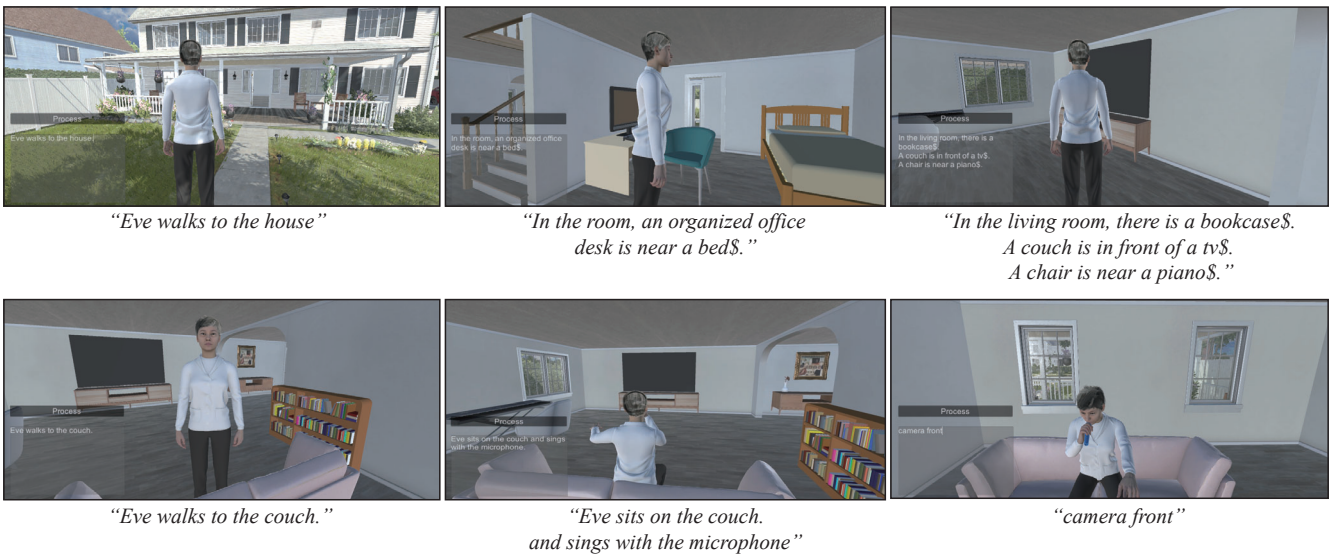
DOI: 10.1111/cgf.14415
Pacific Graphics 2021
E. Eisemann, K. Singh, and F.-L Zhang
(Guest Editors)

Volume 40 (2021), Number 7

Write-An-Animation: High-level Text-based Animation Editing with Character-Scene Interaction

Jia-Qi Zhang[1], Xiang Xu[1], Zhi-Meng Shen[1], Ze-Huan Huang[1], Yang Zhao[1], Yan-Pei Cao[2], Pengfei Wan[2], Miao Wang[1†]

[1]State Key Laboratory of Virtual Reality Technology and Systems, Beihang University, China
[2]Y-tech, Kuaishou Technology



"Eve walks to the house"

"In the room, an organized office desk is near a bed$."

"In the living room, there is a bookcase$. A couch is in front of a tv$. A chair is near a piano$."

"Eve walks to the couch."

"Eve sits on the couch. and sings with the microphone"

"camera front"

Figure 1: *Write-An-Animation interface and animation generation process. After initialization, the user can edit texts in the textbox to specify the virtual scene, character motion, and virtual camera, etc. The dollar sign is added to the end of the script to distinguish scene descriptions from character-motion commands. The 3D animation will be iteratively generated and updated according to the user's input.*

Abstract

3D animation production for storytelling requires essential manual processes of virtual scene composition, character creation, and motion editing, etc. Although professional artists can favorably create 3D animations using software, it remains a complex and challenging task for novice users to handle and learn such tools for content creation. In this paper, we present Write-An-Animation, a 3D animation system that allows novice users to create, edit, preview, and render animations, all through text editing. Based on the input texts describing virtual scenes and human motions in natural languages, our system first parses the texts as semantic scene graphs, then retrieves 3D object models for virtual scene composition and motion clips for character animation. Character motion is synthesized with the combination of generative locomotions using neural state machine as well as template action motions retrieved from the dataset. Moreover, to make the virtual scene layout compatible with character motion, we propose an iterative scene layout and character motion optimization algorithm that jointly considers character-object collision and interaction. We demonstrate the effectiveness of our system with customized texts and public film scripts. Experimental results indicate that our system can generate satisfactory animations from texts.

CCS Concepts

• ***Computing methodologies*** → ***Motion processing;***

1. Introduction

With the pervasive use of mobile phones and social networks, diverse video media such as vlogs, selfies and animated films

† Corresponding author: miaow@buaa.edu.cn

have now become increasingly popular. Compared with traditional texts or pictures, an increasing number of people prefer to watch and share animated films. However, the production of 3D animation or animated films is complicated, which not only requires a variety of professional software but also highly relies on user's skill and experience. With the development of deep learning technology in recent years, researchers have proposed various cost-effective content generation methods such as language-driven sketch colorization [ZMG*19], facial animation [AHK*02], video montage [WYH*19] and animation [HID*14]. Among them, text-driven animation has been a hot but challenging research topic.

Although several animation synthesis methods were proposed [AHKM20, GCO*21], to the best of our knowledge, no prior work has focused on the combination of virtual scene editing with character-motion synthesis. Some work was devoted to solving 3D indoor scene generation [CSM14b, CSM14a], while others aimed to improve virtual character motion realism [HKS17, SZKS19]. We would like to argue that, seperate generations of high-quality virtual scenes or character motions cannot guarantee the animation quality of their combinations. On the one hand, characters can interact with the virtual scene, on the other hand, scene layout can implicitly affect character's behavior like locomotion trajectory.

In this work, we present *Write-An-Animation*, a text-based animation editing system, where users can edit virtual scene layout, control character locomotion in the scene and specify the interaction with scene models, all through texts. Our system supports natural language texts, and allows users to edit texts for scene layout and character motion editing. The interaction interface of Write-An-Animation consists of a textbox and an execution button, which is simple yet effective. The user can iteratively edit the text in the textbox and update the animation result (see Figure 1). We clarify that our text-based animation editing is high-level and friendly to novice users, where users do not need to set any parameters for character pose, walking trajectory, etc. Nevertheless, users can futher fine-tune animation details with the help of commercial software, which is not the foucs of this paper. Our system is compatible to motion capture (mocap) data, as long as the Biovision hierarchical data (BVH) and corresponding action tags are indexed to our system. We demonstrate our system using customized texts and texts from public film scripts for single character animation in indoor scenes. Experimental results including ablation studies and subjective studies indicate that our system can produce satisfactory animation. To summarize, the major contributions of this paper are:

- An interactive, high-level text-based 3D animation editing system that drives human character animation in high-quality virtual scene.
- An algorithm for scene layout and interactive character motion optimization.
- A character-motion blending algorithm from generative locomotion and template action representations. It facilitates the joint optimization of scene layout and character motion and is compatible with new mocap data.

