

# Breathing New Life into 3D Assets with Generative Repainting

Tianfu Wang<sup>1</sup> Menelaos Kanakis<sup>1</sup> Konrad Schindler<sup>2</sup> Luc Van Gool<sup>1,3,4</sup> Anton Obukhov<sup>1→2</sup>  
<sup>1</sup>ETH Zürich, Computer Vision Laboratory <sup>2</sup>ETH Zürich, Photogrammetry and Remote Sensing <sup>3</sup>KU Leuven <sup>4</sup>INSAIT, Sofia

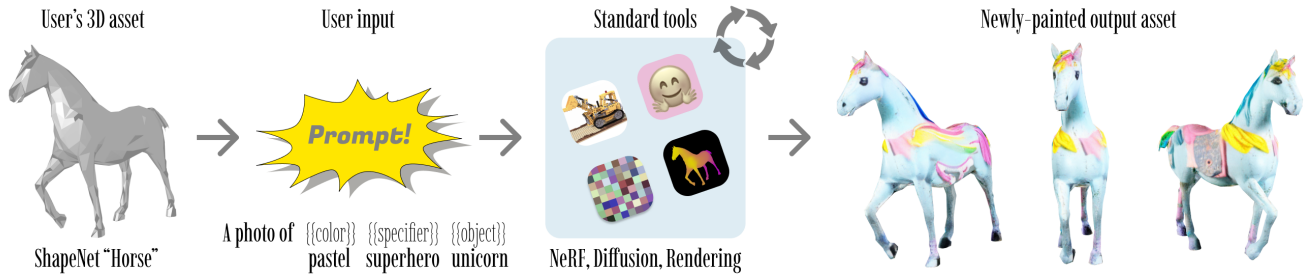


Figure 1. **We present a pipeline for text-guided painting of legacy geometry.** We leverage rich pretrained generative 2D diffusion models to give a fresh look to existing 3D assets, and neural radiance fields to enforce 3D consistency and overcome issues of the legacy representations. Starting from an input geometry and the desired output description, our pipeline orchestrates calls to several generative and modality conversion tools to breathe new life into the input assets. The tools communicate using images instead of gradients with each other, making our pipeline interpretable and amenable to partial upgrades. Project page: [https://www.obukhov.ai/repainting\\_3d\\_assets](https://www.obukhov.ai/repainting_3d_assets).

## Abstract

*Diffusion-based text-to-image models ignited immense attention from the vision community, artists, and content creators. Broad adoption of these models is due to significant improvement in the quality of generations and efficient conditioning on various modalities, not just text. However, lifting the rich generative priors of these 2D models into 3D is challenging. Recent works have proposed various pipelines powered by the entanglement of diffusion models and neural fields. We explore the power of pretrained 2D diffusion models and standard 3D neural radiance fields as independent, standalone tools and demonstrate their ability to work together in a non-learned fashion. Such modularity has the intrinsic advantage of eased partial upgrades, which became an important property in such a fast-paced domain. Our pipeline accepts any legacy renderable geometry, such as textured or untextured meshes, orchestrates the interaction between 2D generative refinement and 3D consistency enforcement tools, and outputs a painted input geometry in several formats. We conduct a large-scale study on a wide range of objects and categories from the ShapeNetSem dataset and demonstrate the advantages of our approach, both qualitatively and quantitatively.*

## 1. Introduction

Creating high-quality 3D assets based on textual descriptions for a diverse range of objects is an endeavor with great potential for digital media and artists. Recently, there has been a rise in denoising diffusion-based (DDPM) [14] text-to-image models [30, 32] producing results of unprecedented quality. The generative power of these 2D image models prompts the question: Can we use them to generate multi-view consistent 3D content? As it turns out, lifting these rich generative priors to 3D is a non-trivial task. In this work, we focus on the problem of text- and geometry-conditioned painting, an adjacent problem of text-to-3D generation.

The overview of our pipeline for generating a diverse multi-view consistent painting from a text description and input geometry is presented in Fig. 1. We bootstrap our pipeline from two crucial components: a pretrained generative text- and depth-conditioned image diffusion model [30] and neural radiance fields (NeRF) [19]. The design of our pipeline separates these components into distinct processes, which communicate using the interface of image files. This is contrary to several recent approaches that rely on gradient flow between the components, either in the form of Score Distillation [18, 26], or differentiable rendering [29]. We rely on traditional rendering techniques to enable communication between the components, including Z-buffer extraction for the rendered views. The image file interface is naturally

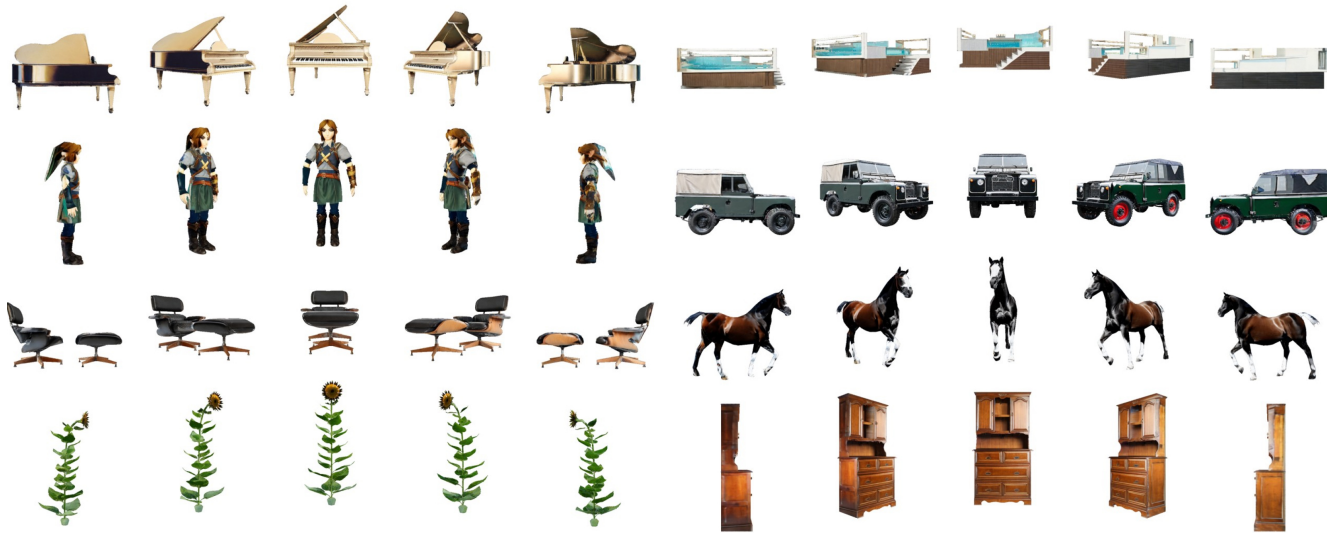


Figure 2. **Texturing the ShapeNetSem [6] dataset with the proposed method.** We discard the original texture and paint objects with our method using the dataset metadata “name” field as a text prompt. We show several objects from 5 views spaced with 45-degree increments around the vertical axis. Our method produces high-quality results from the input text and geometry. More visual results in Figs. 6, 7.

interpretable and better suited for building modular and partially upgradable systems. This is especially important as both DDPM and NeRF research fields advance rapidly.

Prior generative 3D works often employ the UV texture unwrapping [16], a costly operation and a potential point of failure. Since our method requires only Z-buffer queries from the input geometry, the input does not necessarily have to have a UV texture map attached or even be a valid mesh. To support this claim, we experimented with Point-E [21], thus extending our pipeline to a pure text-to-3D setting.

