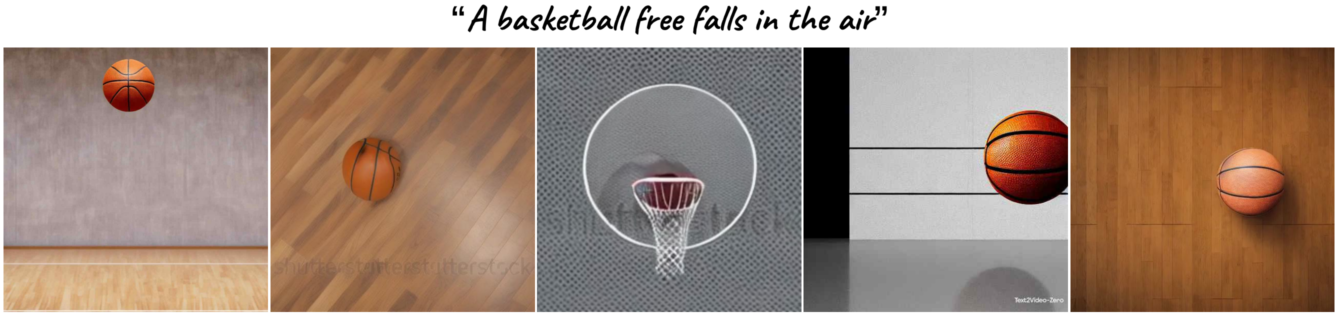
GPT4Motion: Scripting Physical Motions in Text-to-Video Generation via Blender-Oriented GPT Planning

Jiaxi Lv[1,2*] Yi Huang[1,2*] Mingfu Yan[1,2*] Jiancheng Huang[1,2] Jianzhuang Liu[1]
Yifan Liu[1] Yafei Wen[3] Xiaoxin Chen[3] Shifeng Chen[1†]
[1]Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences,
[2]University of Chinese Academy of Sciences, [3]VIVO AI Lab
{jx.lv1, yi.huang, mf.yan, jc.huang, jz.liu, yf.liu2, shifeng.chen}@siat.ac.cn
{yafei.wen, xiaoxin.chen}@vivo.com



GPT4Motion AnimateDiff [13] ModelScope [50] Text2Video-Zero [24] DirecT2V [20]

Figure 1. Comparison of the video results generated by different text-to-video models with the prompt *"A basketball free falls in the air"*. *Best viewed with Acrobat Reader for animation.*

Abstract

Recent advances in text-to-video generation have harnessed the power of diffusion models to create visually compelling content conditioned on text prompts. However, they usually encounter high computational costs and often struggle to produce videos with coherent physical motions. To tackle these issues, we propose GPT4Motion, a training-free framework that leverages the planning capability of large language models such as GPT, the physical simulation strength of Blender, and the excellent image generation ability of text-to-image diffusion models to enhance the quality of video synthesis. Specifically, GPT4Motion employs GPT-4 to generate a Blender script based on a user textual prompt, which commands Blender's built-in physics engine to craft fundamental scene components that encapsulate coherent physical motions across frames. Then these components are inputted into Stable Diffusion to generate a video aligned with the textual prompt. Experimental results on three basic physical motion scenarios, including rigid object drop and collision, cloth draping and swinging, and liquid flow, demonstrate that GPT4Motion can generate high-quality videos efficiently in maintaining motion coherency and entity consistency. GPT4Motion offers new insights in text-to-video research, enhancing its quality and broadening its horizon for future explorations. Our homepage website is `https://GPT4Motion.github.io`.

1. Introduction

In recent years, the computer vision community has shown increasing interest in generative AI. The rise of diffusion models [17, 45–47] has led to significant advancements in high-quality image generation from textual prompts, commonly known as text-to-image (T2I) synthesis [6, 39, 40, 42]. Building upon this success, researchers have explored the extension of T2I diffusion models to the realm of text-to-video (T2V) generation and editing. Earlier efforts primarily focus on directly training T2V diffusion models in pixel [10, 18, 19, 44] or latent spaces [1, 4, 8, 15, 28, 50, 51, 56, 58]. While such approaches yield promising results, their reliance on extensive datasets [3, 52, 55] for training leads to heavy computational costs. In search of more cost-effective video generation methods, some researchers have

Work done during the internship of Lv at VIVO AI Lab.
[*]Equal contributions.
[†]Corresponding author: shifeng.chen@siat.ac.cn

proposed mechanisms that adapt existing T2I models for the video domain. For example, Tune-A-Video [54] considerably reduces the training effort by fine-tuning a pretrained T2I model like Stable Diffusion [40] for video editing. However, it still requires an optimization process for each video generation.

Recent research has shifted towards developing training-free T2V approaches [21, 24] to alleviate the computational burden. For instance, Text2Video-Zero [24] utilizes the pretrained T2I model, Stable Diffusion, to synthesize videos without additional training. While these training-free methods have advanced in reducing resource requirements, they encounter challenges in achieving coherent motions, particularly when using a single user prompt to guide all frames' generation. This limitation can result in videos that lack the continuity of action or miss essential motion details due to the model's limited understanding of the temporal dynamics from a simple abstract description. To address these shortcomings, recent studies [20, 21] have harnessed the descriptive power of large language models (LLMs) [35, 53], such as GPT-4 [34] and PaLM2 [2], to generate frame-by-frame descriptions from a single user prompt, aiming to enrich the narrative across the video sequence. Building upon this foundation, subsequent research [30, 31] has taken a step further by instructing LLMs to generate not only detailed descriptions but also explicit spatiotemporal layouts from a single prompt, which then serve as conditions of the T2I diffusion models to generate videos frame by frame. Although the complemented prompts or dynamic layouts improve the video quality over methods relying on a single prompt, it is substantially challenging to ensure motion coherence particularly when there are large motion shifts.

Motivated by these LLM-assisted methods [9, 20, 21, 30, 31, 36], this paper offers a new perspective to handle the problem of motion incoherence. Specifically, we propose GPT4Motion, a training-free framework that leverages the strategic planning capability of GPT-4, the physical simulation strength of Blender[1], and the excellent image generation ability of Stable Diffusion to enhance the quality of video synthesis. Given a user textual prompt, GPT4Motion begins by deploying GPT-4 to produce Blender scripts that drive the creation of essential video scene elements, including edges and depth maps. Subsequently, these elements are then employed as conditions for Stable Diffusion to generate the final video. This methodology ensures that the resulting video not only faithfully aligns with the textual prompt but also exhibits consistent physical behaviors across all frames, as shown in Figure 1. The contributions of our work are summarized in the following.

- We demonstrate the powerful planning capability of GPT-4 in driving Blender to accurately simulate basic physical motion scenes, showing the potential of LLMs to contribute to physics-based video generation tasks.
- We propose GPT4Motion, a training-free framework that employs scripts generated by GPT-4 for Blender's scene simulation, enabling the generation of temporally coherent videos using the pretrained T2I Stable Diffusion.
- Experimental results on three basic physical motion scenarios demonstrate that GPT4Motion can efficiently generate high-quality videos which maintain motion coherency and entity consistency.

