3D Paintbrush: Local Stylization of 3D Shapes with Cascaded Score Distillation

Dale Decatur
University of Chicago

Itai Lang
University of Chicago

Kfir Aberman
Snap Research

Rana Hanocka
University of Chicago

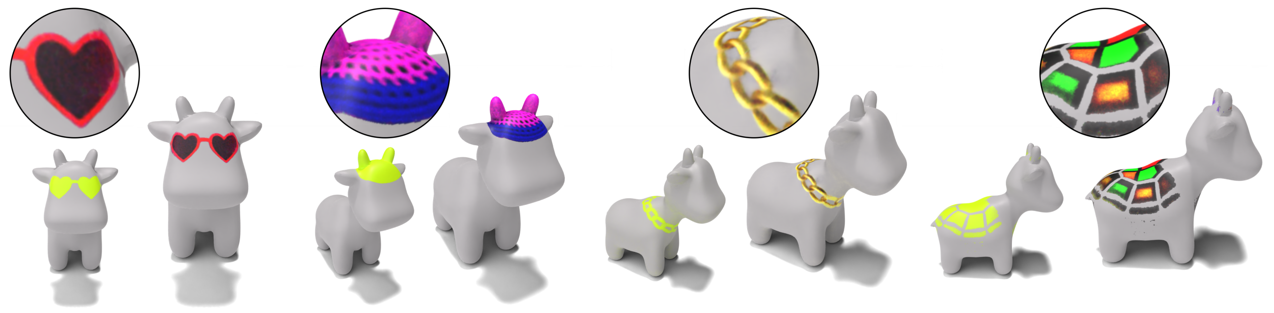

Figure 1. Utilizing only a text prompt as guidance, 3D Paintbrush seamlessly generates local stylized textures on bare meshes. Our approach produces a localization map (yellow regions) and a highly detailed texture map which conforms to it.

Abstract

We present 3D Paintbrush, a technique for automatically texturing local semantic regions on meshes via text descriptions. Our method is designed to operate directly on meshes, producing texture maps which seamlessly integrate into standard graphics pipelines. We opt to simultaneously produce a localization map (to specify the edit region) and a texture map which conforms to it. This approach improves the quality of both the localization and the stylization. To enhance the details and resolution of the textured area, we leverage multiple stages of a cascaded diffusion model to supervise our local editing technique with generative priors learned from images at different resolutions. Our technique, referred to as Cascaded Score Distillation (CSD), simultaneously distills scores at multiple resolutions in a cascaded fashion, enabling control over both the granularity and global understanding of the supervision. We demonstrate the effectiveness of 3D Paintbrush to locally texture different semantic regions on a variety of shapes. Project page: https://threedle.github.io/3d-paintbrush

1. Introduction

The ability to edit existing high-quality 3D assets is a fundamental capability in 3D modeling workflows. Recent works have shown exceptional results for text-driven 3D data creation [32, 38, 48, 53, 58, 59], but focus on making *global* edits. While some progress has been made on local editing using an explicit localization of the edit region [49, 67], these regions are often coarse and lack fine-grained detail. Highly-detailed and accurate localizations are important for constraining the edits to be within a specific region, preventing changes unrelated to the target edit. Furthermore, while meshes with texture maps are the de facto standard in graphics pipelines, existing local editing work does not natively operate on meshes nor produce texture maps for them.

In this work we develop 3D Paintbrush, a method for automatically texturing local semantic regions on meshes via text descriptions. Our method is designed to operate directly on meshes, producing texture maps which seamlessly integrate into standard graphics pipelines. 3D Paintbrush is controlled via intuitive, free-form text input, allowing users to describe their edits using open vocabulary on a wide range of meshes. Specifically, given an input mesh and a text prompt, 3D Paintbrush produces the corresponding high-quality texture map and a localization region to confine it. To enhance the details and resolution of the locally textured area, we introduce Cascaded Score Distillation (CSD) which leverages multiple stages of a cascaded diffusion model. Our explicit localization masks can be used to layer our edit texture onto existing textures.

