2 - \omega_3 - \omega_4) \cdot \\
& \quad U(\overline{u}, c_k, c_{k-1}, \cdots, c_{k-q})\}\}.
\end{aligned}
\tag{4.16}
$$

The recursive representation of the cost function above makes the future step of the optimization process independent from its past step, which is the foundation of dynamic programming. The problem can be converted into a graph theory problem of finding the shortest path in a directed cyclic graph (DAG). The computational complexity of the algorithm is $O(T \times |Z|^{q+1})$ (where $|Z|$ is the cardinality of Z), which depends directly on the value of q. For most cases, q is a small number, so the algorithm is much more efficient than an exhaustive search algorithm with exponential computational complexity.

4.3.4 Fidelity Model Simulation

A fidelity model was used to simulate the audience's recognition and understanding of a character's action in a movie, assuming that the performance of the character's action is

well conducted, it was expected that a typical adult audience can understand the action of a character successfully if the character is completely unobstructed in the video frame. It was discovered that the obscuring of the action might cause a challenge for the audience to recognize and understand the activity of the scene. On the other hand, the camera placements, range, and angle are all critical aspects that may influence the audience's perception. For example, the close-up (CU) camera is only able to capture the character's head and partial upper body in the frame and the camera with the angle from the back of the character is unable to convey visual information of the front side action to the audience, such as facial expressions like 'Smile'. Thus, a fidelity model was proposed to estimate the loss introduced during the film-making process caused by the occlusion of the character, which is highly relevant to the camera placement. The occlusion can be defined as the proportion of the size of the partial characters, in which the audience cannot see them in the frame caused by the obstacles between the camera and characters, to the size of full characters. Without loss of generality, two types of actions are considered in this experiment, self-action and interactive action. The self-action is an action made by the character himself/herself without a receiving object, such as a 'Jump' action, and the interactive action is the action that has one or multiple receiving objects, such as the 'Speak', which requires the camera to capture both the initiator and receiver of the action. Thus, the occlusion is calculated based on the subject (initiator of the action) for the self-action. For the interactive action, the occlusion is calculated based on both the subject and object (receiver of the action).

As mentioned in section 3.2, a video understanding and description neural network model[39] was considered as an objective viewer. An in-house data set with 15500 video clips (3-5 seconds each) was used to train and test our network model. The video includes 20 different character models, and each character model performs 10 different actions in 5

scenes and 5 different positions for each scene. Each of these action performances was captured by 31 default camera placements. The data set was randomly divided into a training set (80%) and a test set (20%) while ensuring that each camera placement had the same amount of data in the training and test set.

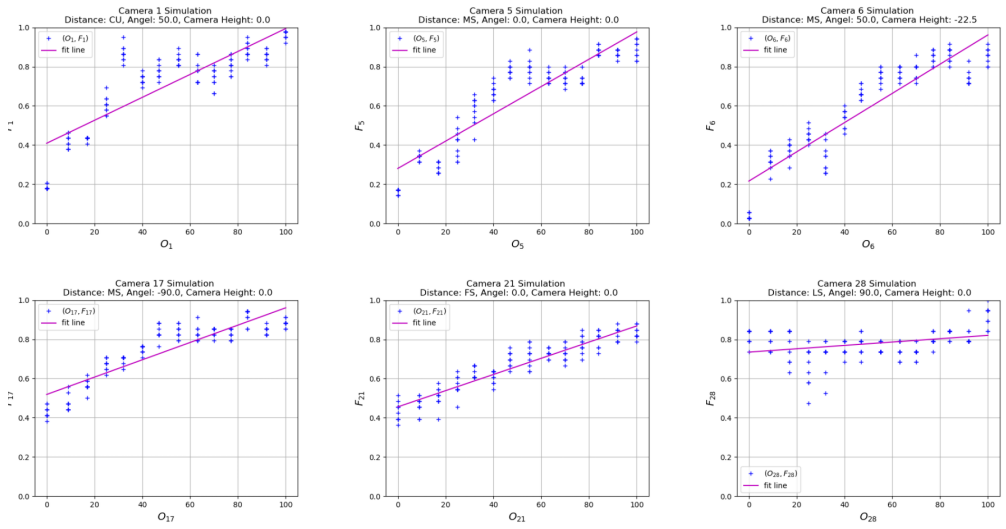


Figure 4.10: Examples of fidelity model simulation: it can be observed how the distance and angle of the camera may affect the simulation curve in various occlusions. The MS camera slightly off the front side (cameras 5 and 6) of the character provides the lowest Fidelity level. When the camera (camera 28) is too far away it is difficult to distinguish the action

In our experiments, the textual description corresponding to each action clip consists of 3 parts, the initiator of the action, the action itself, and the recipient of the action. Therefore, the video understanding network needs to parse out these 3 parts separately from the input video. The input video is first transformed into 30-50 frames (10 frames per second), and then the frames are feature-extracted by CNN, and we use resnet-152[42] as the feature extraction layer. Since most of the actions have temporal and spatial continuity, a single image is not a good judge of the character's actions. the LSTM layer is used to explore this temporal and spatial continuity in the input clips. In the whole frame, the character's action usually takes up only a part of the frame instead of the whole frame, so the attention layer is

effective in finding the part of the frame related to the character's action and discarding the irrelevant part of the frame such as the background, thus improving the accuracy of video understanding.

During the training process, no occlusion was applied to characters; during the testing process, different occlusion patterns were randomly generated and applied to mimic the scenarios of various occlusion degrees of characters. Cross-validation was applied to obtain the average fidelity level and occlusion for all these camera placements.

In Figure. 4.10, the simulation results of representative camera placements were demonstrated. These results all indicated that there is a strong correlation between calculated occlusion level and fidelity level. Interestingly the accuracy of action recognition decreases when the filming camera is either not at an appropriate distance, too far away, or too close to the subject, or not at a proper angle, from the side or back of the subject.

4.4 Experimental Results

In this section, the details of the simulation fidelity model are demonstrated, and the proposed framework is evaluated with the following three groups of experiments: (1) Check whether the automated workflow proposed by T2A adds value to users; (2) Compare the proposed camera optimization framework with the state-of-the-art solution without the fidelity model; (3) Investigate the proposed camera optimization framework in-depth with ablation studies.

We implemented the T2A system by putting together several tools and subsystems: first, a web tool is developed to handle the script analysis and action list generation with the support of AllenNLP [53]; second, an auto-staging and animation tool is built using Unity [78] to

transform the action list into character performance; third, a camera optimization algorithm is implemented in the back-end that can be called by the animation tool; fourth, the video action recognition engine is built following the work of sequence to sequence neural network for video caption [79] with additional attention layer; it is first trained and tested with the MSR-VTT public data set [80] and then trained using our own data set details in the section 4.3.4 for 2000 epochs. The Adam optimizer [81] is utilized with a batch size of 128, and the learning rate starts with 0.0004 and gets decreased every 200 epochs by multiplying it by the decay factor of 0.8. It takes around 8 hours to finish the whole training process based on the NVIDIA GTX 2080 TI GPU.

We have also developed several character models for our tool, and all of them have a standard set of action libraries to support the related animation performances. If an action described in the script or an action similar to it cannot be found in the action library, the animator can manually adjust the action list.

Table 4.1: The time consumption in automation on and off modes (professional users)

Script	Auto-staging		Auto-camera		Reduced Time (%)
	Off	On	Off	On	
TJ-1.1	88 min	18 min	28 min	15 min	83 min (71%)
TJ-1.2	107 min	20 min	29 min	16 min	100 min (74%)
TJ-1.3	80 min	18 min	24 min	13 min	73 min (70%)
TJ-1.4	120 min	22 min	34 min	18 min	114 min (74%)
TJ-1.5	115 min	13 min	34 min	19 min	117 min (78%)

4.4.1 Value of Automation

To evaluate T2A, five professional animators were invited to test drive the system, each of them was assigned a separate script with 1-2 minute screen time, and they were asked

to make a video by using the tool. The purpose of the test is to find out the differences between turning auto-staging and auto-camera functions on and off. In the staging phase, the user is required to select the relevant scenes and characters in our Unity tool according to the script and place the characters in the appropriate locations in the scene, and then set the character's actions, dialogue voice, and other relevant data sequentially by following the content in the script. After the staging phase, the characters in the virtual scene will be able to perform the story according to the script. In the camera setting phase, The user needs to place several virtual cameras in the scene according to chronological order and adjust them to the appropriate position and angle to capture the desired video.

Table 4.2: The time consumption in automation on and off modes (non-professional users)

Script	Auto-staging		Auto-camera		Reduced Time (%)
	Off	On	Off	On	
TJ-1.1	121 min	43 min	48 min	10 min	116 min (68%)
TJ-1.2	143 min	36 min	53 min	3 min	157 min (80%)
TJ-1.3	89 min	42 min	63 min	21 min	89 min (58%)
TJ-1.4	144 min	62 min	55 min	2 min	135 min (67%)
TJ-1.5	135 min	47 min	58 min	11 min	135 min (69%)

As shown in Table 4.1 and Table 4.2. There is a considerable gap in production time consumption in the staging phase between professional animators and general users due to the difference in the level of knowledge about the subject. Most of the non-professional users use the camera track generated by the optimization algorithm directly without making further adjustments, therefore spending less time in this phase. Therefore we will compare the statistics of professional animators in detail. The professional users, on average, spent 34.4 minutes adjusting the character staging and performance and the camera parameters (during auto-staging and auto-camera mode). However, they spent much more time (on average

97.4 minutes) manually selecting character staging and actions and handling the camera placement. The automation flows sped up the manual process by 73.9% on average. For both experiments, the professional animators produced videos according to similar quality criteria. A sample video pair can be reviewed in the video starting from 00:24 https://www.youtube.com/watch?v=MMTJbmWL3gs

To evaluate the value of the fidelity model, we compared the proposed framework with an existing auto-cinematography solution [5] that was optimized mainly using an aesthetic model. The experiments indicate that the proposed solution achieved better performance during the scenarios when multiple characters were conducting interactive activities and when a complete action in length was expected to be captured.

4.4.2 Value of Fidelity Model

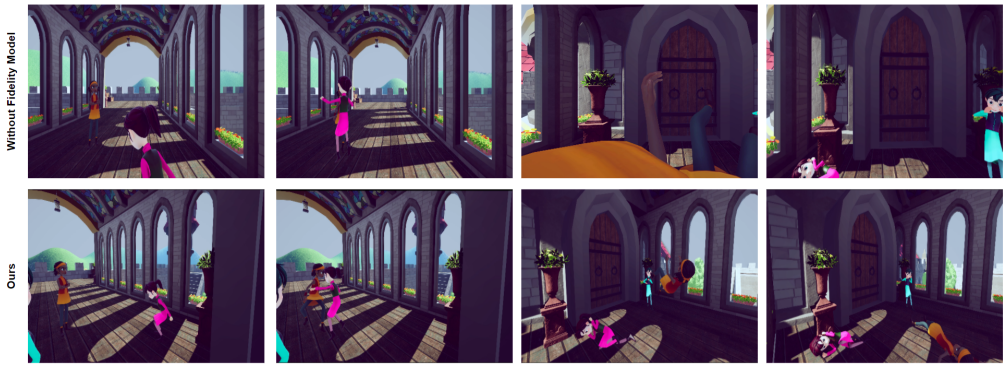


Figure 4.11: Sample frames for the whole action maintaining chronological order from left to right. The optimization using the fidelity model obtains the camera settings (bottom), representing the entire action in the frame

In animation, the character's action can take seconds to complete and thus may cross dozens of frames. With the utilization of the fidelity model, the mismatch between script and recognized action is detected by the video action recognition engine according to equations

4.9 and 4.10, and are simulated by using the model in equation 4.12. Thus the optimization process can select the best representation of the action with a group of video frames captured from selected cameras and associated settings. As shown in Figure. 4.11, the video frames generated without the fidelity model (as shown in the top row) were compared with the ones with the fidelity model (as shown in the bottom row), and the sample frames for the whole action were demonstrated in chronological order from left to right. The bottom row can represent a good abstraction of the entire movement of characters' actions while the top row cannot because there is a referee built into the system to keep the ones easy to understand and filter out others hard to recognize.

To better demonstrate the performance of various models, three example videos of comparison results can be found in the video starting from 02:33 https://www.youtube.com/watch?v=MMTJbmWL3gs to experience the differences between the animation videos with and without the fidelity model in the joint optimization.

4.4.2.1 Multiple Characters Interaction

Figure 4.12: Compare the frames taken by the camera selected by the different optimization approaches. The optimization using the fidelity model obtains the camera settings (bottom figures), which capture more characters in the interactive actions

For narrative films, the interactions among multiple characters, such as chatting, fight-

ing, and collaborating, are indispensable components. When using the aesthetic model, in algorithm [5], the camera is optimized to highlight the acting characters while sometimes ignoring others in passive mode. As shown in Figure. 4.12, the top figures are the frames captured by the aesthetic model optimization for the scenarios of people chatting. As expected, the camera successfully captured the characters who were talking but ignored the other characters who were listening quietly. In contrast, the cameras used by the bottom figures captured all the characters involved in the interaction, which were generated by the proposed algorithm that took into consideration the fidelity model. The key difference between these approaches is whether the produced character performance and scene representation are of high fidelity compared to the original script. In the fidelity model, optimizing D_f forces the system to select the best scene representation that can be recognized by the video action recognition engine and thus can fully represent the activity with all characters involved. This is highly correlated to natural human understanding as well as the recent research on dialogue scenarios [59]. That is, all parties of the interaction covered in the same frame can effectively help the audience understand the topological relationship among characters.

4.4.2.2 Full Action Capture

In animation, the character's action can take seconds to complete and thus may cross dozens of frames. With the utilization of the fidelity model, the mismatch between script and recognized action is detected by the video action recognition engine according to equations 4.9 and 4.10, and are simulated by using the model in equation 4.12. Thus the optimization process can select the best representation of the action with a group of video frames captured from selected cameras and associated settings. As shown in Figure. 4.11, the video frames generated without the fidelity model (as shown in the top row) were compared with the ones

with the fidelity model (as shown in the bottom row), and the sample frames for the whole action were demonstrated in chronological order from left to right. The bottom row can represent a good abstraction of the entire movement of characters' actions while the top row cannot because there is a referee built into the system to keep the ones easy to understand and filter out others hard to recognize.

Table 4.3: Comparison between the camera shots created by different optimization solutions and professional animators. (NoSA denotes the number of shot adjustments required for auto cinematography results compared to the manual camera setting provided by the animator)

	Full-body	Multi	Single	Angle	Distance	Total shot
D_a	32	148	401			549
D_a NoSA	74	95	161	15	83	
T2A	93	219	298			517
T2A NoSA	12	91	73	13	71	
Animator	87	193	290			483

To better demonstrate the performance of various models, three example videos of comparison results can be found in the video starting from 02:33 https://www.youtube.com/watch?v=MMTJbmWL3gs to experience the differences between the animation videos with and without the fidelity model in the joint optimization.

4.4.3 Numerical Comparison with existing Metrics

We compared the number of adjustments made by the animators to the obtained shots for various optimization methods (D_a and D_a + D_f) in 50 different animation scenes in Table. 4.3 below along with the corresponding justifications.

The second row in Table. III indicates the results of the optimization using only D_a as distortion. There are 32 full-body shots, 148 multi-character shots, and 401 single-character shots, for a total of 549 shots (the full-body shot could be either Multi or single-character shot). The fourth row shows the results of the optimization using D_a+D_f. The sixth row

illustrates the results after the animator manually adjusted the results on the automatic camera output.

The third and the fifth rows demonstrate the number of times it takes to adjust the camera settings for each type of shot of the two optimization algorithms before the camera settings are finally approved by the animator. The columns describe the reasons why the animator thinks these adjustments are necessary.

The results show that our proposed method has considerable advantages in capturing full-body movements and single-character scenes, thus reducing the number of camera adjustments the animator needs to make after the optimization. The following is a detailed description of each type of adjustment.

Full-body In animation, some of the character movements need the virtual camera to be able to capture the whole body of the character to provide a better viewing experience. Figure. 4.13 shows one of the examples, we can see that the action "push" of the character is better captured by the full-body camera.

Figure 4.13: Captured "push" action scene from the script: Lead the witchfinder inside, the optimization was done only for D_a (left), the optimization was done based on $D_a + D_f$ (right)

Multi-character Multi-character interaction in animation is very common. Take dialogue scenes as an example, for the audience to understand the relationship between the

characters, it is sometimes necessary (as shown in Figure. 4.14) for both sides of the dialogue to appear in the frame at the same time.