2. Related Work

2.1. Text-to-Video Generation

Text-to-video (T2V) generation targets at the creation of videos from textual descriptions. Although significant progress has been made in text-to-image (T2I) synthesis [6, 12, 23, 33, 39, 40, 42], T2V techniques are still in the early stage. The Video Diffusion Model (VDM) [19] adapts the image diffusion U-Net [41] architecture to a 3D U-Net for joint image and video training, while Make-A-Video [44] introduces a novel approach learning from image-text pairs and unlabelled videos.

While these methods depend on extensive datasets [3, 52, 55] for training, recent research [21, 24] focuses on training-free T2V to reduce training costs. For instance, Text2Video-Zero [24] uses pretrained Stable Diffusion [40] for video synthesis, employing cross-attention with the first frame for frame consistency. DiffSynth [7] proposes a latent in-iteration deflickering framework and a video deflickering algorithm to mitigate flickering and generate coherent videos. Despite their advancements, many T2V models still face challenges such as motion incoherence and entity inconsistency. Addressing these issues, our work introduces a novel approach that integrates the planning power of LLMs with the simulation capability of Blender for T2V synthesis.

2.2. LLM-Assisted Visual Generations

Large language models like GPT-4 [34], PaLM [2], and BLOOM [43] excel in various multimodal tasks[25, 27, 42]. In the field of text-to-image generation, LLMs have been successfully used to generate prompts [5, 14] or to create spatial bounding boxes from textual prompts to control image generation [9, 29, 36]. Inspired by these developments, recent efforts [20, 21, 31] have started to incorporate LLMs into the T2V realm. For instance, Free-bloom [21] leverages LLMs to generate detailed frame-by-frame descriptions from a single prompt, thereby enriching the video's narrative. Similarly, LVD [30] expands this idea by not only producing detailed descriptions but also creating comprehensive spatiotemporal layouts that guide T2I diffusion models in the frame-by-frame video generation pro-

[1] Blender is a popular open-source 3D creation suite that offers a comprehensive set of tools for 3D modeling, animation, and rendering. See https://www.blender.org/ for details.

cess. Different from them, this paper instructs GPT-4 to generate scripts for Blender to generate scene components which further serve as conditions of Stable Diffusion to synthesize videos.

2.3. Blender in Deep Learning

Blender, an open-source 3D creation suite, offers a comprehensive set of tools for 3D modeling, animation, and rendering, enabling the creation of complex and realistic 3D scenes. Beyond these effects, Blender has also played a key role in deep learning, particularly for generating synthetic data crucial for model training [49]. Additionally, 3D-GPT [48] instructs LLMs to drive Infinigen [38], a Python-Blender-based library of generation functions, for procedural 3D modeling. However, the application of Blender on T2V has not yet been explored. Traditional video creation utilizing Blender often requires much professional technical knowledge and involves complex manual procedures such as texturing, rigging, animation, lighting and compositing.

Our GPT4Motion simplifies this process by introducing an innovative framework that employs GPT-4 to generate Blender's script, which bypasses the need for manual interaction and intricate scene setup. This not only simplifies the creation process but also ensures the temporal coherence and textual alignment of the generated videos. Through the integration of GPT-4-driven scripting with Blender's advanced simulation capability, GPT4Motion marks substantial progress in the T2V domain, offering a user-friendly and efficient approach to producing high-quality videos.

3. Method

3.1. Task Formulation

Given a user prompt about some basic physical motion scenario, we aim to generate a physically accurate video. Physical phenomena are often associated with the material of the object. We focus on simulating three common types of object materials encountered in daily life: 1) *Rigid Objects*, such as balls, which maintain their shapes when subjected to forces; 2) *Cloth*, such as flags, characterized by their softness and propensity to flutter; 3) *Liquid*, such as water, which exhibits continuous and deformable motions. Moreover, we give particular attention to several typical motion modes for these materials, including *collisions* (direct impacts between objects), *wind effects* (motion induced by air currents), and *flow* (continuously and easily move in one direction). Simulating these physical scenarios typically involves knowledge of Classical Mechanics [11], Fluid Mechanics [26] and other physical knowledge. Current text-to-video diffusion models struggle to capture this complex physical knowledge through training, thereby failing to produce videos that adhere to physical principles.

To address these challenges, we propose a novel training-free text-to-video generation framework, named GPT4Motion, which is illustrated in Figure 2. The advantage of our approach is that GPT-4's semantic understanding and code generation capabilities are leveraged to translate the user prompt into a Blender Python script. This script can drive Blender's built-in physics engine to simulate the corresponding physical scene. We then introduce ControlNet [57], which takes as input the dynamic results of the Blender simulation and directs Stable Diffusion to generate each frame of the video. This framework ensures that the generated video is not only consistent with the user prompt, but also physically correct. In the next sections, we describe the details of our framework.

3.2. Blender Simulations via GPT-4

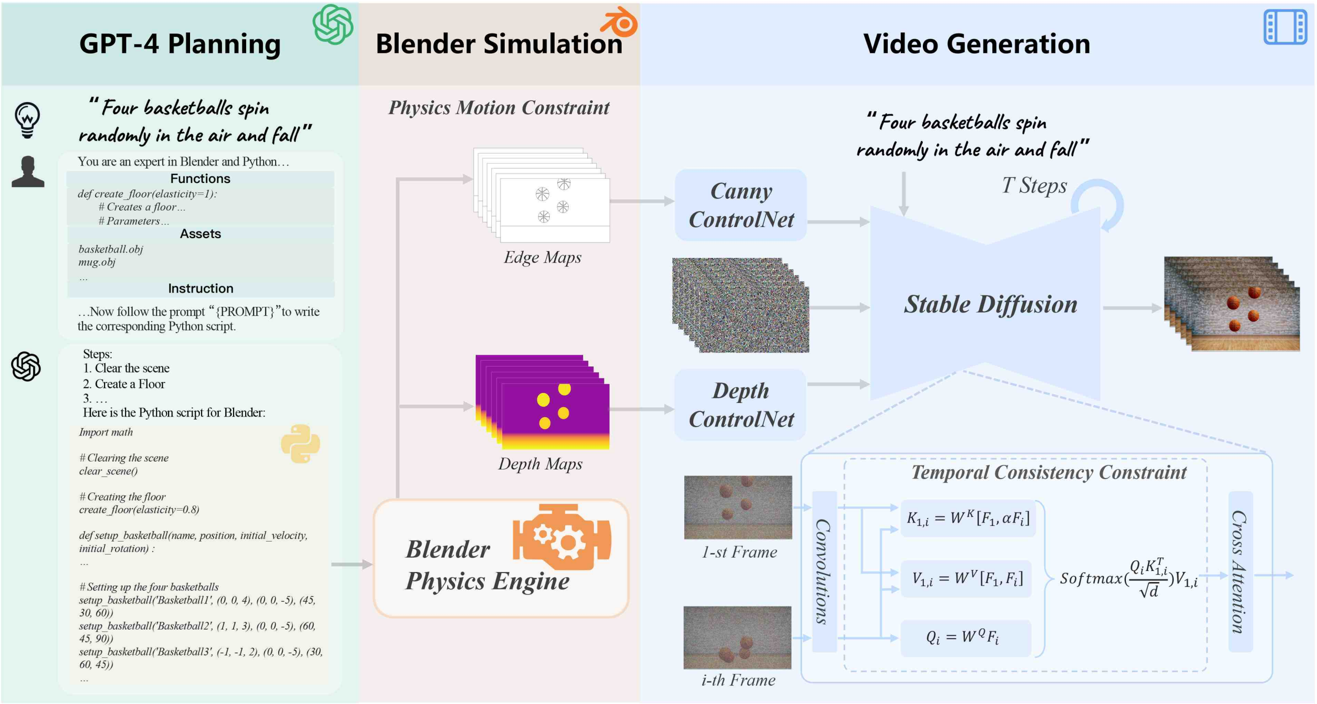
GPT-4 is a large language model pre-trained on huge amounts of Internet data with great capability for semantic understanding and code generation. We have observed that while GPT-4 has a certain knowledge about the Blender Python API, it still struggles with generating Blender Python scripts based on user prompts. On the one hand, asking GPT-4 to create even a simple 3D model (like a basketball) directly in Blender seems to be an overwhelming task [48]. On the other hand, because the Blender Python API has fewer resources and its API version is updated quickly, GPT-4 can easily misuse certain functions or make errors due to version differences. To address these issues, we propose the following schemes:

