Supplementary Material for Dream3D: Zero-Shot Text-to-3D Synthesis Using 3D Shape Prior and Text-to-Image Diffusion Models

Jiale Xu[1,3*] Xintao Wang[1†] Weihao Cheng[1] Yan-Pei Cao[1]
Ying Shan[1] Xiaohu Qie[2] Shenghua Gao[3,4,5†]
[1]ARC Lab, [2]Tencent PCG [3]ShanghaiTech University
[4]Shanghai Engineering Research Center of Intelligent Vision and Imaging
[5]Shanghai Engineering Research Center of Energy Efficient and Custom AI IC

In this supplementary material, We give more details on the 3D shape generator G_S (Sec. A), the shape embedding mapping network G_M (Sec. B), the process of fine-tuning Stable Diffusion (Sec. C), and the process of 3D optimization with shape prior (Sec. D). We also provide additional results on text-guided shape generation (Sec. E) and text-to-3D synthesis (Sec. F). Then, we give some discussions on integrating single-view reconstruction (SVR) methods into our framework (Sec. G). Finally, we give the full list of text prompts we utilized to evaluate the CLIP retrieval precision (Sec. H).

A. Details of 3D Generator G_S

We adopt the architecture of SDF-StyleGAN [18] as our 3D generator. As Fig. 1 shows, it maps a random noise $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ to a latent shape embedding $\boldsymbol{e}_S \in \mathcal{W}$ and synthesizes a 3D feature volume $\boldsymbol{F}_V$, which is an implicit representation of the generated shape. We can query the SDF value at arbitrary position $\boldsymbol{x}$ by feeding the interpolated feature from $\boldsymbol{F}_V$ at $\boldsymbol{x}$ into a jointly trained MLP network. During training, a global discriminator and a local discriminator are used simultaneously to supervise the generated SDF grids at the coarse and fine level respectively. Different from the original SDF-StyleGAN that trains one network for *one shape category*, we train *one* 3D shape generator G_S on *13 categories* of the ShapeNet [1] dataset to enlarge the shape generation capability.

B. Details of Shape Embedding Mapping Network G_M

The shape embedding mapping network G_M is a diffusion-model-based generative network that can generate shape embeddings $\boldsymbol{e}_S$ from the CLIP image embeddings $\boldsymbol{e}_I$ of shape renderings. The network architecture and training strategy of G_M are based on an open-source DALL-E-2 [12] implementation[*]. Specifically, G_M is equivalent to the *diffusion prior network* in DALL-E-2 which generates CLIP image embeddings from CLIP text embeddings. Here we replace the input with CLIP image embeddings of shape renderings and the output with shape embeddings. We use the *DiffusionPrior* class in the codebase to implement G_M and the *train_diffusion_prior.py* script to train G_M. The model and training hyperparameters are listed in Tab. 1.

Model Parameter	Value	Training Parameter	Value
timesteps	100	iterations	500,000
beta_schedule	cosine	max_grad_norm	0.5
predict_x_start	True	batch_size	1024
cond_drop_prob	0	learning_rate	1.1×10^{-4}
dim	512	weight_decay	6.02×10^{-2}
depth	6	ema_beta	0.9999
dim_head	64	ema_update_every	10
heads	8	Adam β_1, β_2	0.9, 0.999

Table 1. Model details and training hyper-parameters of the shape embedding mapping network G_M.

Background	FID ↓
Solid white	60.61
Solid green	71.04
Random-color	**33.71**

Table 2. The Fréchet Inception Distance (FID) between the shape renderings used for fine-tuning and the images synthesized by the fine-tuned Stable Diffusion.

C. Details of Fine-tuning Stable Diffusion

In our framework, we connect the text and image modalities by fine-tuning the Stable Diffusion into a stylized gen-

[*]Work done during an internship at ARC Lab, Tencent PCG.
[†]Corresponding Author.

[*]https://github.com/lucidrains/DALLE2-pytorch

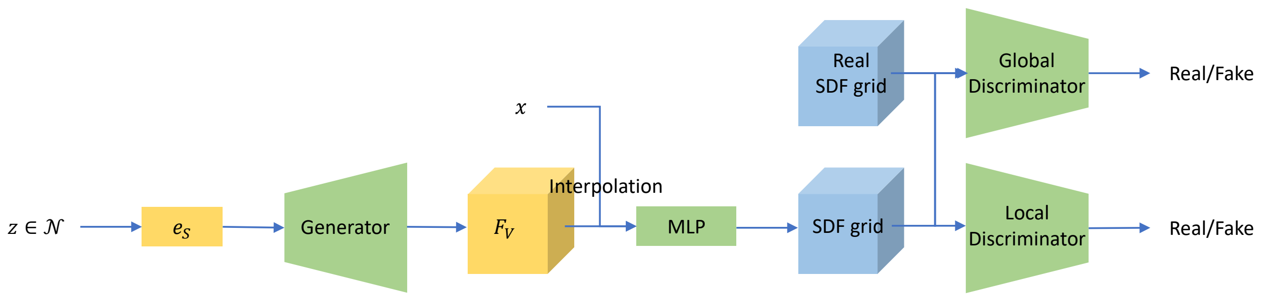
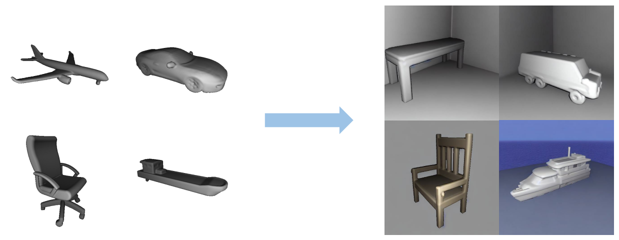
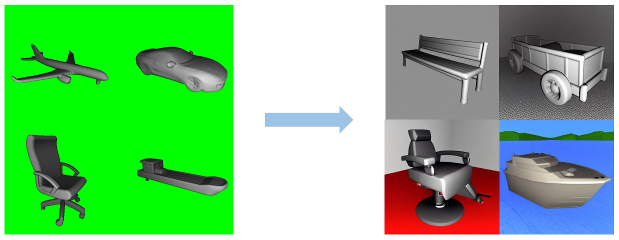