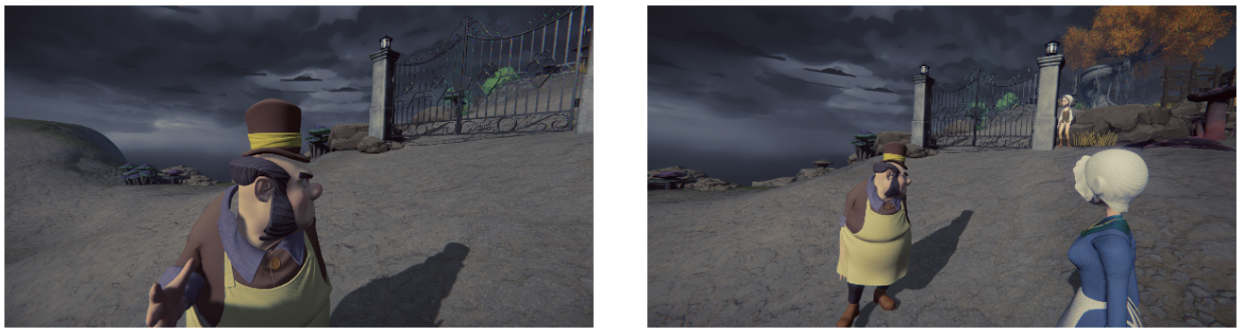


Figure 4.14: Captured dialogue scene from the script: interrupt the conversation, the optimization was done only for D_a (left), the optimization was done based on $D_a + D_f$ (right)

Single-character The use of a single-character shot is also very important in animation. For example, to express the emotion of anger, a single close-up of the character's face is better to convey this emotion to the audience (as shown in Figure. 4.15)

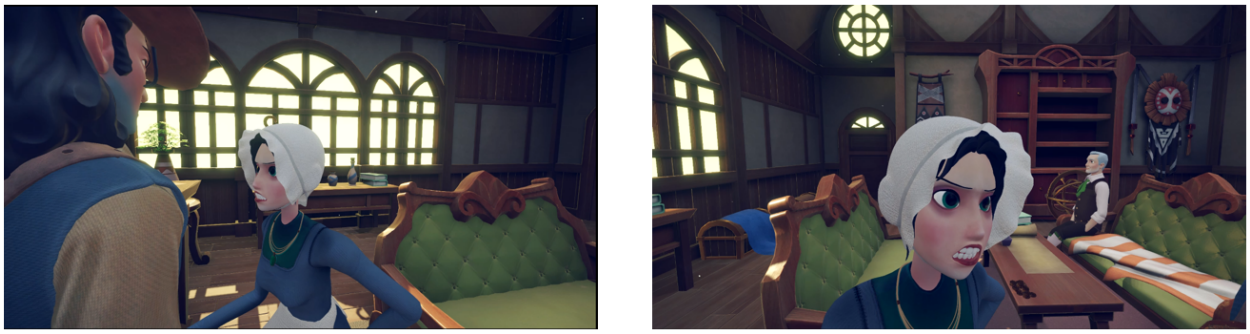


Figure 4.15: Captured "angry" emotion scene from the script: Lead the witchfinder inside, the optimization was done only for D_a (left), the optimization was done based on $D_a + D_f$ (right)

Angle and Distance These two adjustments indicate that the animator believes there is a better choice of angle (yaw, pitch, and roll) or camera distance for the current situation, but not because of the problem we described earlier. For example, the background behind the character is more appropriate after adjusting the camera angle.

4.4.4 Ablation Studies

In this section, the proposed framework is analyzed in depth. The influence of distortion weight coefficients (λ) and main parameters (V, C, S, U) on the optimization results are compared separately for the optimization process. First of all, the role of the λ is investigated. As shown in Figure. 4.16, the normalized value of D_a increases as λ increases, while the normalized value of D_f decreases as λ increases, this is quite aligned with the definition of D in equation .4.14. When λ becomes very small, the cost function in the optimization

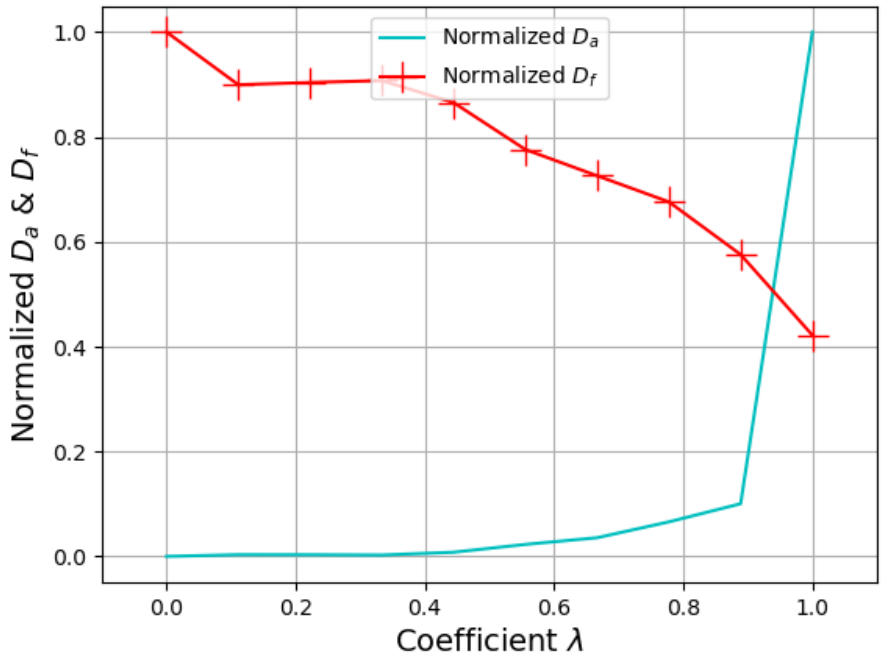


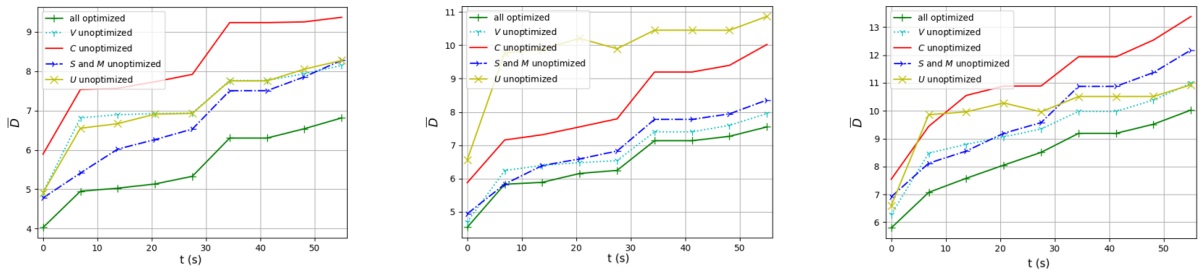
Figure 4.16: The value change of normalized D_a and D_f according to λ

progress is dominated by D_a. Thus the algorithm searches for the path that values more aesthetic factors. On the other hand, when λ becomes very large, the cost function in the optimization process is dominated by D_f. Thus the fidelity factor becomes very important. Clearly, the flexibility provided by adjusting λ provides opportunities for users to shift the

optimization bias for different applications and requirements.

For ablation studies, the key components, such as visibility, camera configuration, screen continuity, shot duration, and fidelity, are investigated to measure their impact on the overall performance of the system. The shot duration cost is only calculated when the camera switches between different positions during the optimization process, and the total D depends on the previous q in the current optimized path. The average distortion of D for the period from $t - 2\overline{u}$ to $t + 2\overline{u}$ is considered for evaluation, which can be expressed in the following equation:

$$\overline{D} = \frac{\sum(D_{t-2\times\overline{u}}, D_{t-2\times\overline{u}+1}, ..., D_{t+2\times\overline{u}})}{4 \times \overline{u}}. \tag{4.17}$$



(a) $\lambda = 0.1$ (b) $\lambda = 0.5$ (c) $\lambda = 0.9$

Figure 4.17: Aesthetic component impact on the optimization results

In Figs. 4.17a, 4.17b, 4.17c, the optimized curves are compared with others with one selected component that is intentionally not optimized, and the setting of λ=0.1, 0.5, and 0.9 are compared to demonstrate the impact of fidelity. It can be observed that the overall impact of the unoptimized component in λ=0.9 is smaller than that of λ=0.1, which is understandable because the former case emphasizes the fidelity factor much more than the aesthetic factor. The figures indicate that camera configuration has a very strong impact on the performance of the system with a gain of 25-32%, shot duration ranks second with a

gain of 16-35%, and visibility has the least impact.

Figure. 4.18 demonstrates the ablation from another angle, where the optimized solution has around 60% gain compared to the fully unoptimized reference, the shot duration component has the largest impact, while the remaining two components have a similar ordered impact with each other. In Figure. 4.19, the generated video frames by the proposed frame-

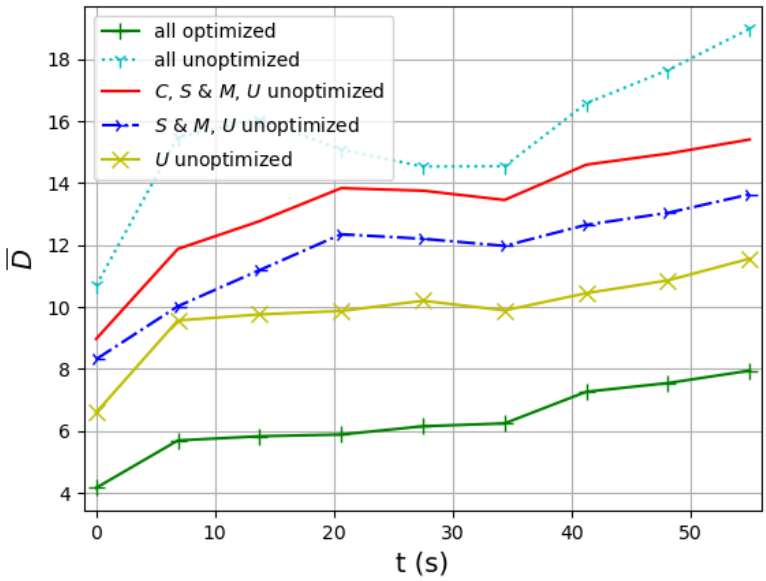


Figure 4.18: Aesthetic distortion component impact on the optimization results from another perspective ($\lambda = 0.5$)

work and several references are demonstrated. By comparing the script and the video frames, the advantage of the proposed framework with respect to both the fidelity and aesthetic aspects is quite apparent. The Row (1) generated from the reference solution of character visibility unoptimized can be found in error for all timestamps by either missing the character or focusing on the wrong character. Row (2) with screen continuity unoptimized does not work well in the second half of the frames. Row (3) with camera configuration unoptimized made wrong camera angle choices in 60% of the cases, for example, at 00:06, it selected a

long shot camera while the camera with such a distance setting is not suggested to shoot this kind of action in the general cinematography guideline; in addition at 00:35 and 00:43, the characters are blocking each other due to the wrong camera angle selection. Row (4) with shot duration unoptimized failed in 60% of the cases, for example, at 00:03, it failed to capture the whole 'fall' action due to lack of duration constraint, and the camera stayed in the same position for too long.

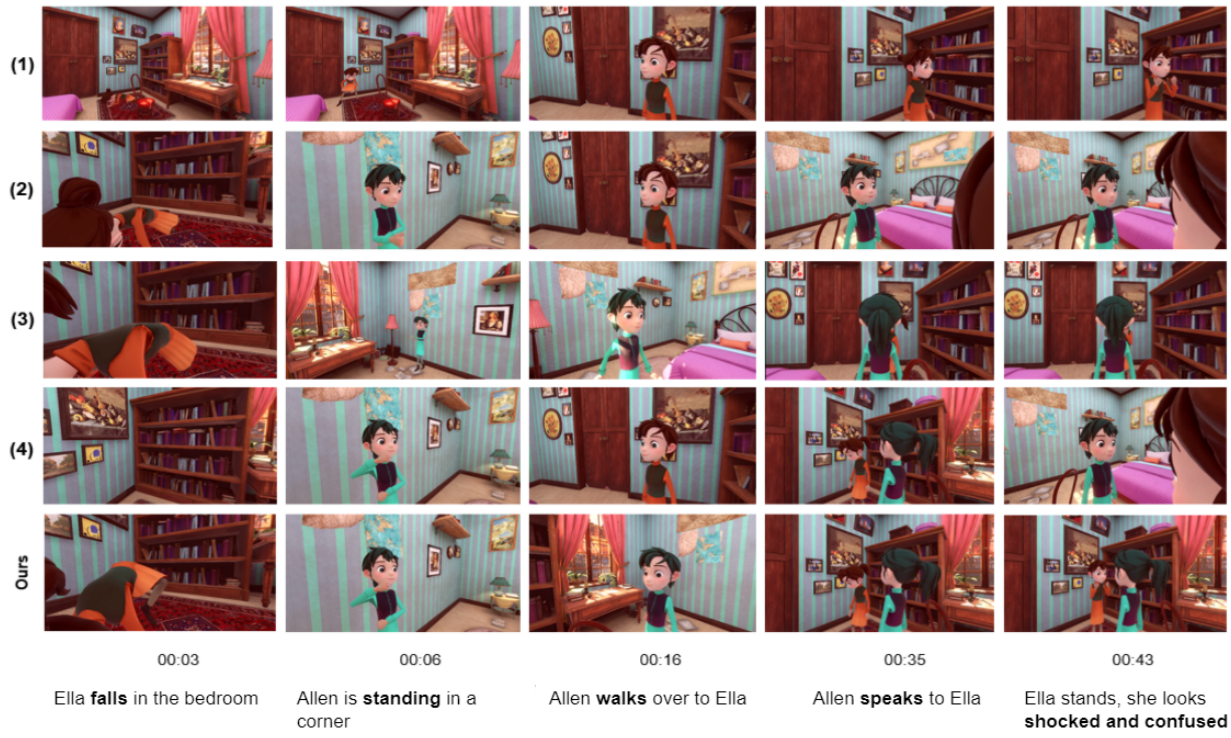


Figure 4.19: Video results of the proposed framework and several references: (1) V unoptimized. (2) $S\&M$ unoptimized. (3) C unoptimized (4) U unoptimized

4.5 Conclusions

3D animation-making typically requires a professional team, sufficient funding and resources, knowledge of cinematography and film editing, and much more. Thus this field is, in general,

not accessible to non-professionals. In this chapter, we presented T2A, an animation production framework that implemented our proposed auto cinematography method. To reduce the barrier for non-professional users, the auto cinematography optimization framework that can choose cameras and their associated settings based on a joint fidelity and aesthetic model, in which the comprehensiveness of visual presentation of the input script and the compliance of generated video with a given cinematography specifications are mapped into a mathematical representation. Although the experimental results indicate both the time consumption and quality advantage of the proposed framework, we believe further investigations on fidelity and aesthetic modeling are needed to make the solution generally applicable to a broader scope of 3D animation and filmmaking tasks.

Chapter 5

Enabling auto cinematography with Reinforcement Learning

5.1 Introduction

In this chapter, we demonstrate our proposed auto cinematography with reinforcement learning, inspired by the idea of mimicking the idiomatic use of lens language by human directors. In this manner, we attempt to address the problem of the rigidity of the lens language used in the rule-based auto cinematography approach.

Cinematography, the art of choosing camera shot types and angles in capturing motion pictures, is an effective way to demonstrate its artistic charm in the film industry, and applying cinematography to content creation requires lots of training and knowledge. In the computer age, it becomes natural to explore the possibility of making robot directors and entrusting them with this challenging task, which is also known as auto cinematography. Some successful efforts, such as [48, 49], have shown that this goal can be partially achieved by translating the rules from the standard cinematography guidelines into a number of cost functions and applying them during an optimization process. Such aesthetic rules-based robot directors can provide valuable references for an entry-level artist without much film experience. In [4] the aesthetic model is further extended to a hybrid model that considers

the fidelity of the output video by comparing it with the textual content in the original script. It is reported in [64] that the mere use of these rules for expression in the film is far from satisfactory. In the practical film industry, human directors are experts in twisting these standard rules in cinematography and developing their unique lens language to express human creativity and imagination [82]. Therefore, although creating robot directors based on common lens language and summarized rules is an interesting idea, it is practically very difficult to meet the expectations of film artists.

As a relevant exploration effort, [51] used a neural network to extract lens language from the existing film and imitated the human director's behavior in filmmaking. However, this effort suffered from an inevitable problem, that is the experimental data is inaccurate and incomplete because the data were estimated from two-dimensional frames with information loss in another dimension. Therefore, such an approach can only be applied in limited scenarios and is unable to comprehensively imitate the director's lens language. Another work [62] demonstrated that such a behavioral imitation problem with a limited amount of training data can be achievable by reinforcement learning (RL) techniques [83] with reward generated from human feedback. In order to develop a robot director that can truly learn human lens language and use it in various scenarios, it needs not only precise data collection during cinematography but also the capacity to continuously improve the model based on external feedback. By using the current auto cinematography algorithms [3–5], it is possible to build a framework for directors to use so that their chosen camera settings can be recorded as training data for reinforcement learning algorithms to learn the lens language models.

This observation inspired us to extend our previous work [4] to the direction of using reinforcement learning because all the needed data for cinematography usage can be collected during the film production process without extra effort with our framework. We reused the

structure of the filmmaking frame T2A proposed in [4], but used a reinforcement learning module for auto cinematography to replace the camera optimization process in T2A, thus the new framework is called RT2A. A director's feedback module is incorporated to correct the camera placements, and such feedback is valuable for the camera agent to learn and imitate the lens language of the target human director. In practical scenarios, the director's feedback can come from stored training data that is collected from real human directors during their routine filmmaking work using our tool [4]. With sufficient data and feedback, the RT2A can effectively learn to support such a robot director that has the potential to become the embodiment of a real director and produce animation with a similar lens language. To the best of our knowledge, this is the first work in the world that applies reinforcement learning to auto cinematography that can effectively imitate the human director's usage of lens language.

The rest of the chapter is organized as follows: Section II overviews the related work in computational cinematography; Section III demonstrates the overall framework of RT2A, the problem formulation, and the solution; Section IV discusses the experimental results and the last section draws the conclusion about the work.

5.2 Related Work

The quality of lens language is a crucial factor that determines the quality of the final film. It requires acumen of artistic sense as well as great knowledge of cinematography when using shots going beyond the basic rules. Luring by the wish to use auto cinematography techniques to reduce film production cost, there are a considerable amount of studies in this area, and they can be roughly categorized into two directions.

The first direction is the cinematography guideline rule-based approach. By defining multiple constraints according to various rules and formulating them into the corresponding loss functions, the optimal shot setup for the current situation can be calculated by minimizing the total loss. Different aesthetic constraints can be applied under separate scenarios, such as dialogue scene [59], cooking [45], and outdoor activities [50]. However, as discussed in [64], it is difficult to judge whether the use of lens language is sufficient under only aesthetic models. In [4], the fidelity model was introduced, which considered the original script content in the optimization process to make sure that the generated video is fully aligned with the script. The advantage of the rule-based approach is that the results are perfectly consistent with the pre-defined constraints, thus guaranteeing that no rule-breaking will occur; however, it also severely limits the freedom and creativity of the artist.