Leveraging External 3D Models. Creating 3D models typically requires professional artists to manually craft them, spending substantial time sculpting details, painting fine texture maps, and optimizing the model topology, which GPT-4 cannot independently accomplish. Fortunately, there is a large amount of 3D models available on the Internet[2]. Hence, we have collected common 3D objects from everyday life and can automatically load the 3D models via scripts corresponding to textual prompts.

Encapsulating Blender Functions. Although GPT-4 possesses the necessary knowledge of the Blender Python API, writing a lengthy script to render an entire scene remains challenging. We note that for our target scenarios, Blender Python scripts typically consist of several fixed steps, including scene initialization, rendering, object creation and import, and physical effects. Thus, we guide GPT-4 to encapsulate these reusable functions (see the supplement material). By doing so, we have greatly simplified the entire process from user prompts to rendering corresponding physical scenarios. These encapsulated functions can be broadly categorized into three types:

- *Scene initialization and rendering functions.* These functions are responsible for clearing the default initial scene

[2] https://www.blenderkit.com/


GPT-4 Planning
Blender Simulation
Video Generation
Physics Motion Constraint
Edge Maps
Depth Maps
Blender Physics Engine
" Four basketballs spin randomly in the air and fall"
T Steps
Canny ControlNet
Depth ControlNet
Stable Diffusion
Temporal Consistency Constraint
1-st Frame
i-th Frame
Convolutions
$K_{1,i} = W^K[F_1, \alpha F_i]$
$V_{1,i} = W^V[F_1, F_i]$
$Q_i = W^Q F_i$
$Softmax(\frac{Q_i K_{1,i}^T}{\sqrt{d}})V_{1,i}$
Cross Attention

" Four basketballs spin randomly in the air and fall"

You are an expert in Blender and Python…

Functions

```
def create_floor(elasticity=1):
    # Creates a floor…
    # Parameters…
```

Assets

```
basketball.obj
mug.obj
…
```

Instruction

…Now follow the prompt "{PROMPT}"to write the corresponding Python script.

Steps:
1. Clear the scene
2. Create a Floor
3. …
Here is the Python script for Blender:

```
Import math

# Clearing the scene
clear_scene()

# Creating the floor
create_floor(elasticity=0.8)

def setup_basketball(name, position, initial_velocity,
initial_rotation) :
…

# Setting up the four basketballs
setup_basketball('Basketball1', (0, 0, 4), (0, 0, -5), (45,
30, 60))
setup_basketball('Basketball2', (1, 1, 3), (0, 0, -5), (60,
45, 90))
setup_basketball('Basketball3', (-1, -1, 2), (0, 0, -5), (30,
60, 45))
…
```

Figure 2. The architecture of our GPT4Motion. First, the user prompt is inserted into our designed prompt template. Then, the Python script generated by GPT-4 drives the Blender physics engine to simulate the corresponding motion, producing sequences of edge maps and depth maps. Finally, two ControlNets are employed to constrain the physical motion of video frames generated by Stable Diffusion, where a temporal consistency constraint is designed to enforce the coherence among frames.

and performing the rendering. In Blender, one can set up the simultaneous image outputs of depth, normal, edge, and segmentation for a video. We find that using edge and depth images yields good performance in our framework, so we render these edge and depth images for video generation.

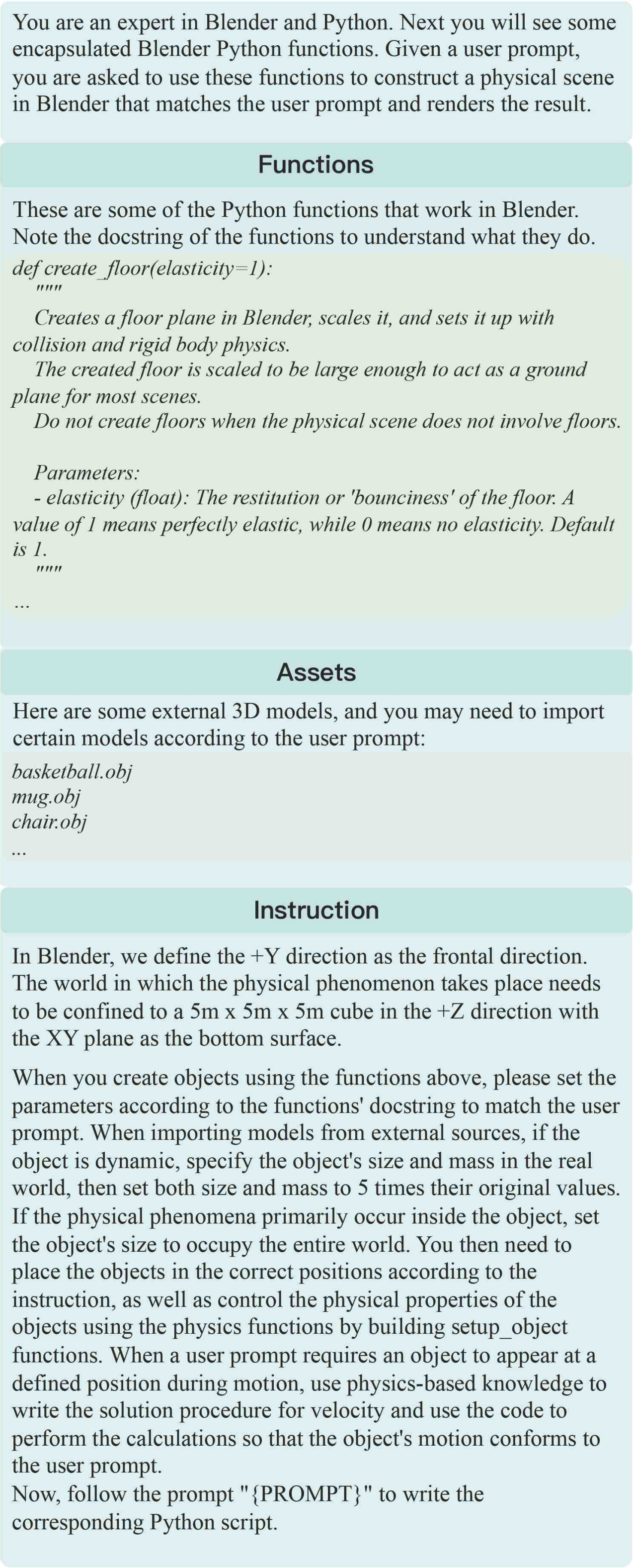
- *Object creation and import functions.* These functions offer the capability to create basic objects (such as viewpoints, floors, cubes, spheres, etc.) within a Blender scene. In addition to creating simple objects, we also provide import functions that allow users to bring external 3D models into Blender.
- *Physics effect functions.* These functions encapsulate the basic physics and material effect settings within Blender. For instance, they can assign different physical types (such as rigid, cloth, or liquid) to objects or set up wind force effects.