The output of our pipeline is a NeRF corresponding to the input geometry, painted in a multi-view consistent manner. The NeRF can be converted into the explicit input format with extra coloring information.

Our pipeline’s performance depends on each component’s performance, so it will keep improving as the components get faster. For example, recent progress in DDPMs [30] led to a tenfold decrease in image generation time; NeRF research has seen similar speedups [20]. Capitalizing on that, we conduct a large-scale study of painting the ShapeNetSem [6] dataset, composed of 12K objects from over 270 categories (Fig. 2). The study shows that our pipeline sets new state-of-the-art results on several generative metrics while attaining proper 3D consistency.

The summary of our contributions is as follows:

- We introduce a novel approach for giving 3D assets a new life, by painting their geometry using text inputs and pretrained generative image diffusion models.
- Our method is unique in that it combines pretrained 2D diffusion models and 3D neural radiance fields as

*standalone* pipelines. The weak coupling of tools is achieved through the interpretable interface of image files and permits partial upgrades.

- We conduct a large-scale study of painting ShapeNetSem [6] dataset and attain the new state-of-the-art on several metrics and perceived 3D consistency.
- Our method is robust to input corruptions and produces the output assets in several formats.

## 2. Related Work

**Generative Text-to-Image Models** Until recently, generative imaging was dominated by unconditional or few-classes-conditional models [5, 12, 35]. With advancements in natural language processing, Contrastive Language-Image Pretraining (CLIP) [27] bridged the gap between visual and text modalities. This opened an avenue for open-category and text-conditioned image generation. Currently, Denoising Diffusion Probabilistic Models (DDPM) [14, 32] dominate the niche of high-quality and affordable text-conditioned generative imaging. Stable Diffusion [30] proposed shifting the diffusion process to a low dimensional latent space, achieving competitive performance while reducing the computation requirements. Subsequent models could further condition the process on various modalities, such as depth maps, images, and inpainting masks. These new modalities and accessible pretrained checkpoints gave rise to new applications of diffusion models, such as image un-cropping [31] and perpetual view generation [4]. Likewise, our method

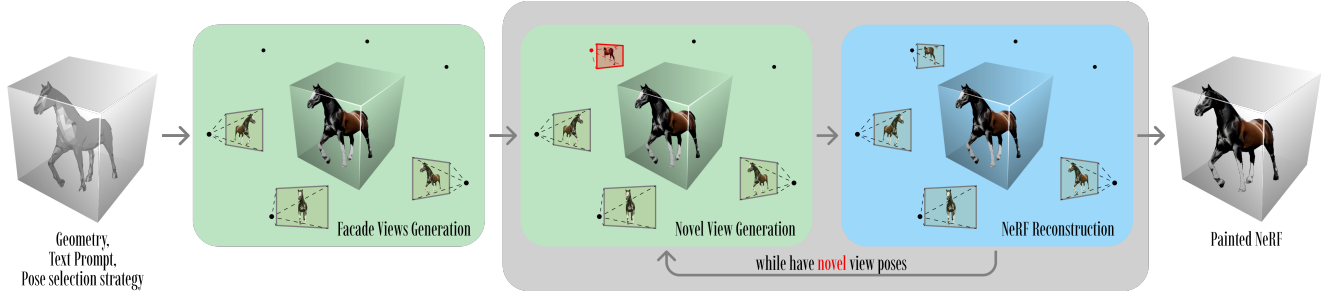


Figure 3. **Geometry painting pipeline that takes the geometry, a text prompt, and outputs a painted NeRF of the model.** We utilize the diffusion image generation process and the 3D reconstruction process of NeRF as standalone procedures. We start by generating the facade views using only diffusion view generation. Our pipeline progressively builds the 3D model by using NeRF to generate view-consistent images and feeding them back to the diffusion process to generate a new input view according to the view selection strategy.

relies on standalone pretrained DDPMs with their various ways of conditioning.

**Neural Radiance Fields** Neural scene representations gained popularity due to their simplicity of usage and ability to capture complex scenes efficiently. Neural Radiance Fields (NeRF) [19] have recently demonstrated their versatility as a solution for 3D reconstruction from posed images. Recently, numerous improvements and variants of NeRF have been developed [7, 20, 24]. In particular, Instant NGP [20] proposed an efficient multi-resolution hash-based grid data structure, which reduces the training time of NeRF from hours to minutes. Similarly to COLMAP [33] for structure from motion, Instant NGP has become the go-to standalone tool for images to NeRF conversion.

**Generative 3D Models** Research on high-quality 3D models and assets generation gained a lot of interest recently [11, 26, 34, 37, 38]. Previous methods leveraged Generative Adversarial Networks (GANs) [12] coupled with 3D-aware learned pipelines, such as differentiable renderers [11], face convolutional neural networks (CNNs) [34], voxel grids [38], and NeRFs [5, 39]. However, most of the methods require training a separate model per category, and thus, the evaluation focuses on a handful of classes, typically “cars” and “chairs”, such as seen in ShapeNet [6]. With the rise of popularity in diffusion models and accessible text conditioning, recent works focused on integrating them into 3D content generation pipelines [26, 37]. DreamFusion [26] proposed score distillation sampling to couple a pretrained text-to-image diffusion model with a NeRF module to form an end-to-end trainable pipeline. Although score distillation cleverly avoids backpropagation through the diffusion model, thus reducing computational costs, it still requires significant computations. Pipelines with surrogate 3D output [21] have also received attention. Mesh-based inpainting schemes such as Latent-Paint [18] and TEXTure [29] employ differentiable rendering to generate a texture image for the input mesh. However, these methods are susceptible to artifacts

introduced during UV texture unwrapping and gradient interaction between the generative model and the texturing target. Another two relevant works appeared recently: Text2Tex [8] utilizes a mesh-based inpainting scheme similar to TEXTure [29]; TextMesh [8] combines NeRF with SDS loss akin to DreamFusion [26]. Our method overcomes the discussed limitations by using NeRF for both scene representation and iterative consistency enforcement.

### 3. Method

The pipeline of our method is outlined in Fig. 3. It takes an input geometry and a text description and generates a NeRF model that adheres to the structure of input geometry but is enhanced with text-guided painting. It paints the geometry progressively: starting from the object facade initialization, it iteratively picks a novel view according to the camera pose selection strategy, generates a novel view, and reconciles it with the previous views using NeRF.

**Prerequisites and Assumptions** Our pipeline is object-centric; hence, we create a virtual scene with the object scaled and positioned in the origin and a camera positioned on a unit sphere, pointing at the origin. We additionally assume that the object surface is opaque, which is required to perform unambiguous queries of the renderer’s Z-buffer. This constraint limits processing models with transparency or with large sprite surfaces (e.g., for trees or flowers), sometimes seen in ShapeNetSem. As discussed in the previous chapters, the input geometry is not required to have UV unwrapping or other properties attached to the geometry. Whenever normals are available, the inpainting procedure can benefit from them through an additional inpainting zoning step; however, this is optional.

We require a pretrained image diffusion model with text and depth conditioning to paint novel views. From the NeRF pipeline, we expect that it can ingest view images, poses, and optional depth maps, and output a model that can be queried at arbitrary poses for color and depth.