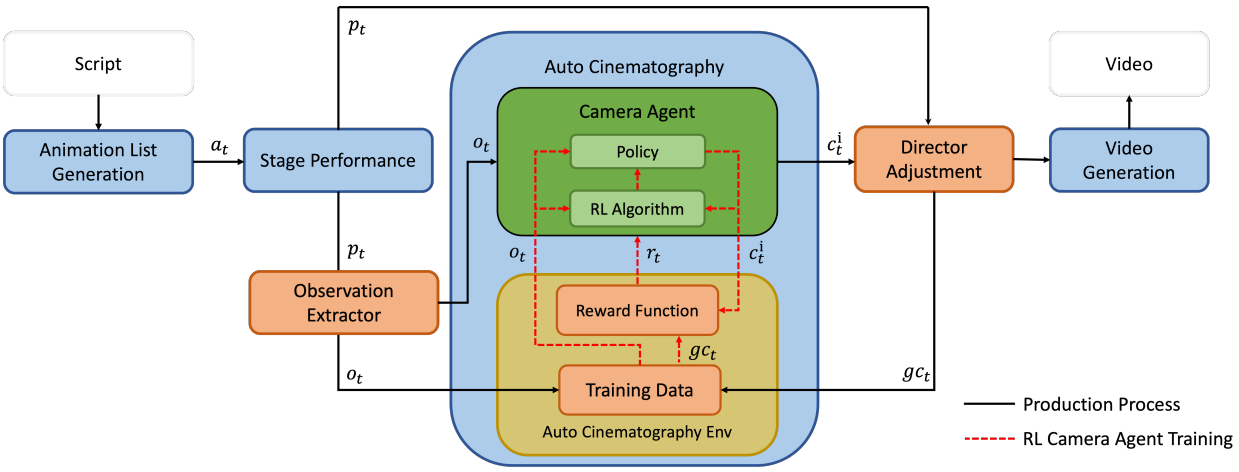


Figure 5.1: Overview of our RT2A animation production framework

The second direction is to learn the behavior model of these cameras directly from the existing movies [51]. The advantage of this approach is that it is a data-driven methodology instead of a rule-driven solution, therefore there is no need to create rules or constraints to guide the algorithm. By using reinforcement learning, such as deep Q-learning (DQN) [84],

Trust Region Methods (TRPO), or policy optimization (PPO) [85], the solution can be found by providing sufficient training data. In literature, there are works in the field of drone photography controlling [62,86], however, no work can be found in the literature using reinforcement learning in the auto cinematography for filmmaking, a possible reason is that to get sufficient precise data from existing movies like in the work of [51] is a non-trivial task.

5.3 Framework Design

The filmmaking process is demonstrated in Figure. 5.1 which contains four major modules, namely, *Action List Generation, Stage Performance, Auto Cinematography*, and *Video Generation*, and two additional modules (in orange color) to support the reinforcement learning workflow. The *Action List Generation* takes the textual script as input, analyzes with NLP techniques [53], and then generates the corresponding action list, which is a chronological list of action objects a_t and can be considered as a special format to represent the content of the original script. Each a_t contains the necessary information for the virtual characters to make the corresponding performance p_t in the *Stage Performance*. The entire script's story can be performed in the *Stage Performance* with a sequence of p_t that follows the order in the action list. At this phase, all the characters appearing in the scene are bound with multiple cameras that can be distinguished by the unique index. The frames captured by cameras also become a part of the p_t. The *Observation Extractor* takes the p_t to generate the corresponding observation o_t (details can be found in section III.A) that is defined in the auto cinematography RL environment for the camera agent to calculate the currently selected camera c_t^i (where i is the index of the camera). In case the initial training data

is insufficient, the acceptance rate of c_t^i computed by the camera agent in the *Auto Cinematography* would be low, thus manual adjustment of c_t^i by the director is required to assure video quality. The modification process is accomplished in *Director Adjustment*. The revised c_t^i is then annotated as the ground truth camera gc_t and added together with o_t into the training data. The p_t and gc_t can be used by *Video Generation* to generate the current output video frame. With the growth of the number of desirable videos generated with this RT2A framework, the training data grows as well. With sufficient training data, *Auto Cinematography* will be able to properly train the camera agent and update its policy with the RL algorithm, thus the workload required for directors to adjust the camera placement would be significantly reduced.

In Figure.5.1, The camera agent training process of the auto cinematography module has been shown with red lines. An observation generator uses the training data to generate the single observation, o_t, at times t. The RT2A camera agent takes o_t as input to obtain the corresponding selected camera, c_t^i. The reward function calculates the reward, r_t, by comparing the c_t^i and the gc_t. The r_t is then used to update the RT2A camera agent policy and parameters by the RL algorithm.

5.3.1 Proposed Auto Cinematography

In the auto cinematography environment, the observation space contains all the information needed by the RT2A camera agent to select the optimal camera setting based on the current policy. The following aspects are included in the observation space.

Character Visibility Character visibility is determined by two factors: (i) the size of the character in the frame compared to the total frame size. (ii) the weights for various groups of characters and camera combinations during the calculation because multiple cameras

are bound to different characters, which reflects the extent to which obstacles obscure the characters from the current camera view.

Camera Configuration When switching cameras, the configuration of the previous camera may also have an impact on the selection of the next camera, such as the shot-reverse-shot commonly used in the dialogue scene. In such cases, the configuration of the previous camera will be included in the observations.

Left to Right Order (LRO) LRO is used to show the positional relationship of multiple characters in the shot frame, and the inclusion of this data has a very significant impact on the 180-degree rule, which is one of the most significant rules in cinematography. It is important to be aware that LROs at t taken by different cameras may not be the same.

Action Type Some character actions, such as "idle", do not affect the shot selection, while others may have a preference for shot selection. For example, the facial action of "anger" is more inclined to be expressed by a close shot. In our experiments, we divide the movements into categories of facial, upper limb, lower limb, whole body, and standby.

Action Start Time and Duration The start time of every character's action is crucial, and generally, the transition of the shot c_i is at the beginning of a certain action. On the other hand, the length of the action is also very crucial. In many cases, the action with a long performance time (e.g., more than 10 seconds) requires a combination of different shots to take it.

Dialogue Start Time and Duration The character's taking action in the dialogue scene is very special. During long conversations, the camera angle is switched between the interlocutors, and often an over-the-shoulder shot is used in such a scenario.

In our auto cinematography environment, the only action in the action space is camera index selection. There are various default cameras to shoot the characters from different

distances and angles. The default cameras cover most of the basic camera settings in the cinematography guideline and each camera has been given a unique index for the agent to select it. the default camera setting can be described with 3 parameters:

1. $d(c)$: distance between the camera and the shooting character, which may include extreme close shot (ECU), close shot (CU), median shot (MS), full body shot (FS), and long shot (LS). By quantifying these distances to numeric representation from 0 to 4, we can calculate the differences between them.
2. $h(c)$: pan (horizontal) angle: form 0° to 360°.
3. $p(c)$: pitch of the camera: form 15° to -15°.

Determining a meaningful reward function is very crucial for the RL algorithm. The reward function is construed only based on attributes of c_t^i and gc_t, where we need to consider the distinction between their settings in detail. The more similar the camera settings the higher the r_t.

The reward function for distance is defined as:

$$r_t^d = \begin{cases} 1 & \text{if } d(c_t^i) = d(gc_t) \\ 1 - \frac{|d(c_t^i) - d(gc_t)|}{4} & \text{otherwise.} \end{cases} \tag{5.1}$$

Similarly, the reward function for the pan and pitch of the camera is defined as:

$$r_t^h = \begin{cases} 1 & \text{if } h(c_t^i) = h(gc_t) \\ 1 - \frac{|h(c_t^i) - h(gc_t)|}{30} & \text{otherwise.} \end{cases} \tag{5.2}$$

When the difference between the agent-selected camera and the ground truth is less than a predefined threshold, δ, an extra reward is added to further boost the learning process.

The larger the δ, the larger the deviation between c_t^i and gc_t that can be tolerated.

$$r_t^p = \begin{cases} 1 & \text{if } p(c_t^i) = p(gc_t) \\ 1 - \frac{|p(c_t^i) - p(gc_t)|}{180} & \text{otherwise.} \end{cases} \tag{5.3}$$

$$r_t^\delta = \begin{cases} 1 & \text{if } \frac{|d(c_t^i) - d(gc_t)|}{4} + \frac{|h(c_t^i) - h(gc_t)|}{30} + \frac{|p(c_t^i) - p(gc_t)|}{180} < \delta \\ 0 & \text{otherwise.} \end{cases} \tag{5.4}$$

The overall reward function is defined as the sum of all the previous rewards.

$$r_t = r_t^d + r_t^h + r_t^p + r_t^\delta. \tag{5.5}$$

5.3.2 Reinforcement Learning Algorithm

An agent starts with observation o_0 and selects the next camera index according to the strategy, *policy* $\pi(o)$, that the agent uses in pursuit of goals which in most cases is to maximize the reward, R. The $\mathcal{R}$ for an episode with T steps is defined as:

$$\mathcal{R} = \sum_0^T \gamma^{t-1} r_t \qquad t = 0, 1, \cdots, T, \tag{5.6}$$

where r_t is immediate rewards at t and γ is a discount factor that defines the importance of r_t versus future rewards, where typically $0 \leq \gamma \leq 1$. The higher the γ value the more important the future rewards.

The RL algorithm aims to find the optimal π^* which can maximize R.

$$\pi^* = \arg\max \mathbb{E}(R|\pi). \tag{5.7}$$

This process is accomplished by iteratively updating the parameter of the *policy*, π_θ, according to the loss function $\mathcal{L}(\centerdot)$ that measures the error between the reward estimation calculated by the current policy, π_{current}, and the previous policy, π_{old}. There are several methods to address the problem raised in Eq. 5.7, and we use PPO, a variant of an Advantage Actor-Critic (A2C) [87], which combines policy-based and value-based RL algorithms. The actor neural network model takes state (or observation) and outputs the action according to the $\pi()$, and the critic neural network model maps each state to its corresponding quality of value the state (i.e., the expected future cumulative discounted return). The advantage $\hat{\mathcal{A}}$ (or discounted return) is used to indicate how good a camera selection is compared to the average camera selection for a specific observation. The $\hat{\mathcal{A}}$ at time t is defined as following:

$$\hat{\mathcal{A}} = -V(o_t) + rt_t + \gamma_{t+1} + \cdots + \gamma^{T-t+1} rt_{T-1} + \gamma^{T-t} V(o_t), \tag{5.8}$$

where the V is the learned state-value function and rt_t is the parameter changing ratio between the π_{current} and π_{old} at step t.

During the training process, PPO needs to update the parameters of both actor and critic neural networks by back-propagation according to two different loss functions. Every update for π is designed to maximize the overall return (i.e. $\max[\mathbb{E}_t(rt_t\hat{\mathcal{A}})]$). However, changing the π_θ needs to be avoided in a single update. Thus, the $\mathcal{L}_{\mathsf{actor}}$ is defined as:

$$\mathcal{L}_{\mathsf{actor}}(\theta) = \min(rt(\theta)\hat{\mathcal{A}}_t,\ \mathsf{clip}(rt(\theta), 1-\epsilon, 1+\epsilon)\hat{\mathcal{A}}_t). \tag{5.9}$$

The $\mathsf{clip}(\centerdot)$ modifies the surrogate objective by clipping the probability ratio, which removes the incentive for moving rt outside of the interval $[1-\epsilon, 1+\epsilon]$. Regardless of the value of the

positive feedback gained according to c_t^i, the PPO will only update the policy based on this result within this limited range. Thus, it can incrementally update π_θ with an appropriate value. However, the penalty based on the negative reward has no limitation.

The critic loss function $\mathcal{L}_{\mathsf{critic}}$ is defined as:

$$\mathcal{L}_{\mathsf{critic}} = \frac{1}{2}\hat{\mathcal{A}}^2. \tag{5.10}$$

5.4 Experimental Results

Figure 5.2: Sample frames for the scene maintaining chronological order from left to right. As shown, our approach better imitates the lens language used by the animators

In this section, the experiment setup and the performance details of our proposed RL auto cinematography method are demonstrated. We develop the auto cinematography environment and implement the RT2A camera agent using OpenAI Gym [88]. It took around 38 hours to finish the whole training process based on the NVIDIA GTX 2080 TI GPU. To

evaluate the advantage of our proposed RL auto cinematography model, we compare RT2A with the reference methods: Aesthetic-based [5], and Aesthetic + Fidelity-Based [4]. The performance is evaluated from two aspects: (i) we compare the camera placement generated by the RT2A camera agent and by the reference algorithms; (ii) we compare the visual quality of videos produced by RT2A and reference algorithms.

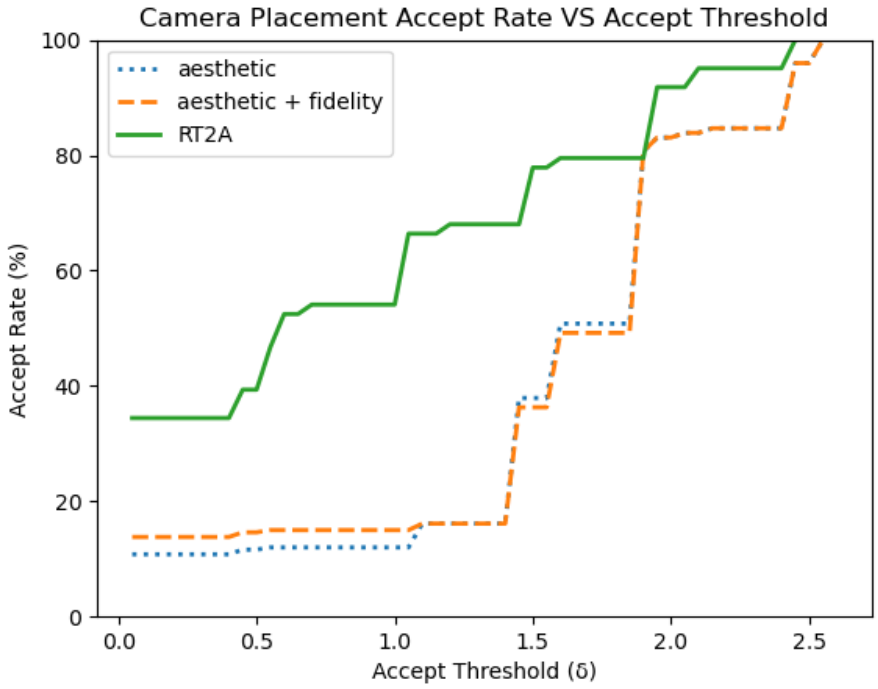


Figure 5.3: Camera placement acceptance rate with different acceptance threshold

Difference in Camera placement In the environment of the experiment, the cameras generated by different methods are sampled every second. By comparing the physical distance between the camera placement of the algorithm with the ground truth (placement manually by directors), and defining several acceptance thresholds (that is, the placement is accepted if the physical distance is less than the threshold), the performance of the algorithms can be measured by the percentage of the cameras being accepted, which is called

acceptance rate. The calculation of the physical distance is based on the equations 5.1, 5.2, and 5.3. As the results are shown in Figure. 5.3, the proposed algorithm constantly outperforms the reference algorithm, and the gain in acceptance rate is up to around 50%. The visual comparison of the generated frames from various algorithms as shown in Figure. 5.2 also indicates that the proposed algorithm has a much higher similarity to the camera selected by directors than the reference methods. This evidence proves that the RT2A can effectively imitate the behaviors of the director's camera usage model after training.

Number of shot A complete shot is a continuous view through a single camera without interruption. The number of shots (i.e., average shot duration) used in a single scene is also an important indicator to reflect the shooting style of the director. Table. 5.1 shows the number of shots of our proposed auto cinematography approach compared to the reference methods in several different scenes selected from the test data set. The results indicate that the number of shots used in each of the test scenes by our proposed method is much closer to the ground truth than the reference methods. Visual results also support the conclusion when the frames generated from the aesthetic model [5] are compared to the frames generated from RT2A. It is important to realize that although it may be possible for the rule-based approach like the aesthetic model algorithm to mimic the director by setting several rules to optimize when compared to the proposed RT2A algorithm, the challenges in adjusting the weights for various parameters and cost functions are much more complicated. In the following text, we will demonstrate how the reference model can achieve similar visual results by adding the corresponding cost functions or adjusting weights, hence it convincingly proved the advantages of the proposed RT2A algorithm using a data-driven methodology, instead of manually crafting many rules.

Particular shot selection As demonstrated in [89], *"the single long shot showing initial*

spatial relations became one portion of the scene, usually coming at the beginning. Its function then became specifically to establish a whole space which was then cut into segments or juxtaposed with long shots of other spaces.", long shots as the establishing shots are used to set a particular tone and mood for what the audience is about to see. As shown in Figure. 5.4, compared to the reference approach, the RT2A camera agent learns this lens language better and uses the establishing shot at the beginning of the scene. It is possible if the reference algorithm desires to achieve a similar outcome, that is, a new cost function determining whether the t represents the first few frames needs to be included in the optimization framework, and the characters captured by the LS shot will be required to occupy the least space in the frame compared to other shot types, which means that the weight of character visibility in cost function needs to be minimized.

Table 5.1: The difference in the number of shots of our proposed and reference methods compared with the director's selected camera placement. The results show the number of shots (also the differences in percentage compared with the selected shot by the director) used by different approaches in a single scene

Script	Aesthetic	Aesthetic + Fidelity	RT2A	Director
1	32 (33%)	35 (46%)	27 (13%)	24
2	50 (150%)	43 (115%)	28 (40%)	20
3	45 (95%)	40 (74%)	31 (40%)	23
4	37 (146%)	39 (160%)	21 (35%)	15
5	15 (150%)	13 (116%)	8 (33%)	6

The Over the Shoulder Shot is widely used in the dialogue for the audience to understand the relationship between the characters and to convey a dramatic tension to the viewers [90]. It is sometimes necessary to use it to assemble the reverse shot in a dialogue scene. As shown in Figure. 5.5, the RT2A camera agent learns this lens language successfully and uses the Over the Shoulder Shot in dialogue scenarios. If the reference model wants to achieve a

similar result, a new cost function needs to be added to determine whether there is enough duration for the current "speak" action to switch between shots. In addition, the weight of the camera placement used to take the over-the-shoulder shot needs to be modified during the optimization process, to lose the requirement of capturing actions from the back of the character.