Translating User Prompts into Physics. Figure 3 shows the general prompt template we design for GPT-4. It includes encapsulated Blender functions, external assets, and instruction. We define the dimensions of the virtual world in the template and provide information about the camera's position and viewpoint. Such information aids GPT-4 in better understanding the layout of the 3D space. Ultimately, the user prompt becomes part of the instruction, directly guiding GPT-4 to generate the corresponding Blender Python script. Finally, with this script, Blender renders the edge and depth image sequences.

3.3. Video Synthesis with Physical Conditions

Our goal is to generate a consistent and realistic video based on the user prompt and corresponding physical motion conditions provided by Blender. We adopt Stable Diffusion XL (SDXL) [37], an upgraded version of Stable Diffusion [40]. We made the following modifications to SDXL.

Physics Motion Constraints. ControlNet [57] is a network architecture that can control the image generation of a pretrained text-to-image diffusion model with additional conditions, such as edge or depth. However, a single ControlNet is limited to one type of condition. The generation of some physical motion videos requires the control of multiple conditions. For example, when generating a video of a basketball in free fall, its edges can accurately reflect its texture changes, but the edges cannot reflect 3D layout of the scene, resulting in the lack of realism in the video. On the other hand, the depth map of the scene helps address this problem but is unable to capture the texture changes of the basketball. Therefore, we leverage a combination of Canny-



You are an expert in Blender and Python. Next you will see some encapsulated Blender Python functions. Given a user prompt, you are asked to use these functions to construct a physical scene in Blender that matches the user prompt and renders the result.

Functions

These are some of the Python functions that work in Blender. Note the docstring of the functions to understand what they do.

```
def create_floor(elasticity=1):
    """
    Creates a floor plane in Blender, scales it, and sets it up with collision and rigid body physics.
    The created floor is scaled to be large enough to act as a ground plane for most scenes.
    Do not create floors when the physical scene does not involve floors.

    Parameters:
    - elasticity (float): The restitution or 'bounciness' of the floor. A value of 1 means perfectly elastic, while 0 means no elasticity. Default is 1.
    """
...
```

Assets

Here are some external 3D models, and you may need to import certain models according to the user prompt:

```
basketball.obj
mug.obj
chair.obj
...
```

Instruction

In Blender, we define the +Y direction as the frontal direction. The world in which the physical phenomenon takes place needs to be confined to a 5m x 5m x 5m cube in the +Z direction with the XY plane as the bottom surface.

When you create objects using the functions above, please set the parameters according to the functions' docstring to match the user prompt. When importing models from external sources, if the object is dynamic, specify the object's size and mass in the real world, then set both size and mass to 5 times their original values. If the physical phenomena primarily occur inside the object, set the object's size to occupy the entire world. You then need to place the objects in the correct positions according to the instruction, as well as control the physical properties of the objects using the physics functions by building setup_object functions. When a user prompt requires an object to appear at a defined position during motion, use physics-based knowledge to write the solution procedure for velocity and use the code to perform the calculations so that the object's motion conforms to the user prompt.
Now, follow the prompt "{PROMPT}" to write the corresponding Python script.

Figure 3. Our prompt template designed for GPT-4. It contains information about functions, external assets, and instruction. The user prompt is inserted into the placeholder "{PROMPT}".

edge-based ControlNet and depth-based ControlNet to precisely control the generation of the video. Specifically, we add the intermediate results of the two ControlNets together to serve as the final conditions for SDXL.

Temporal Consistency Constraint. To ensure temporal consistency across different frames of a video, we modify the self attention (SA) in the U-Net of SDXL into cross-frame attention (CFA). Specifically, the self attention in the U-Net uses linear projections W^Q, W^K, and W^V to project the feature F_i of the i-th frame (for simplicity, we ignore the time-step t) into $Q_i = W^Q F_i$, $K_i = W^K F_i$, and $V_i = W^V F_i$, and perform the self attention calculation:

$$SA(Q_i, K_i, V_i) = \text{Softmax}(Q_i K_i^T / \sqrt{d}) V_i, \tag{1}$$

where d is a scaling factor. To obtain the cross-frame attention, we concatenate the feature of the frame F_i, $i \neq 1$, with the first frame F_1 for K and V, while keeping Q unchanged:

$$Q_i = W^Q F_i,\ K_{i,1} = W^K[F_1, \alpha F_i],\ V_{i,1} = W^V[F_1, F_i], \tag{2}$$

and the cross-frame attention operation is:

$$CFA(Q_i, K_{i,1}, V_{i,1}) = \text{Softmax}(Q_i K_{i,1}^T / \sqrt{d}) V_{i,1}, \tag{3}$$

where $[\cdot, \cdot]$ denotes the concatenation, and $\alpha \in [0, 1]$ is a hyperparameter. We find that increasing α improves the fidelity of the moving object but at the same time brings more flickering; on the contrary, decreasing α reduces the flickering but also decreases the fidelity of the moving object. The cross-frame attention has the effect that the i-th frame pays attention to not only itself but also the first frame. Surprisingly, by this cross-frame attention design, the generated video frames exhibit remarkable content consistency. Additionally, we employ the same initial noise for SDXL to generate all the frames of the video, which further enhances the temporal consistency.