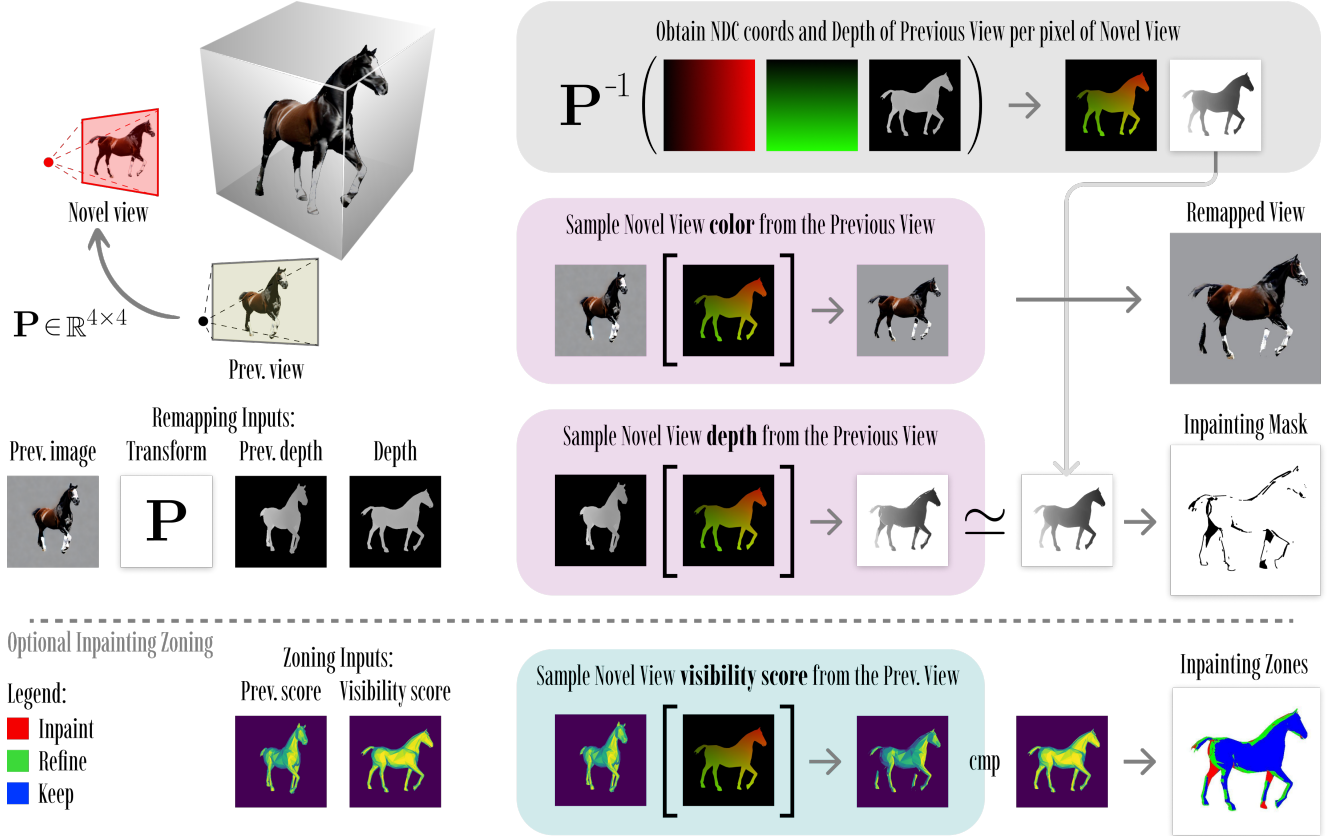


Figure 4. **Novel view remapping from a previous view.** Multi-view consistency is enforced by remapping the previous view into the novel view and preparing the inpainting mask of the unseen areas. The remapping procedure consists of three steps: (1) obtaining sampling coordinates of the previous view in the novel view, (2) sampling the novel view from the previous view, and (3) obtaining the inpainting mask by analyzing occlusions. An optional inpainting zoning step provides better control of inpainting for inputs with surface normals.

**Initialization** The first view generation defines and constrains the object’s overall painting and style. To obtain the first painted view, we render the object’s depth map and give it together with the text prompt to the depth-to-image pipeline. At this point, it is possible to query the user if the generated initialization is according to expectation and make early alterations by changing text or the pipeline seed.

**Novel View Remapping** Multi-view consistency is crucial for generating meaningful geometry painting. However, it is tricky to achieve in a pipeline with disentangled stages applied one after another, such as our design. To this end, we employ an occlusion-aware backward remapping scheme for image view reprojection from a previously-painted view to the novel one (Fig. 4).

At its core is the view transformation  $\mathbf{P} = \mathbf{K}\mathbf{E}\mathbf{K}^{-1}$ , which transforms normalized device coordinates (NDC) of the previous view into the novel view, where  $\mathbf{K}$  is the projection from world to NDC space and  $\mathbf{E} = [\mathbf{R}|\mathbf{T}]$  is the relative transformation of camera poses in world coordinates.

As a first step, we use the inverse transform  $\mathbf{P}^{-1}$  to map

the novel view NDC coordinates with  $z$ -values assigned from the Z-buffer of the novel view rendering into the previous view. This gives us an  $xy$ -map (depicted as a green-red tile) of pixels of the novel view and their source locations directly in the previous view. The transform also gives us the depth map of the source locations as seen from the previous view, which is used for the occlusion test.

Secondly, we obtain the previous view’s backward remapping into the novel view using the bilinear interpolation of the previous view at the  $xy$ -map locations. Compared to the direct application of the transform  $\mathbf{P}$  to the previous view image, the backward remapping is continuous by design and guarantees the absence of seams or holes in the remapped image.

However, an additional occlusion mask is required to identify areas of the novel view that are not visible from the previous view to handle these areas properly. Thus, as a third step, we obtain this mask by comparing the previous view depth map resampled using our  $xy$ -map, with the  $z$ -values obtained from the transformation on the first step. Evidently, the positions with agreeing depth are visible in both views

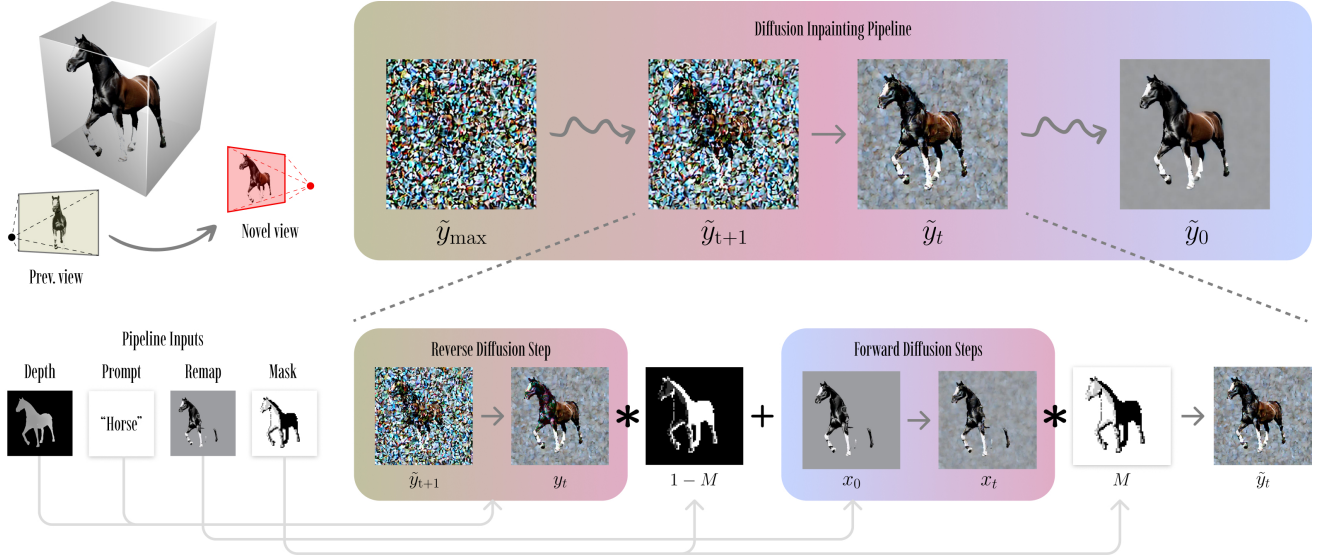


Figure 5. **Our text- and depth-conditioned latent diffusion inpainting pipeline for constrained novel view synthesis.** It is inspired by both the inpainting pipeline that takes an inpainting mask and applies it to the latents, and the text- and depth-conditioned generation pipeline from the Stable Diffusion distribution [30]. At each diffusion time step, the latents are composed from the forward diffusion step over the inpainting constraints (“Remap” in the figure), and the reverse diffusion step, conditioned on the input text prompt and depth.

under the assumptions we declared in the prerequisites. The final remapped view is thus obtained by combining the outputs of the previous two steps.