2. Related Work

Text-based Image and Video Synthesis. Recently, image and video applications with natural languages as inputs have attracted increasing attention, due to the advance of natural language processing. Zhu et al. [ZGE*07] proposed a text-to-picture synthesis system to augment the input text with retrieved multiple images. While their goal was to augment the communication, the generated results were separated images rather than a composed image for storytelling. Built upon neural networks, Fu et al. [ZMG*19] proposed a text-based sketch colorization system. With simple text instructions, the input sketch image was colorized step by step. Recently, Wang et al. [WYH*19] proposed a text-based video montage system to retrieve video clips from datasets and compose videos that followed cinematographic guidelines. Text-based editing of talking-head video [FTZ*19] was proposed to produce high-quality portrait videos, especially for interview videos. The work indicates that utilizing natural languages as inputs is a promising direction for image and video synthesis.

Text-based Scene Generation. 3D scene generation methods from various input types such as RGB-D images [CLW*14], sketches [XCF*13], and languages [MPF*18] have been proposed. Among them, text-based scene synthesis has been a hot topic. Chang et al. [CSM14b, CSM14a] transformed texts into graphs, and then extracted the spatial knowledge of objects, such as occurrence, hierarchy, surface and relative position, to generate a reasonable layout. Chang et al. [CMS*15] further extended this work by introducing a dataset with natural language descriptions and learning mappings from data to texts. Rui et al. [MPF*18] described objects in scenes as semantic scene graphs to retrieve scenes in the dataset that best matched the text. Although this approach depends on semantic scene graph construction for the dataset, it can ensure reasonable scene layout, which is convenient to add or delete models in scenes. For more work on scene synthesis, please refer to the surveys [ZZLH19, WLLZ20].

Text-based Animation Creation. Text-based animation offers dynamic illustration of text-form storytellings. Irene et al. [AHK*02] proposed a rule-based method to convert text into an animation sequence of facial expressions, which is synchronized with speech. Hayashi et al. [HID*14] proposed a text-to-CG animation generation method based on TV program Making Language (TVML), but the generated characters only perform pre-defined sets of actions at fixed positions. Abrami et al. [AHKM20] proposed a text-to-scene method that allows users to interactively build a virtual scene during immersive exploration via VR headset and controllers. Ghosh et al. [GCO*21] proposed a text-driven skeleton animation method based on deep neural networks, which can handle short sentences composed by a single action or long sentences composed by a combination of multiple consecutive actions. However, their method may fail to generate continuous actions that are not adequately distributed in the dataset.

Generation of 3D Character Motion. Mocap solutions are usually used to customize realistic character animation in industry. It is laborious and time-consuming to deploy mocap apparatuses and conduct the experiments. It is always expected to simplify or even automate the process [RLA*18, LZL20]. Dietmar et al. [SPH11] simultaneously generated speech and CG animation based on the Hidden Markov Model. Kapadia et al. [KFS*16] proposed an optimization algorithm to automatically fill in missing narration details for given key plot points in a story to synthesize a complete 3D animation.

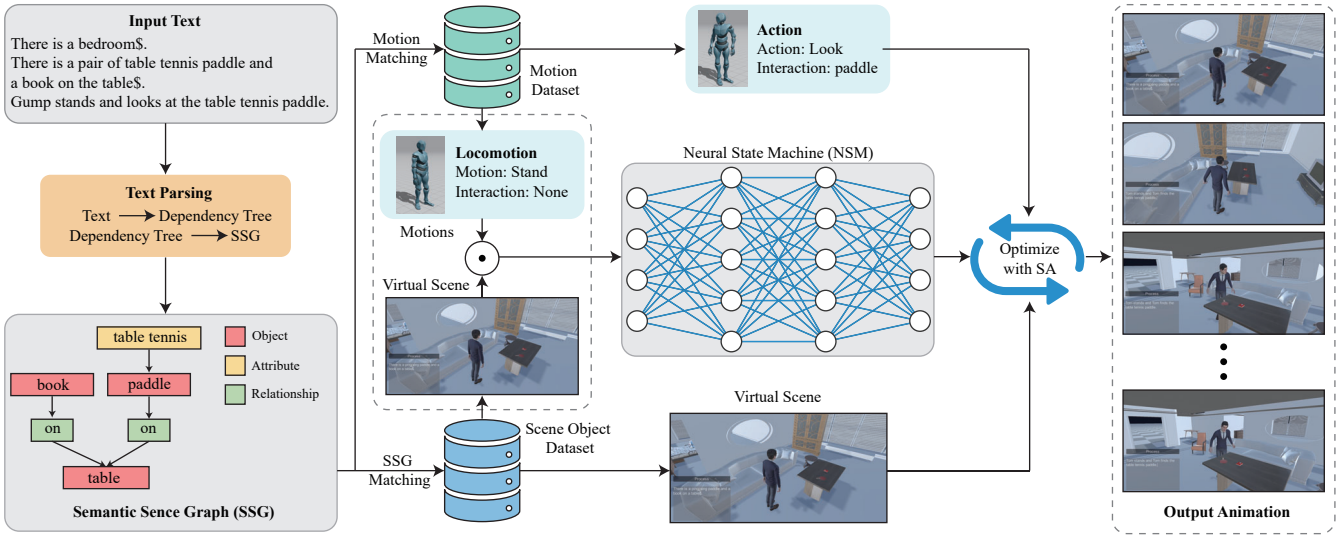


Figure 2: *The pipeline of the our system. The input texts are first parsed into semantic scene graphs by CoreNLP [MSB*14] and LDS [MPF*18]. From the semantic scene graphs, our system extracts objects and template action motions from the scene dataset and motion dataset respectively, and generates locomotion using NSM [SZKS19]. Finally, the virtual scene, locomotion and template action motions are jointly optimized to create the final animation.*

Recently, deep neural networks [CWY*20,HYNP20,ZWP*21] and reinforcement learning methods [HHC*19, PRL*19] were used to synthesize character animation. Fragkiadaki et al. [FLFM15] also proposed LSTMs for mocap generation. However, the training for sequence modelling using LSTMs could be challenging [BKK18]. Holden et al. [HKS17] proposed the phase-functioned neural networks (PFNN) architecture, which handles sharp movements of animation better than sequential convolution. Compared with the single fixed cyclic function of PFNN, Sebastian et al. [SZKS19] proposed the neural state machine (NSM) that allows the network to automatically learn phase functions for multiple actions. After that, Sebastian et al. [SZKZ20] proposed the local motion phases (LMP) that extends network capabilities to the generation of human-to-human interaction. In this paper, we employ NSM as a character lomotion generation method, and further formulate an optimization model of joining scene layout and character motion.

3. The *Write-An-Animation* System

3.1. Overview

Write-An-Animation provides a text-to-animation solution that allows users to edit 3D animation by simply writing and editing texts. Our design objectives are multi-fold: first, we aim to generate virtual scenes close to the daily life where objects can be interacted with; second, we expect the character to be aware of the scene context and adaptively locomote in the scene; third, diverse character motions should be supported, such as waving arm angrily, playing the piano, etc. To accomplish the objectives, we build our system upon Language-Driven Synthesis [MPF*18] and NSM-based character locomotion [SZKS19] to jointly optimize the scene layout and character motion, and further generate vivid 3D animation by editing texts.

The text-based virtual scene is synthesized by LDS, while character motions are generated by the NSM. However, the NSM has two shortcomings: First of all, the performance of NSM is sensitive to the data, which will cause the network to generate unreasonable motions if the scale of test models changes by more than 10%. For example, when the height of the testing chair model is different from the training data, the character usually first sits on the border of the chair, then gradually moves to the center of the chair. Secondly, in order to control the character to perform 7 different actions, the NSM captured 94 minutes of motion and divided it into 452 motion clips. When the number of actions rose to several hundred, the labeled data and training time required by the NSM network greatly increased. In order to avoid these issues, we propose to optimize the layout of virtual scene objects based on generated motion trajectory, and incorporate template actions with NSM-based character locomotion. Template actions are easily collected from off-the-shelf mocap data in advance, which greatly reduces the work of data collection and labeling.

