Latent-NeRF for Shape-Guided Generation of 3D Shapes and Textures

Gal Metzer* Elad Richardson* Or Patashnik Raja Giryes Daniel Cohen-Or

Tel Aviv University

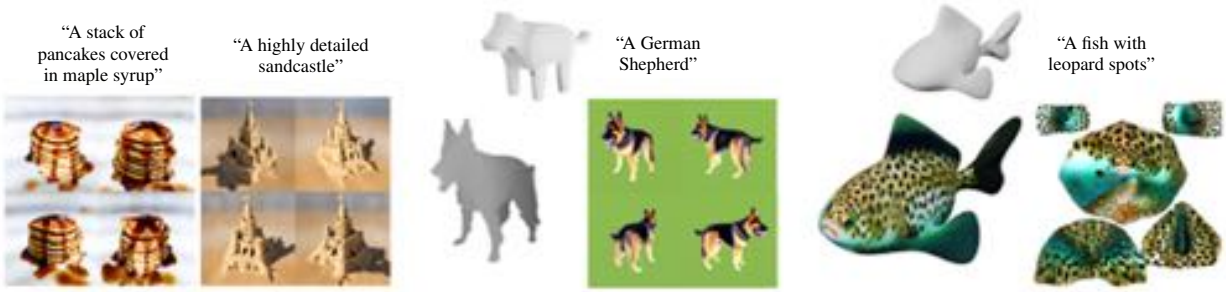

Figure 1. Our three text-guided models: a purely text-guided Latent-NeRF, Latent-NeRF with Sketch-Shape guidance for more exact control over the generated shape, and Latent-Paint for texture generation for explicit shapes. The top row represents the models' inputs.

Abstract

Text-guided image generation has progressed rapidly in recent years, inspiring major breakthroughs in text-guided shape generation. Recently, it has been shown that using score distillation, one can successfully text-guide a NeRF model to generate a 3D object. We adapt the score distillation to the publicly available, and computationally efficient, Latent Diffusion Models, which apply the entire diffusion process in a compact latent space of a pretrained autoencoder. As NeRFs operate in image space, a naïve solution for guiding them with latent score distillation would require encoding to the latent space at each guidance step. Instead, we propose to bring the NeRF to the latent space, resulting in a Latent-NeRF. Analyzing our Latent-NeRF, we show that while Text-to-3D models can generate impressive results, they are inherently unconstrained and may lack the ability to guide or enforce a specific 3D structure. To assist and direct the 3D generation, we propose to guide our Latent-NeRF using a Sketch-Shape: an abstract geometry that defines the coarse structure of the desired object. Then, we present means to integrate such a constraint directly into a Latent-NeRF. This unique combination of text and shape guidance allows for increased control over the generation process. We also show that latent score distillation can be successfully applied directly on 3D meshes. This allows for generating high-quality textures on a given geometry. Our experiments validate the power of our different forms of guidance and the efficiency of using latent rendering.

1. Introduction

Text-guided image generation has seen tremendous success in recent years, primarily due to the breathtaking development in Language-Image models [25, 28, 36] and diffusion models [14, 21, 37–40]. These breakthroughs have also resulted in fast progression for text-guided shape generation [9,29,53]. Most recently, it has been shown [35] that one can directly use score distillation from a 2D diffusion model to guide the generation of a 3D object represented as a Neural Radiance Field (NeRF) [30].

While Text-to-3D can generate impressive results, it is inherently unconstrained and may lack the ability to guide or enforce a 3D structure. In this paper, we show how to introduce shape-guidance to the generation process to guide it toward a specific shape, thus allowing increased control over the generation process. Our method builds upon two models, a NeRF model [30], and a Latent Diffusion Model (LDM) [39]. Latent Models, which apply the entire diffusion process in a compact latent space, have recently gained popularity due to their efficiency and publicly available pretrained checkpoints. As score distillation was previously applied only on RGB diffusion models, we first present two key modifications to the NeRF model that are better paired with guidance from a latent model. First, instead of representing our NeRF in the standard RGB space, we propose a *Latent-NeRF* which operates directly in the latent space of the LDM, thus avoiding the burden of encoding a rendered RGB image to a latent space for each and every guid-

ing step. Secondly, we show that after training, one can easily transform a Latent-NeRF back into a regular NeRF. This allows further refinement in RGB space, where we can also introduce shading constraints or apply further guidance from RGB diffusion models [40]. This is achieved by introducing a learnable linear layer that can be optionally added to a trained Latent-NeRF, where the linear layer is initialized using an approximate mapping between the latent and RGB values [45].

Our first form of shape-guidance is applied using a coarse 3D model, which we call a *Sketch-Shape*. Given a Sketch-Shape, we apply soft constraint during the NeRF optimization process to guide its occupancy based on the given shape. Easily combined with Latent-NeRF optimization, the additional constraint can be tuned to meet a desired level of strictness. Using a Sketch-Shape allows users to define their base geometry, where Latent-NeRF then refines the shape and introduces texture based on a guiding prompt.

We further present *Latent-Paint*, another form of shape-guidance where the generation process is applied directly on a given 3D mesh, and we have not only the structure but also the exact parameterization of the input mesh. This is achieved by representing a texture map in the latent space and propagating the guidance gradients directly to the texture map through the rendered mesh. By doing so, we allow for the first time to colorize a mesh using guidance from a pretrained diffusion model and enjoy its expressiveness.

We evaluate our different forms of guidance under a variety of scenarios and show that together with our latent-based guidance, they offer a compelling solution for constrained shape and texture generation.