Additionally, the occlusion mask is stored for the future inpainting stage. Since most inpainting methods permit varying inpainting strength per pixel, we additionally compute inpainting zones map (similar to “trimaps” in TEXTure [29]), whenever the input geometry has surface normals. Specifically, we assign the visibility score to each fragment as a dot product between the surface normal and the unit vector originating in the camera origin and pointing at the fragment. By comparing visibility scores between the previous and novel views’ fragments, we classify zones into areas that are kept intact, areas of full inpainting, or refinement. As we identify in the ablation study, inpainting zoning helps with multi-view consistent painting details.

Finally, as we expand the painted area of the input, more views become available for color transfer to a novel view. The described procedure is thus easily extended to perform remapping from multiple previous views.

**Novel View Inpainting** We employ a custom text- and depth-conditioned latent diffusion inpainting pipeline to complete novel views after the remapping. The pipeline inherits from the previous works on inpainting with diffusion models [9, 17] and is largely based on the pretrained Stable Diffusion [30]. The input to the pipeline is the same as for image generation, with the addition of a mask that defines the inpainting area and the remapped image constraint (Fig. 5).

The mask  $M$  is taken from the remapping stage and down-

sampled to match the latent diffusion resolution. Upon availability, inpainting zoning additionally assigns an intermediate weight value for the refined areas.

At each denoising step  $t$ , we take the latent representation of the remapped image  $x_0$  and inject noise through  $t$  forward diffusion steps to obtain  $x_t$ . At the same time, we perform a single reverse diffusion step to obtain  $y_t$  from the more noisy  $\tilde{y}_{t+1}$  step, at which point we use the depth and the text prompt as conditions. We now blend the denoised latent  $y_t$  with remapped condition  $x_t$  using the inpainting mask  $M$ :  $\tilde{y}_t = (1 - M)y_t + Mx_t$ . This process starts with  $\tilde{y}_{max} \sim \mathcal{N}(0, 1)$  and is repeated until obtaining  $\tilde{y}_0$ , which is then decoded into the inpainting output. Notably, latent diffusion is the primary source of inconsistency between the inpainted images and their remapped constraints, which calls for a solution to enforce multi-view consistency globally.

**NeRF Reconstruction** Using the remapping and inpainting techniques introduced above, we can ensure the soft consistency of a subset of proximal views. However, we aim for global multi-view consistency, which requires considering all the generated views simultaneously. To this end, we employ a flavor of NeRF to resolve multi-view conflicts and reconcile painting from all viewpoints. Since the standard NeRF formulation supports different colors of the same 3D location depending on the viewpoint, we disable such view-dependent effects and fit the NeRF to predict view-invariant colors instead. Starting with a set of facade views and until there are no more unvisited poses, we submit all the generated images, their respective camera poses, and depth maps,

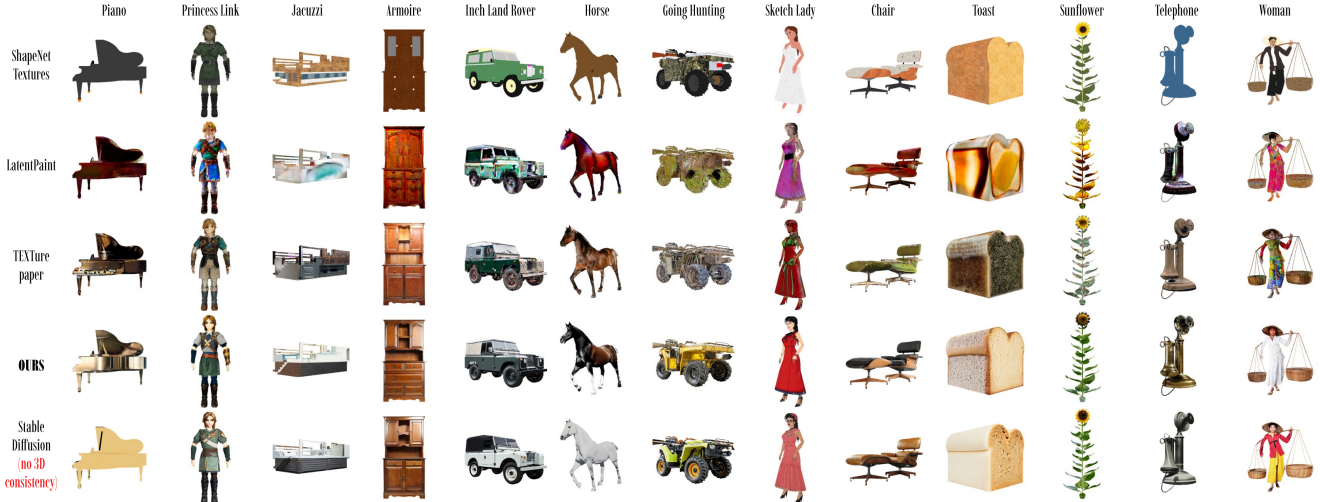


Figure 6. **Qualitative Comparisons** of our method to TEXTure [29], Latent-Paint [18], the original texturing from ShapeNetSem [6], and the “upper-bound quality” generative prior applied to each individual view without 3D consistency constraints. As can be seen, our method generates noise- and seam-free texturing with a high degree of detail.

as inputs to NeRF. Once the scene is fitted, all painted training images are replaced with renders from the fitted NeRF, so that our subsequent remapping steps always start from multi-view consistent inputs.

## 4. Experiments

As a first step towards painting ShapeNetSem, we chose a few hyperparameters for our pipeline. To paint each model, we rely on 9 views regularly spaced around the object in the horizontal plane ( $40^\circ$  increment). Starting from the front view, we generate 5 facade views using just the remapping and inpainting procedures. This facade configuration maximizes the coverage of the input geometry within the range of efficiency of our remapping technique. Before generating each subsequent view, we perform NeRF reconstruction. Our pose selection strategy picks the next view from the clockwise and counter-clockwise increments in alternating steps. We remap two of the closest painted views from the left and right paths around the model each time. This technique helps minimize the content gap in the last view, where the clockwise and counter-clockwise painting paths meet.

**Text Prompting** The base is set to “A photo of  $\{\{object\}\}$ ”. An additional “ $\{\{dir\}\}$  view” modifier specifies the coarse relation of the viewpoint and the object, helping with 3D consistency. Other modifiers are discussed in the Appendix.

**NeRF Setup** We chose Instant NGP [20] as a standalone NeRF backbone for its high degree of configurability and great performance. Additionally, we leverage depth supervision in NeRF training to facilitate faster convergence and obtain higher-quality reconstruction.

Our setting slightly differs from the default NeRF objec-

tive because our training images are generated from diffusion and can have soft view conflicts. As mentioned previously, the purpose of NeRF in our pipeline is to bring multi-view painting to agreement rather than to simulate light transport. We disable view-dependent effects in the NeRF configuration to align with this purpose. Additionally, we adjust the parameters for the grid encoding settings. We found that a higher number of levels (5) and encoded features (16) achieve good rendering fidelity while keeping a sufficiently smooth and continuous NeRF surface.