3.2. System Pipeline

Our system includes major components of dataset preparation, text parsing, initial generation of scene and character motion, and scene layout and character motion optimization. The pipeline is shown in Figure 2.

Dataset Preparation. We build datasets for 3D indoor scene object retrieval and template character action motion retrieval respectively. Our 3D indoor scene dataset includes SceneNN [HPN*16], SceneSynth [FRS*12] and 3D-FRONT [FCG*20]. All models in the scene dataset are annotated with the form used in ShapeNetSem [CFG*15] for object's category, attributes and relationship with other objects. Each scene is saved as a Seman-

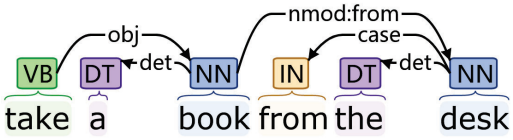


Figure 3: *An example of dependency parsing by using CoreNLP. Each node is tagged to perform part-of-speech, VB=verb, NN=noun, IN=preposition, DT=determiner. Edges represent dependencies, such as obj=object and nmod=nominal modifier.*

tic Scene Graph (SSG), which records the model position, attributes and relationship. The template action dataset contains NSM [SZKS19] motion dataset, Mixamo 3D character motion [Inc21] and AMASS [MGT*19] dataset. In total, we collected 10 characters, 149 template action motions (101 AMASS actions and 48 mixamo actions) and 1883 indoor objects in 133 scenes.

Text Parsing. The part-of-speech (POS) of input text is first tagged by CoreNLP [MSB*14] and transformed into a dependency tree, including all entities and their corresponding attributes and relationships which correspond to human actions, as shown in Figure 3. The entity represents characters, the attribute corresponds to an action, and relationships record interactive models of the action. Then, the dependency tree is converted into an SSG, which saves descriptions of each object in the input texts and relationships between objects. In order to facilitate processing, we assume the user appends a "$" sign at the end of a text to indicate a scene description sentence. Moreover, our system also supports verbal commands. We provide a set of weather, virtual camera and motion control commands to facilitate animation creation. For example, command "camera left" sets the virtual camera to the left of the character. The user can iteratively write and edit texts to update the results.

Initial Generation of Scene and Character Motion. We use the subgraph of the parsed text to match SSGs in the scene dataset for scene object retrieval. We represent character motion as locomotion and action motion (see Section 4.2), where the locomotion is generated by a pre-trained NSM model and serves to perform basic motions and determine character positioning in the scene, for instance, to walk, stand, run, etc. Action motions are mocap clips retrieved from dataset, and can be performed during locomotion. We fuse locomotion and action motion to produce diverse combinations of character motions.

Scene Layout and Character Motion Optimization. In order to ensure compatibility between scene layout and character motion, especially for character-object-interaction, we propose a method that takes the locomotion, action motion and scene layout as inputs, and iteratively optimizes the scene layout. In each iteration, our method tries to avoid unnecessary collisions between the character body and virtual objects in the scene, meanwhile ensuring body parts (i.e., finger tips) are close enough to the points of objects (i.e., keyboard). We formulate cost functions and employ the simulated annealing algorithm for iterative optimization of scene layout and character animation generation.

Algorithm 1: Object Matching for Scene Composition.

Input: Query SSG G_q, Dataset SSG $\mathbb{G}_{db}$
Output: Match map M between G_q and $\mathbb{G}_{db}$, Match Score S
$S \leftarrow 0$
for *relation node u in* G_q **do**
  for *relation node v in* $\mathbb{G}_{db}$ **do**
    $G_q^{'} = \{(u.active, u), (u, u.anchor)\}$
    $G_{db}^{'} = \{(v.active, v), (v, v.anchor)\}$
    if $G_q^{'} \subseteq G_{db}^{'}$ **then**
      $m(u,v) \rightarrow M$
      $m(u.active, v.active) \rightarrow M$
      $m(u.anchor, v.anchor) \rightarrow M$
      $S_{reward} = CalculateMatchScore(M)$
      $S \leftarrow S + S_{reward}$
for *object node u in* G_q **do**
  for *object node v in* $\mathbb{G}_{db}$ **do**
    if $u \notin M$ && $v \notin M$ && $v = u$ **then**
      $m(u,v) \rightarrow M$
      $S_{reward} = CalculateMatchScore(M)$
      $S \leftarrow S + S_{reward}$
return M, S

4. Technical Details

In this section, we describe more technical details of major components in our system.

4.1. Virtual Scene Composition

Our virtual scene composition method builds upon and extends the original LDS [MPF*18] method. We first parse the texts as a low-level dependency tree using the CoreNLP framework, then convert the dependency tree into a semantic scene graph, similar to the LDS method. Finally, we match the semantic scene graph with the graphs saved in the scene dataset, and extract corresponding objects from multiple matched candidates to compose a new scene. Specifically, when matching semantic scene graphs, LDS matches individual objects as the first step, then matches the spatial relationships from the top returned objects. Despite that the LDS method can find a suitable solution through a huge scene dataset and optimize positions of the objects by commands, semantic relationships could be lost, as shown in the first row of Figure 4. By contrast, our method injects the relationship between objects to the matching rule so that more accurate results can be found.

Each SSG contains object nodes and relation nodes, where the relation node stores information of the spatial relationship between two object nodes. Let (u,v) be the relation node between object node u and v. For example, if a scene contains a cup on a table, the preposition "on" indicates the spatial relation between the cup and the table, where $u.active$ denotes all incoming nodes, and $u.anchor$ denotes all outgoing nodes.

Let G_q denote the query SSG and $\mathbb{G}_{db}$ indicate a dataset of SSGs. Our method matches the relation nodes of query SSG G_q with the

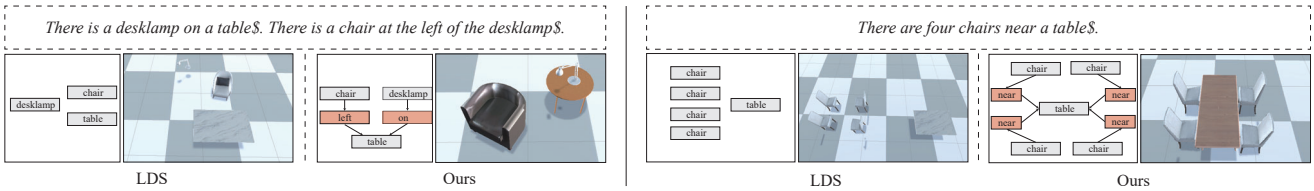


Figure 4: *SSG matching comparison between LDS [MPF*18] and our method. Top: results from LDS. Bottom: our results.*

dataset SSGs $\mathbb{G}_{db}$, and puts each matched subgraph into a matching map M that stores the location of the scene SSG corresponding to each text SSG node. If no matched subgraphs are found, our method directly matches individual object nodes of G_q in $\mathbb{G}_{db}$, and saves successfully matched objects into M. The pseudocode is provided in Algorithm 1. Scene composition results of our method are shown in the second row of Figure 4. LDS directly matches relationships from top returned objects. However, the relationships between the table and desklamp, and the chair and desklamp are not included. As the relationship between the table and chairs is ignored, objects are separately placed in the scene. Scene composition results of our own method correct objects and relationships.