2. Related Work

3D Shape Generation 3D shape synthesis is a long-standing problem in computer graphics and computer vision. In recent years, with the emergence of neural networks, the research in 3D modeling has immensely advanced. The most conventional supervision type is applied directly with 3D shapes, through different representations such as implicit functions [10, 20, 33], meshes [19, 50] or point clouds [27, 49]. As 3D supervision is often difficult to obtain, other works use images to guide the generative task [6, 7, 32]. In fact, even when 3D data is available, 2D renderings are sometimes chosen as the supervising primitive [5, 8, 18]. For example, in GET3D [18], two generators are trained, one generates a 3D SDF, and the other a texture field. The output textured mesh is then obtained in a differentiable manner by utilizing DMTet [42]. These generators are adversarially trained with a dataset of 2D images. In [47] a diffusion model has been used to generate multiple views of a given input image. Yet, it has been trained in a supervised manner on a multi-view dataset, unlike our work which does not require a dataset.

Text-to-3D with 2D Supervision Recently, the success of text-guided synthesis in numerous domains [1, 2, 17, 34, 44], has motivated a surge of works that use Language-Image models to guide 3D scenes representations. CLIP-Forge [41] consists of two separate components, an implicit autoencoder conditioned on shape codes, and a normalizing flow model that is trained to generate shape codes according to CLIP embeddings. CLIP-Forge exploits the fact that CLIP has a joint text-image embedding space to train on image embeddings and infer on text embeddings, achieving text-to-shape capabilities. Text2Mesh [29] introduced mesh colorization and geometric fine-tuning by optimizing an initial mesh through differential rendering and CLIP [36] guidance. TANGO [9] follows a similar optimization scheme, while improving results by considering an explicit shading model. CLIP-Mesh [26] optimizes an initial spherical mesh according to a target text prompt, using a modified CLIP loss that accounts for the gap and ambiguity between image/text CLIP embeddings. Similarly to our method, they also use UV texture mapping to bake colors into the mesh. DreamFields [24] employs CLIP guidance as well, but uses NeRFs to represent the 3D object instead of an explicit triangular mesh, together with a dedicated sparsity loss. CLIPNeRF [46] pretrains a disentangled NeRF representation network on rendered object datasets, which is then used to constraint a NeRF scene optimization under CLIP loss, between random renderings of the NeRF and target image or text CLIP embedding. DreamFusion [35] introduced, for the first time, the use of largely successful pretrained 2D diffusion models for text-guided 3D object generation. DreamFusion uses a proprietary 2D diffusion model [40] to supervise the generation of 3D objects represented by NeRFs. To guide a NeRF scene using the pretrained diffusion model, the authors derive a *Score-Distillation* loss, see Section 3.1 for more details.

Neural Rendering The recent rapid progression of neural networks has immensely advanced the performance of differential renderers. Particularly NeRF [4, 30, 31] have shown astounding performance on novel view generation and relighting, also extending to other applications like 3D reconstruction [51]. Thanks to their differentiable nature, it has been recently shown that one can introduce different neural objectives during training to guide the 3D modeling.

3. Method

Here we present our shape-guidance solution. We describe the Latent-NeRF framework, presented in Figure 2, and then introduce different guidance controls that can be combined with Latent-NeRF for controlling its generation. Yet, before showing our solution, we provide a quick overview of two recently proposed techniques that are highly relevant to our method.

3.1. Preliminaries

A **latent diffusion model (LDM)** [39] is a specific form of a diffusion model that is trained to denoise *latent codes* of a pretrained autoencoder, instead of the high-resolution images directly. First, an autoencoder composed of an encoder $\mathcal{E}$, and a decoder $\mathcal{D}$ is trained to reconstruct natural images $x \sim X$, where X is the image training dataset, in the following form: $\tilde{x} = \mathcal{D}(\mathcal{E}(x))$. The autoencoder is trained with a reconstruction loss, perceptual loss [54] and a patch-based adversarial loss [22]. Then, given the trained autoencoder, a denoising diffusion probabilistic model (DDPM) [21] is trained to generate a spatial latent z from noise, according to the distribution $z = \mathcal{E}(x)$ $s.t.$ $x \sim X$. In order to generate a novel image, a latent $\tilde{z}$ is sampled from this learned distribution, using the trained DDPM, and passed to the decoder to obtain the final image $\mathcal{D}(\tilde{z})$. Operating in the latent space requires less compute, and leads to faster training and sampling, which makes LDM widely popular. In fact, the recent Stable Diffusion model is also an LDM.

Score Distillation is a method that enables using a diffusion model as a critic, *i.e.*, using it as a loss without explicitly back-propagating through the diffusion process. It has been introduced in DreamFusion [35] for guiding 3D generation using the Imagen model [40]. To perform score distillation, noise is first added to a given image (e.g., one view of the NeRF's output). Then, the diffusion model is used to predict the added noise from the noised image. Finally, the difference between the predicted and added noises is used for calculating per-pixel gradients. For NeRF, the gradients are back-propagated for updating the 3D NeRF model.

Going into more detail, at each iteration of the score distillation optimization, a rendered image x is noised to a randomly drawn time step t,

$$x_t = \sqrt{\bar{\alpha}_t}x + \sqrt{1-\bar{\alpha}_t}\epsilon, \tag{1}$$

where $\epsilon \sim \mathcal{N}(0, I)$, and $\bar{\alpha}_t$ is a time-dependent constant specified by the diffusion model. Then, the per-pixel score distillation gradients are taken to be

$$\nabla_x L_{SDS} = w(t)(\epsilon_\phi(x_t, t, T) - \epsilon), \tag{2}$$

where ϵ_ϕ is the diffusion model's denoiser (which approximates the noise to be removed), ϕ are the denoiser's parameters, T is an optional guiding text prompt, and $w(t)$ is a constant multiplier that depends on α_t. During training, gradients are propagated from the pixel gradients to the NeRF parameters and gradually change the 3D object. Please refer to [35] for the complete details and derivation of Score Distillation. Note that DreamFusion uses the proprietary Imagen [40] model that is very computationally demanding. We rely on the publicly available Stable Diffusion model and the re-implementation of DreamFusion [43] (it operates in the RGB space and not the latent as we propose).