Figure 5.4: Captured establishing shot frames from the camera generated by the aesthetic model (left) and the camera generated by RT2A (right)

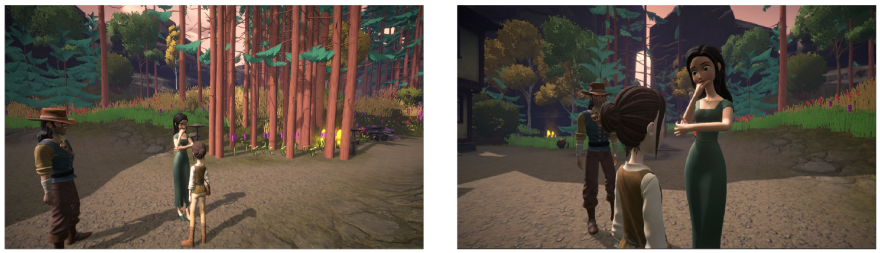


Figure 5.5: Captured dialogue frames from the camera generated by the aesthetic model (left) and the camera generated by RT2A (right)

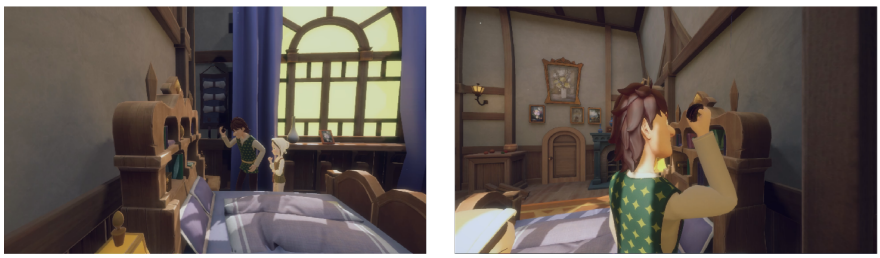


Figure 5.6: Captured single action frames from the camera generated by the aesthetic model (left) and the camera generated by RT2A (right)

Sometimes actions of particular characters need to be shot from a close distance and appropriate angle for the audience to better understand the content. As shown in Figure. 5.6, the frames captured by the reference model selected camera do not properly illustrate the actions of "inspect the item" very well. In order to let the aesthetic model achieve a similar result, it is required to manually modify the weights of different camera configurations for some particular actions according to the interpretation of the story.

5.5 Conclusions

The creativity and imagination demonstrated by the use of lens language in 3D animation and filmmaking require tremendous professional knowledge and talent. It would benefit entry-level artists if the auto cinematography algorithm could learn from professional directors and thus mimic the camera languages used in successful films. However, a major obstacle is that there is insufficient accurate training data in this area. In this work, we presented the RT2A framework, which produces accurate data for training the auto cinematography system with the support of directors. By learning the lens language from these directors, RT2A can select the right distance and camera angle for the auto cinematography process.

Chapter 6

Automated Adaptive Cinematography in Open World

6.1 Introduction

In this chapter, we extend the auto-cinematography to the open-world environment. By leveraging Generative Adversarial Networks (GANs) and optimized camera placement spaces (such as toric surfaces), automatic film cinematography techniques can deliver satisfactory results in more open and uncertain environments. Such advancements facilitate cost savings in production and enhance the audience's sense of immersion.

The gaming industry's progression has been persistently propelled by the aspiration to provide users with increased autonomy in gameplay, which serves to elevate their pleasure and absorption within the game experience [91]. This has resulted in a trend towards the creation of open-world games, which provide users with a more expansive and unrestricted environment and, as a result, a more enriching experience. Cinematography is a crucial element in conveying character emotions and advancing the plot in media such as games and films, and it can significantly enhance the immersion of the audience if utilized effectively. Research on user emotions and cognition in video games has demonstrated that virtual cinematography is closely connected to the user experience. The connection between user

and camera transcends the norms of conventional cinematography, as virtual cinematography predominantly centers on the character during entire utilization [92, 93].

In cinematography, the use of lens language techniques is contingent upon several intricate factors, such as character emotions, actions, character count, and the environment. In conventional film and narrative-driven games, both the story and character development adhere to a predetermined script. The film directors can reference the script to strategize and select suitable lenses during production, which simplifies the process of optimizing the viewer's experience. Employing this approach enables film directors to generate a streamlined camera script, which subsequently facilitates the production of aesthetically pleasing shot compositions. Moreover, this method significantly reduces associated costs [3, 56]. In other works [4, 94], classical lens language techniques were condensed into a set of camera rules. Because these techniques are simple to implement, they have facilitated the development of automated cinematography methodologies based on these principles. When supplied with adequate information, these approaches can generate satisfactory outcomes for the average viewer. They also contribute to a significant reduction in production expenses.

Open-world environments are characterized by a high degree of uncertainty (as shown in Table 6.1). As anticipated, the enhanced freedom afforded by these environments introduces novel challenges for the implementation of cinematography. Traditional rule-based automatic cinematography approaches prove unsuitable for such settings due to their unpredictable nature. Users in these environments tend to favor the capacity to craft and generate their unique narratives via interactive gameplay, rather than being constrained by pre-established storylines.

Simultaneously, artificial intelligence has facilitated the responsiveness and adaptability of in-game characters and objects to alterations within the virtual environment. As a result,

actions and interactions of virtual characters and objects in open-world scenarios are no longer constrained by scripts. This freedom allows users to indulge in unique and personalized experiences. In addition, user-controlled characters or avatars are no longer mandated to adhere to a predetermined script. They can engage with virtual world objects in real time, interactions that transcend the limitations of pre-defined choices. These interactions can assume a variety of forms, supported by the employment of portable devices for facial and action capture.

Such novel interaction modalities have considerably amplified the uncertainty surrounding aspects such as narrative development, character relationships, and the context of character interactions. While rule-based automatic cinematography can still be employed to a degree, utilizing limited, predetermined camera angles as evidenced in [95], it is not without limitations. Rule-based methods may repetitively deploy the same perspectives. Coupled with their inability to effectively portray the relationships between characters and objects, these methods can potentially compromise user immersion within the virtual environment [96].

Despite the mounting demand for freedom in interactive experiences, research on automatic cinematography in open-world contexts remains limited. In recent years, AI technology has been applied to the field of automatic cinematography [4, 94]. This development allows film directors to glean lens language usage from data and extend its application to a broader array of scenarios. Therefore, the utilization of AI-based automatic cinematography holds significant potential in open-world environments, as it can effectively bolster the immersive gaming and viewing experience.

One of the key challenges we encountered was determining the most effective camera placement in the algorithm that can achieve high levels of shot quality. The toric surface

has been suggested as a suitable selection space for automatic camera placement in previous studies [74]. However, incorporating character actions and object states in open-world environments requires further consideration. To tackle this challenge, we expand the toric surface to accommodate character emotions, actions, and the environment, enabling appropriate camera selections for various situations within open-world settings. Additionally, our system framework is designed to be flexible, open to alternative options, and not just limited to the toric surface for camera placement possibilities.

Moreover, maintaining consistency in character movement, emotions [97], and camera movement is crucial for achieving cohesive and effective results [98–100]. Recent studies have demonstrated that automatic camera movement, which focuses on individual characters, effectively conveys their emotions and movements [101]. This overcomes the limitations of approaches that track a character's head movement without considering their emotional state [102]. In our approach, we further expand upon this work and apply it to open-world scenarios.

To augment the aesthetic value of the frames captured by automatic cinematography, we integrated data from professional film directors' shot selections into the loss function employed in our automated camera generation model [103]. The model's performance has been enhanced, enabling it to learn how professional film directors utilize shot language to capture frames in diverse situations.

To the best of our knowledge, this paper is the first effort in the automated cinematography field that generates camera movement for multi-character interactions in an open-world environment.

The contribution of this paper can be summarized as follows:

1. To address the challenge of open-world character interactions, we propose a novel

auto cinematography framework called AACOGAN based on Generative Adversarial Networks (GANs). AACOGAN is designed to generate camera movements that are consistent with the interactive actions of the characters in the open-world environment.

2. In open-world scenarios, we develop comprehensive quality assessment metrics tailored for automatic cinematography. These metrics effectively aid in evaluating the quality of generated camera movements.
3. To guarantee consistency between character and camera movements and tackle the challenge of camera movements in multi-character interaction scenes, a unique input feature and generator architecture are employed. This design promotes precise alignment of generated camera movements with diverse character interactions, which is particularly effective in multi-character scenes.

The rest of the paper is organized as follows: Section 6.2 provides a literature review on computational cinematography, video editing, and video understanding. Section 6.3 presents our proposed framework, problem formulation, and dynamic programming solution. Section 6.4 discusses experimental results and the potential impact of video understanding errors on the framework. Section 6.5 concludes the work.

Table 6.1: A comparison of three different types of user experiences (traditional movie, purpose-driven interactive movie or game, and open world) in various aspects as perceived by the audience

	Traditional Movie	**Interactive Movie/Game**	**Open World**
Protagonist control	NO	Yes	Yes
Script form	Single-line	Branching	Free-form
Interactive distance	Fixed	Fixed	Varies
Interactive choice	None	Limited	Any
Interactive objects	Plot-based	Selectable	Any in scene
Lens usage	Preset	Idiom-based	Not preset
Emotions by script	Yes	Yes	No
Camera movement	Director's choice	Director's choice	Fixed (view)
User experience	Watch	Script-driven	Freely decide

6.2 Related Work

Cinematography is crucial in fostering immersion and realism for users in open-world games. Dynamic camera movements and carefully crafted shots direct the user's attention toward specific objects or events [100, 104], and can generate a sense of momentum during action sequences. Moreover, cinematography establishes the game world's mood, tone, and atmosphere while conveying information and emotions to users. However, manual shot production demands considerable time and cost. Consequently, automation in cinematography techniques, aimed at expediting production and reducing expenses, has garnered substantial attention in recent decades.

Initially, the film industry attempted to tackle the challenge of generating suitable camera movements using a film director-led approach, wherein a camera script was written based on the story's plot [102, 105]. This method provides precise control over camera movements through multiple constraint elements. However, it necessitates mastering a specialized camera control script format and demands considerable cinematography expertise and manual effort from the creator for each shot.

Recent studies [47, 106] identified cinematic idioms as frequently used camera movements in specific scene types. An automated cinematography program can align content with relevant cinematic idioms for filming, based on the content that needs capturing. This approach and its advanced derivatives [4, 49, 59] have been developed for films or games with clear narrative structures, reducing the need for filmmaker cinematography expertise and enhancing efficiency and convenience. The connection between lens language and cinematography has been scrutinized in [95, 96]. These studies illustrate that employing various idioms based on game content can effectively guide user emotions, leading to increased user immersion.

However, this approach may be unsuitable for open-world scenarios, as camera shot selection depends on character actions, emotions, and object interactions. These variables complicate the determination of idioms.

An alternative automated cinematography approach, as described in [94, 107], might be more fitting for open-world scenarios. This method trains the model to mimic human directors' lens language in various contexts, integrating all relevant lens language factors into the training data. This results in more flexible and adaptive outcomes, as lens language can vary based on character actions and object interactions in open-world environments. The GAN-based automated cinematography [101] shows it can effectively execute camera shots focused on a single character while considering actions and emotions. However, this system fails to establish relationships between characters and objects, which is crucial for lens language selection in open-world environments.

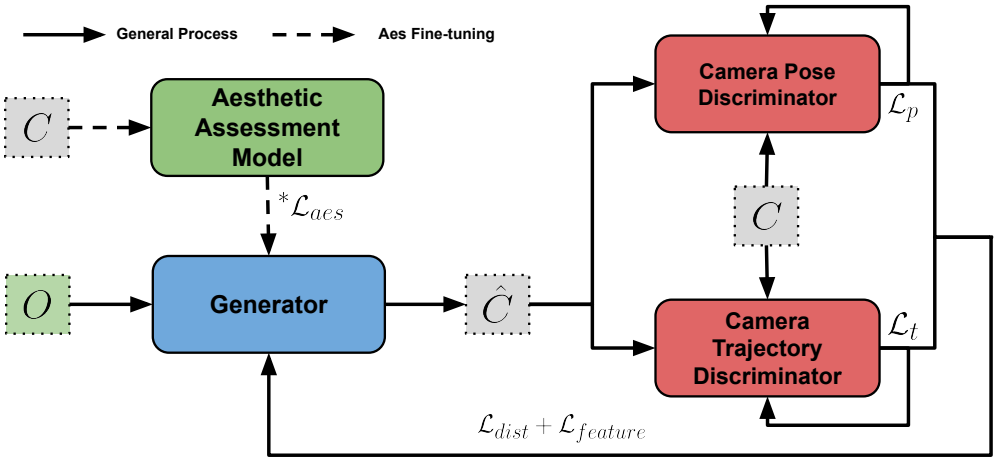


Figure 6.1: AACOGAN architecture overview, where O is the input feature, C is the ground truth camera trajectory, $\hat{C}$ is the generated camera trajectory and $\mathcal{L}$ is the loss

The selection space for camera positions is vital in open-world scenarios, as the numerous camera placement options render most of the space redundant. Restricting camera movement through a widely accepted camera movement space reduces computational complexity,

simplifying calculations while ensuring high-quality results [74, 108].

Automated cinematography often lacks objective evaluation criteria, resulting in less widely accepted outcomes. A potential solution is incorporating an aesthetic assessment model that evaluates frames based on commonly accepted aesthetic standards [109–111]. Integrating this model into AI training for automatic cinematography has the potential to significantly improve the general quality of captured frames.

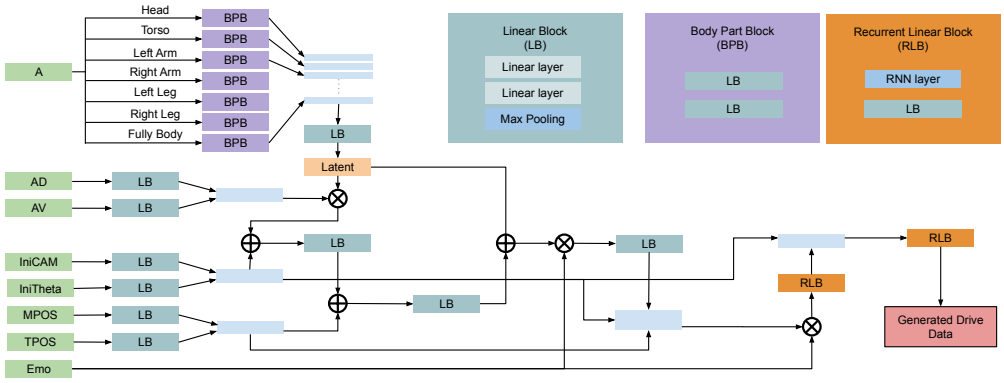


Figure 6.2: AACOGAN generator architecture overview. The green blocks are the input for the generator model including skeletal key point position (A), point speed (AV), point acceleration (AD), initial camera parameters (CAM), initial theta value on camera moving surface (IniTheta), user-controlled character position (MPOS) and target objects position (TPOS). The block with "Latent" is the model to generate the latent representation through feature extraction. The detailed explanation of this overview is in section III.B

6.3 Problem Formulation

Evaluating the quality of lenses generated by automated photography technology presents a complex challenge, given the absence of universally acknowledged objective evaluation standards [64]. Traditional user research, employed for outcome assessment, may be hampered by biases and limitations in addressing dynamic user demands, potentially impeding system enhancements. Therefore, the development of objective evaluation criteria is vital for improving research results.

6.3.1 Quality Measurement

As shown in Table 6.1, open-world environments significantly differ from traditional shooting settings, making it challenging to directly apply automatic photography techniques designed for conventional film productions. These environments involve factors such as user-controlled character exploration, spatial relationships between objects and characters, varying interactive methods, and fluctuating numbers of participating individuals. In these contexts, user-directed character movements are often primarily focused on the character itself rather than the entire scene. To effectively address these unique requirements for automatic cinematography in open-world contexts, we propose an equation with rational metrics to evaluate the quality of the automatically generated camera q, as detailed below:

$$q = Q(D_{\mathsf{corr}}(C, A),\ R(C),\ S_{\mathsf{aes}}(C)), \tag{6.1}$$

where $Q(\cdot)$ is the quality function, C is the generated camera trajectory, and A is the position and rotation of the skeletal bone during character interactive movement. $D_{\mathsf{corr}}(\cdot)$ is the function that calculates the similarity between the camera and character movement trajectories. $R(\cdot)$ is the function that calculates the ratio of the all-character captured frame during the interaction, and $S_{\mathsf{aes}}(\cdot)$ calculates the aesthetic score of the frames captured by the given C.

Undoubtedly, the employment of cinematic language as an essential component of the artistic domain is contingent upon subjective assessments. Consequently, a preferable approach might encompass enabling automatic cinematography systems to learn from human film directors in the gaming industry [94], rather than relying exclusively on fixed algorithms for camera movement generation. By leveraging pertinent data to mimic the cinematic lan-

guage habits of directors and iteratively refining the generated algorithm based on user feedback, neural network technology could effectively replicate directorial expertise. During the system training phase of our experiment, the preceding quality measurement defined in Equation 6.1 can be expanded to Q_{ref} as follows:

$$q_{\mathsf{ref}} = Q_{\mathsf{ref}}(D_{\mathsf{corr}}(C, A), R(C), S_{\mathsf{aes}}(C), \mathsf{Dis}(C, \hat{C})), \tag{6.2}$$

where $\mathsf{Dis}(\cdot)$ represents the distance metric that measures the dissimilarity between the C and the ground truth camera motion $\hat{C}$ obtained from human film directors, who authored the actual in-game camera movements.

It should be noted that the approach we use to define Q_{ref} is not necessarily a standard one. Moving forward, we will provide a detailed explanation of each component that makes up this Q_{ref}.

