SceneScape: Text-Driven Consistent Scene Generation

Supplementary Material

Paper ID: 4362

Anonymous Author(s)
Affiliation
Address
email

1 Implementation Details

We provide implementation details for our framework and finetuning/generation regime.

Runtime. We use the Stable Diffusion [8] model that was additionally finetuned on the inpainted task to perform inpainting. We use DDIM scheduler [9] with 50 sampling steps for each generated frame. Synthesizing 50 frame-long videos with our full method takes approximately 2.5 hours on an NVIDIA TeslaV100 GPU. Specifically, Table 3 reports runtime required for each frame.

Rendering	Inpainting	Depth model finetuning	Decoder finetuning
~40 sec	~5 sec	~40 sec	~60 sec

Table 3: **Runtime per-frame.** Number of seconds required for each step of the framework. Note that Rendering includes antialiasing and floating artifacts fix steps.

Depth prediction model and LDM decoder finetuning. We use MiDaS-DPT Large [6] as our depth prediction model. For each generated frame, we finetune it for 300 epochs, using Adam optimizer [2] with a learning rate of $1e-7$. Additionally, we revert the weights of the depth prediction model to the initial state, as discussed in Sec. 3.3. We finetune the LDM decoder for 100 epochs on each generation step using Adam optimizer with a learning rate of $1e-4$.

Camera path. Our camera follows a rotational motion combined with translation in the negative depth direction. Starting from a simple translation for k frames, every n frames we randomly sample a new rotation direction in the x-z plane (panning), that camera follows for n frames. In our experiments, we set $k=5$ and $n=5$. We use PyTorch3D [7] to render and update our unified 3D representation.

Mask handling. We observed that the scene's geometry sometimes induces out-of-distribution inpainting masks for the Stable Diffusion inpainting model. To address this issue, we perform a morphological open operation on the inpainting mask: $\boldsymbol{M}_O = Open(M)$ with kernel size 3. Then we inpaint the mask difference $\boldsymbol{M} - \boldsymbol{M}_O$ using Telea [11], while the inpainting model operates on $\boldsymbol{M}_O$ afterward.

1.1 Mesh update.

As discussed in Sec. 3.5 in the paper, we update the mesh as follows: given an image $\boldsymbol{I_{t+1}}$, a mask of pixels to unproject $\boldsymbol{M}$, a corresponding depth map $\boldsymbol{D_{t+1}}$ and a camera pose $\boldsymbol{C_{t+1}}$, we unproject the

Submitted to 37th Conference on Neural Information Processing Systems (NeurIPS 2023). Do not distribute.

(a) No antialiasing (a) With antialiasing (c) No stretch removal (d) With stretch removal

Figure 8: **Rendering improvements.** Left: effect of rendering without antialiasing (a) and with antialiasing (b). Right: effect of rendering without stretched triangles removal (a) and with stretched triangles removal (b).

content in a process that is denoted in the main paper by $\tilde{\mathcal{M}}_{t+1} = \mathit{UnProj}\left(\boldsymbol{M}, \boldsymbol{I_{t+1}}, \boldsymbol{D_{t+1}}, \boldsymbol{C_{t+1}}\right)$. First, each pixel center is unprojected by its depth value and camera pose into a 3D mesh vertex with the pixel's color as its vertex color. Then, each unprojected four neighboring pixels with coordinates $(i, j), (i+1, j), (i, j+1), (i+1, j+1)$ are used to define two adjacent triangle faces.

To prevent holes in the generated mesh we connect the existing mesh $\mathcal{M}_t$ to the newly unprojected part $\tilde{\mathcal{M}}_{t+1}$ by inserting additional triangles. To do this, we first extract the boundary pixels surrounding the inpainting mask $\boldsymbol{M}$, and find the faces from $\mathcal{M}_t$ that are projected to the current view $\boldsymbol{C}_{t+1}$. For each such face, we select the 3D point closest to the current camera center: $p^* = \underset{p \in (p_1, p_2, p_3)}{\arg\min} \|p - c\|^2$, where c is the camera center of $\boldsymbol{C}_{t+1}$, and (p_1, p_2, p_3) are the points of a given triangle. Finally, we add the selected points to the triangulation scheme described above, which automatically creates the required triangles connecting $\mathcal{M}_t$ and $\tilde{\mathcal{M}}_{t+1}$.