**ShapeNetSem Processing** We demonstrate that our method can be applied to a wide range of object categories and shapes by conducting a study of texturing a significant subset of the ShapeNet [6] dataset called ShapeNetSem, which contains 12K models in over 270 categories. We preprocess each model by orienting it using the up and front vectors from the metadata, centering, and scaling to fit the unit sphere. We take the text prompt’s “*object*” part from the name field of the dataset metadata.

Each model has a list of associated categories attached to it. We compute frequencies of all categories in the entire dataset and assign each model a primary category. These primary categories are used for both qualitative and quantitative studies. We demonstrate high-quality painting results on a select set of categories, including electronics, animals, and game characters, in Fig. 2. See Figs. 6, 7 for more results.

**Comparison with Other Methods** We compare our method quantitatively with two recent mesh texturing methods, Latent-Paint [18] and TEXTure [29] (Fig. 6). We ran both pipelines on the ShapeNetSem [6] dataset using the same 360-degree camera views and text prompts. While the



Figure 7. **Large-Scale Comparison of ShapeNetSem Texturing** with the original textures [6], Latent-Paint [18], TEXTure [29], and our method. We present spin-views of  $\sim 12K$  models from over 270 categories. The models are grouped by category and sorted by group size. Categories, IDs, and model names (prompts) are specified under the corresponding video tiles. *Tip:* Use timecodes to conveniently skip to categories of interest.

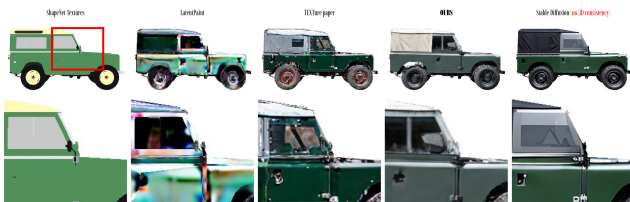


Figure 8. **A Closer Look** reveals that our method produces more realistic results with invisible seams, while other methods often exhibit texture filtering issues and lower realism.

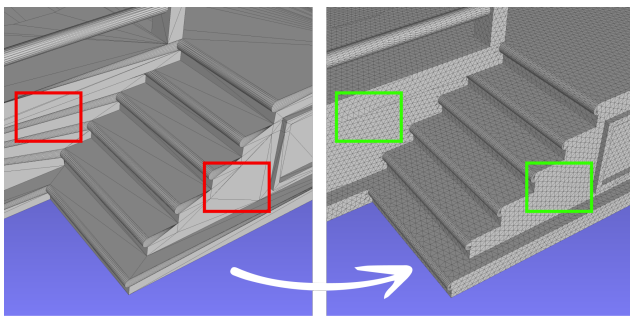


Figure 9. **Exporting NeRF as Mesh.** Given the input mesh and the painted NeRF, we remesh the input almost isotropically with planarity constraints and sample vertex colors from the NeRF. This technique does not require a UV texture map for the input geometry.

TEXTure method handles well-defined camera trajectories, Latent-Paint requires way more views to perform decently; otherwise, we kept their default settings and ensured alignment of the cameras. We rendered interpolated views of the output models and compared the results of the two pipelines.

To facilitate the quantitative study, we additionally gener-

ated painting results for the evaluation views using only the Stable Diffusion [30] depth-to-image model. Although this set of images completely lacks 3D consistency, it provides a useful upper bound on the image fidelity that is attainable with the generative model.

After processing all models with the selected methods, we render their 360-degree spin views using synchronized camera setups and aggregate them in the video gallery (Fig. 7).













A closer look at the output renders in the video (also the car model in Fig. 8) reveals discernible quality differences between different geometry painting methods. We can see that the original ShapeNet [6] textures are rather primitive. Latent-Paint [18] exhibits blurred and overall coarse texturing. TEXTure [29] produces much more realism and details; however, compared to our method, its output contains spurious artifacts and texture filtering issues. This effect is prevalent in complex meshes containing many fine-grain geometry details. We observe that both prior methods have distinct artifacts that stem from the effective resolution of the UV texture maps, texture atlas patch discontinuities, and imperfect UV unwrapping. These issues are further exacerbated when differential rendering is employed. Our method is free of these issues; refer to the Appendix for discussion.

**Compute Requirements** Unlike the other two methods, whose memory footprint fluctuates depending on the 3D model complexity and requires at least 16GB GPU RAM, our method’s resources are defined purely by NeRF configuration and are fixed across the whole dataset to 12GB RAM. Our pipeline configured as stated above takes  $\sim 15$ -20 min to complete, which is on par with the competition.

**Quantitative Evaluation** We execute our pipeline, collect the output NeRF, and sample it at 8 different evaluation views at  $45^\circ$  increments. Using collections of these views obtained for all models in the dataset, we compared distribution metrics between each method and the reference (no 3D consistency) for the whole dataset and several primary categories. Through this evaluation, we aim to understand how close we can get to the upper bound of lifting the learned generative prior in 3D while maintaining 3D consistency by design. Frechet Inception Distance (FID) [13] and Kernel Inception Distance (KID) [2] are the standard metrics for comparing distributions of images: natural or sampled from generative models.

Following the footsteps of [15], we report  $FID_{CLIP}$  with the CLIP feature extractor. We additionally propose two new metrics:  $FID_{DINOv2}$  and  $KID_{DINOv2}$ , which utilize the novel self-supervised feature extraction techniques [25]. Unlike the decade-old Inception backbone and CLIP, which focuses on named entities, DINOv2 is a powerful self-supervised feature extractor trained on natural images. All metrics are computed through a verified evaluation protocol of torch-fidelity [23]. The results of this quantitative study

Table 1. Comparison of geometry painting with various methods on ShapeNetSem [6] dataset measured through Frechet Inception Distance (FID ↓) [13] metric with various feature extractors. Lower values are better. Results with Kernel Inception Distance [2] metric are in Tab. 2.