In gaming scenarios, participants can assert control over the in-game environment and status through the prescribed character's interactive functionalities, which is a process that can also affect consequential alterations to camera trajectories based on rules [99]. Within open-world contexts, user influence on character control resembles that in gaming, where these interactive actions predominantly change open-world environment and status. As pointed out in [112], preserving consistency in character control, which supports expected character and camera movements, can mitigate users' sense of disorientation. Consequently, the congruence between the trajectories of key skeletal nodes of characters along axes during interactions and the camera motion trajectory serves as a metric for evaluating their coherence. The similarity difference can be computed using the correlation distance d_{corr} for each

position and rotation axis between the two trajectories, as follows:

$$
\begin{aligned}
d_{\mathsf{corr}} &= D_{\mathsf{corr}}(C, A) \\
&= \sqrt{1 - \frac{\left(\sum_{t=1}^{n-1}(f_t - \bar{f})(p_t - \bar{p})\right)^2}{\sum_{t=1}^{n-1}(f_t - \bar{f})^2 \sum_{t=1}^{n-1}(p_t - \bar{p})^2}},
\end{aligned}
\tag{6.3}
$$

f_t and p_t are the frame and position coordinates of the t-th point on the trajectory C, n is the number of frames for the action and camera movement, and $\bar{f}$ and $\bar{p}$ are the mean values of the f and p coordinates, respectively.

The absence of a predefined script in open-world environments poses a significant challenge to automated filming techniques, especially when it comes to focusing on multiple points [74]. In scenarios involving multiple parties, it is crucial for the camera to capture the entire interaction process and all characters comprehensively, not solely the character currently under user control. To address this challenge, a potential strategy is to facilitate the camera's extended capture of all involved characters throughout the camera movement process. An effective metric for evaluating the efficacy of capturing all involved characters r throughout the camera movement process is the ratio of the number of frames in which all interactive characters appear in the frame to the total number of frames n used for the interaction, which can be calculated by:

$$
r = R(C) = \frac{\sum_{t=0}^{t=n} R_{\mathsf{frame}}(f_t)}{n} \times 100\%, \tag{6.4}
$$

where $R_{\mathsf{frame}}(\cdot)$ result equals 1 when all the interactive characters are present within ft, otherwise it equals 0.

Although objective criteria for evaluating the quality of imagery generated by automated

cinematography techniques remain elusive, aesthetic evaluation models have been widely acknowledged for images as presented in [111,113]. As a sequence of images, the video captured during camera movement can be evaluated objectively in terms of aesthetics by calculating the aesthetic score of each frame captured during the camera movement process. Integrating aesthetic models into the automated cinematography system can improve the conformity of captured imagery with objective standards. This aesthetic score can be calculated by:

$$s_{\mathsf{aes}} = S_{\mathsf{aes}}(C) = \frac{\sum_{t=1}^{t=n} \mathsf{AES}(f_t)}{n}, \tag{6.5}$$

where f_t is the visual content captured by the C at t-th frame, AES is the model for image aesthetic evaluation, and n is the number of frames for the camera movement.

The emotional state of characters is a crucial factor that significantly influences automated camera movement, as emotions can greatly impact shot selection [100, 101]. Furthermore, previous studies indicate a positive correlation between heightened screen motion intensity and viewer arousal [97]. Consequently, even with identical interactive actions, varying emotional states should yield different camera movement speeds and amplitudes to better convey the characters' current emotional state. The relationship between emotions and camera movement quality cannot be directly evaluated, as each individual possesses distinct standards for camera movement amplification in response to various emotions. This difference will be directly reflected in the actual camera trajectory and can be considered a component of $\mathsf{Dis}(C, \hat{C})$. The $\mathsf{Dis}(\cdot)$ is employed to compute the distance between the generated and real camera drive data, consisting of two parts: MSE and Euclidean distance,

represented as follows:

$$\mathsf{dist} = \mathsf{Dis}(C, \hat{C}) = \mathsf{MSE}(C, \hat{C}) + \mathsf{Euclidean}(C, \hat{C}). \tag{6.6}$$

In response to the aforementioned problems and challenges, a deep-learning-based generative model, AACOGAN, is based on GANs and is developed to enable automatic cinematography in the open-world environment. GANs have demonstrated impressive results in generating synthetic data samples that resemble data from a training dataset, commonly used for generating images [114] and audio [115] signals. The essence of camera movement in cinematography is the variation of the camera's position and rotation along different axis over time, which is comparable to the time-varying intensity of audio signals.

6.3.2 AACOGAN Architecture

Based on the aforementioned factors that influence the generated camera trajectory in AACOGAN, we have designed the input (green block of Figure. 6.2) for the generation model.

Character skeletal animation, also known as skeletal animation (Figure. 6.3), is a widely adopted technique in the animation field that enables the generation of realistic and intricate movement. This technique involves continuously recording A of the skeletal bones during character interactive movement, as well as computing and documenting the speed (AV) and acceleration (AD) of these bones. By including the speed and acceleration of the skeletal bones as input features, the generation model can make more accurate predictions about the future position of the camera, particularly in cases of continuous camera movement. The initial camera position is also significant, as it cannot be forecasted by the generation model.

In addition to the 3D coordinates and rotation ($IniCAM$) of the camera's initial position within the virtual environment, the relative coordinates on a toric surface ($IniTheta$) have been considered as input parameters. This is because the movement of the camera is confined to this surface in the experiment.

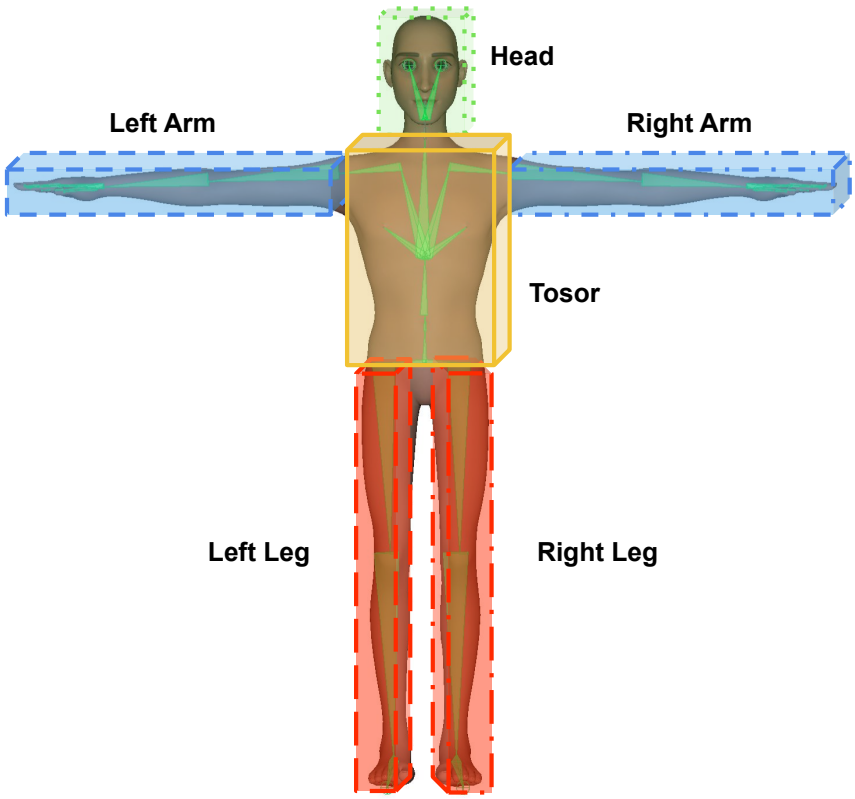


Figure 6.3: 3D Character skeleton illustration. Each tetrahedron corresponds to a bone in the skeletal animation

Character and object position, as well as the position relationship between the user-controlled character ($MPOS$) and the target interactive objects ($TPOS$), are crucial factors in determining the camera's path. In the experiment, the 3D coordinates of both the character and the interactive objects are recorded. It should be noted that there may be multiple interactive objects, and thus a set of vectors representing their positions have been included

in the experiment, with a maximum of five vectors. The character's emotional state *Emo* is a crucial factor in the selection of lens language. As current domestic motion capture technology is inadequate in capturing the user's emotions, they are randomly assigned to each of the character's interactive actions during the experiment. This is represented by numerical values ranging from 0 to 1, indicating the level of impact that the emotion has on the character's movements. Therefore, the input, O, for the generator can be represented as the collection of the above factors.

The architecture of the AACOGAN generator model is represented in Figure. 6.1. The generator is designed to learn the pattern of the ground truth camera drive data sequence, $\hat{C} = [\hat{c_0}, \hat{c_1}, \cdots, \hat{c_T}]$, for each interactive action and generate a corresponding sequence of camera drive data, $C = [c_0, c_1, \cdots, c_T]$, based on the sequence of observed input features, $O = [o_0, o_1, \cdots, o_T]$, where T represents the number of samples of the interactive action over the duration.

As depicted in Figure. 6.2, the AACOGAN generator is a neural network designed to generate a camera drive data sequence based on a given input sequence of observed features, utilizing an encoder-decoder GAN structure [116,117]. The encoder, a function that processes a sequence of partial input features (A), produces a latent representation through feature extraction [118].

Some interactive actions, such as running or walking, involve the entire body and can impact the camera's overall path in the virtual environment. To address potential connections between various body parts and movements, in the proposed AACOGAN architecture (Figure. 6.3), the encoder separates input data into distinct body parts using specialized neural network layers known as Body Part Blocks (BPBs). These BPBs, trained to isolate and encode specific body regions, facilitate the learning of fine-grained representations

for each region, establishing a deeper correlation between character and camera movements. This separation offers multiple benefits, including the effective capture of various body parts' engagement during different actions and enabling the encoder to focus on specific input data segments. This is particularly useful for capturing relevant body parts during intense or vigorous movements.

Subsequently, this collective feature representation undergoes a Linear Block (LB) operation, primarily designed to optimize the input data's structural compatibility with subsequent computations. This process derives a latent representation of the skeletal bones A as per the following equation:

$$A_{\mathsf{latent}} = \mathsf{LB}(\mathsf{BPB}(A_{\mathsf{head}}), \cdots, \mathsf{BPB}(A_{\mathsf{fully}})). \tag{6.7}$$

In particular, the latent representation of A retains the positions and rotations of skeletal bones, encompassing crucial information for generating camera drive data. The remaining data decodes this latent representation at various stages and from distinct perspectives based on its type. Firstly, AV and AD which are derived from A are processed through two independent LBs and concatenated, serving as supplementary information about the skeletal bones' position and rotation during movement. The positional data of the camera, including *IniCAM* and *IniTheta*, are processed separately through two independent LBs and concatenated. These features enhance the model's ability to establish a robust correlation between the initial camera position and camera movements. Features regarding the positions of the character and interactive object(s), *MPOS* and *TPOS*, are involved in the decoding of the latent representation after processing through two other independent LBs.

Emo is a critical component of the proposed model, integrated with other data by the

generator through a single LB. This integration enhances the model's capacity to establish a strong relationship between camera movements and character emotions. Since *Emo* is one-dimensional and remains relatively stable within a single interactive action, it is incorporated as a coefficient on all intermediate outputs during the second fusion for decoding.

The camera's movement is represented by data that continuously varies over time, requiring two Recurrent Linear Blocks (RLBs) in the generator's final portion to decode the time-varying characteristics of the latent representation.

AACOGAN employs two discriminators, inspired by [119], to evaluate the performance of the generator from two distinct perspectives, thus evaluating and enhancing the generator's performance for multiple requirements.

Accurately capturing subtle variations in the camera's position is vital to ensure the generator produces camera driving parameters closely resembling the ground truth. Given that minor differences in the shooting angle can significantly affect the overall outcome (example details shown in the experiment about the aesthetic score in Section IV B), properly evaluating the generator's output is essential. Consequently, the pose discriminator evaluates individual data points, c_t, generated from the input features o_t.

Conversely, the trajectory discriminator assesses the complete camera drive data, C, for an entire interactive action generated by O. This discriminator is pivotal in ensuring the overall coherence and realism of the generated camera movement. By evaluating the entire sequence of camera driving data, it can determine whether the generated camera movement adheres to a plausible and natural trajectory, rather than consisting of unrelated or jarring movements. This aspect is particularly important for maintaining immersion and ensuring a seamless user experience. As such, it is critical that the trajectory discriminator accurately assesses the quality of the generated camera movement, as inaccuracies could lead

to unrealistic or incoherent camera movements.

6.3.3 Loss Functions and Algorithm

Our goal is to maximize the $\mathsf{q}_{\mathsf{ref}}$ while minimizing the network loss during the training. Let θ represent the parameters of this camera drive data generator. Additionally, let ψ_p and ψ_t denote the parameters of the camera pose and trajectory discriminators, respectively. Then the objective function derivatives from Equation 6.2 for the AACOGAN can be expressed as follows:

$$\max_{\theta} Q_{\mathsf{ref}}(\cdot) = \min_{\theta} \max_{\psi_p, \psi_t} \omega_0 \mathcal{L}_{\mathsf{dist}} + \omega_1 \mathcal{L}_{\mathsf{corr}} + \omega_2 \tilde{\mathcal{L}}_{\mathsf{aes}} + \omega_3 \mathcal{L}_p + \omega_4 \mathcal{L}_t, \tag{6.8}$$

where $\omega_0, \cdots, \omega_4$ are the weight factors used to balance the loss terms, the loss functions $\mathcal{L}_{\mathsf{dist}}$ and $\mathcal{L}_{\mathsf{corr}}$ represent the distance function $\mathsf{Dis}(\cdot)$ and the correlation distance function $\mathsf{D}_{\mathsf{corr}}(\cdot)$ for the AACOGAN generator model. They are employed to compute the distance and similarity between C and $\hat{C}$. $\tilde{\mathcal{L}_{\mathsf{aes}}}$ is a loss function representation of $S_{\mathsf{aes}}(\cdot)$ which utilizes an anesthetic assessment model (AES) to evaluate the conformance of the results to general aesthetic standards. This loss function differs from others in that it requires the use of the resulting captured frames for evaluation, whereas the generator only generates camera drive data. As a result, it is utilized as a separate fine-tuning mechanism for the generator model after training. Following the completion of a training phase, the generator is used to generate a set of parameters C for each O in the training set. Each set of these C values is used to capture interactive actions in a virtual environment as a camera shot, resulting in a corresponding video clip. Each video clip can be represented as a sequence of images, and an AES is employed to evaluate these images. Due to the camera trajectories being

pre-designed within the optimization space of a toric surface, as applied in our experiments, we discern important insights from the data analysis of professional directors that we collect. The majority of cases highlighted that the two most critical elements determining the quality of the shot and the camera trajectory are the starting and ending points. Therefore, the beginning and ending frames in this sequence are given more weight in the calculation of the loss value as follows:

$$\tilde{\mathcal{L}_{\mathsf{aes}}} = \frac{1}{T}\sum_{t=1}^{T}\alpha_t(\mathsf{AES}_{\max} - \mathsf{AES}(c_t)), \tag{6.9}$$

where $\mathsf{AES}_{\max}$ is the max score of the employed AES and α_t is the weight factor for different frames over t.

As the number of characters captured by C can only be calculated after the actual video generation, there is no function based on $R(\cdot)$ in the loss function for the AACOGAN generator.

The discriminator loss function has two parts. The pose discriminator loss function, $\mathcal{L}_p$, which is similar to the $\mathcal{L}_{\mathsf{dist}}$ that calculates the pose difference C and $\hat{C}$ at each t. The trajectory discriminator loss function, $\mathcal{L}_t$, which is similar to the $\mathcal{L}_{\mathsf{corr}}$ that calculates the trajectory difference between C and $\hat{C}$. The discriminator loss can be defined as follows:

$$\begin{aligned} L_p = \sum_{t=1}^{T}(&- \mathbb{E}[\log D(c_t, \hat{c_t})] \\ &- \mathbb{E}[\log(1 - D(G(o_t)))]), \end{aligned} \tag{6.10}$$

$$L_t = -\mathbb{E}\left[\log D(C, \hat{C})\right] - \mathbb{E}\left[\log(1 - D(G(O)))\right], \tag{6.11}$$

where the D is the discriminator and G is the generator.

The pseudo-code for training AACOGAN is given in Algorithm 1. In the first and second loops, ψ_p is updated separately, and θ is updated in both loops. The third loop is implemented to refine θ based on aesthetic aspects for fine-tuning.

Algorithm 1 Training Procedure of AACOGAN

Input The extracted features defined in the pre-processing $O_n = [o_{n1}, o_{n2}, \cdots, o_{nT}]$ and the corresponding ground truth camera drive data $C_n = [c_{n1}, c_{n2}, \cdots, c_{nT}]$ for $n = 1, 2, \cdots, N$.

Output Generator parameters θ and two discriminator parameters ψ_p and ψ_t.

---

for $epoch = 1$ to max_epoch **do**
  for $iter_p = 1$ to k_p **do**
    Sample a mini-batch of input features and camera drive data pairs in frame $\{(o_t, c_t)\}$ from the training set.
    Generate a single camera drive data point $\hat{c}_t$.
    Calculate the $\mathcal{L}_{k_p} = \mathcal{L}_{\mathsf{dist}} + \mathcal{L}_{\mathsf{corr}} + \mathcal{L}_p$
    Update ψ_p and θ.
  end for
  for $iter_t = 1$ to k_t **do**
    Sample a mini-batch of input features and camera drive data for the entire interactive action pairs in $\{(O, C)\}$ from the training set.
    Generate camera drive data for the entire integrative action $\hat{C}$.
    Calculate the $\mathcal{L}_{k_t} = \mathcal{L}_{\mathsf{dist}} + \mathcal{L}_{\mathsf{corr}} + \mathcal{L}_t$
    Update ψ_t and θ.
  end for
end for
for $iter_t = 1$ to k_t **do**
  Sample a mini-batch of input features and aesthetic score pair for each interactive action.
  Calculate the $\tilde{\mathcal{L}_{aes}}$
  Update θ.
end for

Due to the precise camera parameters for each frame and the presence of noise during data generation, the resulting camera drive data may exhibit minor fluctuations between frames. These continuous, randomly oriented fluctuations can disrupt image continuity and reduce user immersion. To address this issue, our system's postprocessing applies smoothing to the final output data to create a more continuous curve by minimizing these fluctuations.