4.2. Motion Synthesis

After the construction of SSG, our system extracts all action types and their corresponding interactive objects in parallel to scene composition. We first filter out locomotion types, and then retrieve action motion files (BVH format) in the motion dataset. Let M_l and M_a denote the locomotion and action motion, and the virtual scene constructed by SSG is denoted as E that encodes the spatial position and three-dimensional scale of objects. We feed M_l and E into a pre-trained NSM network to synthesize a preliminary motion $M_p = NSM(M_l, E)$.

Next, we blend the retrieved action motion M_a and preliminary motion M_p to generate a blended motion M_o that represents the motion trajectory of body joints in the entire action. Specifically, we rotate locomotion and action motion skeletons to the standard T-pose respectively, and then get the blended motion $M_o^i = \hat{M}_b^i \times T_o^i$, where i is a joint, and $\hat{M}_b^i$ can be either locomotion T-pose or action motion T-pose. T_o represents the global rotation matrix from T-pose to M_o. In our experiment, we blend the upper body (joints above the hip) of template actions with the lower body (joints below the hip) of generative locomotions. Finally, the inverse kinematics (IK) is used to ensure that the combined action M_o conforms to laws of kinematics.

4.3. Scene Layout and Character Motion Optimization

Although a plausible scene can be generated using the method in Section 4.1, there can be artifacts during character-object interaction, because the scene and character actions are computed separately. For instance, there can be a spatial gap between specific actions and interactive objects at contact points; the generated actions cannot avoid obstacles and are distorted when interacting with distant targets, etc. To make the scene and character motion compatible, we design cost functions and perform joint optimization based on simulated annealing.

Algorithm 2: Scene Layout and Character Motion Optimization.

Input: Virtual Scene Layout E, Blended Motion M_o
Output: Optimized Scene $\hat{E}$, Optimized Motion $\hat{M}$
$R_s \leftarrow SearchRoute(E, M_o)$
$R_g, M_g, E_g \leftarrow NSMGenerate(R_s, E, M_o)$
$S_{old} \leftarrow GetRunStatus(R_g, M_g, E_g)$
$OldError \leftarrow CalculateError()$
for *each step* **do**
  $S_{cur} \leftarrow AddDisturbance(S_{old})$
  $R_g, M_g, E_g \leftarrow NSMGenerate(S_{cur})$
  $S_{cur} \leftarrow GetRunStatus(R_g, M_g, E_g)$
  $error \leftarrow CalculateError()$
  $error \leftarrow AcceptanceProbability(error)$
  $error, OldError, S_{old} \leftarrow$
   $UpdateStatus(S_{cur}, S_{old}, error, OldError)$

Collision Cost. During locomotion, the character should avoid unnecessary collision with scene objects. To this end, we design collision costs to compute the collision level of the character with the scene. Instead of using only one axis-aligned bounding box to represent an object, we use multiple small fixed-size bounding boxes to represent an object, which is approximate to the object's shape. The collision cost is computed as the number of collisions between character body and the bounding boxes during movement:

$$C_D = \frac{1}{|O|} \sum_{o \in O} \frac{n_o^c}{n_o^a}, \tag{1}$$

where O is the set of objects which collided with the character, n_o^c is the number of collided bounding boxes of object o, n_o^a is the total number of bounding boxes of object o.

Contact Cost. There is usually a large gap between the locomotion generated by the NSM and the object model. We define contact cost to penalize distances between action motions and interactive objects. For each interactive object, we follow the NSM to obtain contact points, including the hip, left/right middle fingers, and left/right wrists. In each iteration of the optimization process, we compute the average position of the skeleton joint in the entire cycle of the template action corresponding to a contact point, and

Figure 5: *Iterative creation of animation with customized text.*

compute the average distance for all pairs of skeleton joints and contact points:

$$C_T = \frac{1}{|P|} \sum_{p \in P} \|\overline{j(p)} - p\|_2, \tag{2}$$

where P is the set of contact points of an object, $\overline{j(p)}$ is the average position of a skeleton joint corresponding to p in an action motion.

Optimization. Our goal is to minimize the overall cost function:

$$C = \lambda_D C_D + \lambda_T C_T, \tag{3}$$

where λ_D and λ_T are constant parameters. We use the simulated annealing algorithm to optimize the spatial position and three-dimensional scale of objects E and the motion trajectory of all joints M_o in Section 4.2.