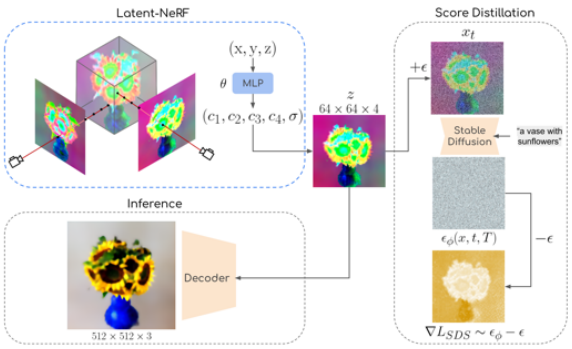

Figure 2. An overview of Latent-NeRF trained with a similar score distillation scheme as proposed by [35]. At each training iteration we render the scene from a random view point to produce a feature map z. Then, z is noised with ϵ according to a random diffusion step t. The noised version of z, *i.e.*, x_t, is denoised using Stable Diffusion [39], with the input text prompt. Finally, the input noise is subtracted from the predicted noise by Stable Diffusion, to approximate per-pixel gradients that are back propagated to the NeRF representation.

3.2. Latent-NeRF

We now turn to describe our Latent-NeRF approach. In this method, a NeRF model is optimized to render 2D feature maps in Stable Diffusion's latent space $\mathcal{Z}$. Latent-NeRF outputs four pseudo-color channels, (c_1, c_2, c_3, c_4), corresponding to the four latent features that stable diffusion operates over, and a volume density σ. Figure 2 illustrates this process. Representing the scene using NeRF implicitly imposes spatial consistency between different views, due to the spatial radiance field and rendering equation. Still, the fact that $\mathcal{Z}$ **can** be represented by a NeRF with spatial consistencies is non-trivial. Previous works [1, 45] showed that super-pixels in $\mathcal{Z}$ depend mainly on individual patches in the output image. This can be attributed to the high resolution (64×64) and low channel-wise depth (4) of this latent space, which encourages local dependency over the autoencoder's image and latent spaces. Assuming $\mathcal{Z}$ is a near patch level representation of its corresponding RGB image makes the latents nearly equivariant to spatial transformations of the scene, which justifies the use of NeRFs for representing the 3D scenes.

Text Guidance The vanilla form of Latent-NeRF is text-guided, with no other constrains for the scene generation. In this setting, we employ the following loss:

$$L = \lambda_{SDS} L_{SDS} + \lambda_{sparse} L_{sparse},$$

where L_{SDS} is the Score-Distillation loss depicted in Figure 2. Note, that the exact value of this loss is not accessible. Instead, the gradients implied by it are approximated

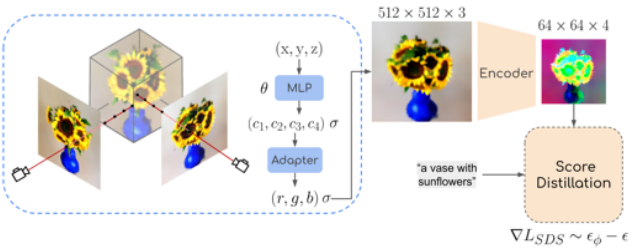

Figure 3. To perform fine-tuning in RGB, we take a Latent-NeRF model that was trained in **latent space** (conventional setting), apply the matrix adapter to the channel output of the model to get an RGB preview, and then continue optimizing the MLP's weights through supervision in RGB to improve the result.

by a single forward pass through the denoiser. These gradients are directly passed to the autograd solver. The loss $L_{sparse} = BE(w_{blend})$ suggested in [43] prevents floating "radiance clouds" by penalizing the binary entropy of ill-defined background masks w_{blend}. Namely, it encourages a strict blending of the object NeRF and background NeRF.

RGB Refinement Using Latent-NeRF, one may successfully learn to represent 3D scenes even when optimizing solely in latent space. Still, in some cases, it could be beneficial to further refine the model by fine-tuning it in pixel space, and have the NeRF model operate directly in RGB. To do so, we must convert the NeRF that was trained in latent space to a NeRF that operates in RGB. This requires converting the MLP's output from the four latent channels to three RGB channels such that the initial rendered RGB image is close to the decoder output, when applied to the rendered latent of the original model. Interestingly, it has been shown [45] that a linear approximation is sufficient to predict plausible RGB colors given a single four-channel latent super pixel, via the following transformation

$$\begin{pmatrix}\hat{r}\\ \hat{g}\\ \hat{b}\end{pmatrix} = \begin{pmatrix}0.298 & 0.187 & -0.158 & -0.184\\ 0.207 & 0.286 & 0.189 & -0.271\\ 0.208 & 0.173 & 0.264 & -0.473\end{pmatrix}\begin{pmatrix}c_1\\ c_2\\ c_3\\ c_4\end{pmatrix}, \tag{3}$$

which was calculated using pairs of RGB images and their corresponding latent codes over a collection of natural images. As our NeRF model is already composed of a set of fully connected layers, we simply add another linear layer that is initialized using the weights in Equation 3. This converts our Latent-NeRF to operate in pixel space and ensures that our refinement process starts from a valid result. The additional layer is then fine-tuned together with the rest of the model, to create the refined and final output. The overall fine-tuning procedure is illustrated in Figure 3.

3.3. Sketch-Shape Guidance

Next, we introduce a novel technique for guiding the Latent-NeRF generation based on a coarse geometry, which we call a Sketch-Shape.