| FID ↓ [13]<br>Features | Methods           | All<br>(11992)  | Misc.<br>(2912)   | Chair<br>(682)  | Lamp<br>(655)   | ChstDrw.<br>(503)   | Table<br>(416)  | Couch<br>(405)  | Computer<br>(241)   | TV (229)  | WallArt<br>(220)  | Bed<br>(218)  | Cabt.<br>(216)  |
|------------------------|-------------------|---|---|---|---|---|---|---|---|---|---|---|---|
|                        |                   |  |  |  |  |  |  |  |  |  |  |  |  |
| Inception<br>[13, 36]  | Orig. texture [6] | 30.10   | 31.82   | 40.79   | 47.61   | 113.6   | 49.18   | 81.39   | 63.02   | 60.89   | 64.92   | 67.37   | 111.9   |
|                        | Latent-Paint [18] | 27.73   | 30.87   | 36.65   | 38.36   | 67.60   | 28.44   | 65.98   | 68.85   | 67.85   | 90.99   | 49.04   | 73.60   |
|                        | TEXTure [29]      | 16.10   | 18.34   | 23.44   | 30.75   | 32.65   | 34.98   | 40.40   | 46.48   | 45.85   | 61.23   | 43.04   | 38.88   |
|                        | <b>Ours</b>       | <b>9.60</b>   | <b>11.05</b>  | <b>16.30</b>  | <b>19.54</b>  | <b>32.64</b>  | <b>22.01</b>  | <b>26.23</b>  | <b>39.96</b>  | <b>29.60</b>  | <b>35.77</b>  | <b>33.13</b>  | <b>36.28</b>  |
| CLIP<br>[15, 27]       | Orig. texture [6] | 18.86   | 18.71   | 24.89   | 27.66   | 40.15   | 25.72   | 33.57   | 20.60   | 27.29   | 18.86   | 28.79   | 37.07   |
|                        | Latent-Paint [18] | 15.84   | 16.42   | 17.08   | 12.29   | 29.51   | 11.34   | 22.22   | 24.50   | 22.47   | 27.30   | 19.35   | 27.83   |
|                        | TEXTure [29]      | 6.85  | 6.85  | 9.62  | 9.37  | 11.29   | 11.00   | 9.48  | 11.28   | 11.38   | 13.17   | 11.09   | 9.79  |
|                        | <b>Ours</b>       | <b>3.24</b>   | <b>3.33</b>   | <b>3.90</b>   | <b>3.47</b>   | <b>7.77</b>   | <b>4.12</b>   | <b>4.69</b>   | <b>8.22</b>   | <b>6.16</b>   | <b>6.18</b>   | <b>5.54</b>   | <b>7.30</b>   |
| DINOv2<br>[25]         | Orig. texture [6] | 588.1   | 585.9   | 620.6   | 787.3   | 1640.   | 883.9   | 1265.6  | 767.1   | 999.4   | 857.9   | 946.3   | 1517.   |
|                        | Latent-Paint [18] | 332.9   | 366.1   | 285.8   | 329.0   | 696.6   | 280.6   | 556.3   | 673.4   | 773.9   | 866.5   | 533.4   | 765.1   |
|                        | TEXTure [29]      | 175.0   | 194.6   | 181.1   | 278.9   | 321.6   | 248.0   | 282.1   | 404.4   | 501.5   | 580.3   | 276.7   | 366.2   |
|                        | <b>Ours</b>       | <b>125.1</b>  | <b>136.7</b>  | <b>130.8</b>  | <b>181.6</b>  | <b>299.4</b>  | <b>173.1</b>  | <b>239.4</b>  | <b>383.0</b>  | <b>333.5</b>  | <b>312.2</b>  | <b>226.3</b>  | <b>320.2</b>  |

are presented in Tab. 1. Evidently, our method achieves state-of-the-art fidelity to the generative prior while maintaining 3D consistency.

**Geometry Export** The output of our pipeline is contained in the final NeRF reconstruction. While NeRF as a 3D asset format gains popularity as hardware acceleration catches up, we take an extra step to transfer the generated painting back into a standard editable format. Since we do not require UV texture maps on the input and want to support use cases such as Point-E discussed below, we opt for transferring colors to the input mesh vertices. However, to ensure sufficient spatial resolution for such a scheme, vertices should be uniformly distributed on the surface of the input, which is usually not the case. To overcome this issue, we designed an algorithm for approximately-isotropic remeshing [22] that preserves the input geometry and only focuses on planar regions (Fig. 9). Using our remeshing technique helps obtain an identical mesh but with sufficient resolution for color transfer. Thanks to unambiguous color querying from our view-invariant NeRF flavor, we directly transfer color onto the remeshed input by sampling NeRF at all vertices locations. We further note, that the output asset files with per-vertex colors occupy significant space, which can be reclaimed by compression techniques such as DRACO [1].

**Pure Text-to-3D via Point-E** We extend our pipeline with Point-E [21], a diffusion-based generative model that produces 3D point clouds from text prompts. Following [21], we convert the point cloud generated by Point-E to a signed distance field and use marching cubes with grid size 64 to obtain the mesh serving as an input to our method. Since the resulting geometry has surface normals of limited quality, we skip inpainting zoning in our method. Fig. 10 demonstrates an overall pipeline that takes only a text prompt as the input and outputs a mesh with improved painting. From the opposite point of view, since Point-E cannot generate

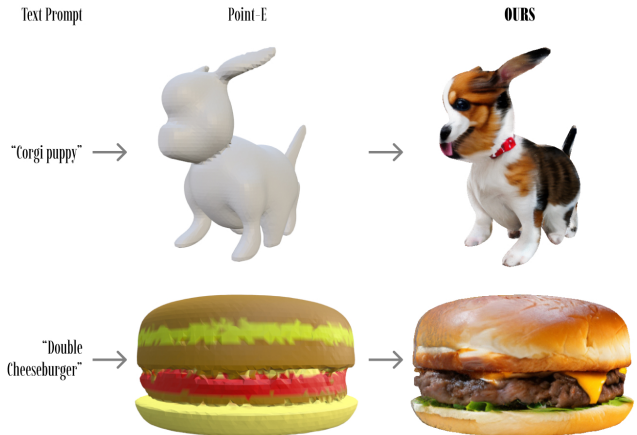


Figure 10. **Painting Point-E** [21]. We extend our pipeline to pure text-to-3D by chaining it after Point-E. The same text prompt is used to generate the geometry and then repaint it with our method.

detailed textures, our method can be seen as a downstream modular extension of Point-E to boost the texture quality of the produced 3D models.

## 5. Discussions and Conclusion

In this work, we presented a novel pipeline combining a generative 2D diffusion prior and 3D neural radiance fields as *standalone* modules and demonstrated their ability to paint the input geometry using a text prompt in a 3D-consistent manner. We conducted a large-scale study on the ShapeNet-Sem [6] dataset and demonstrated the advantages of our approach against several prior art methods on a wide range of object categories. We believe that our pipeline will reach the community of artists, content creators, and game developers and enable quick prototyping of 3D assets, particularly from existing ones, thus giving them a new life.

We thank Shengyu Huang for proofreading this manuscript.



# Breathing New Life into 3D Assets with Generative Repainting

## Supplementary Material

### A. Large-Scale Study of ShapeNetSem

Out of 12,288 models in the dataset, we processed 11,992 with all methods. The remaining 296 models either had flat geometry or could not be processed by the Latent-Paint [18] pipeline, TEXTure [29], or both. The failure cases happened most commonly due to the complex geometry not fitting 16GB GPU RAM within the respective method pipeline or failures in xatlas texture UV unwrapping module [16]. Our method produced results consistently even on these models, but for a fair comparison, we excluded these models completely.

In addition to the FID [13] evaluation from Tab. 1, we provide a quantitative evaluation of all pipelines on ShapeNetSem with the KID metric [2] in Tab. 2.

The ability of our method to handle complex geometry, low memory footprint, weak dependence on the geometry format or the rendering pipeline, and potentially unknown texture coordinates – all these properties make our method a reliable go-to solution for 3D assets revamping.

### B. Subjective User Study

We conducted a limited crowd-sourced perceptual comparison between Latent-Paint [18], TEXTure [29], and our method. The study was based on 50 randomly sampled models from 10 categories, c.f. Tab. 1. Subjects were instructed (Fig. 11, left) to analyze and vote for higher quality and realism after observing a full 360° spin of models painted with a pair of methods, side by side. Each subject submitted 20 votes, plus 2 validation questions with predefined correct answers (Fig. 11, right). 35 subjects participated in our study, of which 29 (83%) passed the validation. 638 votes were collected, ensuring at least 3 votes for every pair, and aggregated into preference scores with the Crowd Bradley-Terry [3] model. The resulting scores were (log-scale, up to additive constant, 95% confidence intervals, higher is better):  $S_{\text{Latent-Paint}} = 0.15_{\pm 0.15}$ ,  $S_{\text{TEXTure}} = 0.30_{\pm 0.11}$ ,  $S_{\text{ours}} = 1.86_{\pm 0.13}$ . The scores agree with the quantitative results.