Algorithm 2 shows the pseudocode of the optimization algo-

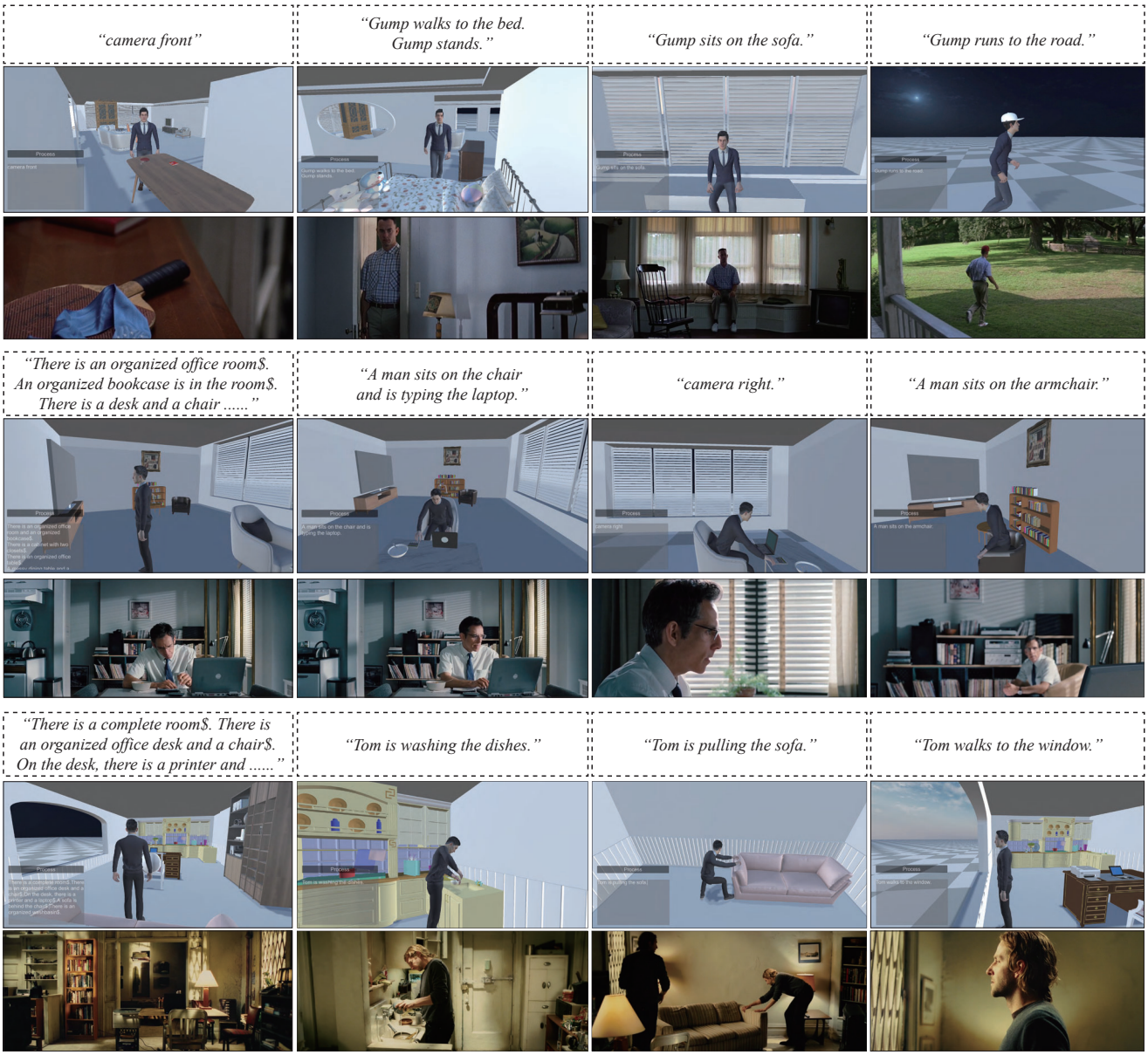


Figure 6: *Animation results from film scripts. In each example, the first row are the input texts, the second row shows results generated by our system, and the third row shows corresponding frames of the* The Secret Life of Walter Mitty*,* Forrest Gump*, and* Limitless*.*

rithm. In the optimization, the vertical positions of objects are constrained to be fixed, while the heights are allowed to be modified by changing the vertical scale of objects. First, we search a feasible locomotion route R_s from the character's position to the interactive object using A* algorithm, and then feed the searched route R_s, virtual environment E and blended motion M_o into the NSM to compute an initial solution $S_{old} = \{R_g, M_g, E_g\}$, where R_g, M_g and E_g are initial locomotion route, blended motion, and virtual scene layout respectively. In each iteration, we update the current scene layout E_g according to strategies introduced below to get a new scene E_d. The current scene E_d and action M_g are fed into the NSM again to get a new solution $S_{cur} = \{R_g, M_g, E_g\}$. After optimization, the new scene $\hat{E}$ and character motion $\hat{M}$ are obtained.

Our algorithm accepts or rejects the solution based on Metropolis-Hastings acceptance rule. As the distance between the character and the interactive object is closer, the character is more likely to collide with and pass through the object model during movement. Therefore, in order to ensure that the character touches interactive objects, we set λ_D to 1.0 and λ_T to 10.0 in our experiments. The initial value of temperature is set as 1, and it decreases by 0.002 in each iteration until it reaches zero. The maximum number of iterations is set to 100.

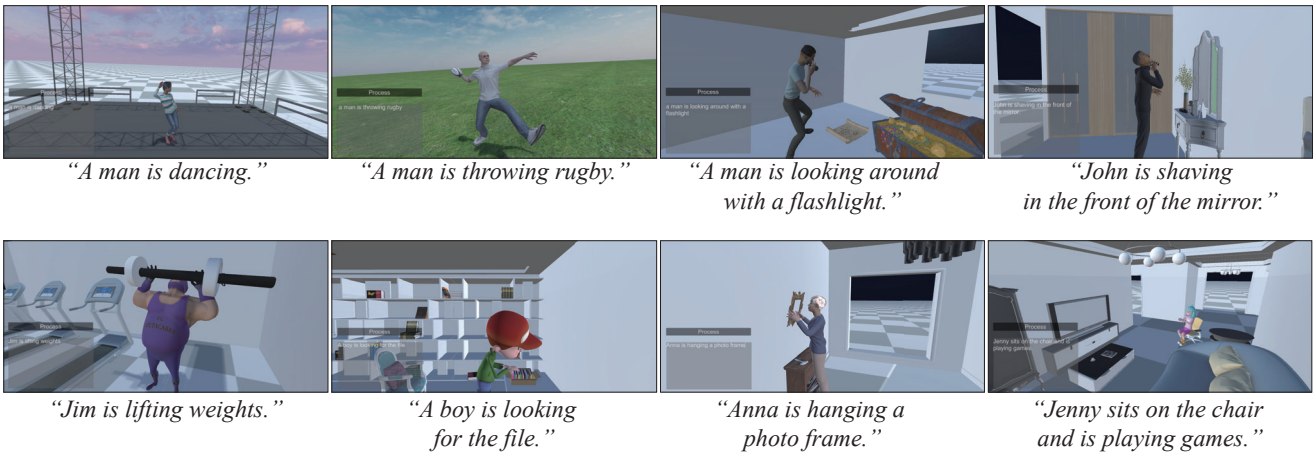


Figure 7: *Actions in different scenes with customized text. With the input scripts, our method can generate animations in daily scenarios.*

We use the following strategy to produce change of status in each iteration. Spatial translations and scales are used to change the contact positions of an object and its adjacents. By adjusting the scale of related objects together, our method can ensure contextual semantics between interactive objects and adjacent objects. We sample positions $(x+\Delta x, z+\Delta z)$ and scales $(s+\Delta s)$ using truncated normal distribution with boundary set to $(-3\sigma, 3\sigma)$ as follows:

$$X \sim \mathcal{N}(\mu, \sigma^2, -3\sigma, 3\sigma), \tag{4}$$

where $\mu = 0$ and $\sigma = 0.02$.

5. Results and Evaluation

In this section, we show and discuss visual results generated by Write-An-Animation. To adequately evaluate the effectiveness of our method, we not only use customized texts, but also run expriments using public film scripts. We invite users to watch animated video clips and evaluate the results of our method from different aspects.

5.1. Implementation Details

In order to allow characters to perform actions in the scene, collision bounding boxes of all scene object models are computed in advance. Meanwhile, contact points of interactive objects are marked. Unlike the original NSM method, we use five contact points corresponding to character model joints, including left and right wrist joints, middle finger joints and a hip joint. The location of contact points are used to determine how the character interacts with the object. In addition, character models used in our experiments are all from Mixamo [Inc21], and the user needs to select an appropriate character model for a virtual scene before the initialization. We use Unity3D as the main development tool and use Blender to edit models and mocap data. We run all experiments on a Windows PC with Intel Core i7-9700k CPU, 32 GB RAM with Intel(R) UHD Graphics 630.

During the test, our system generates an average of 3 minutes animation for each customized text, at a 20 to 30 fps framerate. The average processing time for each input text is about 1 second. The runtime mainly depends on the number of actions and interactive objects in the text.

5.2. Results

We test our system with texts describing *single-character* daily indoor activities and text-form scripts of classic film clips. Figure 5 shows representative frames of five iteratively created animations. Each character is controlled to locomote in an indoor scene and interact with objects. In order to test if our system can work with film scripts and generate correct animations compared to film clips, we search public scripts from the Internet, and add objects related to the script to the scene dataset. We then write and edit text-form scripts to build the scene and generate the animation. Figure 6 shows an example of the generated animation for the film clip, including *Forrest Gump*, *Secret Life of Walter Mitty* and *Limitless*. The results indicate that although the generated virtual scene and character motions are not exactly the same as those in the film, the semantics of the animation is similar to the film clip. In total, we generate 5 animation clips for customized texts and 3 animations for film clip scripts, all driven by text editing via Write-An-Animation.

It is worth noting that NSM should be retrained when dealing with a new model with different skeletons. However, in our system, different character body skeletons are first mapped to Unity's avatar skeleton through retargeting. Then the fused motion is mapped to the corresponding skeleton of the avatar. Note that during the mapping process, our method takes into account the geometry differences of skeletons. Figure 7 shows the thumbnails of actions in 8 different scenes with different characters.