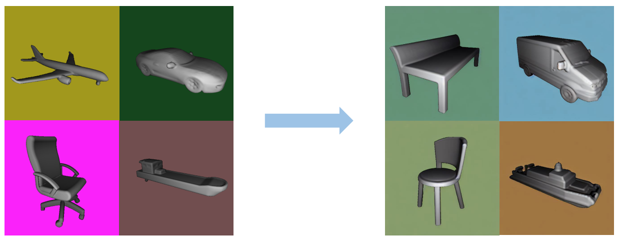
Figure 1. The network architecture of the 3D generator based on SDF-StyleGAN [18].



(a) Solid white background.



(b) Solid green background.



(c) Random-color background.

Figure 2. The results of fine-tuning Stable Diffusion using shape renderings with different backgrounds. For each type of background, we visualize the shape renderings used to fine-tune Stable Diffusion on the left and the images synthesized by the fine-tuned Stable Diffusion on the right.

erator with a set of shape renderings $\{\boldsymbol{I}_S^j\}_{j=1}^{N_S}$ and give it the ability to synthesize images in the "rendering" style. In experiments, we find it crucial to utilize a large set of shape renderings for fine-tuning and to augment the backgrounds of the shape renderings with random colors.

We tried fine-tuning Stable Diffusion using shape renderings with three different types of backgrounds: 1) solid white background, 2) solid green background and 3) random-color background. We visualize the shape renderings used for fine-tuning and the images synthesized by the fine-tuned Stable Diffusion in Fig. 2. As Fig. 2a and Fig. 2b show, although shape renderings with solid-white or solid-green backgrounds can make the fine-tuned Stable Diffusion capture the "rendering" style of the object successfully, the backgrounds in the synthesized images are out of control, *i.e.*, the fine-tuned Stable Diffusion fails to synthesize images with solid-color backgrounds. This will increase the difficulty of separating the foreground objects from the backgrounds and affect the stability of the subsequent image-to-shape generation since the shape embedding mapping network G_M is trained on shape renderings with solid-color backgrounds. In comparison, augmenting the backgrounds of the shape renderings with random colors leads to a stable stylized generator that can synthesize solid-color-background images consistently, as Fig. 2c shows.

During fine-tuning, we indeed expect the Stable Diffusion model to capture two types of styles: 1) the "rendering" style of the foreground object and 2) the "solid-color" style of the background. Similar to the observation that the foreground "rendering" style requires a large set of rendered images to learn, we consider that a single-color background is too few to be recognized as a "solid-color background style" by the Stable Diffusion model, while showing a lot of different solid-color examples to the model can make it notice the solid-color background style and capture it during fine-tuning.

To better demonstrate the importance of the random-color background augmentation, we also evaluate the Fréchet Inception Distance (FID) between the shape renderings used for fine-tuning and the images synthesized by the fine-tuned Stable Diffusion in Tab. 2. For each type of background, we render 1000 images with that background for each ShapeNet [1] category, forming a shape rendering dataset containing 13000 images in total (denoting as D_S). Then we leverage D_S to fine-tune the Stable Diffusion model for 5000 steps, and utilize the fine-tuned Stable Diffusion to synthesize 100 images for each shape category using the text prompt *"a CLS in the style of *"*, leading to a set of 1300 generated images (denoting as D_{gen}). Finally, we compute the FID between D_S and D_{gen}. As Tab. 2 shows, augmenting the backgrounds of shape renderings with random colors significantly boosts the FID, which demonstrates its effectiveness.

D. Details of 3D Optimization with 3D Shape Prior

D.1. DVGO-based Volume Rendering

In the optimization stage, we adopt DVGO [15] as our 3D scene representation which represents NeRF [9] with a density voxel grid $\boldsymbol{V}_{\text{density}} \in \mathbb{R}^{N_x \times N_y \times N_z}$ and a shallow color MLP network f_{rgb} for efficient optimization. Given a 3D position $\boldsymbol{x}$, we query its density σ and color c by:

$$\tilde{\sigma} = \text{interp}(\boldsymbol{x}, \boldsymbol{V}_{\text{density}}), \tag{1a}$$

$$\sigma = \text{softplus}(\tilde{\sigma}) = \log(1 + \exp(\tilde{\sigma} + b)), \tag{1b}$$

$$c = f_{\text{rgb}}(\gamma(\boldsymbol{x})), \tag{1c}$$

where $\gamma(\cdot)$ denotes a positional encoding function. $\text{interp}(\cdot)$ denotes the trilinear interpolation. The shifted softplus function $\text{softplus}(\cdot)$ is applied to transform the raw density value $\tilde{\sigma}$ into activated density value σ (*i.e.*, a mapping of $\mathbb{R} \to \mathbb{R}_{\geq 0}$), the shift b is a hyperparameter. To be noted, the density grid $\boldsymbol{V}_{\text{density}}$ stores the raw density values instead of the activated ones. DVGO [15] calls the scheme of interpolating on the raw density values first and then performing softplus activation as "post-activation" and demonstrates its advantages on producing sharper shape boundaries over other choices.