The smoothing process, which aims to refine the data points and generate a smoother curve in a 2D coordinate system, is achieved using a Savitzky-Golay filter [120]. To ensure the accuracy of the smoothed result in representing the underlying data, a window size of 5 is employed for the filter. The outcome is then evaluated using four different polynomials of varying degrees, ranging from 2 to 5. The smoothed result that most closely resembles the original data points is selected, thus preserving data curvature while reducing potential information loss due to the smoothing process. This step can be represented as follows:

$$C_{\mathsf{smooth}} = \min_{p=2}^{5} \mathsf{MSE}\{\mathsf{savitzky}_{\mathsf{golay}}(G(O), 5, p), G(O)\}. \tag{6.12}$$

6.4 Experiment

In this section, the evaluation of the proposed AACOGAN is carried out utilizing both objective and subjective metrics. To the best of our knowledge, there is no prior study on automatic cinematography in open-world scenarios, so comparisons are drawn between our results and conventional automatic cinematography techniques [94,101] commonly employed in general games and films. These two reference works have tackled the problem of automatic cinematography using different approaches. In Yu2022 [94], a rule-based language is utilized to train a Recurrent Neural Network (RNN) for capturing the essence of cinematography, which can be applied to new scenes. While in Wu2023 [101], the authors focused on individual characters and also used a GAN-based model to generate the camera movement according to the characters' emotions and actions. More experiment example video footage can be found at https://youtu.be/vhvgvE-DU2Y.

6.4.1 MineStory Dataset

Given the complexity and dynamism of user interactions within open-world environments, we developed a novel interactive action dataset named the MineStory dataset. This dataset, created using motion capture techniques, trains the AACOGAN to adapt to the diverse and ever-evolving nature of user actions in such virtual environments.

The MineStory dataset encompasses a total of 546 distinct actions, including most actions typically used in animation production and some actions specially tailored for our product. Each action is captured by a standard protocol that involves 25 joint nodes distributed throughout the character's entire body. The positional and rotational data of each node is captured at a rate of 30 frames per second, yielding a comprehensive skeletal animation.

To train the AACOGAN, we need a set of camera trajectory data for each action, as directed by a human operator in an open-world environment. We employ the toric surface method, as introduced in [74], to simplify the process of generating this data. This method conceptualizes the toric surface as a two-dimensional plane for creating camera movement trajectories, which can then be projected into a three-dimensional virtual space. This approach allows for the efficient generation of extensive camera trajectory data with a limited number of human operators. In our experiments, we generated 2 to 8 distinct sets of camera trajectory data for each interactive action. This data generation employed a randomized combination of parameters tailored to the unique aspects of the interactive activities, such as the characters' emotional states and the distance of interaction.

6.4.2 Objective Numerical Comparison

The comparisons between the AACOGAN and the baseline model have been performed with regard to several important aspects of automatic photography technology in open-world scenarios.

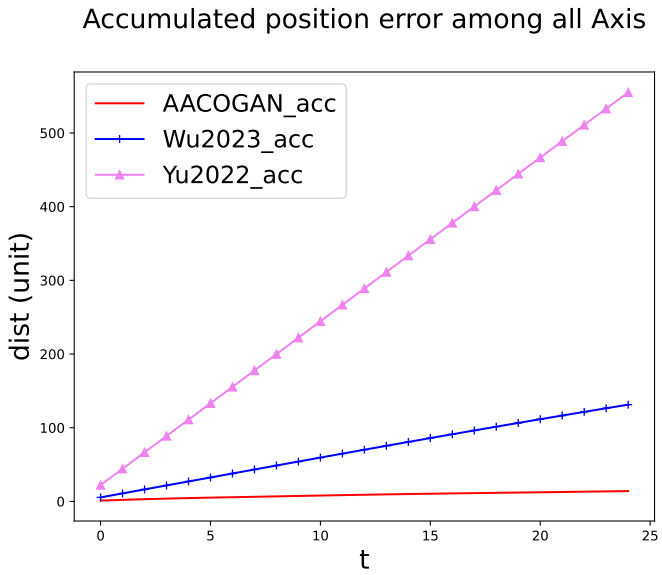


Figure 6.4: The accumulated difference between parameters of the generated camera positions and the ground truth camera position calculated over time, with the error distance expressed in unit distance

Precise Difference in Camera Position and Rotation The camera's location and orientation in the environment are specified by a set of six parameters, three of which pertain to position and three to rotation. This metric directly compares the difference between the generated camera position and rotation parameters with the ground truth for each approach. Figure. 6.4 and Figure. 6.5 present the sum of differences of all the x, y, and z axes between the generated camera parameters and the ground truth parameters for position parameters and rotation parameters are presented, respectively, over time. Compared to other methods, AACOGAN exhibits the least deviation in position from the ground truth in terms of results. The results show that there has been a reduction in the average positional error of 1.56 units

(93.6%) and an average reduction in the rotational error of 1.11 radians (55%), where the 'unit' for distance measurement is the unit distance for object positioning in the virtual environment.

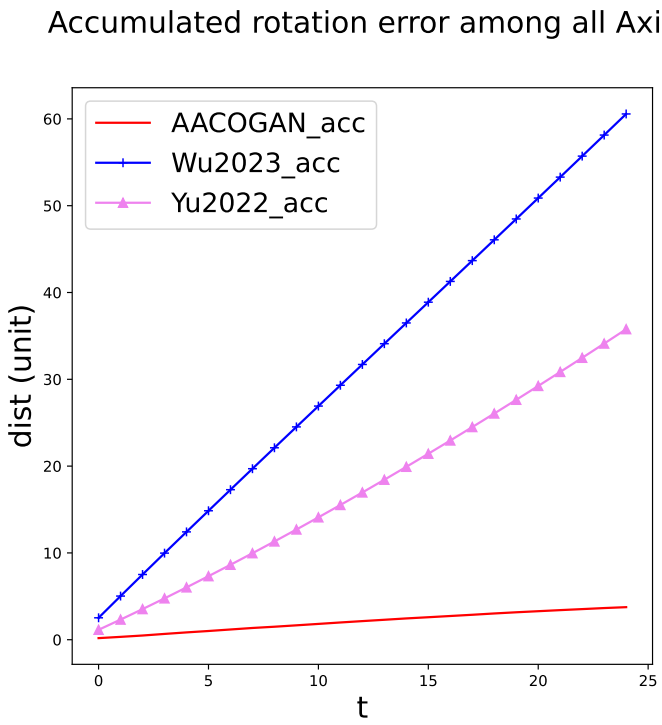


Figure 6.5: The accumulated differences between the generated camera parameters and the ground truth over time are shown for all the rotation parameters. The error distance is expressed in radians

Figure.6.6 offers a comprehensive depiction of the accumulated distance error for position parameters along the x, y, and z axes. The upper section of Figure.6.6 displays the accumulated difference between generated camera parameters and ground truth over time, for each axis individually. AACOGAN exhibits the most minor accumulated discrepancies across all scenarios in comparison to other methods. It is crucial to recognize that mere similarity in camera parameter numerical values does not guarantee similarity in camera movement. The camera's trajectory in three-dimensional space throughout the shot is of utmost significance. Consequently, we extended our evaluation by comparing the variation curves of

generated camera parameters for each method along each axis over time. The bottom section of Figure. 6.6 demonstrates the variation of camera rotation parameters over time, with the visual similarity between the AACOGAN-generated camera parameters and ground truth parameters being more distinct than the two baseline models.

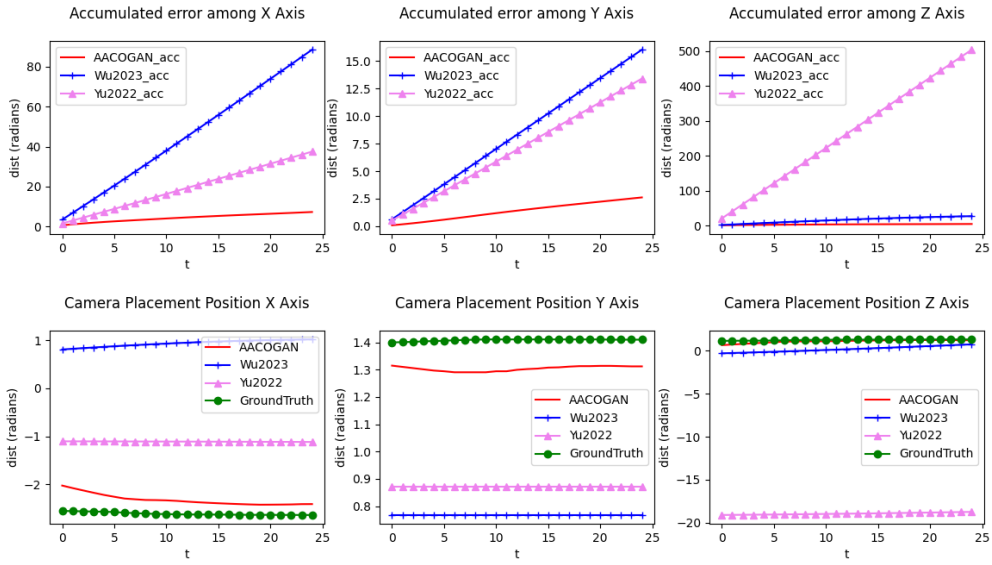


Figure 6.6: Top: The accumulated differences between the generated camera parameters by different methods and the ground truth ones are displayed over time. The parameters for the x, y, and z-axis are shown individually. Bottom: the exact values of the parameters generated by different approaches for each axis at each time point are presented

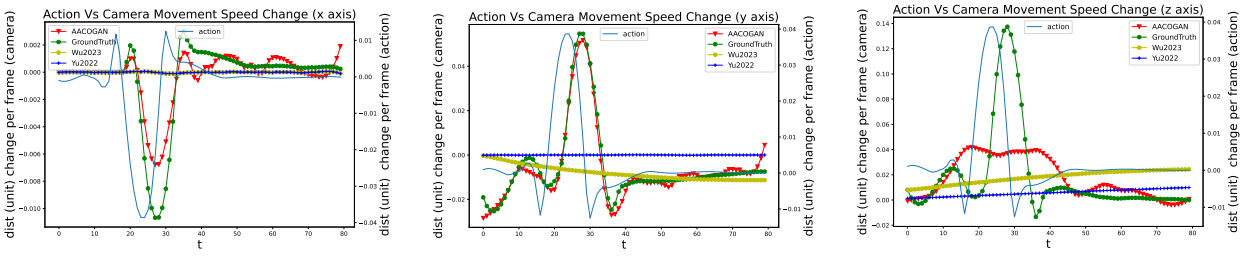
Figure. 6.7 displays a visual comparison of generated camera shots from different methods in terms of shooting position and angle. When contrasted with ground truth camera shots, the AACOGAN-generated frames exhibit a higher degree of similarity, corroborating the numerical results.

In summary, the camera movement generated by the AACOGAN model more closely approximates the ground truth, indicating that our method is more adept at learning the film director's lens language in open-world scenarios.

Correlation of Trajectories The correlation distance metric is employed to calculate the similarity between generated camera motion and the actual motion of subjects during

interactive actions. As previously mentioned in [98, 99], when camera movement closely mirrors the subjects' movements, it can enhance the sense of immersion for the audience. The significance of body parts in real action is often gauged by their range of motion and velocity.

Figure 6.7: Compare the frames captured by the virtual camera generated by different methods. From the visual inspection of these frames, we can intuitively observe the differences between the generated camera shots and the ground truth, in terms of position and orientation



(a) Change curves of the x-axis parameters

(b) Change curves of the y-axis parameters

(c) Change curves of the z-axis parameters

Figure 6.8: Interactive action comparisons of the change curves of the camera position parameters generated by different methods over a certain period with the corresponding skeletal animation motion curves of the interactive action

In this experiment, the joint exhibiting the largest range of motion and fastest movement

was selected as a reference to assess the correlation between subject and camera movements. Table 6.2 presents the similarity between various camera parameter curves generated by different automatic cinematography methods and the ground truth, quantified using correlation distance. The camera movement generated by AACOGAN is more similar to the actual subject movement than other methods, especially near peaks of subject movement.

The similarity between the camera trajectory generated by AACOGAN and the manually created camera exhibits the lowest discrepancy at 27%, while other methods display a minimum discrepancy of 95.3%. Compared to the other two references, the camera trajectory produced by AACOGAN demonstrates a decrease of 0.78 (73%) in the average correlation distance. This synchronized movement between the camera and the action can provide a superior experience for the user.

Table 6.2: Correlation distance between the various generated camera parameters and actual skeleton animation movement among different axis for a single action

	Wu2023	Yu2022	Manual	AACOGAN
X axis D_{corr}	0.84	0.95	0.43	0.55
Y axis D_{corr}	1.25	1.18	0.005	0.066
Z axis D_{corr}	1.78	1.1	0.17	0.25
Average D_{corr}	1.29	1.07	0.2	0.288

Figure. 6.8a, 6.8b and 6.8c illustrates the comparison between the camera transport mirror trajectory generated by different methods and the skeletal animation motion trajectory for the x-axis in the experiment environment. The y-axis in these figures represents the position of the character's skeletal joint (right y-axis) or the camera position (left y-axis) relative to the previous time step for each time step. It can be observed that the trajectory generated by AACOGAN has a higher similarity to the motion trajectory. The corresponding example video footage can be found at https://youtu.be/M24bHDvDnqk.

Multi-focus In open-world settings, interactive actions may not be solely centered on specific characters or objects. In such instances, the lens language employed by automatic cinematography technology should encompass as many related objects as feasible. This metric gauges the proportion of time during which camera movements, generated by various methods, successfully capture specified objects in multi-focus scenes.

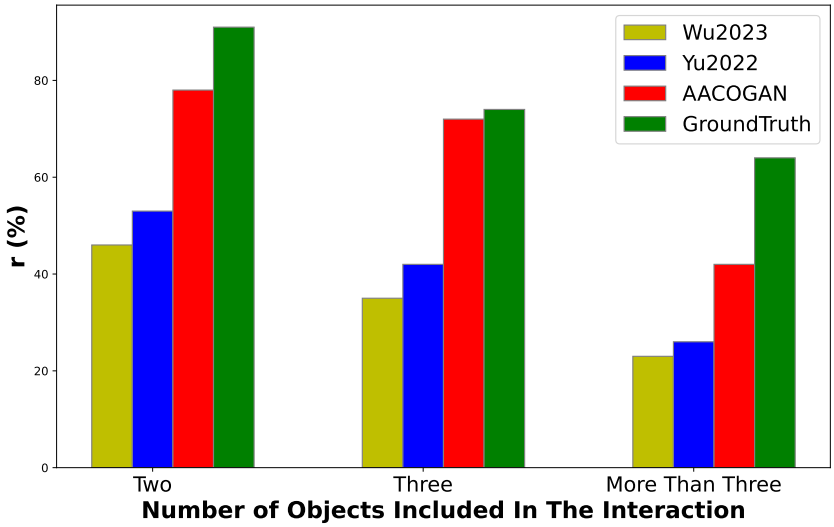


Figure 6.9: Comparison of different methods for calculating the r value under varying numbers of individuals and scenes

Higher proportions generally imply superior performance and user experience. A comparison of the capture ratio for designated characters or objects during interactive actions between AACOGAN and alternative methods is depicted in Figure. 6.9. The results reveal that AACOGAN captures more relevant characters over extended durations, thereby augmenting users' overall viewing experience. In comparison to other reference methods, AACOGAN demonstrates an average improvement of 22% and up to 32.9% in multi-focal scene image capturing, contingent upon content.

Figure. 6.10 displays frames captured by distinct automatic cinematography techniques

within a given multi-person interactive dance scene. Observing the image, the lens language generated by AACOGAN captures more comprehensive character images within the scene. The corresponding video footage example can be accessed at https://youtu.be/3KImvj9wabg.

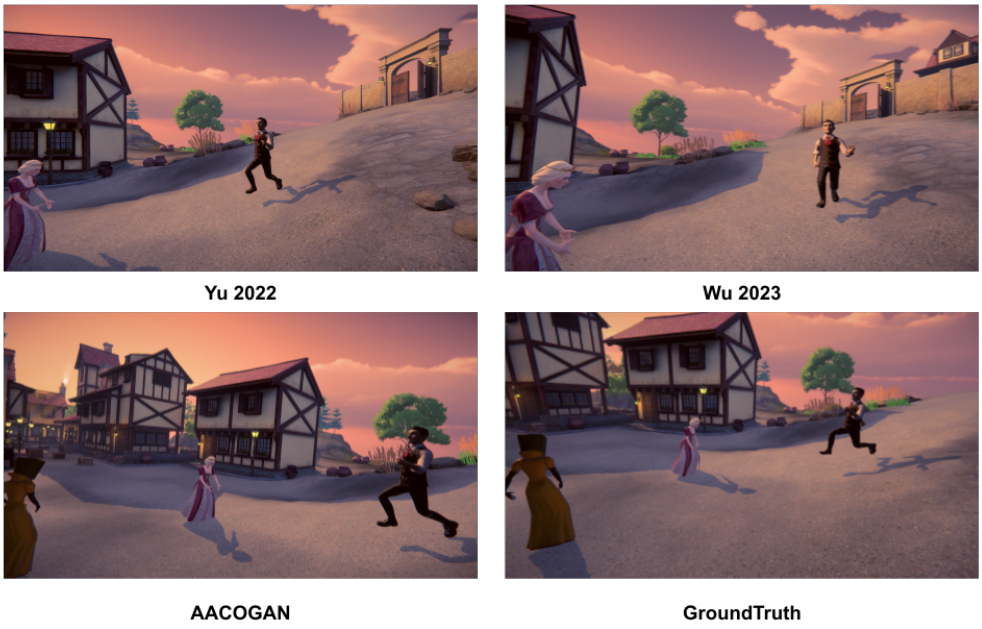


Figure 6.10: Frames captured by different automatic cinematography techniques in a dance scene

Aesthetic Score The AES [111, 113] offers an objective evaluation of images and assigns scores based on their aesthetic quality, with a higher score indicating a higher level of aesthetic appeal. This model is widely utilized in the realm of image and video content generation and provides valuable insights through artistic analysis of the output generated by these technologies. To the best of our knowledge, AACOGAN is the first work to apply this type of model in the field of automatic cinematography for shot generation. By incorporating aesthetic scores as part of the input for fine-tuning the generator of the AACOGAN, the resulting shots have higher aesthetic ratings than the original model without aesthetic-related adjustments.