A Sketch-Shape is an abstract coarse alignment of simple 3D primitives like spheres, boxes, cylinders, etc., that together depict an outline of a more complex object. Figures 9, 10, 11 illustrate such simple shapes. Ideally, we would like the output density of our MLP to match that of the Sketch-Shape, such that the output Latent-NeRF result resembles the input shape. Nevertheless, we would also like the new NeRF to have the capacity to create new details and geometries that match the input text prompt and improve the fidelity of the shape. To achieve this lenient constraint, we encourage the NeRF's occupancy to match the winding-number [3, 23] indicator of the Sketch-Shape, but with decaying importance near the surface to allow new geometries. This loss reads as

$$L_{Sketch-Shape} = CE(\alpha_{NeRF}(p), \alpha_{GT}(p)) \cdot (1 - e^{-\frac{d^2}{2\sigma_S}}). \tag{4}$$

This loss implies that the occupancy should be well constrained away from the surface, and free to be set by score distillation near the surface. This loss is applied in addition to the Latent-NeRF loss, over the entire point set p that is used by the NeRF's volumetric rendering. d represents the distance of p from the surface, and σ_S is a hyperparameter that controls how lenient the loss is, *i.e.*, lower σ_S values imply a tighter constraint to the input Sketch-Shape. Applying the loss only on the sampled point-set p, makes it more efficient as these points are already evaluated as part

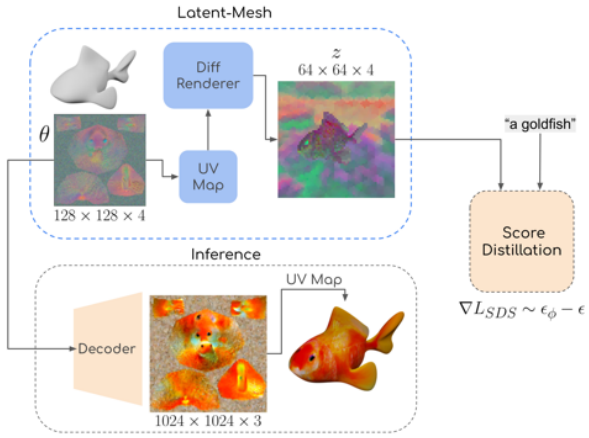

Figure 4. An overview of Latent-Paint texture generation, trained with a similar score distillation loss [35]. The object is represented as a textured mesh, with a latent texture image that has four pseudo-color channels. At each iteration, the mesh is rendered with a differential renderer, to create a feature map from a random viewpoint. Score distillation is applied to the rendered feature map, and the gradients back-propagate to the latent texture image through the differential renderer. See Section 3.4.

of the Latent-NeRF rendering process.

3.4. Latent-Paint of Explicit Shapes

We now move to a more strict constraint, where the guidance is based on an exact structure of a given shape, e.g., provided in the form of a mesh. We call this approach Latent-Paint, which leads to the generation of novel textures for a given shape. Our method generates texture over a UV texture map, which can either be supplied by the input mesh, or calculated on-the-fly using XAtlas [52]. To color a mesh, we first initialize a random latent texture image of size $H \times W \times 4$, where H and W can be chosen according to the desired texture granularity. We set them both to be 128 in our experiments.

Figure 4 presents the training process. At each score distillation iteration, we render the mesh with a differentiable renderer [15] to obtain a $64 \times 64 \times 4$ feature map that is *pseudo-colored* by the latent texture image. Then, we apply the score distillation loss from Equation 2 in the same way it is applied for Latent-NeRF. Yet, instead of back-propagating the loss to the NeRF's MLP parameters, we optimize the deep texture image by back-propagating through the differentiable renderer. To get the final RGB texture image, we simply pass the **latent texture** image through Stable Diffusion's decoder $\mathcal{D}$ once, to get a larger high-quality RGB texture.

4. Evaluation

We now validate the effectiveness of our different forms of guidance through a variety of experiments.

Implementation Details We use the Stable Diffusion implementation by HuggingFace Diffusers, with the `v1-4` checkpoint. For score distillation, we use the code-base provided by [43], with Instant-NGP [31] as our NeRF model.

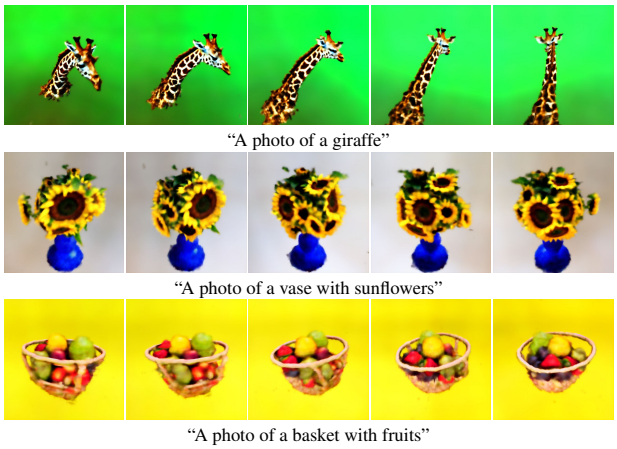

Figure 5. Latent-NeRF results from different viewpoints.

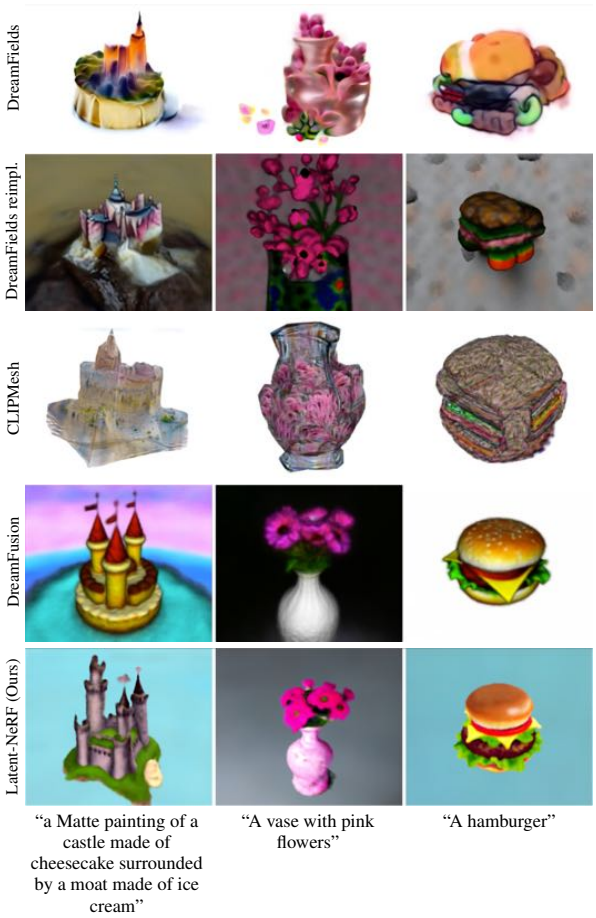

Figure 6. Qualitative comparison with other text-to-3D methods.