To render the color of a pixel $\hat{C}(r)$, we cast the ray r from the camera center through the pixel, and sample K points between the pre-defined near and far planes. We then query the densities and colors of the K ordered sampled points $\{(\sigma_i, c_i)\}_{i=1}^{K}$ using Eq. (1). Finally, we accumulate the K queried results into a single color with the volume rendering process:

$$\hat{C}(r) = \left(\sum_{i=1}^{K} T_i \alpha_i c_i\right) + T_{K+1} c_{bg}, \tag{2a}$$

$$\alpha_i = \text{alpha}\left(\sigma_i, \delta_i\right) = 1 - \exp\left(-\sigma_i \delta_i\right), \tag{2b}$$

$$T_i = \prod_{j=1}^{i-1} \left(1 - \alpha_j\right), \tag{2c}$$

where α_i denotes the opacity representing the probability of termination at point i, T_i denotes the accumulated transmittance from the near plane to point i, δ_i denotes the distance to the adjacent sampled point, and c_{bg} demotes a pre-defined background color.

Following DVGO, all values in $\boldsymbol{V}_{\text{density}}$ are initialized as

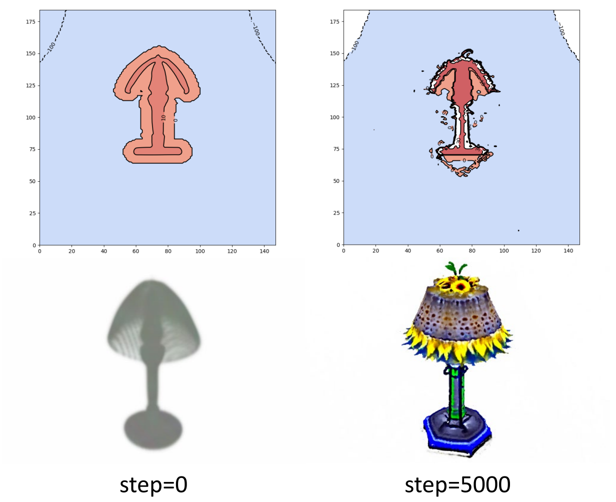


Figure 3. Illustration on 3D optimization with 3D shape prior. The text prompt is *"A lamp imitating sunflower"*. We visualize the contours of the density grid (up) and the volume-rendered images (bottom) at the 0^{th} and 5000^{th} optimization steps to show how the density grid is initialized and optimized.

0 and the bias term in Eq. (1b) is set to

$$b = \log\left(\left(1 - \alpha_{\text{init}}\right)^{-\frac{1}{s}} - 1\right), \tag{3}$$

where α_{init} is a hyperparameter and is set to 10^{-6} in practice. With such an initialization, the accumulated transmittance T_i is decayed by $1 - \alpha_{\text{init}} \approx 1$ for a ray that traces forward a distance of a voxel size s, making the scene "transparent" at the beginning of optimization.

D.2. Shape Prior Initialization and Optimization

A big difference between our text-guided 3D synthesis framework and previous methods [4–6] is that we use an explicit *3D shape prior* to initialize the CLIP-guided optimization process, instead of optimizing from a *randomly-initialized* 3D representation. Given a 3D shape prior S represented by an SDF grid $\tilde{\boldsymbol{V}}_{\text{sdf}} \in \mathbb{R}^{N_x \times N_y \times N_z}$, we use it to initialize the density voxel grid $\boldsymbol{V}_{\text{density}}$ with the following equations [11, 15, 17]:

$$\boldsymbol{\Sigma} = \frac{1}{\beta} \,\text{sigmoid}\left(-\frac{\tilde{\boldsymbol{V}}_{\text{sdf}}}{\beta}\right), \tag{4a}$$

$$\boldsymbol{V}_{\text{density}} = \max(0, \text{softplus}^{-1}(\boldsymbol{\Sigma})), \tag{4b}$$

where $\text{sigmoid}(x) = 1/(1 + e^{-x})$ and $\text{softplus}^{-1}(x) = \log(e^x - 1)$. Eq. (4a) converts SDF values to activated density values (equivalent to the σ in Eq. (1b)), where $\beta > 0$ controls the sharpness of the shape boundary, and smaller β leads to a sharper shape boundary. We set $\beta = 0.05$ in our experiments. Eq. (4b) further transforms the activated

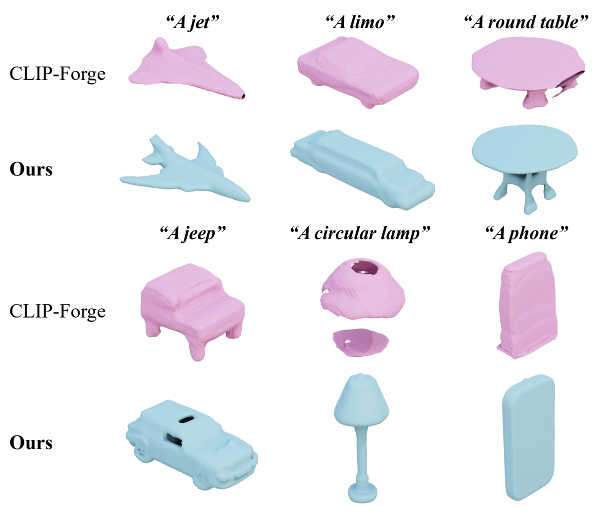


Figure 4. Additional text-guided shape generation results. All meshes are extracted at 64^3 resolution.

density values into raw density values (equivalent to the $\tilde{\sigma}$ in Eq. (1a)).