In our experiments, employing AES led to a 9% average increase in the aesthetic score of images captured by the AACOGAN. For actions like over-wall jumping, illustrated in Figure. 6.11, the right-side camera shot generator captures more decadent lighting, background, and environmental information compared to the left-side perspective. This result is achieved

Figure 6.11: The use of aesthetic scores as a part of the input for the generator of AACOGAN resulted in improved visuals, as seen in the comparison between the original camera shooting direction (left) and the fine-tuned version camera shooting direction (right) after incorporating aesthetic considerations

by slightly lowering the camera position and raising the shooting angle. This distinction can be ascribed to the use of aesthetic scores in the fine-tuning process of AACOGAN's generator, causing a preference for content-rich camera angles over monotonous ones, such as those oriented towards the sky or ground. Aesthetic assessment models typically favor images with more content. The corresponding video footage example can be accessed at https://youtu.be/_je_Gg_QQG0.

The benefit of this additional aesthetic fine-tuning for the generator is twofold. Firstly, it enhances the aesthetic quality of the automatically generated camera movement. Secondly,

it preserves the quality of shots produced by the AACOGAN. This results in shots that better align with the preferences of a broader audience, without compromising the utilized lens language.

Real-time Performance In Table 6.3, we present a detailed comparison of the performance of the AACOGAN model at various latency levels ranging from 5 to 30 frames. For each latency level, the table provides data on the frames per second (FPS) and floating point operations per second (FLOPS) metrics. We use average D_{corr} to represent the quality of the generated camera position. The data clearly demonstrate an increase in latency, which effectively allows the model to leverage more frame data, and improves the predictive capability of the model for the subsequent camera position. However, this is achieved at the expense of an increase in computational complexity, as shown by the higher FLOPS. Thus, it becomes a trade-off between real-time responsiveness and the quality of camera position prediction, necessitating careful tuning according to the specific demands of the gaming environment.

Table 6.3: The real-time performance metrics for the AACOGAN model with different input latency. The number in parentheses after the model indicates the number of delayed frames. The FLOPs unit is in a million flops per second

Model	FLOPs(M)	FPS	Latency	Average D_{corr}
AACOGAN(5)	64.33	78.9	0.012	0.393
AACOGAN(10)	113.93	39.63	0.025	0.356
AACOGAN(20)	217.97	20.79	0.048	0.307
AACOGAN(30)	317.28	14.2	0.071	0.288

6.4.3 Subjective Evaluation

Emotional Reinforcement The lens language also plays an essential role in expressing the atmosphere and emotions of characters. The emotion of a character can be analyzed through the user's facial expressions [121], speech [122], or body movements [123]. In different

scenarios or characters' emotional states, the camera movement should reflect the emotions of the characters by varying to a greater or lesser degree.

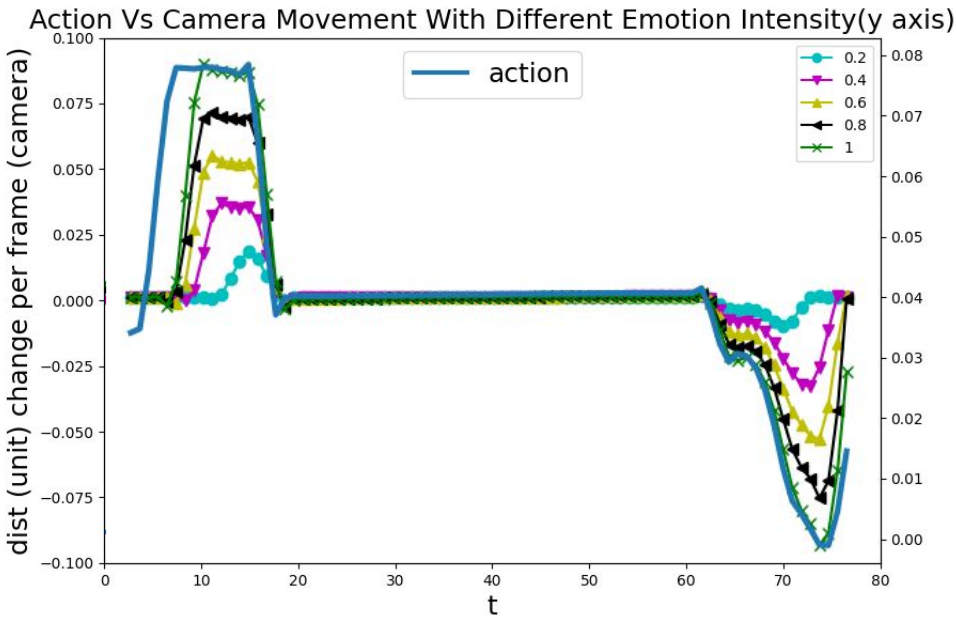


Figure 6.12: Comparison of the change curves of the y-axis position parameter of the camera trajectory generated by different emotional states

This metric assesses the influence of a character's emotional state on camera trajectory, where stronger emotions should yield a more significant impact on camera movement. The comparison of the character's emotional state's effect on camera movements generated for identical interactive actions is depicted in Figure. 6.12. The y-axis value, representing emotion, denotes the stability of the character's emotional state; a value closer to 1 indicates an unstable or extremely unstable emotion (such as anger), while a value closer to 0 signifies a calmer state.

The y-axis in Figure. 6.12 represents the character's skeletal joint position (right y-axis) or the camera position (left y-axis) relative to the previous time step for each time step. As the character's emotional state intensifies, a more pronounced impact on camera movement is

evident. Consequently, the camera movement generated by AACOGAN more accurately reflects the characters' actual emotions and creates a superior atmosphere compared to reference methods. The corresponding video footage example can be found athttps://youtu.be/HfmotyfEWHw.

Human Evaluation It is crucial to acknowledge that the previously mentioned evaluation methods do not entirely capture the superiority of AACOGAN's generated camera shots in open-world contexts compared to other automatic cinematography techniques concerning actual user experience. Consequently, enlisting real users to assess the generated shots is indispensable. In addition to a general ranking-based human evaluation, participants will be asked to appraise the shots in distinct aspects: 1) shot quality (Frames Quality), 2) shot consistency with the actual interaction (Consistence), 3) representation of characters and objects involved in the interaction (Completeness), and 4) the enhancement of emotions conveyed by the characters within the shot (Emotional Enhancement).

Table 6.4: The results of the subjective evaluation. The scores for the four aspects of the shot quality are on a scale of 1-5, with 5 being the best. The final row shows the ranking of the four shots created by different methods, with a lower score indicating a better ranking

	Wu2023	Yu2022	GroundTruth	AACOGAN
Frames Quality	4	4	**4.67**	4.5
Consistency	3.16	3	4.5	**4.67**
Completeness	2.3	2.16	**3.83**	**3.83**
Emotional	3.83	2	**4.6**	4.16
Overall Ranking	3.16	3.67	**1.33**	1.83

As shown in Table 6.4, the results of the subjective evaluation indicate that compared to the baseline method, AACOGAN received the highest ranking in the sorting task. In comparison to other methods across various aspects, AACOGAN also received higher evaluations.

In summary, our experimental outcomes indicate that AACOGAN effectively generates

camera parameters and produces shots more akin to those captured by human operators during interactive actions. This is demonstrated by comparing AACOGAN-generated parameters to baseline methods using metrics such as camera pose similarity, motion similarity, and character and object coverage. Moreover, aesthetic assessments and subjective evaluations conducted by human participants corroborate AACOGAN's superiority in creating visually appealing and interaction-aligned shots. Consequently, these findings suggest the potential of the AACOGAN method to enhance the quality of automatic cinematography in open-world interactive scenarios.

6.5 Conclusions

The advent of multimedia technology in the entertainment industry has bestowed upon consumers an unprecedented level of autonomy in their media consumption experiences. Within virtual open-world environments, automatic cinematography has emerged as an instrumental factor in delivering immersive experiences, catering to the users' growing demand for more engaging forms of interaction. In this study, we introduce AACOGAN, a technique for automatic cinematography designed for user-initiated interactions within open-world scenarios, leveraging GANs. This model incorporates various elements such as skeletal animations of interactive actions, positional relationships between interactive objects and characters, as well as character emotions, facilitating the effective generation of suitable camera movements for a wide array of interactive actions. Moreover, the integration of aesthetic scores into the generator's training process significantly enhances the quality of the shots generated. The results of our experiments substantiate the efficiency of the AACOGAN approach in producing camera shots that rival the quality of human-generated shots, demanding minimal

input, thus leading to a more engaging user experience and substantially reducing costs.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

In this dissertation, we have articulated a comprehensive framework for automated animation production, offering an in-depth exploration of two pivotal technologies—lip-sync speech animation and auto cinematography—that constitute this framework.

Initially, we delineated the foundational modules embedded within the automatic animation production framework and elaborated upon their respective functionalities. Although human intervention is still required in the transitional journey from script to animation, our framework has notably ameliorated time consumption and expertise prerequisites. Each module integrates technologies from disparate domains; given the future maturation of these technologies, our framework holds the potential for completely autonomous animation production.

In addressing the challenges of lip-sync speech animation, we propose RealPRNet, an innovative deep-learning-based real-time phoneme recognition network. By leveraging spatial and temporal patterns in raw audio data and incorporating a long short-term memory (LSTM) stack block, RealPRNet facilitates competitive real-time speech animation within the *Stage Performance* module of our framework. Empirical evaluations confirm that RealPRNet outperforms extant algorithms, registering a 20% improvement in Phoneme Error Rate (PER)

and a 4% enhancement in Block Error Distance (BDE) in optimal cases.

Subsequently, our research delves into auto cinematography, presenting two distinct strategies: a rule-based approach (T2A) and a method mimicking human lens language behavior (RT2A). T2A efficiently expedites the script-to-video creation process by capitalizing on advancements in computational cinematography and video understanding. Utilizing fidelity and aesthetic models in tandem, we have formulated camera placement in 3D environments as an optimization problem, solvable through dynamic programming. Our experimental data substantiates that T2A can reduce manual animation production efforts by approximately 74% and enhance the perceptual quality of output videos by up to 35%. Alternatively, RT2A records directorial decisions regarding camera settings for future training of auto cinematography agents. A well-engineered reward function aids the algorithm in mimicking the human director's decision-making, resulting in significant gains in camera placement acceptance rate and the rhythm of camera switching.

Moreover, we have adapted our auto cinematography techniques to comply with the evolving demands of the modern entertainment sector, where users increasingly seek to create unique animation experiences in open-world scenarios. To this end, we introduce AACOGAN, a novel Generative Adversarial Network (GAN)-based methodology designed for these user-driven interactive experiences. This innovative model synthesizes various elements, including skeletal animations, spatial relationships among interactive entities, and expressive character emotions, thereby systematizing the generation of suitable camera movements across a wide range of interactive sequences. Aesthetic metrics are also integrated during the training phase to elevate the quality of the resulting shots. Experimental validation confirms AACOGAN's efficacy, with generated camera angles and movements rivaling those crafted by human professionals.

In summary, the framework and technologies explored in this dissertation endeavor to democratize animation production, significantly reducing both the time and expertise required. Notably, our innovations in automatic cinematography allow for the archival of current directors' lens language patterns, thereby preserving their unique cinematic contributions for future AI-driven projects.

7.2 Future Work

The quest for full automation in animation production remains a focal point of ongoing research. While each module in our presented framework has already demonstrated substantial utility, there exists a compelling scope for further advancements as technological landscapes evolve. Drawing upon our comprehensive analyses in the realms of lip-sync speech animation and auto cinematography, we propose several promising avenues for future investigations:

1. Influence of Emotional Context on Viseme Dynamics - Emotional states have a non-negligible impact on phoneme articulation. Notably, even identical phonemes can manifest divergent visemes under varying emotional conditions. This observation underscores the importance of incorporating emotional contexts into viseme animation algorithms to enhance the realism and emotional expressiveness of animated characters.
2. Adaptive Auto Cinematography - Much like the impossibility of a single directorial vision universally appealing to all audiences, there exists no monolithic cinematographic language that can satiate diverse viewer preferences. As auto cinematography technologies mature, an intriguing direction for future research lies in the development of adaptive, or personalized, auto cinematography. Here, identical scripts could be rendered through divergent cinematographic lenses, tailored to individual viewer predilections,

thereby enhancing audience engagement and re-watchability.

3. Unconstrained Camera Placement in Auto Cinematography - Existing auto cinematography solutions often circumscribe the spectrum of potential camera placements and orientations due to computational limitations. For example, the utilization of toric surfaces in AACOGAN serves as a case in point. While these limitations expedite computational processes and uphold a baseline quality of results, they also stifle the creative potential for novel and inventive camera placements. Thus, devising techniques that transcend these restrictive boundaries, thereby allowing cameras to operate within a 'free space,' stands as a fruitful line of inquiry.

BIBLIOGRAPHY

[1] S. Taylor, T. Kim, Y. Yue, M. Mahler, J. Krahe, A. G. Rodriguez, J. Hodgins, and I. Matthews, "A deep learning approach for generalized speech animation," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 4, p. 93, 2017.

[2] Y. Zhou, Z. Xu, C. Landreth, E. Kalogerakis, S. Maji, and K. Singh, "Visemenet: Audio-driven animator-centric speech animation," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 4, p. 161, 2018.

[3] H. Subramonyam, W. Li, E. Adar, and M. Dontcheva, "Taketoons: Script-driven performance animation," in *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology*, 2018, pp. 663–674.

[4] Z. Yu, E. Guo, H. Wang, and J. Ren, "Bridging script and animation utilizing — a new automatic cinematography model," in *Submitted to MIPR 2022*.

[5] L.Sun and H.Wang, "Director-hint based auto-cinematography," in *US Patent 11,120,638*, 2021.

[6] F. Tao and C. Busso, "End-to-end audiovisual speech recognition system with multitask learning," *IEEE Transactions on Multimedia*, 2020.

[7] M. Mehrabani, S. Bangalore, and B. Stern, "Personalized speech recognition for internet of things," in *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*. IEEE, 2015, pp. 369–374.

[8] M. Dawodi, J. A. Baktash, T. Wada, N. Alam, and M. Z. Joya, "Dari speech classification using deep convolutional neural network," in *2020 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS)*. IEEE, 2020, pp. 1–4.

[9] M. La Mura and P. Lamberti, "Human-machine interaction personalization: a review on gender and emotion recognition through speech analysis," in *2020 IEEE International Workshop on Metrology for Industry 4.0 & IoT*. IEEE, 2020, pp. 319–323.

[10] G. Llorach, A. Evans, J. Blat, G. Grimm, and V. Hohmann, "Web-based live speech-driven lip-sync," in *2016 8th International Conference on Games and Virtual Worlds for Serious Applications (VS-GAMES)*. IEEE, 2016, pp. 1–4.

[11] Y. Xu, A. W. Feng, S. Marsella, and A. Shapiro, "A practical and configurable lip sync method for games," in *Proceedings of Motion on Games*. ACM, 2013, pp. 131–140.

[12] P. Edwards, C. Landreth, E. Fiume, and K. Singh, "Jali: an animator-centric viseme model for expressive lip synchronization," *ACM Transactions on Graphics (TOG)*, vol. 35, no. 4, p. 127, 2016.

[13] C. G. Fisher, "Confusions among visually perceived consonants," *Journal of speech and hearing research*, vol. 11, no. 4, pp. 796–804, 1968.

[14] J. Michalek and J. Vaněk, "A survey of recent dnn architectures on the timit phone recognition task," in *International Conference on Text, Speech, and Dialogue*. Springer, 2018, pp. 436–444.

[15] S. Kapadia, V. Valtchev, and S. J. Young, "Mmi training for continuous phoneme recognition on the timit database," in *1993 IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 2. IEEE, 1993, pp. 491–494.

[16] I. Bromberg, Q. Qian, J. Hou, J. Li, C. Ma, B. Matthews, A. Moreno-Daniel, J. Morris, S. M. Siniscalchi, Y. Tsao *et al.*, "Detection-based asr in the automatic speech attribute transcription project," in *Eighth Annual Conference of the International Speech Communication Association*, 2007.

[17] J. Morris and E. Fosler-Lussier, "Combining phonetic attributes using conditional random fields," in *Ninth International Conference on Spoken Language Processing*, 2006.

[18] J. Park and H. Ko, "Real-time continuous phoneme recognition system using class-dependent tied-mixture hmm with hbt structure for speech-driven lip-sync," *IEEE Transactions on Multimedia*, vol. 10, no. 7, pp. 1299–1306, 2008.

[19] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. J. Lang, "Phoneme recognition using time-delay neural networks," *IEEE transactions on acoustics, speech, and signal processing*, vol. 37, no. 3, pp. 328–339, 1989.

[20] J. Kim, K. Hwang, and W. Sung, "X1000 real-time phoneme recognition vlsi using feed-forward deep neural networks," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2014, pp. 7510–7514.

[21] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE, 2013, pp. 6645–6649.

[22] T. N. Sainath, O. Vinyals, A. Senior, and H. Sak, "Convolutional, long short-term memory, fully connected deep neural networks," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015, pp. 4580–4584.

[23] M. McAuliffe, M. Socolof, S. Mihuc, M. Wagner, and M. Sonderegger, "Montreal forced aligner: Trainable text-speech alignment using kaldi." in *Interspeech*, 2017, pp. 498–502.

[24] P. L. Jackson, "The theoretical minimal unit for visual speech perception: Visemes and coarticulation." *The Volta Review*, 1988.