### C. ShapeNet Rendering Settings

To facilitate a fair comparison of different methods on the ShapeNetSem dataset [6], we choose the mesh rendering settings in all pipelines such that the output result is adequate for all methods. Notably, TEXTure [29] relies on mesh normals to determine inpainting regions. However, a subset of ShapeNetSem [6] meshes have faces with inappropriately



Figure 11. **Subjective Study.** Left: User instruction with quality and realism judgment examples; **Right:** Two validation questions “Which one is better?” shared among all subjects to ensure engagement (the right column answers were expected for a pass).

oriented surface normals. For these meshes, directly passing them as input to TEXTure [29] produces corrupt texturing.

To address this issue, we utilize back-face culling of mesh to disable the rendering of mesh faces that are oriented away from the camera. We build our method on top of PyTorch3D [28], which provides a built-in implementation of back-face culling. However, since both Latent-Paint [18] and TEXTure [29] pipelines rely on the Kaolin renderer [10], which did not implement back-face culling as of the time of writing, we implemented back-face culling in software. This allowed us to address the rendering discrepancy and level the settings for all pipelines.

We experimented with double-face rendering as an alternative approach to resolving face orientation issues. However, the result of using double-face rendering is worse than that of using back-face culling, as seen in Fig. 12 (left). We suspect this is due to areas of the mesh having overlapping front-facing faces in the double-face rendering setting, thereby negatively affecting texture back-projection in the TEXTure [29] method. Overall, our rendering protocol is chosen to maximize the output quality of the pipelines relying on differential rendering under complex geometry.

### D. Ablation: Inpainting Zoning

Inpainting zoning works in areas of the mesh that face away from the camera in one generated view so that they can be further refined in the subsequent views. Fig. 12 (middle) shows that our refinement scheme brings more details to the areas of the model with challenging visibility constraints.

Table 2. Comparison of geometry painting with various methods on ShapeNetSem [6] dataset measured through Kernel Inception Distance (KID ↓) [2] metric with various feature extractors. Standard deviations are given in small font for all values. Lower values are better.













| KID ↓ [2]<br>Features<br>Multiplier | Methods           | All<br>(11992)  | Misc.<br>(2912)   | Chair<br>(682)  | Lamp<br>(655)   | ChstDrw.<br>(503)   | Table<br>(416)  | Couch<br>(405)   | Computer<br>(241)   | TV (229)  | WallArt<br>(220)  | Bed<br>(218)  | Cabt.<br>(216)  |
|-------------------------------------|-------------------|---|---|---|---|---|---|--|---|---|---|---|---|
|                                     |                   |  |  |  |  |  |  |  |  |  |  |  |  |
| Inception<br>[2, 13, 36]<br>×0.01   | Orig. texture [6] | 1.19 ±0.04  | 1.18 ±0.09  | 1.40 ±0.20  | 2.03 ±0.38  | 7.89 ±0.79  | 1.61 ±0.26  | 4.47 ±0.56   | 2.76 ±0.47  | 3.04 ±0.40  | 1.84 ±0.52  | 2.72 ±0.36  | 5.98 ±0.69  |
|                                     | Latent-Paint [18] | 1.31 ±0.05  | 1.37 ±0.11  | 2.02 ±0.25  | 1.95 ±0.36  | 4.52 ±0.54  | 1.05 ±0.21  | 4.26 ±0.39   | 3.84 ±0.35  | 4.17 ±0.34  | 3.81 ±0.58  | 2.14 ±0.32  | 4.19 ±0.47  |
|                                     | TEXTure [29]      | 0.75 ±0.04  | 0.71 ±0.08  | 1.19 ±0.20  | 1.61 ±0.35  | 1.79 ±0.38  | 1.97 ±0.32  | 2.37 ±0.48   | 2.14 ±0.33  | 2.36 ±0.31  | 2.10 ±0.41  | 1.90 ±0.32  | 1.54 ±0.29  |
|                                     | <b>Ours</b>       | <b>0.44 ±0.03</b>   | <b>0.38 ±0.06</b>   | <b>0.65 ±0.18</b>   | <b>0.80 ±0.24</b>   | <b>1.88 ±0.48</b>   | <b>0.94 ±0.22</b>   | <b>1.14 ±0.30</b>  | <b>1.74 ±0.36</b>   | <b>1.06 ±0.24</b>   | <b>0.53 ±0.16</b>   | <b>1.09 ±0.22</b>   | <b>1.32 ±0.41</b>   |
| CLIP<br>[15, 27]<br>×0.01           | Orig. texture [6] | 9.33 ±0.26  | 8.92 ±0.50  | 13.1 ±1.38  | 14.6 ±1.38  | 21.1 ±1.71  | 13.0 ±1.19  | 18.7 ±1.73   | 8.36 ±1.12  | 14.3 ±1.41  | 5.89 ±1.22  | 14.2 ±1.32  | 17.7 ±1.76  |
|                                     | Latent-Paint [18] | 7.87 ±0.18  | 7.87 ±0.37  | 9.36 ±0.57  | 6.44 ±0.67  | 17.1 ±1.23  | 5.47 ±0.48  | 13.1 ±0.85   | 12.1 ±0.81  | 11.2 ±0.67  | 10.7 ±0.99  | 10.5 ±0.91  | 15.7 ±1.20  |
|                                     | TEXTure [29]      | 3.18 ±0.09  | 3.04 ±0.21  | 5.12 ±0.46  | 4.84 ±0.62  | 5.81 ±0.67  | 5.67 ±0.52  | 4.82 ±0.60   | 4.68 ±0.43  | 4.63 ±0.47  | 3.99 ±0.65  | 5.21 ±0.50  | 4.40 ±0.48  |
|                                     | <b>Ours</b>       | <b>1.36 ±0.06</b>   | <b>1.30 ±0.13</b>   | <b>1.69 ±0.32</b>   | <b>1.51 ±0.27</b>   | <b>3.73 ±0.70</b>   | <b>1.90 ±0.34</b>   | <b>2.07 ±0.39</b>  | <b>2.98 ±0.61</b>   | <b>2.14 ±0.48</b>   | <b>0.96 ±0.27</b>   | <b>2.09 ±0.36</b>   | <b>3.12 ±0.62</b>   |
| DINOv2<br>[25]<br>×1.0              | Orig. texture [6] | 2.40 ±0.06  | 2.36 ±0.15  | 3.65 ±0.37  | 4.95 ±0.47  | 11.9 ±0.98  | 5.94 ±0.61  | 14.1 ±1.46   | 5.42 ±0.85  | 9.80 ±0.96  | 3.75 ±0.75  | 7.56 ±0.69  | 9.74 ±0.79  |
|                                     | Latent-Paint [18] | 1.01 ±0.02  | 1.04 ±0.05  | 1.85 ±0.25  | 2.09 ±0.25  | 5.51 ±0.51  | 1.62 ±0.25  | 6.61 ±0.83   | 5.49 ±0.54  | 8.87 ±0.71  | 3.37 ±0.49  | 4.26 ±0.50  | 5.21 ±0.52  |
|                                     | TEXTure [29]      | 0.53 ±0.02  | 0.50 ±0.03  | 1.18 ±0.24  | 1.76 ±0.27  | 2.56 ±0.43  | 1.67 ±0.27  | 3.32 ±0.62   | 3.72 ±0.50  | 3.97 ±0.61  | 2.17 ±0.44  | 2.21 ±0.31  | 2.37 ±0.39  |
|                                     | <b>Ours</b>       | <b>0.38 ±0.01</b>   | <b>0.35 ±0.03</b>   | <b>0.63 ±0.21</b>   | <b>1.07 ±0.21</b>   | <b>2.01 ±0.48</b>   | <b>1.03 ±0.21</b>   | <b>1.90 ±0.64</b>  | <b>3.14 ±0.66</b>   | <b>2.24 ±0.43</b>   | <b>0.86 ±0.19</b>   | <b>1.08 ±0.23</b>   | <b>1.58 ±0.37</b>   |



Figure 12. **Ablations.** **Left:** Rendering Settings Comparison with TEXTure [29] method. Employing back-face culling achieves the best result compared with the original<sup>1</sup> and double-face rendering settings. **Middle:** Visibility Score Refinement produces more realistic details on surfaces seen at sharp angles, such as the ear. **Right:** Low number of generated views (4) leads to poor coverage of the input geometry, 18 results in over-smoothing, and 9 is a trade-off.