With such an initialization, the density values on the shape surface will be close to $\frac{1}{2\beta}\,(\log(\exp(\frac{1}{\beta}\cdot\text{sigmoid}(0))-1)\approx\frac{1}{2\beta})$. The area inside the shape surface will have larger density values ($>\frac{1}{2\beta}$), and the density values outside the shape will decrease with the distance from the shape surface. We set the minimum density value outside the shape to 0 so the area far from the shape surface has the same initialization as the original DVGO.

As Fig. 3 shows, at the beginning of the 3D optimization process (step=0), the 3D shape prior is visible due to the larger density values around the shape surface. As a result, the area around the shape surface will dominate the volume rendering, and the density/color values in this area will be updated faster than the area far from the surface. Based on the initialization, Then subsequent CLIP-guided optimization process further provides more flexibility and is able to synthesize more diverse structures and textures.

E. Additional Results on Text-to-Shape Generation

We show additional qualitative text-guided 3D shape generation results in Fig. 4. Compared to CLIP-Forge [13], our method produces more plausible 3D shapes thanks to the high-quality 3D generator, while the shapes generated by [13] suffer from rough surfaces and discontinuities.

Besides, we also provide more quantitative comparisons with CLIP-Forge on text-to-shape generation. We generate 3 shapes for each text prompt in the text prompt set provided by CLIP-Forge and measure three metrics: 1) Fréchet Inception Distance (FID) [3] between 5 rendered images for each shape with different camera poses and a set of images rendered from the ground truth shapes in the ShapeNet dataset with the same camera poses. 2) Fréchet Point Distance (FPD) [14], for each generated shape and each ground truth shape in the ShapeNet test set, we extract the mesh at 64^3 resolution and sample 2048 points from the mesh surface, then pass the points to a DGCNN [16] backbone network pre-trained on the point cloud classification task and use the feature of the last layer to compute this metric. 3) Maximum Measure Distance (MMD), for each generated shape represented by a 32^3 occupancy grid, we match a shape in the ShapeNet test set based on the highest IOU, and then average the IOU across all the text queries. As Tab. 3 shows, our text-to-shape generation method outperforms CLIP-Forge on all three metrics.

Method	FID ↓	FPD ↓	MMD ↑
CLIP-Forge [13]	112.38	6.896	0.670
Ours	**40.83**	**1.301**	**0.725**

Table 3. Additional quantitative results compared with CLIP-Forge on text-guided shape generation.

F. Additional Results on Text-to-3D Synthesis

In this section, we show additional qualitative comparison results on text-to-3D synthesis with baseline methods in Fig. 5 and more diversified generation results of our method in Fig. 6. It can be seen that our method can synthesize plausible 3D structures with the help of 3D shape priors.