4. Experiments

4.1. Implementation Details

In our experiments, we use the Stable Diffusion XL 1.0-base model[3], along with Canny-edge-based ControlNet[4] and depth-based ControlNet[5]. The α in the rigid object, cloth, and liquid experiments are set to 0.9, 0.75, and 0.4, respectively. We use the DDIM sampler [46] with classifier-free guidance [16] and 50 sampling steps in our experiments on one NVIDIA A6000 GPU. The version of the Blender is

[3] https://huggingface.co/stabilityai/stable-diffusion-xl-base-1.0

[4] https://huggingface.co/diffusers/controlnet-canny-sdxl-1.0

[5] https://huggingface.co/diffusers/controlnet-depth-sdxl-1.0

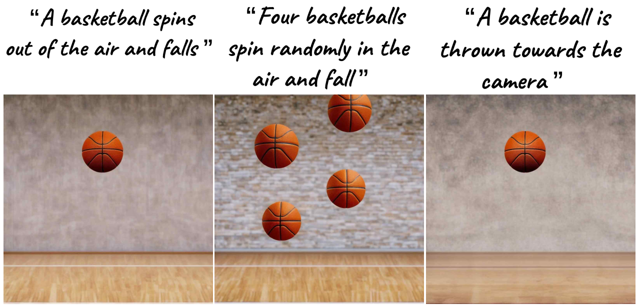

"A basketball spins out of the air and falls"
"Four basketballs spin randomly in the air and fall"
"A basketball is thrown towards the camera"

Figure 4. GPT4Motion's results on basketball drop and collision. Our homepage website is https://GPT4Motion.github.io.

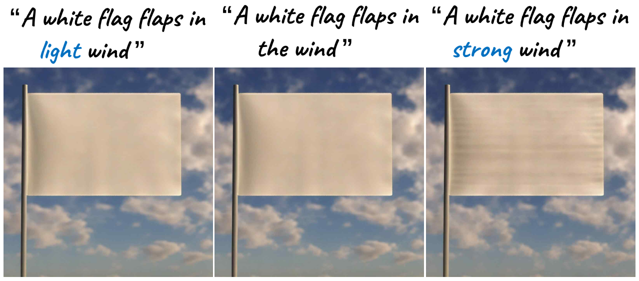

"A white flag flaps in light wind"
"A white flag flaps in the wind"
"A white flag flaps in strong wind"

Figure 5. GPT4Motion's results on a fluttering flag.

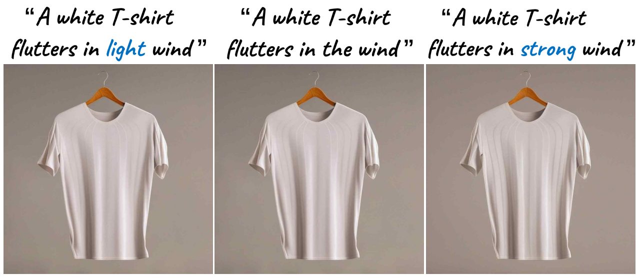

"A white T-shirt flutters in light wind"
"A white T-shirt flutters in the wind"
"A white T-shirt flutters in strong wind"

Figure 6. GPT4Motion's results on a fluttering T-shirt.

3.6. We generate 80-frame sequences of edge and depth maps at a resolution of 1920 × 1080 for each prompt. Theoretically, our method can generate motion video of any length and resolution. For conciseness, in this paper, we show the cropped video with 1080 × 1080 resolution. By the way, the videos in this experimental section may look slow, which is because too many videos are displayed at the same time on the same page. To view the motion in these videos, please use Acrobat Reader[6]. The original videos can be found in our supplementary material.

4.2. Controlling Physical Properties

We demonstrate the generative capabilities of our method across three physical scenarios, highlighting how it enables control over specific physical properties through user prompts to influence the overall generation results.

Basketball Drop and Collision. Figure 4 displays basketball motion videos generated by our method with three prompts. In Figure 4 (left), the basketball maintains a high degree of realism in its texture while spinning, and accurately replicates the bouncing behavior after collision with the floor. Figure 4 (middle) demonstrates that our method can precisely control the number of basketballs and efficiently generate the collisions and bounces that occur when multiple basketballs land. Impressively, as shown in Figure 4 (right), when the user requests that the basketball is thrown towards the camera, GPT-4 calculates the necessary initial velocity of the basketball based on its fall time in the generated script, thereby achieving a visually convincing effect. This demonstrates that our approach can be combined with the physical knowledge that GPT-4 has to control the content of the video generation (see the supplementary material for more details).

Cloth Fluttering in Wind. Figures 5 and 6 validate our method's capability in generating the motion of cloth objects influenced by wind. Utilizing existing physics engines for simulation, GPT4Motion generates the fluctuations and waves of cloth under different wind strengths. In Figure 5, we present the generated results of a flag fluttering. The flag exhibits complex ripple and wave patterns under different wind strengths. Figure 6 shows the motion of an irregular cloth object, T-shirt, under different wind strengths. Influenced by the physical properties of the fabric, such as elasticity and weight, the T-shirt undergoes flapping and twisting, with visible changes in creases and wrinkles.

Water Pouring into a Mug. Figure 7 shows three videos of water of different viscosities being poured into a mug. When the viscosity is low, the flowing water collides and merges with the water in the mug, creating complex turbulence on the surface. As the viscosity increases, the flow becomes slower and the water begins to stick together.

4.3. Comparisons with Baselines

We compare our GPT4Motion against four baselines: 1) *AnimateDiff* [13], which combines Stable Diffusion with a motion module, augmented by Realistic Vision DreamBooth[7]; 2) *ModelScope* [50], incorporating spatial-temporal convolution and attention mechanisms into Stable Diffusion for T2V tasks; 3) *Text2Video-Zero* [24], which leverages Stable Diffusion's image-to-image capabilities for generating videos through cross-attention and modified latent code sampling; 4) *DirecT2V* [20], employing a LLM for frame-level descriptions from prompts, with rotational value mapping and dual-softmax for continuity. To maintain the size

[6] https://www.adobe.com/acrobat/pdf-reader.html

[7] https : / / civitai . com / models / 4201 ? modelVersionId=29460

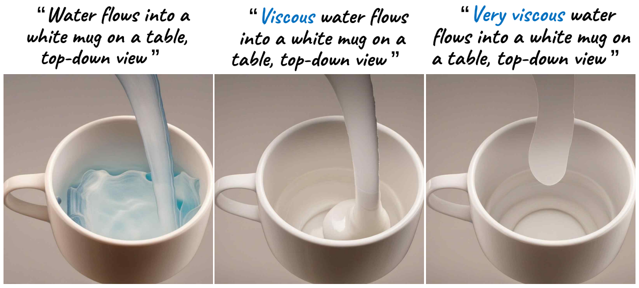

"Water flows into a white mug on a table, top-down view"
"Viscous water flows into a white mug on a table, top-down view"
"Very viscous water flows into a white mug on a table, top-down view"

Figure 7. GPT4Motion's results on the water pouring.

of the paper, we only compare GPT4Motion with these baselines on three examples. More comparisons are given in the supplementary material.

A Basketball Free Falls in the Air. As shown in Figure 1, the baselines' results do not match the user prompt. DirecT2V and Text2Video-Zero face challenges in texture realism and motion consistency, whereas AnimateDiff and ModelScope improve video smoothness but struggle with consistent textures and realistic movements. In contrast to these methods, GPT4Motion can generate smooth texture changes during the falling of the basketball, and bouncing after collision with the floor, which appear more realistic.