Latent-NeRF usually takes less than 15 minutes to converge on a single V100, while using an RGB-NeRF with Stable Diffusion [43] takes about 30 minutes, due to the increased overhead from encoding into the latent space. Note that DreamFusion [35] takes about 1.5 hours on 4 TPUs. This clearly shows the computational advantage of Latent-NeRF.

4.1. Text-Guided Generation

Qualitative Results We begin by demonstrating the effectiveness of the latent rendering approach with Latent-NeRF. In Figs. 1, 5 and 7, we show several results obtained by our method. In the supplementary material we provide additional results of different objects, including video visualizations. In Figure 5, we show the consistency of our learned shapes from several viewpoints. Next, we use the baseline set by DreamFusion [35] to qualitatively compare our approach (with the proposed RGB refinement) against other methods. As can be seen in Figure 6, Latent-NeRF achieves significantly better results than DreamFields [24] and CLIPMesh [26]. We believe that the better quality of DreamFusion can be attributed to the high quality of Ima-

Figure 7. RGB refinement combined with (i) Latent-NeRF (ice-cream and temple), and (ii) Sketch-Shape (lego and car). For each shape we also show the normals direction.

gen [40], but unfortunately, we cannot validate this as the model is not publicly available to the community.

RGB Refinement Figure 7 shows the quality improvement achieved by our RGB refinement method. It reveals that RGB refinement is mostly useful for complex objects or for regions with detailed textures. Refinement iterations in the RGB space are about $\times 2$ slower than iterations in latent space, thus, increasing the runtime to more than 30 minutes. Thanks to our linear mapping from a Latent-NeRF to an RGB-NeRF, practitioners may apply the refinement method only after the 3D shape has already converged with the more efficient latent training. This allows a fast exploration of 3D shapes, and an optional polishing step with RGB refinement.

Textual-Inversion As our Latent-NeRF is supervised by Stable-Diffusion, we can also use *Textual Inversion* [16] tokens as part of the input text prompt. This allows conditioning the object generation on specific objects and styles, defined only by input images. Results using *Textual Inversion* are presented in Figure 8.

4.2. Sketch-Shape Guidance

Figure 9 shows different Sketch-Shape results with the same conditioning mesh. The different text prompts are able to guide the shape toward refined geometries that better match the text prompt. The rough Sketch-Shape in this

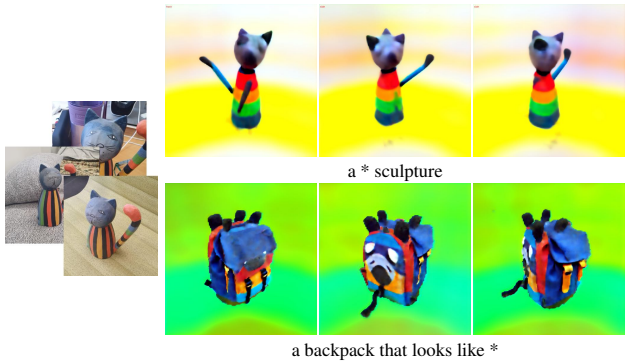

Figure 8. Results of Latent-NeRF using a token learned from the images on the left with Textual Inversion [16].

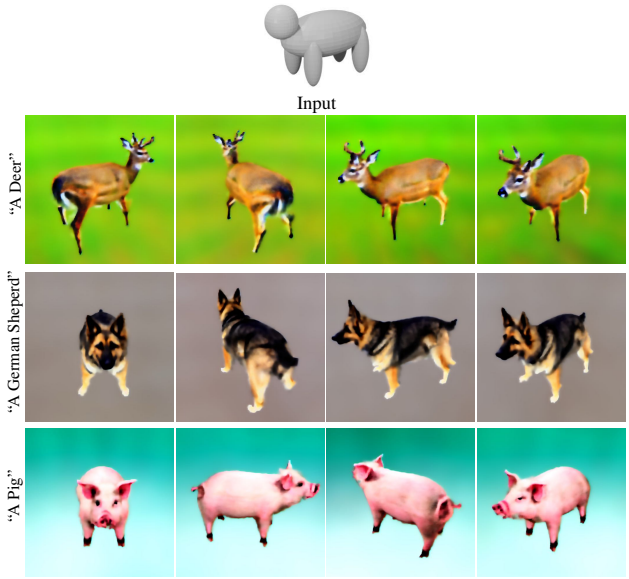

Figure 9. Sketch-Shape results using different prompts. This examples shows how the same Sketch-Shape can influence different objects according to different text prompts.

figure, was quickly designed in Blender [12] and allows us to easily define a coarse shape that guides the Latent-NeRF. Moreover, Figure 17 depicts an ablation over the lenient parameter σ_S from Eq. (4). When σ_S is set to 0.05, the generated mesh takes the form of the conditioning shape (shown in Figure 9). As σ_S grows, more details are added on top of the base shape, until little to no resemblance is observed at $\sigma_S = 1.5$. Figure 10 contains additional results of different shapes generated with the same conditioning mesh, here a coarse house shape. The normals visualization (bottom row) shows that our method can add fine geometric details.

Figure 11 demonstrates that our proposed approach can successfully work with a variety of different Sketch-Shapes. Notice that our method handles a variety of different shapes and also works well with shapes extruded from 2D sketches.

Figure 10. Sketch-Shape results, conditioned on a low-poly house shape. Our RGB Refinement was applied to improve detail quality.

We also exhibit in Figure 14 the effectiveness of shape-guidance, by showing results of the same prompts with and without the shape loss.