## E. Ablation: Number of Input Views

We show a qualitative comparison between models painted using various numbers of input views in Fig. 12 (right). With just 4 input views, we find holes and artifacts on the object’s surface. With 18 views, the shape is smooth, but the generated color lacks detail. The choice of 9 views achieves the best quality.

## F. Prompt Augmentation

Our method transparently exposes the style guidance functionality of the underlying generative models. It permits prompt augmentation, enabling greater variety in the generated painting while preserving 3D consistency. Specifically, our pipeline extends the input object description prompt as follows: “A photo of a {{modifier}} {{object}}, {{dir}} view”. The “{{modifier}}” style specifier term could be the color or the material of the object. In the same vein as text guides image generation models, the texture of our 3D models changes according to the modifier, as shown in Fig. 13.



Figure 13. **Style Specifier Guidance.** In the given examples, prompts take the following form: “A photo of a {{material}} dresser” (top) and “A photo of a {{color}} dragon” (bottom).

## References

- [1] Google Draco authors. Draco: 3d data compression, 2017. Release v1.5.6, <https://github.com/google/draco>. 8
- [2] Mikołaj Bińkowski, Dougal J. Sutherland, Michael Arbel, and Arthur Gretton. Demystifying MMD GANs. In *ICLR*, 2018. 7, 8, 9, 10
- [3] Ralph Allan Bradley and Milton E Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952. 9
- [4] Shengqu Cai, Eric Ryan Chan, Songyou Peng, Mohamad Shahbazi, Anton Obukhov, Luc Van Gool, and Gordon Wetzstein. Diffdreamer: Consistent single-view perpetual view generation with conditional diffusion models. *arXiv:2211.12131*, 2022. 2
- [5] Shengqu Cai, Anton Obukhov, Dengxin Dai, and Luc Van Gool. Pix2nerf: Unsupervised conditional p-gan for single image to neural radiance fields translation. In *CVPR*, 2022. 2, 3
- [6] 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:1512.03012*, 2015. 2, 3, 6, 7, 8, 9, 10
- [7] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *ECCV*, 2022. 3
- [8] Dave Zhenyu Chen, Yawar Siddiqui, Hsin-Ying Lee, Sergey Tulyakov, and Matthias Nießner. Text2tex: Text-driven texture synthesis via diffusion models. *arXiv:2303.11396*, 2023. 3
- [9] Guillaume Couairon, Jakob Verbeek, Holger Schwenk, and Matthieu Cord. Diffedit: Diffusion-based semantic image editing with mask guidance. *arXiv:2210.11427*, 2022. 5
- [10] 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 Lebedev. Kaolin: A pytorch library for accelerating 3d deep learning research, 2022. 9
- [11] Jun Gao, Tianchang Shen, Zian Wang, Wenzheng Chen, Kangxue Yin, Daiqing Li, Or Litany, Zan Gojic, and Sanja Fidler. Get3d: A generative model of high quality 3d textured shapes learned from images. In *NeurIPS*, 2022. 3
- [12] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NeurIPS*, 2014. 2, 3
- [13] 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. In *NeurIPS*, 2017. 7, 8, 9, 10
- [14] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, 2020. 1, 2
- [15] Tuomas Kynkäänniemi, Tero Karras, Miika Aittala, Timo Aila, and Jaakko Lehtinen. The role of imagenet classes in fréchet inception distance. In *ICLR*, 2023. 7, 8, 10
- [16] Bruno Lévy, Sylvain Petitjean, Nicolas Ray, and Jérôme Maillot. Least squares conformal maps for automatic texture atlas generation. *ACM Transactions on Graphics (ToG)*, 21(3):362–371, 2002. 2, 9
- [17] Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. Repaint: Inpainting using denoising diffusion probabilistic models. In *CVPR*, 2022. 5
- [18] Gal Metzer, Elad Richardson, Or Patashnik, Raja Giryes, and Daniel Cohen-Or. Latent-nerf for shape-guided generation of 3d shapes and textures. *arXiv:2211.07600*, 2022. 1, 3, 6, 7, 8, 9, 10
- [19] 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, 3
- [20] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics (ToG)*, 41(4):1–15, 2022. 2, 3, 6
- [21] Alex Nichol, Heewoo Jun, Prafulla Dhariwal, Pamela Mishkin, and Mark Chen. Point-e: A system for generating 3d point clouds from complex prompts. *arXiv:2212.08751*, 2022. 2, 3, 8
- [22] Anton Obukhov. Fast almost-isotropic planar remeshing algorithm, 2023. Commit SHA 56413d1, <https://github.com/toshas/remesh-isotropic-planar>. 8
- [23] Anton Obukhov, Maximilian Seitzer, Po-Wei Wu, Semen Zhydenko, Jonathan Kyl, and Elvis Yu-Jing Lin. High-fidelity performance metrics for generative models in pytorch, 2020. Version: 0.3.0, DOI: 10.5281/zenodo.4957738. 7
- [24] Anton Obukhov, Mikhail Usvyatsov, Christos Sakaridis, Konrad Schindler, and Luc Van Gool. Tt-nerf: Tensor train neural fields. *arXiv:2209.15529*, 2022. 3
- [25] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. DINOv2: Learning robust visual features without supervision. *arXiv:2304.07193*, 2023. 7, 8, 10
- [26] Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv:2209.14988*, 2022. 1, 3
- [27] 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 *ICML*, 2021. 2, 8, 10
- [28] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3d deep learning with pytorch3d. *arXiv:2007.08501*, 2020. 9
- [29] Elad Richardson, Gal Metzer, Yuval Alaluf, Raja Giryes, and Daniel Cohen-Or. Texture: Text-guided texturing of 3d shapes. *arXiv:2302.01721*, 2023. 1, 3, 5, 6, 7, 8, 9, 10

- [30] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022. 1, 2, 5, 7
- [31] Chitwan Saharia, William Chan, Huiwen Chang, Chris Lee, Jonathan Ho, Tim Salimans, David Fleet, and Mohammad Norouzi. Palette: Image-to-image diffusion models. In *ACM SIGGRAPH 2022 Conference Proceedings*, 2022. 2
- [32] 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. *arXiv:2205.11487*, 2022. 1, 2
- [33] Johannes L Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *CVPR*, 2016. 3
- [34] Yawar Siddiqui, Justus Thies, Fangchang Ma, Qi Shan, Matthias Nießner, and Angela Dai. Texturify: Generating textures on 3d shape surfaces. In *EECV*, 2022. 3
- [35] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *ICML*, 2015. 2
- [36] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, 2016. 8, 10
- [37] Daniel Watson, William Chan, Ricardo Martin-Brualla, Jonathan Ho, Andrea Tagliasacchi, and Mohammad Norouzi. Novel view synthesis with diffusion models. *arXiv:2210.04628*, 2022. 3
- [38] Jiajun Wu, Chengkai Zhang, Tianfan Xue, Bill Freeman, and Josh Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *NeurIPS*, 2016. 3
- [39] Yinghao Xu, Sida Peng, Ceyuan Yang, Yujun Shen, and Bolei Zhou. 3d-aware image synthesis via learning structural and textural representations. In *CVPR*, 2022. 3