A White Flag Flaps in the Wind. As shown in Figure 8 (1st row), the videos generated by AnimateDiff and Text2Video-Zero exhibit artifacts/distortions in the flags, whereas ModelScope and DirecT2V are unable to smoothly generate the gradual transition of flag fluttering in the wind. However, as shown in the middle of Figure 5, the video generated by GPT4Motion can show the continuous change of wrinkles and ripples on the flag under the effect of gravity and wind.

Water Flows into a White Mug on a Table, Top-Down View. As shown in Figure 8 (2nd row), all the baselines' results fail to align with the user prompt. While the videos from AnimateDiff and ModelScope reflect changes in the water flow, they cannot capture the physical effects of water pouring into a mug. The videos generated by Text2Video-Zero and DirecT2V, on the other hand, show a constantly jittering mug. In comparison, as shown in Figure 7 (left), GPT4Motion generates the video that accurately depicts the surge of water as it collides with the mug, offering a more realistic effect.

Quantitative Evaluation and User Study. We select three metrics for quantitative comparisons: Motion Smoothness [22], which represents the fluidity of video motion and reflects the physical accuracy to some extent; CLIP scores [32], indicative of the alignment between the prompt and the video; and Temporal Flickering [22], which illustrates the flickering level of the generated videos. Please refer to the supplementary material for details on each metric. The results, as shown in Table 1, demonstrate that our GPT4Motion, leveraging GPT-4 for understanding and invoking Blender to simulate physical scenes, outperforms the other four methods on all the metrics. However, these metrics might not encompass the entire scope of video generation quality, leading us to undertake a user study for a more comprehensive evaluation. We also conduct a user study with 30 participants, where we show videos generated by different methods under the same prompt and ask the participants to vote for the best video based on three evaluation criteria: physical accuracy, text-video alignment, and the least amount of video flickering. Remarkably, our GPT4Motion's results obtain 100% of the votes.

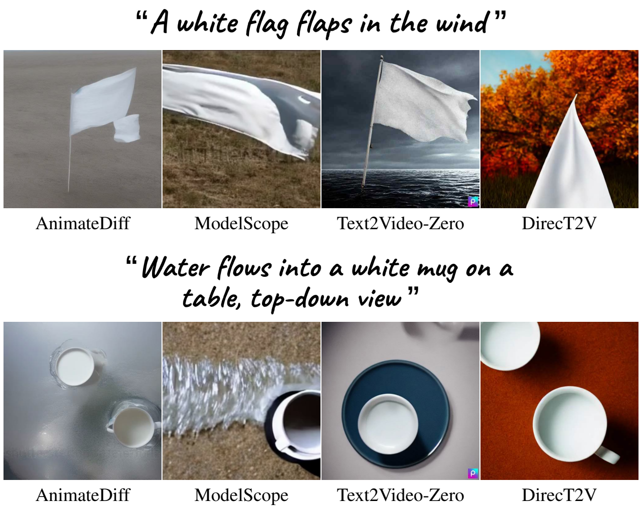

"A white flag flaps in the wind"
AnimateDiff
ModelScope
Text2Video-Zero
DirecT2V
"Water flows into a white mug on a table, top-down view"
AnimateDiff
ModelScope
Text2Video-Zero
DirecT2V

Figure 8. Videos generated by four text-to-video baselines with two user prompts.

Method	Motion↑	CLIP↑	Flickering↑
GPT4Motion	**0.993 ± 0.003**	**0.260 ± 0.022**	**0.990 ± 0.006**
AnimateDiff	0.991 ± 0.002	0.257 ± 0.020	0.988 ± 0.002
ModelScope	0.937 ± 0.051	0.252 ± 0.036	0.924 ± 0.059
Text2Video-Zero	0.946 ± 0.015	0.252 ± 0.024	0.928 ± 0.009
DirecT2V	0.879 ± 0.067	0.253 ± 0.033	0.870 ± 0.071

Table 1. Quantitative comparison across various methods. The best performances are denoted in bold.

4.4. Ablation Study

We perform an ablation study to evaluate the importance of control conditions, cross-frame attention, and α values in Eq. 2, analyzing the effect of each design separately. Experiments are conducted with the user prompt "A white flag flaps in the wind", and the video of the complete model is shown in Figure 5 (middle).

Control Conditions. Figure 9 exhibits the results across frames under different controlling conditions, which shows that the model without the edge condition (*w/o edge*) fails

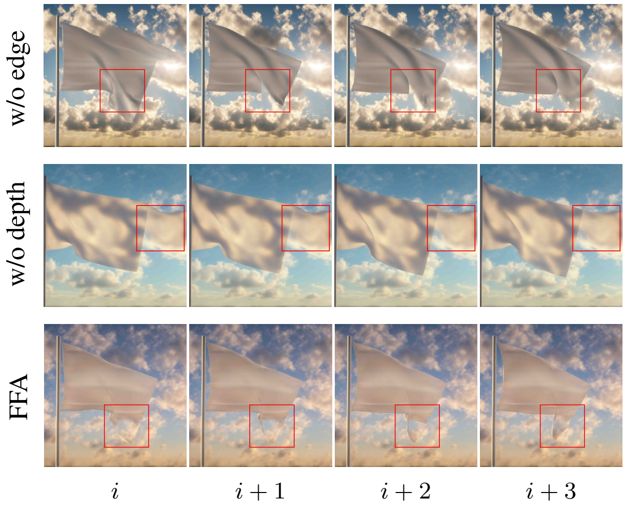


Figure 9. Ablation experiments on various control conditions and cross-frame attention. Four consecutive frames are shown.

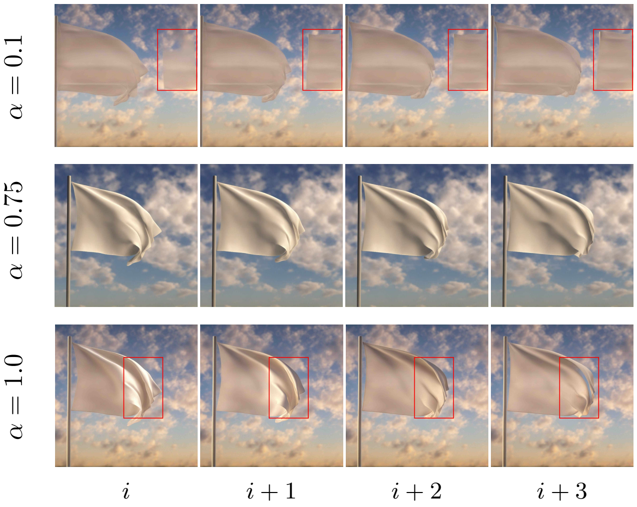


Figure 10. Ablation experiments on different α values. Four consecutive frames are shown.

to generate correct object edges (see the first row). Additionally, the model without the depth condition (*w/o depth*) not only adds extra cloth to the flag, but also mixes the flag and the cloud due to the lack of depth-of-field information. The result of Figure 5 (middle) demonstrates that the joint use of both control conditions preserves the integrity of the object edges and well handles the problem of mixing up the flag with the sky.