1.2 Rendering

Antialiasing. Rendering an image from a mesh requires projecting points and faces to a 2D plane. The rasterization process often creates aliasing artifacts (Fig. 8, left), especially when a high-resolution mesh content from an earlier frame is rendered to a later frame, resulting in a large amount of triangle that needs to be rasterized into a low number of pixels. To avoid these artifacts, we apply antialiasing, similar to the image resize antialiasing method - we render the mesh at x2 higher resolution, apply Gaussian blur, and resize it to the required resolution.

Stretched triangles removal. As described in Sec. 3.6, the stretched triangles are forming between close and far away content along regions of depth discontinuities (Fig. 8, right), and we would like to remove them. Following Liu et al. [3], we apply the Sobel filter on the depth map D_t to detect regions of depth discontinuities and threshold the values below the threshold of 0.3. Then we find triangles that are projected to the selected edge regions and filter them based on their normals. Specifically, we keep only the following triangles: $\{tr_i | (\text{center}(tr_i) - c)^T n < \epsilon\}$, where ϵ is the threshold, n is the normal of a triangle, and c is the camera center of $\boldsymbol{C}_{t+1}$. In practice, we set $\epsilon = -0.05$.

Floating artifacts fix. As described in Sec 3.6, content at the border of the current frame can be mapped towards the interior of the next frame due to parallax. This creates "floating artifacts," shown in Fig. 3 in the paper. To overcome it, we pad the previous depth map D_t with border depth values to 1.5x the rendering resolution, as if content exists beyond the image borders. Then, after we warp the padded depth to the next camera location, we get a new mask, M_{pad}, that contains the content beyond the image borders. We use this mask to mask out the floating regions and thus enable the inpainting model to fill those holes with closer content. We perform this procedure on the image that was already rendered in 2x resolution due to the antialiasing step described above.

2 Baseline Comparison Details

VideoFusion. Since VideoFusion [4] does not have an explicit way to control the motion presented in a video, we append "zoom out video" and "camera moving backward" to the input prompt to encourage the generated videos to follow a backward camera motion. We generate 1000 videos of 16

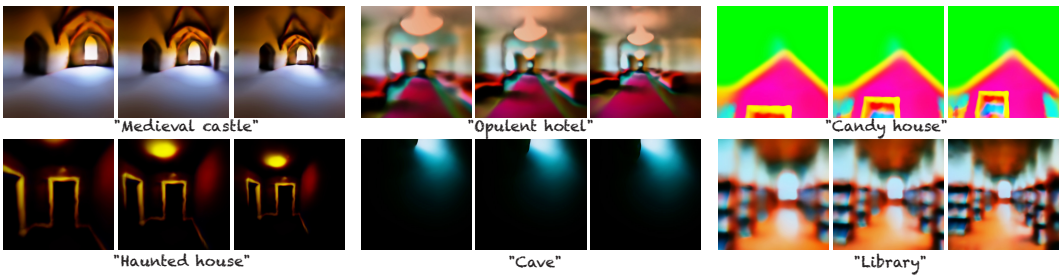

Figure 9: **StableDreamFusion sample results.** We provide it our camera trajectory and a text prompt "*a DSLR photo of the inside a* *". Full results can be found in the SM html file.

frames each, in 256 resolution. When comparing our method to theirs, we downsample our videos to 256 resolution. In this comparison, we use the full set of 10 prompts:

1. "*indoor scene, interior, candy house, fantasy, beautiful, masterpiece, best quality*"
2. "*POV, haunted house, dark, wooden door, spider webs, skeletons*"
3. "*walkthrough, an opulent hotel with long, carpeted hallways, beautiful photo, masterpiece, indoor scene*"
4. "*A dimly lit library, with rows upon rows of leather-bound books and dark wooden shelves*"
5. "*walkthrough, inside a medieval castle, metal, beautiful photo, masterpiece, indoor scene*"
6. "*Simple museum, pictures, paintings, artistic, best quality, dimly lit*"
7. "*walkthrough, sci-fi ship interiors, corridors,amazing quality, masterpiece, beautiful scenery, best quality*"
8. "*A grand, marble staircase spirals up to a vaulted ceiling in a grand entrance hall of a palace.A warm glow on the intricately designed floor*"
9. "*POV, cave, pools, water, dark cavern, inside a cave, beautiful scenery, best quality*"
10. "*inside a castle made of ice, beautiful photo, masterpiece*"