5.3. Ablation Studies

We conduct ablation studies to evaluate the key components of our system. In Section 4.1 and Figure 4, we have demonstrated the effectiveness of the refined SSG matching method that can reflect the relationships between scene objects. Moreover, we also perform

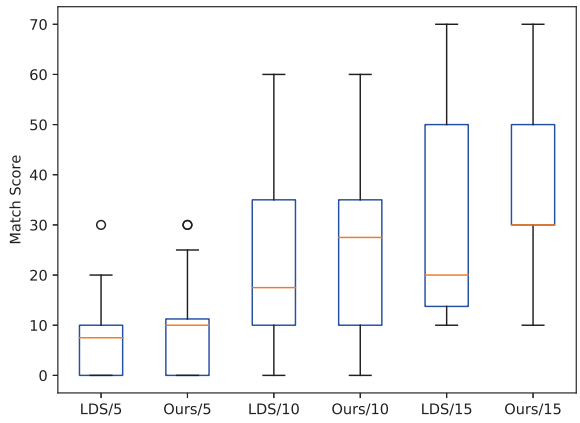


Figure 8: *Box plots of SSG matching scores between LDS [MPF*18] and our method for 5, 10 and 15 query SSGs respectively.*

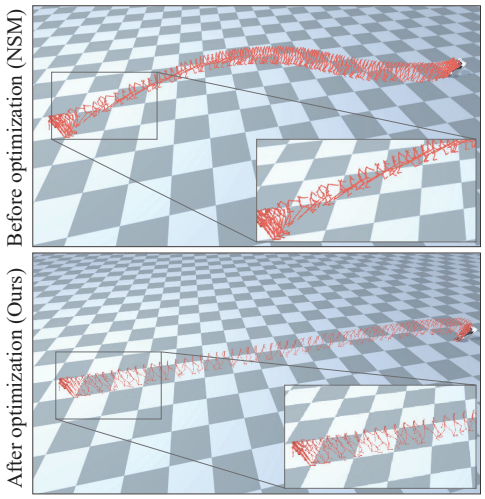


Figure 9: *Effect of the joint optimization on motion distortion avoidance during distant locomotion. Sampled character motions are illustrated during the locomotion to a chair.*

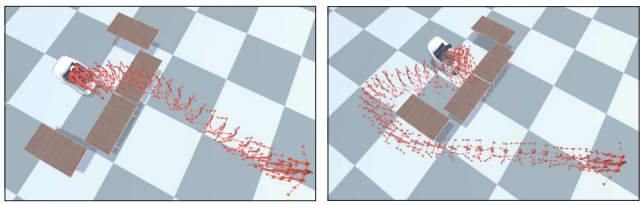


Figure 10: *Effect of the joint optimization on obstacle collision avoidance during locomotion. Sampled character motions are illustrated during the locomotion to an occluded chair.*

quantitative comparisons. Specifically, we randomly select 5, 10 and 15 SSGs out of 20 scripts to test the matching results of the LDS and our method with the matching scores plotted in Figure 8. We find that our method can obtain better matching scores than LDS, and this improvement becomes more obvious as the number of SSGs increases.

The joint scene layout and character motion optimization play a key role in producing correct animations. When the distance between the character and the interactive object (e.g., a chair) is very large, there the character motion can be distorted during the character's locomotion towards the object. Figure 9 shows character locomotion trajectories before and after optimization. Without the optimization, the locomotion generated by NSM can collide with objects in the scene. On the contrary, in our result, the character locomotion is aware of the objects in the scene, and can hence avoid unnecessary collisions, see Figure 10. Figure 11 shows the character-object interaction results before and after the joint scene layout and character motion optimization. In the top row, there are gaps between the character's hand and objects (i.e., the piano and the laptop). With the optimization, the character-object interactions in the bottom row are correct. We illustrate the cost changes during optimization of the character typing in Figure 12. The red curve represents the collision between characters and objects, and the green curve represents the distance between the character joint and contact points of interactive objects. The blue curve indicates the overall loss.

5.4. User Study

Subjective experience of animation results are important for evaluation. We conducted a web-based user study to evaluate our results.

We invited 30 participants (16 males and 14 females) to evaluate our system generated results. Their average age is 23 years old, with the youngest being 20 years old and the oldest being 28 years old. Animations corresponding to 5 customized texts and 3 film clip scripts were generated interactively through our system, and saved as videos. The film clips are from *The Secret Life of Walter Mitty*, *Forrest Gump*, and *Limitless*.

In our web-based user study, each text and the corresponding animation video generated by our system are displayed on web pages. Additionally, the original film clip for film script are shown. All participants were asked to rate the 8 test samples using 11-point Likert scales (0-worst, 10-best) from the following aspects:

- *Semantics*: The semantic matching between input text and generated animation.
- *Diversity*: The diversity of generated scene models.
- *Reality*: The similarity between the generated character motion and real human motion.
- *Fluency*: The smoothness of motion in generated animation.
- *Accuracy*: The accuracy of motion details in generated animation.

Throughout the evaluation process, users can fast forward or slow down the video playback at will, and can change the scores of videos that have been evaluated at any time. The entire evaluation process takes an average of 30 minutes. After the evaluation was completed, the participants were thanked and payed.

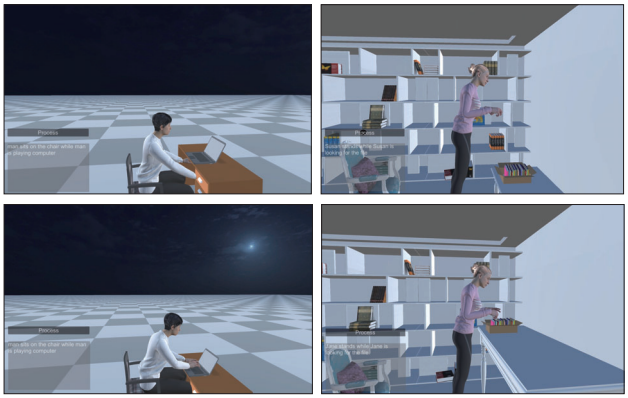


Figure 11: *Character-object interaction results before and after the joint optimization. The first row presents the results before optimization, and the second row shows the results after optimization.*

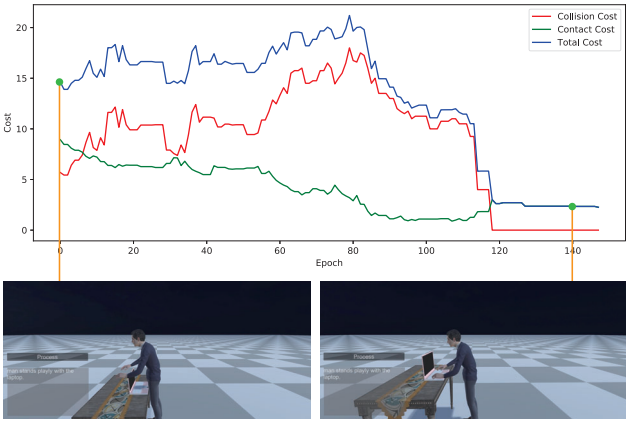


Figure 12: *Cost function values for character interacting with a laptop during optimization. Two intermediate results are visualized on the bottom.*

We compute the avarage ratings for customized texts, film scripts, before and after optimization, respectively. Figure 13 (a) and (b) show the visualizations of user study results, which are conducted in 5 customized texts and 3 film scripts. Participants generally thought that our system generated correct animations for the corresponding text, and created diverse scene objects for both customized texts (Semantics: 8.05, Diversity: 7.79) and film scripts (Semantics: 8.12, Diversity: 7.84). Although participants only on average rated 7.45 points for Reality and 7.34 points for Fluency of generated animations, the results are acceptable to the participants. Figure 13 (c) and (d) show the participant's evaluation of the eight actions before and after optimization (see Figure 7), which indicates a significant user rating improvement after optimization. In addition, participants generally agreed that the details of animations can be further improved.