First-Frame Attention (FFA). In this setting, $K_{i,1}$ is replaced with $K_1 = W^K F_1$, and $V_{i,1}$ is replaced with $V_1 = W^V F_1$ during the generation of the i-th frame in Eq. 3. This means that the i-th frame only attends to the first frame (without paying attention to itself). As shown in Figure 9 (3rd row), the model *FFA* results in incomplete flag generation, where part of the flag merges with the sky and white clouds. Conversely, our cross-frame attention allows the i-th frame during its generation to focus not only on the features of the first frame but also on its own characteristics, thereby maintaining temporal consistency and ensuring the completeness of the generated object.

Different α Values. To explore the balance of the first frame and current frame in keeping temporal consistency, we select three different α values for comparison. Figure 10 presents four generated consecutive frames. It is clear that when the α value is too small, the generated results suffer from distortion, while a large α value causes flickering (inconsistent flag color intensity). By adjusting the α value to an appropriate level (i.e., 0.75), the generated results maintain the fidelity of the flag and reduce the flickering.

5. Limitations and Future Work

Although GPT4Motion advances the field of T2V synthesis, it has several limitations that set the directions for future research. While GPT4Motion successfully handles basic physical motions related to specific object materials, we have not extended it to more complex motion scenarios. We hypothesize that complex motions could be decomposed into a series of basic motions, requiring more refined instructions for LLMs. Another limitation is that sometimes the generated videos still have flickering in some frames. Despite these limitations, we believe that GPT4Motion provides a promising way for T2V generation.

6. Conclusions

This paper proposes GPT4Motion, a new training-free framework that effectively combines the advanced planning capability of Large Language Models (LLMs) with the robust simulation tool, Blender, for efficient text-to-video (T2V) synthesis. By generating Blender's scripts via GPT-4, GPT4Motion significantly simplifies the video generation process, making it more accessible and less reliant on extensive manual effort or a deep, specialized technical knowledge in 3D modeling. Experimental results on three basic physical motion scenarios, including rigid object drop and collision, cloth draping and swinging, and liquid flow, demonstrate GPT4Motion's impressive capability to efficiently generate high-quality videos with temporal coherence, surpassing previous T2V methods. GPT4Motion opens up new perspectives for T2V generation. Its integration of LLM-driven scripting and advanced Blender simulation paves a promising path for tackling more complex scenes in future research.

Acknowledgement This work was supported by Shenzhen Science and Technology Innovation Commission (JSGG20220831105002004, JCYJ20200109114835623).

References

[1] Jie An, Songyang Zhang, Harry Yang, Sonal Gupta, Jia-Bin Huang, Jiebo Luo, and Xi Yin. Latentshift: Latent diffusion with temporal shift for efficient text-to-video generation. *arXiv preprint arXiv:2304.08477*, 2023. 1

[2] Rohan Anil, Andrew M Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, et al. Palm 2 technical report. *arXiv preprint arXiv:2305.10403*, 2023. 2

[3] Max Bain, Arsha Nagrani, Gül Varol, and Andrew Zisserman. Frozen in time: A joint video and image encoder for end-to-end retrieval. In *ICCV*, 2021. 1, 2

[4] Andreas Blattmann, Robin Rombach, Huan Ling, Tim Dockhorn, Seung Wook Kim, Sanja Fidler, and Karsten Kreis. Align your latents: High-resolution video synthesis with latent diffusion models. In *CVPR*, 2023. 1

[5] Tim Brooks, Aleksander Holynski, and Alexei A Efros. Instructpix2pix: Learning to follow image editing instructions. In *CVPR*, 2023. 2

[6] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. In *NeurIPS*, 2021. 1, 2

[7] Zhongjie Duan, Lizhou You, Chengyu Wang, Cen Chen, Ziheng Wu, Weining Qian, Jun Huang, Fei Chao, and Rongrong Ji. Diffsynth: Latent in-iteration deflickering for realistic video synthesis. *arXiv preprint arXiv:2308.03463*, 2023. 2

[8] Patrick Esser, Johnathan Chiu, Parmida Atighehchian, Jonathan Granskog, and Anastasis Germanidis. Structure and content-guided video synthesis with diffusion models. In *ICCV*, 2023. 1

[9] Weixi Feng, Wanrong Zhu, Tsu-jui Fu, Varun Jampani, Arjun Akula, Xuehai He, Sugato Basu, Xin Eric Wang, and William Yang Wang. Layoutgpt: Compositional visual planning and generation with large language models. *arXiv preprint arXiv:2305.15393*, 2023. 2

[10] Songwei Ge, Seungjun Nah, Guilin Liu, Tyler Poon, Andrew Tao, Bryan Catanzaro, David Jacobs, Jia-Bin Huang, Ming-Yu Liu, and Yogesh Balaji. Preserve your own correlation: A noise prior for video diffusion models. In *ICCV*, 2023. 1

[11] Herbert Goldstein, Charles Poole, and John Safko. Classical mechanics, 2002. 3

[12] Shuyang Gu, Dong Chen, Jianmin Bao, Fang Wen, Bo Zhang, Dongdong Chen, Lu Yuan, and Baining Guo. Vector quantized diffusion model for text-to-image synthesis. In *CVPR*, 2022. 2

[13] Yuwei Guo, Ceyuan Yang, Anyi Rao, Yaohui Wang, Yu Qiao, Dahua Lin, and Bo Dai. Animatediff: Animate your personalized text-to-image diffusion models without specific tuning. *arXiv preprint arXiv:2307.04725*, 2023. 1, 6

[14] Yaru Hao, Zewen Chi, Li Dong, and Furu Wei. Optimizing prompts for text-to-image generation. *arXiv preprint arXiv:2212.09611*, 2022. 2

[15] Yingqing He, Tianyu Yang, Yong Zhang, Ying Shan, and Qifeng Chen. Latent video diffusion models for high-fidelity video generation with arbitrary lengths. *arXiv preprint arXiv:2211.13221*, 2022. 1

[16] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022. 5

[17] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, 2020. 1

[18] Jonathan Ho, William Chan, Chitwan Saharia, Jay Whang, Ruiqi Gao, Alexey Gritsenko, Diederik P Kingma, Ben Poole, Mohammad Norouzi, David J Fleet, et al. Imagen video: High definition video generation with diffusion models. *arXiv preprint arXiv:2210.02303*, 2022. 1

[19] Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. In *NeurIPS*, 2022. 1, 2

[20] Susung Hong, Junyoung Seo, Sunghwan Hong, Heeseong Shin, and Seungryong Kim. Large language models are frame-level directors for zero-shot text-to-video generation. *arXiv preprint arXiv:2305.14330*, 2023. 1, 2, 6

[21] Hanzhuo Huang, Yufan Feng, Cheng Shi, Lan Xu, Jingyi Yu, and Sibei Yang. Free-bloom: Zero-shot text-to-video generator with llm director and ldm animator. In *NeurIPS*, 2023. 2

[22] Ziqi Huang, Yinan He, Jiashuo Yu, Fan Zhang, Chenyang Si, Yuming Jiang, Yuanhan Zhang, Tianxing Wu, Qingyang Jin, Nattapol Chanpaisit, et al. Vbench: Comprehensive benchmark suite for video generative models. *arXiv preprint arXiv:2311.17982*, 2023. 7