GEN-1 To compare to GEN-1 [1], we used the RealEstate10K dataset [12], consisting of curated Internet videos and corresponding camera poses. We filtered 22 indoor videos that follow a smooth temporal backward camera motion to adapt it to our method's setting. To do that, we filter videos that adhere to the following constraints:

$$\frac{(c_{t+1} - c_t)^T v_t}{\|(c_{t+1} - c_t)\| \cdot \|v_t\|} \geq 0.9 \quad \forall t = 1, \ldots, \text{n\_frames},$$

where c_t is the center of the camera at frame t, and v_t is the viewing direction (last column of the rotation matrix). We subsample the filtered videos to contain 25 frames and reverse the video if it had a positive average displacement $(c_{t+1} - c_t)$ in the z direction.

Since MiDaS produces depth maps that have a different range compared to the depth assumed in the RealEstate10K videos, we can't directly take the camera extrinsics from the RealEstate10K [12] data. To align those ranges, we need to find a scaling factor to multiply the MiDaS predictions with. To do that we run COLMAP and compute depth maps, using its dense reconstruction. Our scaling factor is then the ratio of median depth values of COLMAP and MiDaS:

$$r = \frac{\text{median}(\{D_t^C\}_{t=1}^N)}{\text{median}(\{D_t^M\}_{t=1}^N)}, \tag{1}$$

where D_t^C and D_t^M are the COLMAP and MiDaS depth maps for frame t respectively, and N is the number of frames in the video.

For comparison to GEN-1 we used prompts #1-#5 from the above prompt list that allow reasonable editing of a video, depicting an interior of a house.

StableDreamFusion. In addition, we compare our method to StableDreamfusion [10], an open-source text-to-3D model capable of generating an implicit function of a scene (NeRF [5]) from text prompts. We provide it our camera trajectory and a simple text prompt such as "*a DSLR photo of the inside a cave*" (when providing prompts from our usual prompts set, StableDreamFusion was unable to converge to meaningful results). The generated scenes contain a lot of blur and unrealistic artifacts, as can be seen in Figure 9 since to achieve good visual quality, NERF requires multiple viewpoints of a scene from different angles. Full video results of StableDreamFusion can be found in the SM HTML file.

3 Broader impact

Our framework is a test-time optimization method, which does not require any training data. Nevertheless, our framework uses two pre-trained models: a depth prediction model and a text-to-image diffusion model [8], which are susceptible to biases inherited to their training data (as discussed in Sec. 5 in [8]). In our context, these models are used to generate 3D static scenes. To avoid harmful content, we consider only text prompts that describe general scenery and objects, while avoiding prompts that involve sensitive content such as humans.

References

[1] Patrick Esser, Johnathan Chiu, Parmida Atighehchian, Jonathan Granskog, and Anastasis Germanidis. 2023. Structure and content-guided video synthesis with diffusion models. *arXiv preprint arXiv:2302.03011* (2023).

[2] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *CoRR* abs/1412.6980 (2014).

[3] Andrew Liu, Richard Tucker, Varun Jampani, Ameesh Makadia, Noah Snavely, and Angjoo Kanazawa. 2021. Infinite nature: Perpetual view generation of natural scenes from a single image. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 14458–14467.

[4] Zhengxiong Luo, Dayou Chen, Yingya Zhang, Yan Huang, Liang Wang, Yujun Shen, Deli Zhao, Jingren Zhou, and Tieniu Tan. 2023. VideoFusion: Decomposed Diffusion Models for High-Quality Video Generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

[5] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. 2020. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *European Conference on Computer Vision*.

[6] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. 2021. Vision Transformers for Dense Prediction. *ICCV* (2021).

[7] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. 2020. Accelerating 3D Deep Learning with PyTorch3D. *arXiv:2007.08501* (2020).

[8] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 10684–10695.

[9] Jiaming Song, Chenlin Meng, and Stefano Ermon. 2020. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502* (2020).

[10] Jiaxiang Tang. 2022. Stable-dreamfusion: Text-to-3D with Stable-diffusion. https://github.com/ashawkey/stable-dreamfusion.

[11] Alexandru Cristian Telea. 2004. An Image Inpainting Technique Based on the Fast Marching Method. *Journal of Graphics Tools* 9 (2004), 23 – 34.

[12] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. 2018. Stereo Magnification: Learning View Synthesis using Multiplane Images. In *SIGGRAPH*.