4.3. Latent-Paint Generation

We tested Latent-Paint on a variety of input shapes shown in Figs. 12 and 13. As all of the shapes in these figures do not contain precomputed UV parameterization, we use XAtlas [52] to compute such parameterization automatically. In contrast, the fish mesh in Figure 15 (obtained from [13]) already contains high quality UV parameterization, which we are also able work with. Figure 13 compares our Latent-Paint approach to two closely related methods, Tango [9] and CLIPMesh [26]. As can be seen, our approach achieves more precise textures thanks to the guidance from the diffusion model.

Note that Latent-Paint can work without assuming or computing any UV-map by simply optimizing per-face latent attributes. Yet, we found it better to use a UV-map for two main reasons: (i) The UV map makes the texture granularity independent of the geometric resolution, *i.e.*, coarse geometries do not imply course colorization; and (ii) texture maps are easier to use with downstream applications like MeshLab [11] and Blender [12].

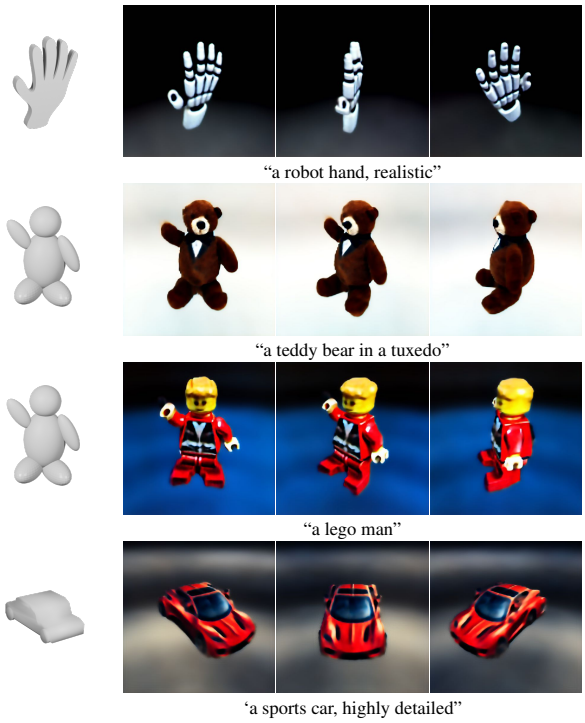

Figure 11. Additional Sketch-Shape-guided results.

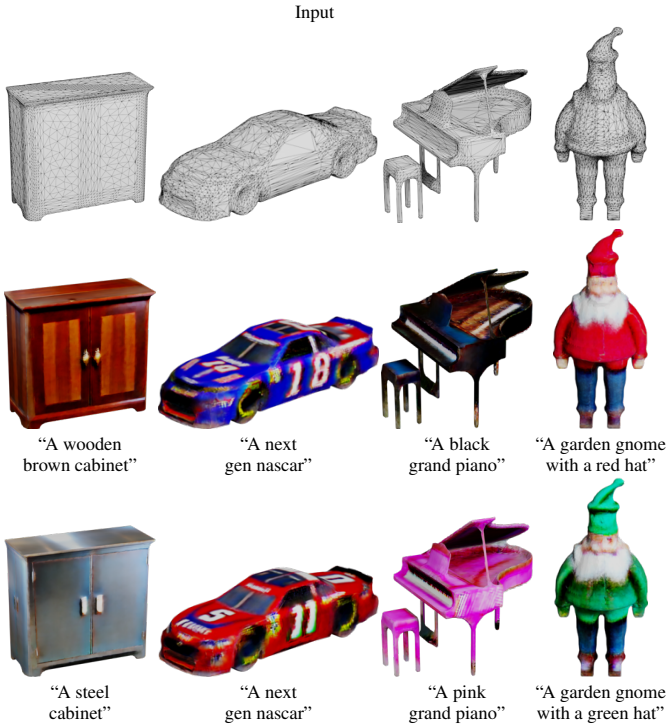

Figure 12. *Latent-Paint* results on shapes from ModelNet40 [48]. UV parameterization was not given for any of the meshes in this figure. For the Nascar shapes, the same text prompt was used with a different initialization.

Figure 13. Generating different shoe textures over the same input mesh by only changing the conditioning text prompt. The surface parameterization was computed using XAtlas [52].

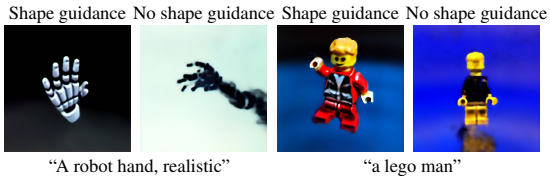

Figure 14. Results for the same prompts with and without shape-guidance. The guiding shapes are the same as in Figure 11.

Figure 15. *Latent-Paint* results for an input mesh containing pre-computed UV parameterization. Model courtesy of "Keenan's 3D Model Repository" [13].

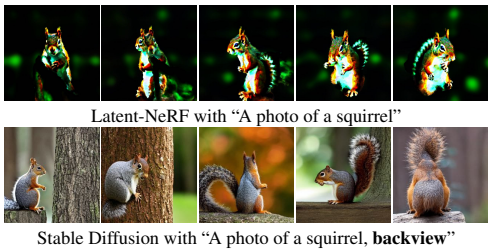

Figure 16. note that the generated squirrel in the first row contains two faces from some view directions; see the second to the left view. This problem may be attributed to the fact that Stable Diffusion fails to generate back views of a squirrel; see bottom row.

5. Limitations

Our presented technique is yet a preliminary step towards the challenging goal of a comprehensive text-to-shape model that uses no 3D supervision. Still, the proposed latent framework has its limitations. To attain a plausible 3D shape, we use the same "prompt tweaking" used by DreamFusion [35], *i.e.*, adding a directional text-prompt (e.g., "front", "side" with respect to the camera) to the input text prompt. We find that this assistance tends to fail with our approach when applied to certain objects. Moreover, we find that even Stable Diffusion tends to generate unsatisfactory images when specifying the desired direction as shown in Figure 16. Additionally, similar to most works that employ diffusion models, there is a stochastic behavior to the results, such that the quality of the results may significantly vary between different seeds.