[25] H. L. Bear and R. Harvey, "Phoneme-to-viseme mappings: the good, the bad, and the ugly," *Speech Communication*, vol. 95, pp. 40–67, 2017.

[26] C. Bregler, M. Covell, and M. Slaney, "Video rewrite: driving visual speech with audio." in *Siggraph*, vol. 97, 1997, pp. 353–360.

[27] E. Bozkurt, C. E. Erdem, E. Erzin, T. Erdem, and M. Ozkan, "Comparison of phoneme and viseme based acoustic units for speech driven realistic lip animation," in *2007 3DTV Conference*. IEEE, 2007, pp. 1–4.

[28] P. Waddell, G. Jones, and A. Goldberg, "Audio/video synchronization standards and solutions a status report," *Advanced Television Systems Committee*, vol. 21, 1998.

[29] A. C. Younkin and P. J. Corriveau, "Determining the amount of audio-video synchronization errors perceptible to the average end-user," *IEEE Transactions on Broadcasting*, vol. 54, no. 3, pp. 623–627, 2008.

[30] A. Graves, N. Jaitly, and A.-r. Mohamed, "Hybrid speech recognition with deep bidirectional lstm," in *2013 IEEE workshop on automatic speech recognition and understanding*. IEEE, 2013, pp. 273–278.

[31] Y. Fan, Y. Qian, F.-L. Xie, and F. K. Soong, "Tts synthesis with bidirectional lstm based recurrent neural networks," in *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

[32] A. Pearce, B. Wyvill, G. Wyvill, and D. Hill, "Speech and expression: A computer solution to face animation," in *Graphics Interface*, vol. 86, 1986, pp. 136–140.

[33] S. A. King and R. E. Parent, "Creating speech-synchronized animation," *IEEE Transactions on visualization and computer graphics*, vol. 11, no. 3, pp. 341–352, 2005.

[34] K. Chen and Q. Huo, "Training deep bidirectional lstm acoustic model for lvcsr by a context-sensitive-chunk bptt approach," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 7, pp. 1185–1193, 2016.

[35] O. Abdel-Hamid, A.-r. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu, "Convolutional neural networks for speech recognition," *IEEE/ACM Transactions on audio, speech, and language processing*, vol. 22, no. 10, pp. 1533–1545, 2014.

[36] S. Zhang, S. Zhang, T. Huang, and W. Gao, "Speech emotion recognition using deep convolutional neural network and discriminant temporal pyramid matching," *IEEE Transactions on Multimedia*, vol. 20, no. 6, pp. 1576–1590, 2017.

[37] P. Zhan and A. Waibel, "Vocal tract length normalization for large vocabulary continuous speech recognition," Carnegie-Mellon Univ, School of Computer Science, Tech. Rep., 1997.

[38] M. J. Gales, "Maximum likelihood linear transformations for hmm-based speech recognition," *Computer speech & language*, vol. 12, no. 2, pp. 75–98, 1998.

[39] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[40] T. N. Sainath, B. Kingsbury, A.-r. Mohamed, G. E. Dahl, G. Saon, H. Soltau, T. Beran, A. Y. Aravkin, and B. Ramabhadran, "Improvements to deep convolutional neural networks for lvcsr," in *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*. IEEE, 2013, pp. 315–320.

[41] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *Fifteenth annual conference of the international speech communication association*, 2014.

[42] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz *et al.*, "The kaldi speech recognition toolkit," in *IEEE 2011 workshop on automatic speech recognition and understanding*, no. CONF. IEEE Signal Processing Society, 2011.

[43] H.Wang, "Write-a-movie: Unifying writing and shooting," in *US Patent filed Oct.*, 2020.

[44] Y. Niu and F. Liu, "What makes a professional video? a computational aesthetics approach," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 7, pp. 1037–1049, 2012.

[45] A. Truong, F. Berthouzoz, W. Li, and M. Agrawala, "Quickcut: An interactive tool for editing narrated video," in *Proc. 29th Annual Symposium on User Interface Software and Technology*, 2016, pp. 497–507.

[46] X. Yang, T. Zhang, and C. Xu, "Text2video: An end-to-end learning framework for expressing text with videos," *IEEE Transactions on Multimedia*, vol. 20, no. 9, pp. 2360–2370, 2018.

[47] M. Wang, G.-W. Yang, S.-M. Hu, S.-T. Yau, and A. Shamir, "Write-a-video: computational video montage from themed text." *ACM Trans. Graph.*, vol. 38, no. 6, pp. 177–1, 2019.

[48] Q. Galvane, "Automatic cinematography and editing in virtual environments." Ph.D. dissertation, Université Grenoble Alpes (ComUE), 2015.

[49] A. Louarn, M. Christie, and F. Lamarche, "Automated staging for virtual cinematography," in *Proceedings of the 11th Annual International Conference on Motion, Interaction, and Games*, 2018, pp. 1–10.

[50] I. Arev, H. S. Park, Y. Sheikh, J. Hodgins, and A. Shamir, "Automatic editing of footage from multiple social cameras," *ACM Transactions on Graphics (TOG)*, vol. 33, no. 4, pp. 1–11, 2014.

[51] H. Jiang, B. Wang, X. Wang, M. Christie, and B. Chen, "Example-driven virtual cinematography by learning camera behaviors," *ACM Transactions on Graphics (TOG)*, vol. 39, no. 4, pp. 45–1, 2020.

[52] L. Gao, Z. Guo, H. Zhang, X. Xu, and H. T. Shen, "Video captioning with attention-based lstm and semantic consistency," *IEEE Transactions on Multimedia*, vol. 19, no. 9, pp. 2045–2055, 2017.

[53] M. Gardner, J. Grus, M. Neumann, O. Tafjord, P. Dasigi, N. F. Liu, M. Peters, M. Schmitz, and L. S. Zettlemoyer, "Allennlp: A deep semantic natural language processing platform," 2017.

[54] S. Kim, K. Cha, M. Kim, and E. Lee, "3d animation using visual script language," in *The Fifth International Workshop on Distributed Multmedia Systems, Taiwan.* Citeseer, 1998, pp. 109–113.

[55] D. C. Petriu, X. L. Yang, and T. E. Whalen, "Behavior-based script language for anthropomorphic avatar animation in virtual environments," in *2002 IEEE International Symposium on Virtual and Intelligent Measurement Systems (IEEE Cat. No. 02EX545).* IEEE, 2002, pp. 105–110.

[56] C. Liang, C. Xu, J. Cheng, W. Min, and H. Lu, "Script-to-movie: a computational framework for story movie composition," *IEEE transactions on multimedia*, vol. 15, no. 2, pp. 401–414, 2012.

[57] M. Hayashi, S. Inoue, M. Douke, N. Hamaguchi, H. Kaneko, S. Bachelder, and M. Nakajima, "T2v: New technology of converting text to cg animation," *ITE Transactions on Media Technology and Applications*, vol. 2, no. 1, pp. 74–81, 2014.

[58] K. Lee, L. He, M. Lewis, and L. Zettlemoyer, "End-to-end neural coreference resolution," *arXiv preprint arXiv:1707.07045*, 2017.

[59] M. Leake, A. Davis, A. Truong, and M. Agrawala, "Computational video editing for dialogue-driven scenes." *ACM Trans. Graph.*, vol. 36, no. 4, pp. 130–1, 2017.

[60] N. Joshi, W. Kienzle, M. Toelle, M. Uyttendaele, and M. F. Cohen, "Real-time hyperlapse creation via optimal frame selection," *ACM Transactions on Graphics (TOG)*, vol. 34, no. 4, pp. 1–9, 2015.

[61] G. Abdollahian, C. M. Taskiran, Z. Pizlo, and E. J. Delp, "Camera motion-based analysis of user generated video," *IEEE Transactions on Multimedia*, vol. 12, no. 1, pp. 28–41, 2009.

[62] M. Gschwindt, E. Camci, R. Bonatti, W. Wang, E. Kayacan, and S. Scherer, "Can a robot become a movie director? learning artistic principles for aerial cinematography," *arXiv preprint arXiv:1904.02579*, 2019.

[63] J. Wang, M. Xu, L. Jiang, and Y. Song, "Attention-based deep reinforcement learning for virtual cinematography of 360° videos," *IEEE Transactions on Multimedia*, 2020.

[64] M. Radut, M. Evans, K. To, T. Nooney, and G. Phillipson, "How good is good enough? the challenge of evaluating subjective quality of ai-edited video coverage of live events." in *WICED@ EG/EuroVis*, 2020, pp. 17–24.

[65] S. Zhao, Y. Liu, Y. Han, R. Hong, Q. Hu, and Q. Tian, "Pooling the convolutional layers in deep convnets for video action recognition," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, no. 8, pp. 1839–1849, 2017.

[66] P. Wang, Y. Cao, C. Shen, L. Liu, and H. T. Shen, "Temporal pyramid pooling-based convolutional neural network for action recognition," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 27, no. 12, pp. 2613–2622, 2016.

[67] H. Wu, X. Ma, and Y. Li, "Spatiotemporal multimodal learning with 3d cnns for video action recognition," *IEEE Transactions on Circuits and Systems for Video Technology*, 2021.

[68] T. V. Nguyen, Z. Song, and S. Yan, "Stap: Spatial-temporal attention-aware pooling for action recognition," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 25, no. 1, pp. 77–86, 2014.

[69] B. Wang, L. Ma, W. Zhang, and W. Liu, "Reconstruction network for video captioning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7622–7631.

[70] S. Chen, Q. Jin, J. Chen, and A. G. Hauptmann, "Generating video descriptions with latent topic guidance," *IEEE Transactions on Multimedia*, vol. 21, no. 9, pp. 2407–2418, 2019.

[71] W. Xu, J. Yu, Z. Miao, L. Wan, Y. Tian, and Q. Ji, "Deep reinforcement polishing network for video captioning," *IEEE Transactions on Multimedia*, 2020.

[72] D. Shao, Y. Zhao, B. Dai, and D. Lin, "Finegym: A hierarchical video dataset for fine-grained action understanding," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 2616–2625.

[73] C. Yang, Y. Xu, J. Shi, B. Dai, and B. Zhou, "Temporal pyramid network for action recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 591–600.

[74] C. Lino and M. Christie, "Efficient composition for virtual camera control," 2012.

[75] R. Thompson and C. Bowen, *Grammar of the Shot*. Taylor & Francis, 2009.

[76] J. E. Cutting, J. E. DeLong, and C. E. Nothelfer, "Attention and the evolution of hollywood film," *Psychological science*, vol. 21, no. 3, pp. 432–439, 2010.

[77] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proc. 2014 conf. EMNLP*, 2014, pp. 1532–1543.

[78] J. K. Haas, "A history of the unity game engine," 2014.

[79] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell, "Long-term recurrent convolutional networks for visual recognition and description," in *Proc. IEEE conf. compu. vision and pattern recog.*, 2015, pp. 2625–2634.

[80] J. Xu, T. Mei, T. Yao, and Y. Rui, "Msr-vtt: A large video description dataset for bridging video and language," in *Proc. IEEE conf. on computer vision and pattern recognition*, 2016, pp. 5288–5296.

[81] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[82] G. Mercado, *The filmmaker's eye: Learning (and breaking) the rules of cinematic composition*. Routledge, 2013.

[83] Y. Li, "Deep reinforcement learning: An overview," *arXiv preprint arXiv:1701.07274*, 2017.

[84] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 30, no. 1, 2016.

[85] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[86] N. Passalis and A. Tefas, "Deep reinforcement learning for controlling frontal person close-up shooting," *Neurocomputing*, vol. 335, pp. 37–47, 2019.

[87] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*. PMLR, 2016, pp. 1928–1937.

[88] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[89] D. Bordwell, J. Staiger, and K. Thompson, *The classical Hollywood cinema: Film style & mode of production to 1960*. Columbia University Press, 1985.

[90] M. Svanera, S. Benini, N. Adami, R. Leonardi, and A. B. Kovács, "Over-the-shoulder shot detection in art films," in *2015 13th International Workshop on Content-Based Multimedia Indexing (CBMI)*. IEEE, 2015, pp. 1–6.

[91] P. Sweetser and D. Johnson, "Player-centered game environments: Assessing player opinions, experiences, and issues," in *International Conference on Entertainment Computing*. Springer, 2004, pp. 321–332.

[92] H. P. Martínez, A. Jhala, and G. N. Yannakakis, "Analyzing the impact of camera viewpoint on player psychophysiology," in *2009 3rd International Conference on Affective Computing and Intelligent Interaction and Workshops*. IEEE, 2009, pp. 1–6.

[93] P. Burelli and G. N. Yannakakis, "Towards adaptive virtual camera control in computer games," in *International symposium on Smart Graphics*. Springer, 2011, pp. 25–36.

[94] Z. Yu, H. Wang, A. K. Katsaggelos, and J. Ren, "A novel automatic content generation and optimization framework," *IEEE Internet of Things Journal*, 2023.

[95] P. Burelli, "Game cinematography: From camera control to player emotions," in *Emotion in Games*. Springer, 2016, pp. 181–195.

[96] ——, "Virtual cinematography in games: investigating the impact on player experience," in *Foundations of Digital Games: The 8th International Conference on the Foundations of Digital Games*. Society for the Advancement of the Science of Digital Games, 2013.

[97] R. F. Simons, B. H. Detenber, T. M. Roedema, and J. E. Reiss, "Emotion processing in three systems: The medium and the message," *Psychophysiology*, vol. 36, no. 5, pp. 619–627, 1999.

[98] M. Haigh-Hutchinson, *Real time cameras: A guide for game designers and developers*. CRC Press, 2009.

[99] M. Christie, P. Olivier, and J.-M. Normand, "Camera control in computer graphics," in *Computer Graphics Forum*, vol. 27, no. 8. Wiley Online Library, 2008, pp. 2197–2218.

[100] B. Tomlinson, B. Blumberg, and D. Nain, "Expressive autonomous cinematography for interactive virtual environments," in *Proceedings of the fourth international conference on Autonomous agents*, 2000, pp. 317–324.

[101] X. Wu, H. Wang, and A. K. Katsaggelos, "The secret of immersion: actor driven camera movement generation for auto-cinematography," 2023.

[102] Q. Galvane, R. Ronfard, M. Christie, and N. Szilas, "Narrative-driven camera control for cinematic replay of computer games," in *Proceedings of the Seventh International Conference on motion in games*, 2014, pp. 109–117.

[103] Y. Deng, C. C. Loy, and X. Tang, "Image aesthetic assessment: An experimental survey," *IEEE Signal Processing Magazine*, vol. 34, no. 4, pp. 80–106, 2017.

[104] A. McMahan, "Immersion, engagement, and presence: A method for analyzing 3-d video games," in *The video game theory reader*. Routledge, 2013, pp. 67–86.

[105] N. Tandon, G. Weikum, G. d. Melo, and A. De, "Lights, camera, action: Knowledge extraction from movie scripts," in *Proceedings of the 24th International Conference on World Wide Web*, 2015, pp. 127–128.

[106] D. B. Christianson, S. E. Anderson, L.-w. He, D. H. Salesin, D. S. Weld, and M. F. Cohen, "Declarative camera control for automatic cinematography," in *AAAI/IAAI, Vol. 1*, 1996, pp. 148–155.

[107] Y. Dang, C. Huang, P. Chen, R. Liang, X. Yang, and K.-T. Cheng, "Path-analysis-based reinforcement learning algorithm for imitation filming," *IEEE Transactions on Multimedia*, 2022.

[108] C. Lino and M. Christie, "Intuitive and efficient camera control with the toric space," *ACM Transactions on Graphics (TOG)*, vol. 34, no. 4, pp. 1–12, 2015.

[109] X. Lu, Z. Lin, H. Jin, J. Yang, and J. Z. Wang, "Rating image aesthetics using deep learning," *IEEE Transactions on Multimedia*, vol. 17, no. 11, pp. 2021–2034, 2015.

[110] X. Tian, Z. Dong, K. Yang, and T. Mei, "Query-dependent aesthetic model with deep learning for photo quality assessment," *IEEE Transactions on Multimedia*, vol. 17, no. 11, pp. 2035–2048, 2015.

[111] H. Talebi and P. Milanfar, "Nima: Neural image assessment," *IEEE transactions on image processing*, vol. 27, no. 8, pp. 3998–4011, 2018.

[112] M. Haigh-Hutchinson, "Fundamentals of real-time camera design," in *GDC*, vol. 5, 2005, p. 20.

[113] L. Zhao, M. Shang, F. Gao, R. Li, F. Huang, and J. Yu, "Representation learning of image composition for aesthetic prediction," *Computer Vision and Image Understanding*, vol. 199, p. 103024, 2020.

[114] A. Brock, J. Donahue, and K. Simonyan, "Large scale gan training for high fidelity natural image synthesis," *arXiv preprint arXiv:1809.11096*, 2018.

[115] C. Donahue, J. McAuley, and M. Puckette, "Adversarial audio synthesis," *arXiv preprint arXiv:1802.04208*, 2018.

[116] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1125–1134.

[117] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2223–2232.

[118] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *Communications of the ACM*, vol. 63, no. 11, pp. 139–144, 2020.

[119] C. Hardy, E. Le Merrer, and B. Sericola, "Md-gan: Multi-discriminator generative adversarial networks for distributed datasets," in *2019 IEEE international parallel and distributed processing symposium (IPDPS)*. IEEE, 2019, pp. 866–877.

[120] A. Savitzky and M. J. Golay, "Smoothing and differentiation of data by simplified least squares procedures." *Analytical chemistry*, vol. 36, no. 8, pp. 1627–1639, 1964.

[121] P. Tarnowski, M. Kołodziej, A. Majkowski, and R. J. Rak, "Emotion recognition using facial expressions," *Procedia Computer Science*, vol. 108, pp. 1175–1184, 2017.

[122] S. G. Koolagudi and K. S. Rao, "Emotion recognition from speech: a review," *International journal of speech technology*, vol. 15, pp. 99–117, 2012.

[123] F. Ahmed, A. H. Bari, and M. L. Gavrilova, "Emotion recognition from body movement," *IEEE Access*, vol. 8, pp. 11 761–11 781, 2019.