Comparing animations generated from customized texts and film scripts, participants on average rated customized text with 7.44 points for Fluency and 7.29 points for Accuracy. For animations

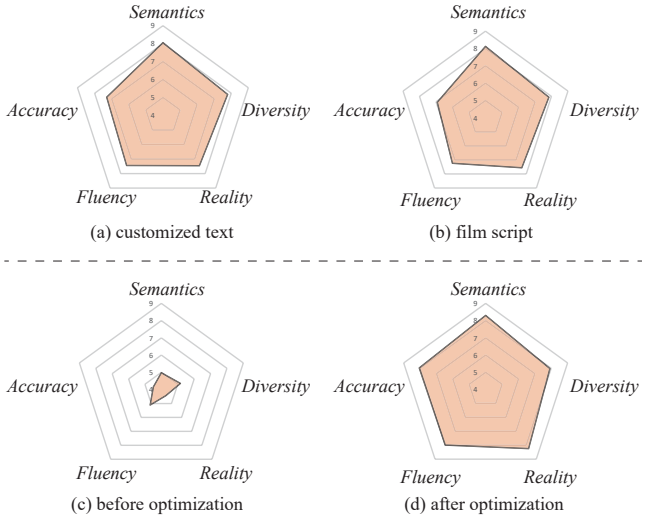


Figure 13: *Visualization of the user study result. (a) Avarage ratings of animations generated by customized text; (b) Average ratings of animations generated by film scripts; (c) Avarage ratings of animations before optimization; (d) Average ratings of animations after optimization.*

generated from film scripts, participants on average rated 7.24 points for Fluency and 6.92 points for Accuracy. Although the average rating of Fluency and Accuracy are lower than other metrics, participants still rated more than 6.9 points and were generally satisfied with the animations.

6. Conclusions, Limitations and Future Work

In this paper, we present a text-to-animation framework *Write-An-Animation*. With Write-An-Animation, the user can iteratively write and edit texts to animate a human-like character in an indoor virtual scene. We employ a refined scene object matching method to constrain spatial relationships during scene object retrieval and scene composition. By dividing character motion as generative locomotion and template action, our method can work with new template mocap data. To make the scene layout compatible with the character motion, we propose a joint optimization method of the scene and character motion and generate optimized animation results. Visual comparisons, ablation studies and user study results varified the effectiveness of our method.

Limitations and Future work. While Write-An-Animation is a novel tool for novice users to produce plausible animations, we acknowledge that it is a prototype system and has a few limitations for future reseach.

First, the quality of animation results are highly dependent on the datasets. In our system, we build a scene object dataset and a template action motion dataset for scene object and motion retrieval respectively. If the object or motion corresponding to the text does not exist in the datasets, our method can fail to generate correct scene or animation. We have included several recognized datasets

(a) Locomotion collision (b) Action collision

Figure 14: *Limitation: character-object collision. (a) The character collide with the chair during locomotion. (b) The template action of the character collide with the chair.*

such as 3D-FRONT [FCG*20] and AMASS [MGT*19], and can incorporate with other datasets in the future.

Second, our system at current only supports single-character animation and several character-object interaction types. Multiple character interaction and object-object interaction are not included. Besides, for each character-object interaction, the contact points of the interactive object and the corresponding skeleton joints of the character should be annotated in advance. For example, when the character is expected to pick up a cup, the contact points on the cup need to be annotated first. Automatic contact point annotation or inference for character-object interaction is a promising research direction.

Third, currently in our system, once composited, objects are fixed in the scene. This can cause inevitable collisions between the character and objects in some cases. For example, if the objective animation is "Emily sits on the chair and plays keyboard", the optimized scene layout guarentees the correct spatial relationship between the finger tips and the keyboard, and between the hip and the chair. However, if the character walks to the chair and sits down, collision will happen. Moreover, because our method does not manipulate template actions, the template action can conflict with objects. Figure 14 shows examples of typical collision artifacts. Such artifacts can be solved in the future by introducing movable objects. We regard it as a future work.

Fourth, as a text-based tool mainly designed for novice users, Write-An-Animation avoids professional operations that are difficult to learn and use. In our implementation, a dollar sign is appended to the end of the script to distinguish a scene description from a character-motion command. For example, the script "Put a vase and a teapot on the table$" is executed by the system to synthesize the scene instead of animating the character. However, precise controls of scene layout and character motion can be lost. Combinations of text editing and other editing operations from professional software for animation generation are expected.

Last but not least, our system processes each text with a multi-stage pipeline. Nevertheless, we expect end-to-end generations of character animations in the virtual scene. Text parsing can be proceeded by transformers [VSP*17] in natural language processing, scene layout can be generated from graph neural networks, and action generation can be learned from the local motion phases network [SZKZ20]. How to efficiently fuse the networks and modules is a promising research direction.

Acknowledgments

The authors wish to thank the anonymous reviewers for their helpful advices. This work was supported by the National Natural Science Foundation of China (Project Number: 61902012 and 61932003).

References

[AHK*02] ALBRECHT I., HABER J., KAHLER K., SCHRODER M., SEIDEL H.-P.: " may i talk to you?:-)"-facial animation from text. In *10th Pacific Conference on Computer Graphics and Applications, 2002. Proceedings.* (2002), IEEE, pp. 77–86.

[AHKM20] ABRAMI G., HENLEIN A., KETT A., MEHLER A.: Text2scenevr: Generating hypertexts with vannotator as a pre-processing step for text2scene systems. In *Proceedings of the 31st ACM Conference on Hypertext and Social Media* (New York, NY, USA, 2020), HT '20, Association for Computing Machinery, p. 177–186.

[BKK18] BAI S., KOLTER J. Z., KOLTUN V.: An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271* (2018).

[CFG*15] CHANG A. X., FUNKHOUSER T., GUIBAS L., HANRAHAN P., HUANG Q., LI Z., SAVARESE S., SAVVA M., SONG S., SU H., XIAO J., YI L., YU F.: *ShapeNet: An Information-Rich 3D Model Repository*. Tech. Rep. arXiv:1512.03012 [cs.GR], Stanford University, Princeton University, Toyota Technological Institute at Chicago, 2015.

[CLW*14] CHEN K., LAI Y.-K., WU Y.-X., MARTIN R., HU S.-M.: Automatic semantic modeling of indoor scenes from low-quality rgb-d data using contextual information. *ACM Trans. Graph. 33*, 6 (Nov. 2014).

[CMS*15] CHANG A., MONROE W., SAVVA M., POTTS C., MANNING C. D.: Text to 3d scene generation with rich lexical grounding. *arXiv preprint arXiv:1505.06289* (2015).

[CSM14a] CHANG A., SAVVA M., MANNING C. D.: Interactive learning of spatial knowledge for text to 3d scene generation. In *Proceedings of the Workshop on Interactive Language Learning, Visualization, and Interfaces* (2014), pp. 14–21.