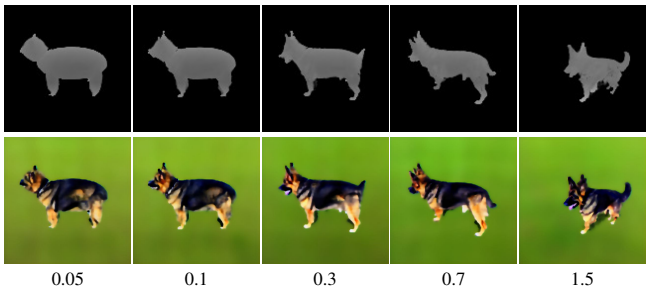

Figure 17. Sketch-Shape ablation over σ_S values (Eq. (4)) and input shape from Figure 9. As σ_S grows, the constraint becomes more lenient, and enables the shape to evolve into new geometries.

6. Conclusions

In this work, we introduced a latent framework for generating 3D shapes and textures using different forms of text and shape guidance. We first adapted the score distillation loss for LDMs, enabling the use of recent, powerful and publicly available text-to-image generation models for 3D shape generation. Successfully applying score distillation on LDMs results in a fast and flexible object generation framework. We then introduced shape-guided control on the generated model. We showed two versions of shape-guided generation, Sketch-Shape and Latent-Paint, and demonstrated their effectiveness for providing additional control over the generation process.

Typically, the notion of rendering refers to generating an image in pixel space. Here, we have presented a method that renders directly into the latent space of a neural model. We believe that our Latent-NeRF approach opens the avenue for more latent space rendering methods, which can gain from a compact and effective latent representation, and advance the use of neural models that operate in latent space rather than pixel space. Furthermore, our novel approach and its ease-of-use nature would encourage further research toward effective text-guided shape generation.

This work was supported by the European research council (ERC-StG 757497 PI Giryes), and the Israel Science Foundation (grants no. 2492/20 and 3441/21).

References

[1] Omri Avrahami, Ohad Fried, and Dani Lischinski. Blended latent diffusion. *arXiv preprint arXiv:2206.02779*, 2022. 2, 3

[2] Omer Bar-Tal, Dolev Ofri-Amar, Rafail Fridman, Yoni Kasten, and Tali Dekel. Text2live: Text-driven layered image and video editing. *arXiv preprint arXiv:2204.02491*, 2022. 2

[3] Gavin Barill, Neil G Dickson, Ryan Schmidt, David IW Levin, and Alec Jacobson. Fast winding numbers for soups and clouds. *ACM Transactions on Graphics (TOG)*, 37(4):1–12, 2018. 4

[4] Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5855–5864, 2021. 2

[5] Miguel Angel Bautista, Pengsheng Guo, Samira Abnar, Walter Talbott, Alexander Toshev, Zhuoyuan Chen, Laurent Dinh, Shuangfei Zhai, Hanlin Goh, Daniel Ulbricht, Afshin Dehghan, and Josh Susskind. Gaudi: A neural architect for immersive 3d scene generation. *arXiv*, 2022. 2

[6] Eric Chan, Marco Monteiro, Petr Kellnhofer, Jiajun Wu, and Gordon Wetzstein. pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5795–5805, 2021. 2

[7] Eric R Chan, Connor Z Lin, Matthew A Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas J Guibas, Jonathan Tremblay, Sameh Khamis, et al. Efficient geometry-aware 3d generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16123–16133, 2022. 2

[8] Xuelin Chen, Daniel Cohen-Or, Baoquan Chen, and Niloy J. Mitra. Towards a neural graphics pipeline for controllable image generation. *Computer Graphics Forum*, 40(2), 2021. 2

[9] Yongwei Chen, Rui Chen, Jiabao Lei, Yabin Zhang, and Kui Jia. Tango: Text-driven photorealistic and robust 3d stylization via lighting decomposition. *arXiv preprint arXiv:2210.11277*, 2022. 1, 2, 7, 8

[10] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling, 2018. 2

[11] Paolo Cignoni, Massimiliano Corsini, and Guido Ranzuglia. Meshlab: an open-source 3d mesh processing system. *ERCIM News*, 2008(73), 2008. 7

[12] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. 6, 7

[13] Keenan Crane. Keenan's 3d model repository. https://www.cs.cmu.edu/ kmcrane/Projects/ModelRepository/, 2022. 7, 8

[14] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems*, 34:8780–8794, 2021. 1

[15] Clement Fuji Tsang, Maria Shugrina, Jean Francois Lafleche, Towaki Takikawa, Jiehan Wang, Charles Loop, Wenzheng Chen, Krishna Murthy Jatavallabhula, Edward Smith, Artem Rozantsev, Or Perel, Tianchang Shen, Jun Gao, Sanja Fidler, Gavriel State, Jason Gorski, Tommy Xiang, Jianing Li, Michael Li, and Rev Lebaredian. Kaolin: A pytorch library for accelerating 3d deep learning research. https://github.com/NVIDIAGameWorks/kaolin, 2022. 5

[16] Rinon Gal, Yuval Alaluf, Yuval Atzmon, Or Patashnik, Amit H Bermano, Gal Chechik, and Daniel Cohen-Or. An image is worth one word: Personalizing text-to-image generation using textual inversion. *arXiv preprint arXiv:2208.01618*, 2022. 6

[17] Rinon Gal, Or Patashnik, Haggai Maron, Amit H Bermano, Gal Chechik, and Daniel Cohen-Or. Stylegan-nada: Clip-guided domain adaptation of image generators. *ACM Transactions on Graphics (TOG)*, 41(4):1–13, 2022. 2

[18] Jun Gao, Tianchang Shen, Zian Wang, Wenzheng Chen, Kangxue Yin, Daiqing Li, Or Litany, Zan Gojcic, and Sanja Fidler. Get3d: A generative model of high quality 3d textured shapes learned from images. *arXiv preprint arXiv:2209.11163*, 2022. 2