[23] Betker James, Goh Gabriel, Jing Li, Brooks Tim, Wang Jianfeng, Li Linjie, Ouyang Long, and et.al. Improving image generation with better captions. 2023. 2

[24] Levon Khachatryan, Andranik Movsisyan, Vahram Tadevosyan, Roberto Henschel, Zhangyang Wang, Shant Navasardyan, and Humphrey Shi. Text2video-zero: Text-to-image diffusion models are zero-shot video generators. In *ICCV*, 2023. 1, 2, 6

[25] Yuma Koizumi, Yasunori Ohishi, Daisuke Niizumi, Daiki Takeuchi, and Masahiro Yasuda. Audio captioning using pre-trained large-scale language model

guided by audio-based similar caption retrieval. *arXiv preprint arXiv:2012.07331*, 2020. 2

[26] Pijush K Kundu, Ira M Cohen, and David R Dowling. *Fluid mechanics*. Academic press, 2015. 3

[27] Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *ICML*, 2022. 2

[28] Xin Li, Wenqing Chu, Ye Wu, Weihang Yuan, Fanglong Liu, Qi Zhang, Fu Li, Haocheng Feng, Errui Ding, and Jingdong Wang. Videogen: A reference-guided latent diffusion approach for high definition text-to-video generation. *arXiv preprint arXiv:2309.00398*, 2023. 1

[29] Long Lian, Boyi Li, Adam Yala, and Trevor Darrell. Llm-grounded diffusion: Enhancing prompt understanding of text-to-image diffusion models with large language models. *arXiv preprint arXiv:2305.13655*, 2023. 2

[30] Long Lian, Baifeng Shi, Adam Yala, Trevor Darrell, and Boyi Li. Llm-grounded video diffusion models. *arXiv preprint arXiv:2309.17444*, 2023. 2

[31] Han Lin, Abhay Zala, Jaemin Cho, and Mohit Bansal. Videodirectorgpt: Consistent multi-scene video generation via llm-guided planning. *arXiv preprint arXiv:2309.15091*, 2023. 2

[32] Yaofang Liu, Xiaodong Cun, Xuebo Liu, Xintao Wang, Yong Zhang, Haoxin Chen, Yang Liu, Tieyong Zeng, Raymond Chan, and Ying Shan. Evalcrafter: Benchmarking and evaluating large video generation models. *arXiv preprint arXiv:2310.11440*, 2023. 7

[33] Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. Glide: Towards photorealistic image generation and editing with text-guided diffusion models. In *ICML*, 2022. 2

[34] OpenAI. Gpt-4 technical report. *arXiv 2303.08774*, 2023. 2

[35] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. In *NeurIPS*, 2022. 2

[36] Quynh Phung, Songwei Ge, and Jia-Bin Huang. Grounded text-to-image synthesis with attention refocusing. *arXiv preprint arXiv:2306.05427*, 2023. 2

[37] Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. Sdxl: Improving latent diffusion models for high-resolution image synthesis. *arXiv preprint arXiv:2307.01952*, 2023. 4

[38] Alexander Raistrick, Lahav Lipson, Zeyu Ma, Lingjie Mei, Mingzhe Wang, Yiming Zuo, Karhan Kayan, Hongyu Wen, Beining Han, Yihan Wang, et al. Infinite photorealistic worlds using procedural generation. In *CVPR*, 2023. 3

[39] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022. 1, 2

[40] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022. 1, 2, 4

[41] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015. 2

[42] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S Sara Mahdavi, Rapha Gontijo Lopes, et al. Photorealistic text-to-image diffusion models with deep language understanding. In *NeurIPS*, 2022. 1, 2

[43] Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*, 2022. 2

[44] Uriel Singer, Adam Polyak, Thomas Hayes, Xi Yin, Jie An, Songyang Zhang, Qiyuan Hu, Harry Yang, Oron Ashual, Oran Gafni, et al. Make-a-video: Text-to-video generation without text-video data. In *ICLR*, 2023. 1, 2

[45] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *ICML*, 2015. 1

[46] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *ICLR*, 2021. 5

[47] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *ICLR*, 2021. 1

[48] Chunyi Sun, Junlin Han, Weijian Deng, Xinlong Wang, Zishan Qin, and Stephen Gould. 3d-gpt: Procedural 3d modeling with large language models. *arXiv preprint arXiv:2310.12945*, 2023. 3

[49] Hui Tang and Kui Jia. A new benchmark: On the utility of synthetic data with blender for bare supervised learning and downstream domain adaptation. In *CVPR*, 2023. 3

[50] Jiuniu Wang, Hangjie Yuan, Dayou Chen, Yingya Zhang, Xiang Wang, and Shiwei Zhang. Mod-

elscope text-to-video technical report. *arXiv preprint arXiv:2308.06571*, 2023. 1, 6

[51] Yaohui Wang, Xinyuan Chen, Xin Ma, Shangchen Zhou, Ziqi Huang, Yi Wang, Ceyuan Yang, Yinan He, Jiashuo Yu, Peiqing Yang, et al. Lavie: High-quality video generation with cascaded latent diffusion models. *arXiv preprint arXiv:2309.15103*, 2023. 1

[52] Yi Wang, Yinan He, Yizhuo Li, Kunchang Li, Jiashuo Yu, Xin Ma, Xinyuan Chen, Yaohui Wang, Ping Luo, Ziwei Liu, et al. Internvid: A large-scale video-text dataset for multimodal understanding and generation. *arXiv preprint arXiv:2307.06942*, 2023. 1, 2

[53] Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. Finetuned language models are zero-shot learners. In *ICLR*, 2021. 2

[54] Jay Zhangjie Wu, Yixiao Ge, Xintao Wang, Stan Weixian Lei, Yuchao Gu, Yufei Shi, Wynne Hsu, Ying Shan, Xiaohu Qie, and Mike Zheng Shou. Tune-a-video: One-shot tuning of image diffusion models for text-to-video generation. In *ICCV*, 2023. 2

[55] Hongwei Xue, Tiankai Hang, Yanhong Zeng, Yuchong Sun, Bei Liu, Huan Yang, Jianlong Fu, and Baining Guo. Advancing high-resolution video-language representation with large-scale video transcriptions. In *CVPR*, 2022. 1, 2

[56] Shengming Yin, Chenfei Wu, Huan Yang, Jianfeng Wang, Xiaodong Wang, Minheng Ni, Zhengyuan Yang, Linjie Li, Shuguang Liu, Fan Yang, et al. Nuwa-xl: Diffusion over diffusion for extremely long video generation. *arXiv preprint arXiv:2303.12346*, 2023. 1

[57] Lvmin Zhang and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. *arXiv preprint arXiv:2302.05543*, 2023. 3, 4

[58] Daquan Zhou, Weimin Wang, Hanshu Yan, Weiwei Lv, Yizhe Zhu, and Jiashi Feng. Magicvideo: Efficient video generation with latent diffusion models. *arXiv preprint arXiv:2211.11018*, 2022. 1