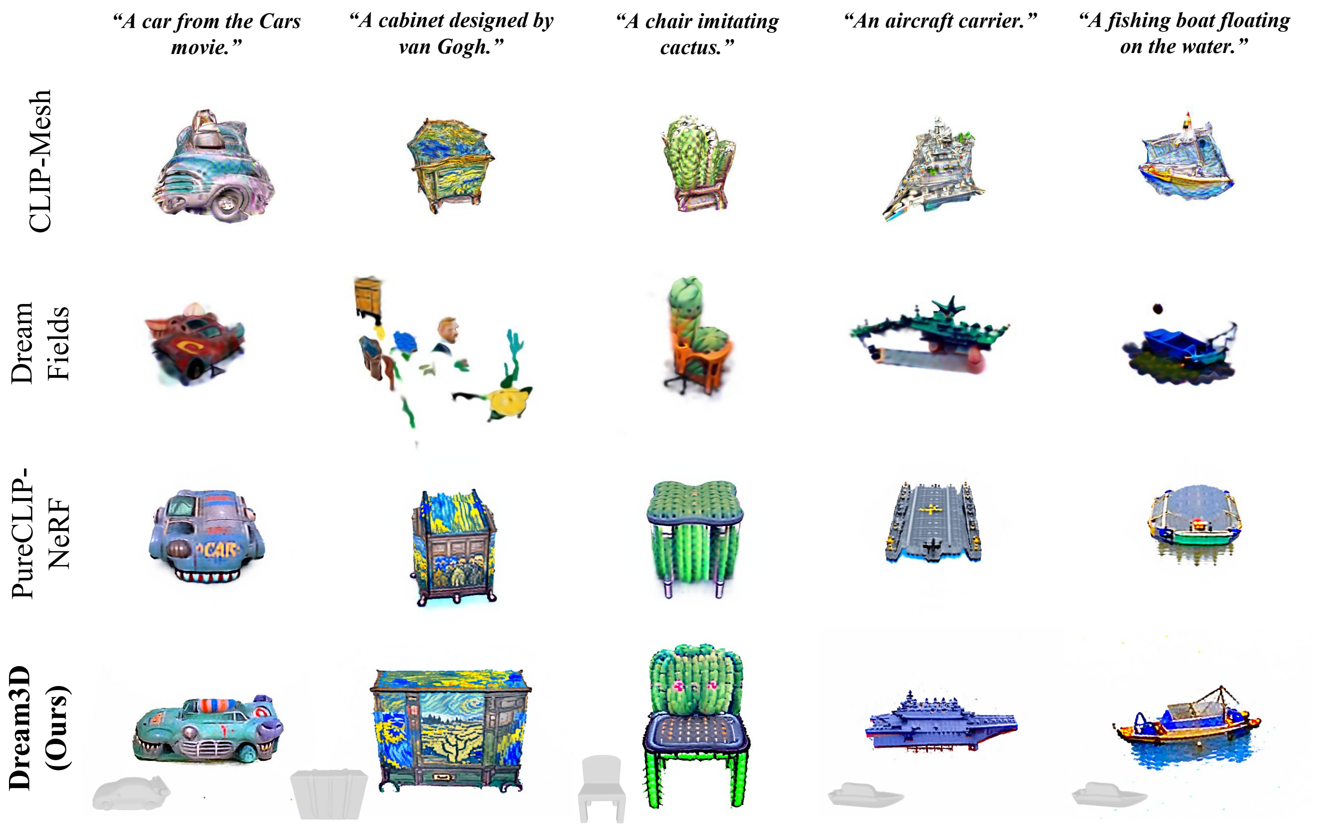

"A car from the Cars movie."
"A cabinet designed by van Gogh."
"A chair imitating cactus."
"An aircraft carrier."
"A fishing boat floating on the water."
CLIP-Mesh
Dream Fields
PureCLIP-NeRF
Dream3D (Ours)

Figure 5. Additional qualitative comparisons on text-guided 3D synthesis. For each of our results (the last row), we also visualize the 3D shape prior used to initialize the CLIP-guided optimization process below it.