[19] Lin Gao, Jie Yang, Tong Wu, Yu-Jie Yuan, Hongbo Fu, Yu-Kun Lai, and Hao Zhang. Sdm-net: Deep generative network for structured deformable mesh. *ACM Transactions on Graphics (TOG)*, 38(6):1–15, 2019. 2

[20] Amir Hertz, Or Perel, Raja Giryes, Olga Sorkine-Hornung, and Daniel Cohen-Or. Spaghetti: Editing implicit shapes through part aware generation. *arXiv preprint arXiv:2201.13168*, 2022. 2

[21] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020. 1, 3

[22] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017. 3

[23] Alec Jacobson, Ladislav Kavan, and Olga Sorkine-Hornung. Robust inside-outside segmentation using generalized winding numbers. *ACM Transactions on Graphics (TOG)*, 32(4):1–12, 2013. 4

[24] Ajay Jain, Ben Mildenhall, Jonathan T. Barron, Pieter Abbeel, and Ben Poole. Zero-shot text-guided object generation with dream fields. 2022. 2, 5

[25] Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc Le, Yun-Hsuan Sung, Zhen Li, and Tom Duerig. Scaling up visual and vision-language representation learning with noisy text supervision. In *International Conference on Machine Learning*, pages 4904–4916. PMLR, 2021. 1

[26] Nasir Mohammad Khalid, Tianhao Xie, Eugene Belilovsky, and Popa Tiberiu. Clip-mesh: Generating textured meshes from text using pretrained image-text models. *SIGGRAPH Asia 2022 Conference Papers*, 2022. 2, 5, 7, 8

[27] Chun-Liang Li, Manzil Zaheer, Yang Zhang, Barnabas Poczos, and Ruslan Salakhutdinov. Point cloud gan. *arXiv preprint arXiv:1810.05795*, 2018. 2

[28] Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. *arXiv preprint arXiv:2201.12086*, 2022. 1

[29] Oscar Michel, Roi Bar-On, Richard Liu, Sagie Benaim, and Rana Hanocka. Text2mesh: Text-driven neural stylization for meshes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13492–13502, 2022. 1, 2

[30] 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. *Communications of the ACM*, 65(1):99–106, 2021. 1, 2

[31] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4):102:1–102:15, July 2022. 2, 5

[32] Michael Niemeyer and Andreas Geiger. Campari: Camera-aware decomposed generative neural radiance fields. *2021 International Conference on 3D Vision (3DV)*, pages 951–961, 2021. 2

[33] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 165–174, 2019. 2

[34] Or Patashnik, Zongze Wu, Eli Shechtman, Daniel Cohen-Or, and Dani Lischinski. StyleCLIP: Text-driven manipulation of StyleGAN imagery. *arXiv preprint arXiv:2103.17249*, 2021. 2

[35] Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv preprint arXiv:2209.14988*, 2022. 1, 2, 3, 4, 5, 8

[36] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR, 2021. 1, 2

[37] 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

[38] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. *arXiv:2102.12092*, 2021. 1

[39] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2021. 1, 3

[40] 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, Tim Salimans, Jonathan Ho, David J Fleet, and Mohammad Norouzi. Photorealistic text-to-image diffusion models with deep language understanding, 2022. 1, 2, 3, 6

[41] Aditya Sanghi, Hang Chu, Joseph G Lambourne, Ye Wang, Chin-Yi Cheng, and Marco Fumero. Clip-forge: Towards zero-shot text-to-shape generation. *arXiv preprint arXiv:2110.02624*, 2021. 2

[42] Tianchang Shen, Jun Gao, Kangxue Yin, Ming-Yu Liu, and Sanja Fidler. Deep marching tetrahedra: a hybrid representation for high-resolution 3d shape synthesis. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. 2

[43] Jiaxiang Tang. A pytorch implementation of the text-to-3d model dreamfusion, powered by the stable diffusion text-to-2d model. https://github.com/ashawkey/stable-dreamfusion, 2022. 3, 4, 5

[44] Guy Tevet, Brian Gordon, Amir Hertz, Amit H Bermano, and Daniel Cohen-Or. Motionclip: Exposing human motion generation to clip space. *arXiv preprint arXiv:2203.08063*, 2022. 2

[45] Kevin Turner. Decoding latents to rgb without upscaling. https://discuss.huggingface.co/t/decoding-latents-to-rgb-without-upscaling/23204, 2022. 2, 3, 4

[46] Can Wang, Menglei Chai, Mingming He, Dongdong Chen, and Jing Liao. Clip-nerf: Text-and-image driven manipulation of neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3835–3844, 2022. 2

[47] Daniel Watson, William Chan, Ricardo Martin-Brualla, Jonathan Ho, Andrea Tagliasacchi, and Mohammad Norouzi. Novel view synthesis with diffusion models. *arXiv:2210.04628*, 2022. 2

[48] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015. 7

[49] Guandao Yang, Xun Huang, Zekun Hao, Ming-Yu Liu, Serge Belongie, and Bharath Hariharan. Pointflow: 3d point cloud generation with continuous normalizing flows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4541–4550, 2019. 2

[50] Jie Yang, Kaichun Mo, Yu-Kun Lai, Leonidas J Guibas, and Lin Gao. Dsg-net: Learning disentangled structure and geometry for 3d shape generation. *ACM Transactions on Graphics (TOG)*, 2022. 2

[51] Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume rendering of neural implicit surfaces. In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021. 2

[52] Jonathan Young. Xatlas: Mesh parameterization / uv unwrapping library. https://github.com/jpcy/xatlas, 2022. 5, 7, 8

[53] Xiaohui Zeng, Arash Vahdat, Francis Williams, Zan Gojcic, Or Litany, Sanja Fidler, and Karsten Kreis. Lion: Latent point diffusion models for 3d shape generation. *arXiv preprint arXiv:2210.06978*, 2022. 1

[54] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018. 3