[CSM14b] CHANG A., SAVVA M., MANNING C. D.: Learning spatial knowledge for text to 3d scene generation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (2014), pp. 2028–2038.

[CWY*20] CHEN W., WANG H., YUAN Y., SHAO T., ZHOU K.: Dynamic future net: Diversified human motion generation. In *Proceedings of the 28th ACM International Conference on Multimedia* (New York, NY, USA, 2020), MM '20, Association for Computing Machinery, p. 2131–2139.

[FCG*20] FU H., CAI B., GAO L., ZHANG L., LI C., ZENG Q., SUN C., FEI Y., ZHENG Y., LI Y., LIU Y., LIU P., MA L., WENG L., HU X., MA X., QIAN Q., JIA R., ZHAO B., ZHANG H.: 3d-front: 3d furnished rooms with layouts and semantics. *arXiv preprint arXiv:2011.09127* (2020).

[FLFM15] FRAGKIADAKI K., LEVINE S., FELSEN P., MALIK J.: Recurrent network models for human dynamics. In *Proceedings of the IEEE International Conference on Computer Vision* (2015), pp. 4346–4354.

[FRS*12] FISHER M., RITCHIE D., SAVVA M., FUNKHOUSER T., HANRAHAN P.: Example-based synthesis of 3d object arrangements. In *ACM SIGGRAPH Asia 2012 papers* (2012), SIGGRAPH Asia '12.

[FTZ*19] FRIED O., TEWARI A., ZOLLH"OFER M., FINKELSTEIN A., SHECHTMAN E., GOLDMAN D. B., GENOVA K., JIN Z., THEOBALT C., AGRAWALA M.: Text-based editing of talking-head video. *ACM Trans. Graph. 38*, 4 (July 2019), 68:1–68:14.

[GCO*21] GHOSH A., CHEEMA N., OGUZ C., THEOBALT C., SLUSALLEK P.: Synthesis of compositional animations from textual descriptions. *arXiv preprint arXiv:2103.14675* (2021).

[HHC*19] HONG S., HAN D., CHO K., SHIN J. S., NOH J.: Physics-based full-body soccer motion control for dribbling and shooting. *ACM Trans. Graph. 38*, 4 (July 2019).

[HID*14] HAYASHI M., INOUE S., DOUKE M., HAMAGUCHI N., KANEKO H., BACHELDER S., NAKAJIMA M.: T2v: New technology of converting text to cg animation. *ITE Transactions on Media Technology and Applications 2*, 1 (2014), 74–81.

[HKS17] HOLDEN D., KOMURA T., SAITO J.: Phase-functioned neural networks for character control. *ACM Trans. Graph. 36*, 4 (July 2017).

[HPN*16] HUA B.-S., PHAM Q.-H., NGUYEN D. T., TRAN M.-K., YU L.-F., YEUNG S.-K.: Scenenn: A scene meshes dataset with annotations. In *2016 Fourth International Conference on 3D Vision (3DV)* (2016), IEEE, pp. 92–101.

[HYNP20] HARVEY F. G., YURICK M., NOWROUZEZAHRAI D., PAL C.: Robust motion in-betweening. *ACM Trans. Graph. 39*, 4 (July 2020).

[Inc21] INC A. S.: Mixamo, 2021. URL: https://www.mixamo.com.

[KFS*16] KAPADIA M., FREY S., SHOULSON A., SUMNER R. W., GROSS M. H.: Canvas: computer-assisted narrative animation synthesis. In *Symposium on Computer Animation* (2016), pp. 199–209.

[LZL20] LI M., ZHOU Z., LIU X.: 3d hypothesis clustering for cross-view matching in multi-person motion capture. *Computational Visual Media 6*, 2 (2020), 147–156.

[MGT*19] MAHMOOD N., GHORBANI N., TROJE N. F., PONS-MOLL G., BLACK M. J.: AMASS: Archive of motion capture as surface shapes. In *International Conference on Computer Vision* (Oct. 2019), pp. 5442–5451.

[MPF*18] MA R., PATIL A. G., FISHER M., LI M., PIRK S., HUA B.-S., YEUNG S.-K., TONG X., GUIBAS L., ZHANG H.: Language-driven synthesis of 3d scenes from scene databases. *ACM Trans. Graph. 37*, 6 (Dec. 2018).

[MSB*14] MANNING C. D., SURDEANU M., BAUER J., FINKEL J., BETHARD S. J., MCCLOSKY D.: The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations* (2014), pp. 55–60.

[PRL*19] PARK S., RYU H., LEE S., LEE S., LEE J.: Learning predict-and-simulate policies from unorganized human motion data. *ACM Trans. Graph. 38*, 6 (Nov. 2019).

[RLA*18] ROBERTS R., LEWIS J. P., ANJYO K., SEO J., SEOL Y.: Optimal and interactive keyframe selection for motion capture. In *SIGGRAPH Asia 2018 Technical Briefs*. 2018, pp. 1–4.

[SPH11] SCHABUS D., PUCHER M., HOFER G.: Simultaneous speech and animation synthesis. In *ACM SIGGRAPH 2011 Posters* (New York, NY, USA, 2011), SIGGRAPH '11, Association for Computing Machinery.

[SZKS19] STARKE S., ZHANG H., KOMURA T., SAITO J.: Neural state machine for character-scene interactions. *ACM Trans. Graph. 38*, 6 (Nov. 2019).

[SZKZ20] STARKE S., ZHAO Y., KOMURA T., ZAMAN K.: Local motion phases for learning multi-contact character movements. *ACM Trans. Graph. 39*, 4 (July 2020).

[VSP*17] VASWANI A., SHAZEER N., PARMAR N., USZKOREIT J., JONES L., GOMEZ A. N., KAISER L., POLOSUKHIN I.: Attention is all you need. *arXiv preprint arXiv:1706.03762* (2017).

[WLLZ20] WANG M., LYU X.-Q., LI Y.-J., ZHANG F.-L.: VR content creation and exploration with deep learning: A survey. *Computational Visual Media 6*, 1 (2020), 3–28.

[WYH*19] WANG M., YANG G.-W., HU S.-M., YAU S.-T., SHAMIR A.: Write-a-video: Computational video montage from themed text. *ACM Trans. Graph. 38*, 6 (Nov. 2019).

[XCF*13] XU K., CHEN K., FU H., SUN W.-L., HU S.-M.: Sketch2scene: Sketch-based co-retrieval and co-placement of 3d models. *ACM Trans. Graph. 32*, 4 (July 2013).

[ZGE*07] ZHU X., GOLDBERG A. B., ELDAWY M., DYER C. R., STROCK B.: A text-to-picture synthesis system for augmenting communication. In *AAAI* (2007), vol. 7, pp. 1590–1595.

[ZMG*19] ZOU C., MO H., GAO C., DU R., FU H.: Language-based colorization of scene sketches. *ACM Trans. Graph. 38*, 6 (Nov. 2019).

[ZWP*21] ZHENG Q., WU W., PAN H., MITRA N., COHEN-OR D., HUANG H.: Inferring object properties from human interaction and transferring them to new motions. *Computational Visual Media* (2021), 1–18.

[ZZLH19] ZHANG S.-H., ZHANG S.-K., LIANG Y., HALL P.: A survey of 3d indoor scene synthesis. *Journal of Computer Science and Technology 34*, 3 (2019), 594–608.