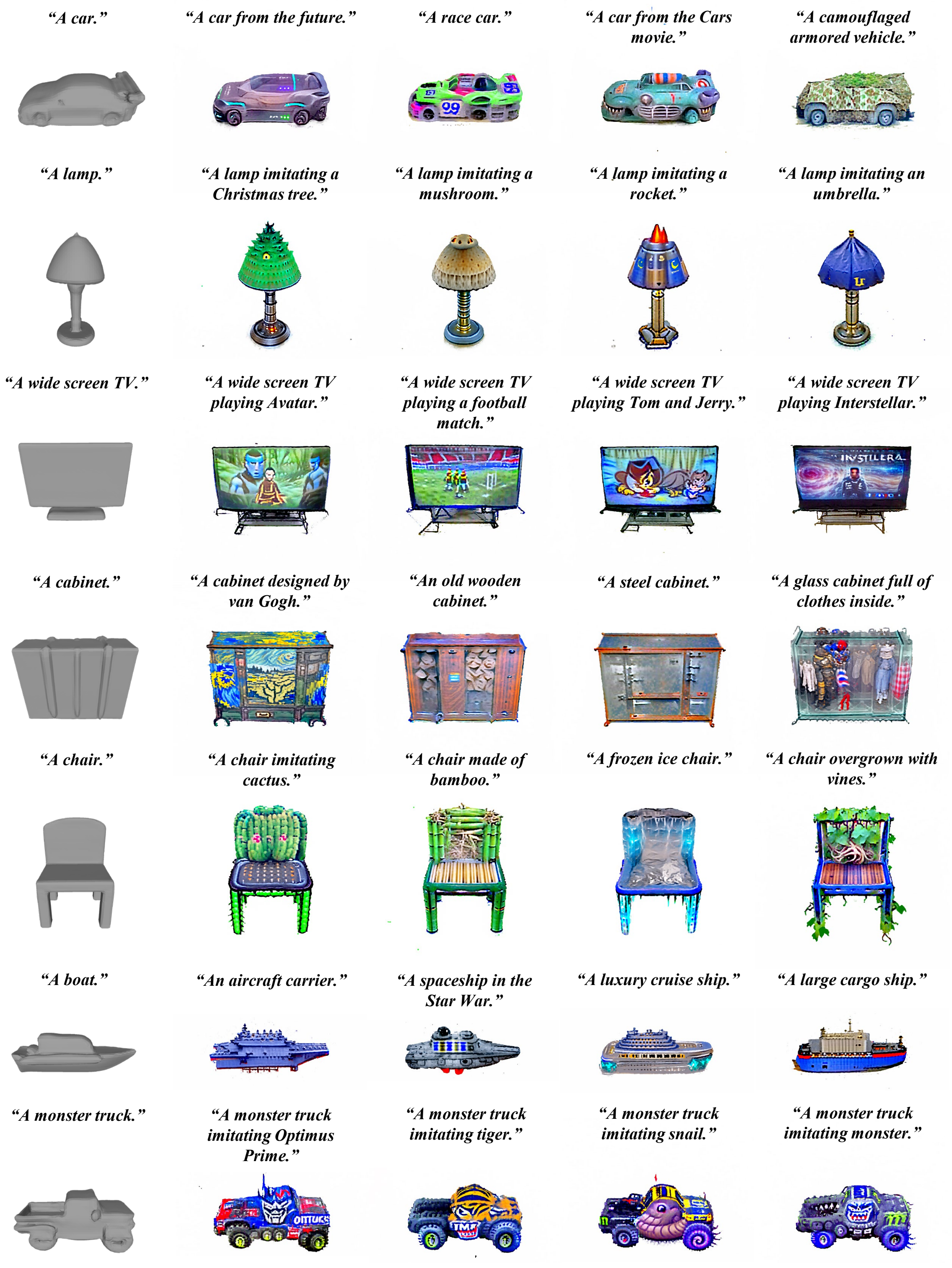

"A car."
"A car from the future."
"A race car."
"A car from the Cars movie."
"A camouflaged armored vehicle."
"A lamp."
"A lamp imitating a Christmas tree."
"A lamp imitating a mushroom."
"A lamp imitating a rocket."
"A lamp imitating an umbrella."
"A wide screen TV."
"A wide screen TV playing Avatar."
"A wide screen TV playing a football match."
"A wide screen TV playing Tom and Jerry."
"A wide screen TV playing Interstellar."
"A cabinet."
"A cabinet designed by van Gogh."
"An old wooden cabinet."
"A steel cabinet."
"A glass cabinet full of clothes inside."
"A chair."
"A chair imitating cactus."
"A chair made of bamboo."
"A frozen ice chair."
"A chair overgrown with vines."
"A boat."
"An aircraft carrier."
"A spaceship in the Star War."
"A luxury cruise ship."
"A large cargo ship."
"A monster truck."
"A monster truck imitating Optimus Prime."
"A monster truck imitating tiger."
"A monster truck imitating snail."
"A monster truck imitating monster."

Figure 6. Additional text-to-3D synthesis results. We visualize the 3D shape prior used for optimization in the first column.

G. Integration with SVR Models

G.1. Text-to-Shape Generation using SVR models

Single-view reconstruction (SVR) models can reconstruct a 3D shape from a single input image. We then ask, can we use an SVR model directly as the image-to-shape module in our framework? To answer this question, we conduct the same fine-tuning process to fine-tune a Stable Diffusion model with the ShapeNet renderings provided by Choy *et al*. [2] on which many common SVR models are trained. We find that although the shape renderings in Choy *et al*. [2] have more complex textures, the fine-tuned model can still capture the style successfully and synthesize novel images imitating the style. With such a fine-tuned Stable Diffusion, we can solve the text-to-shape generation in a precise way: synthesize an image using the fine-tuned Stable Diffusion with text prompt in the format of *"a CLS in the style of *"*, and then directly feed the synthesized image into the SVR model. We show some text-guided shape generation results using two SVR methods, *i.e*., occupancy networks [8] and DVR [10], in Fig. 7 and Fig. 8, respectively. Both methods are trained with the shape renderings provided by Choy *et al*. [2]. The occupancy networks only predict shape, while DVR can predict both shape and color. As Fig. 7 and Fig. 8 show, we achieve text-to-shape generation successfully with the synthesized images, which proves the strong generation ability of Stable Diffusion and the effectiveness of our proposed fine-tuning pipeline.

A recent work named ISS [7] also utilizes an SVR model to perform text-to-shape generation. However, the pipeline of ISS is much more complicated. It trains a mapper network to map CLIP features to the latent space of the SVR model, which requires a two-stage fine-tuning to align the text and shape feature spaces. At inference time, ISS needs to fine-tune the mapper network for each text prompt, which is redundant in our pipeline. With the help of the fine-tuned Stable Diffusion, we can directly generate an image from the text prompt and feed the image into the SVR model to synthesize a 3D shape. Besides, thanks to the strong generation ability of Stable Diffusion, we can enjoy a much larger generation diversity and synthesize as many 3D shapes as we want for each text prompt.

G.2. Text-to-3D Synthesis using SVR models

Despite the success in text-guided shape generation with SVR models, we find that current SVR models are very sensitive to the input images. Although we can successfully capture the style of the shape renderings using the fine-tuned Stable Diffusion, some minor flaws in the synthesized images such as offsets of the objects from the image center and unrealistic artifacts (*e.g*., a chair lacks a leg) are inevitable. These minor flaws may lead to failed shape reconstructions, whose quality affects 3D shape priors. This

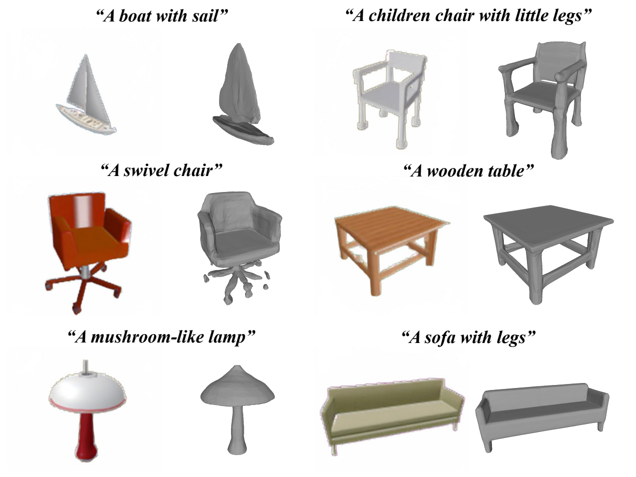


Figure 7. Text-guided shape generation using fine-tuned Stable Diffusion and Occupancy Networks [8]. For each text prompt, we visualize the image synthesized by the fine-tuned Stable Diffusion on the left and the reconstructed shape on the right.

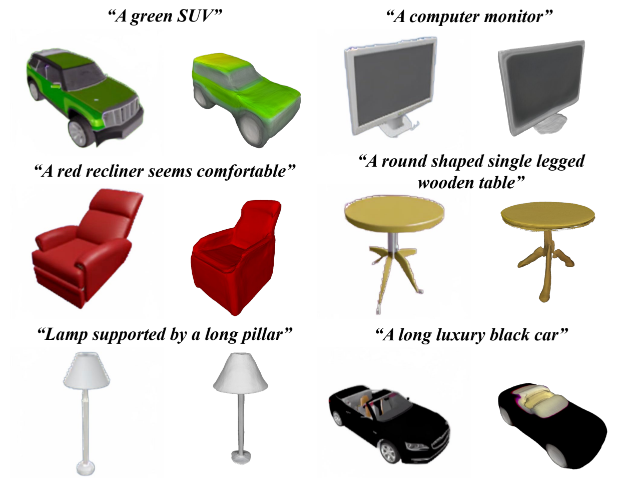

"A green SUV"
"A computer monitor"
"A red recliner seems comfortable"
"A round shaped single legged wooden table"
"Lamp supported by a long pillar"
"A long luxury black car"

Figure 8. Text-guided shape generation using fine-tuned Stable Diffusion and DVR [10]. For each text prompt, we visualize the image synthesized by the fine-tuned Stable Diffusion on the left and the reconstructed shape on the right.

sensitiveness makes the 3D prior generation in the first stage of our framework unstable. Therefore, we choose to use a 3D generator associated with a shape embedding mapping network to generate 3D shapes in the latent shape embedding space, instead of directly using an SVR model in our framework.

We visualize six text-to-3D synthesis results using 3D shape priors produced by the occupancy networks [8] in Fig. 9. The successful results in the first two rows show the probability of integrating as SVR model into our framework. In the last row, we show two failure cases in which the SVR model fails to reconstruct plausible 3D shape priors to illustrate the drawbacks of using SVR models. We can observe that the discontinuity in the "bedside lamp"

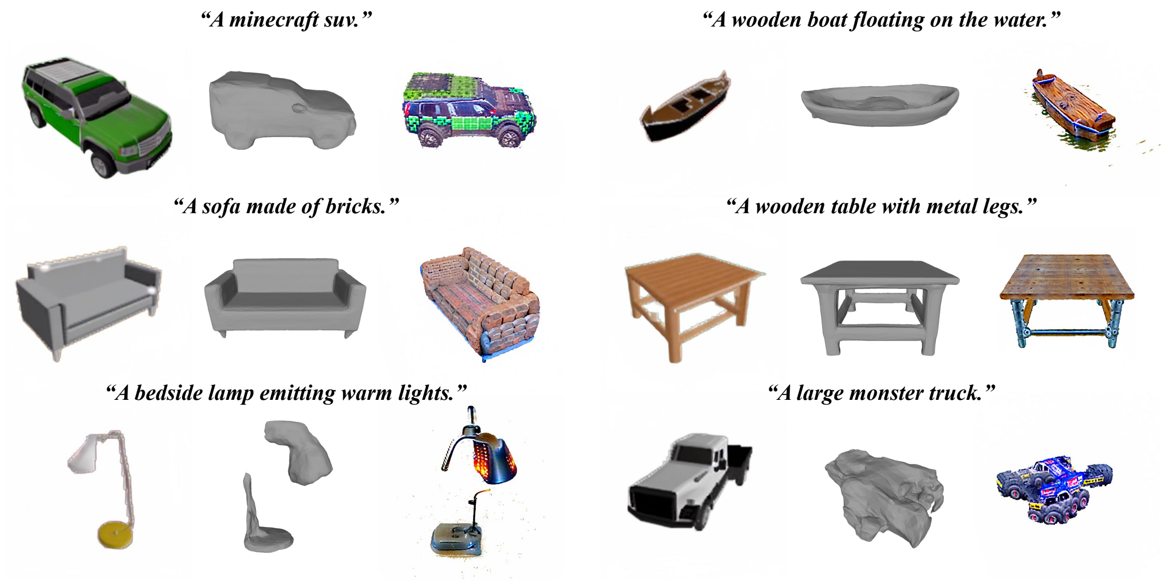

"A minecraft suv."
"A wooden boat floating on the water."
"A sofa made of bricks."
"A wooden table with metal legs."
"A bedside lamp emitting warm lights."
"A large monster truck."

Figure 9. Text-to-3D synthesis results using occupancy networks. For each text prompt, we visualize the shape rendering image synthesized by the fine-tuned Stable Diffusion on the left, the shape reconstructed by the SVR model in the middle, and the optimization result on the right. The last row shows two failure cases.

shape leads to discontinuity in the final optimization result, while the failed truck shape results in total chaos.

H. Text Prompt for Evaluation

We design a set of text prompt containing common objects in ShapeNet categories to evaluate the CLIP retrieval precision metric for baseline methods and our approach (Table 1 in the main manuscript). The full set of 92 text prompts we utilized is listed in Tab. 4.

A park bench overgrown with vines.	A round chair designed by Van Gogh.	A comfortable sofa is burning.
A park bench overgrown with roses.	A round chair designed by Einstein.	A minecraft comfortable sofa.
A park bench covered with snow.	A round chair designed by Van Gogh and Einstein.	A wide screen TV playing cartoon show of Tom and Jerry.
A park bench covered with tiger skin.	A round chair is burning.	A wide screen TV playing weather forecast.
A park bench designed by Van Gogh.	A minecraft round chair.	A wide screen TV playing breaking news.
A park bench designed by Einstein.	A throne overgrown with vines.	A wide screen TV playing The Avengers.
A park bench designed by Van Gogh and Einstein.	A throne overgrown with roses.	A wide screen TV playing a horror movie.
A park bench is burning.	A throne covered with snow.	A bus covered with assorted colorful graffiti on the side of it.
A minecraft park bench.	A throne covered with tiger skin.	A bus covered with advertisement for coca cola on the side of it.
A car overgrown with vines.	A throne designed by Van Gogh.	A bus covered with an apple symbol on the side of it.
A car overgrown with roses.	A throne designed by Einstein.	A bus covered with American flag.
A car covered with snow.	A throne designed by Van Gogh and Einstein.	A fighter jet is flying at a fast speed.
A car covered with tiger skin.	A throne is burning.	A fighter jet is firing ahead.
A car designed by Van Gogh.	A minecraft throne.	A fighter jet performing tricks with smoke coming off of it.
A car designed by Einstein.	The Iron Throne in Game of Thrones.	There are some colorful lights hanging from a street lamp.
A car designed by Van Gogh and Einstein.	A desk overgrown with vines.	There are some warm lights hanging from a street lamp.
A car is burning.	A desk overgrown with roses.	There are some cold lights hanging from a street lamp.
A minecraft car.	A desk covered with snow.	There are some colorful lights hanging from a bedside lamp.
An airplane overgrown with vines.	A desk covered with tiger skin.	There are some warm lights hanging from a bedside lamp.
An airplane overgrown with roses.	A desk designed by Van Gogh.	There are some cold lights hanging from a bedside lamp.
An airplane covered with snow.	A desk designed by Einstein.	A fisherman stands on a fishing boat floating on the water.
An airplane covered with tiger skin.	A desk designed by Van Gogh and Einstein.	The dishes are neatly arranged behind the glass cabinet doors.
An airplane designed by Van Gogh.	A desk is burning.	A park bench sits under a tree with the sun shining.
An airplane designed by Einstein.	A minecraft desk.	A table for playing foosball.
An airplane designed by Van Gogh and Einstein.	A comfortable sofa overgrown with vines.	A table for playing ping pong.
An airplane is burning.	A comfortable sofa overgrown with roses.	A cat is sleeping comfortably on the cushion of chair.
A minecraft airplane.	A comfortable sofa covered with snow.	A lamp imitating sunflower.
A round chair overgrown with vines.	A comfortable sofa covered with tiger skin.	A plane imitating an eagle.
A round chair overgrown with roses.	A comfortable sofa designed by Van Gogh.	A plane imitating a dragon.
A round chair covered with snow.	A comfortable sofa designed by Einstein.	A truck imitating a crocodile.
A round chair covered with tiger skin.	A comfortable sofa designed by Van Gogh and Einstein.	

Table 4. The text prompt dataset used to evaluate the CLIP retrieval precision metric.

References

[1] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. 1, 2

[2] Christopher B Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016. 7

[3] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017. 4

[4] Ajay Jain, Ben Mildenhall, Jonathan T. Barron, Pieter Abbeel, and Ben Poole. Zero-shot text-guided object generation with dream fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 867–876, June 2022. 3

[5] Nasir Mohammad Khalid, Tianhao Xie, Eugene Belilovsky, and Popa Tiberiu. Clip-mesh: Generating textured meshes from text using pretrained image-text models. December 2022. 3

[6] Han-Hung Lee and Angel X Chang. Understanding pure clip guidance for voxel grid nerf models. *arXiv preprint arXiv:2209.15172*, 2022. 3

[7] Zhengzhe Liu, Peng Dai, Ruihui Li, Xiaojuan Qi, and Chi-Wing Fu. Iss: Image as stetting stone for text-guided 3d shape generation. *arXiv preprint arXiv:2209.04145*, 2022. 7

[8] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 7

[9] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020. 3

[10] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 7

[11] Roy Or-El, Xuan Luo, Mengyi Shan, Eli Shechtman, Jeong Joon Park, and Ira Kemelmacher-Shlizerman. Stylesdf: High-resolution 3d-consistent image and geometry generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13503–13513, 2022. 3

[12] 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

[13] Aditya Sanghi, Hang Chu, Joseph G. Lambourne, Ye Wang, Chin-Yi Cheng, Marco Fumero, and Kamal Rahimi Malekshan. Clip-forge: Towards zero-shot text-to-shape generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18603–18613, June 2022. 4

[14] Dong Wook Shu, Sung Woo Park, and Junseok Kwon. 3d point cloud generative adversarial network based on tree structured graph convolutions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019. 4

[15] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5459–5469, June 2022. 3

[16] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog)*, 38(5):1–12, 2019. 4

[17] Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume rendering of neural implicit surfaces. *Advances in Neural Information Processing Systems*, 34:4805–4815, 2021. 3

[18] Xin-Yang Zheng, Yang Liu, Peng-Shuai Wang, and Xin Tong. Sdf-stylegan: Implicit sdf-based stylegan for 3d shape generation. In *Comput. Graph. Forum (SGP)*, 2022. 1, 2