We opt to represent both our localization map and texture map as neural fields encoded by multi-layer perceptions. Our method synthesizes both a fine-grained localization mask and high-quality texture in tandem. Simultane-

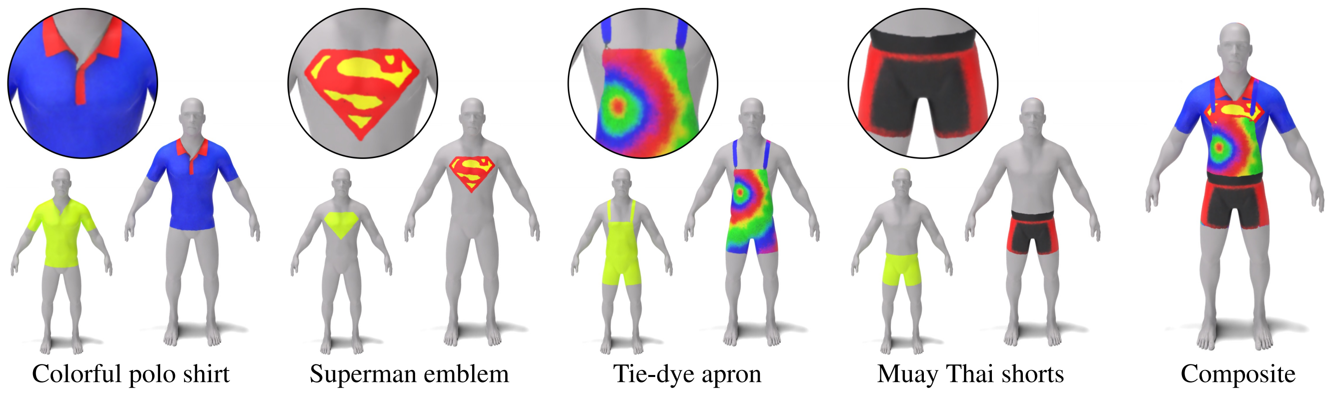

Figure 2. **Precise composition of multiple local textures**. 3D Paintbrush produces highly-detailed textures that effectively adhere to the predicted localizations. This enables seamlessly compositing local textures without unwanted fringes (right).

ously generating the localization and texture maps improves the quality of each. The texture map drives the localization to become more detailed and intricate. The localization explicitly masks the texture, ensuring a coherent local style which respects the localization boundary.

Our local stylization operates in small regions, necessitating higher resolution supervision compared to global generative techniques. Existing approaches leverage pre-trained text-to-image diffusion models with Score Distillation Sampling (SDS) to supervise text-driven optimizations [31, 58]. Text-to-image diffusion models often contain multiple cascaded stages in order to achieve high resolution [21], but standard SDS only utilizes the first low-resolution stage of the cascaded model. Our technique, referred to as Cascaded Score Distillation (CSD), simultaneously distills scores at multiple resolutions in a cascaded fashion, enabling control over both the granularity and global understanding of the supervision. Since cascaded stages are trained entirely independently, our insight is to formulate a distillation loss that incorporates all stages in tandem.

In summary, our method enables local text-driven stylization of meshes. By explicitly learning a localization in tandem with the texture, we ensure that our edits are bounded by the localized region. Using our CSD, which leverages all stages of the diffusion model, we can control the granularity and global understanding of the supervision achieving higher resolution textures and localizations than standard SDS. We demonstrate that 3D Paintbrush yields diverse local texturing on a variety of shapes and semantic regions and outperforms baselines both qualitatively and quantitatively.

2. Related Work

A large body of work has studied stylization and analysis of 3D content. Existing work uses neural networks and optimization [6, 15, 19, 22, 23, 30, 33, 37–39, 41, 42, 51, 60, 62] for mesh stylization. Other works use a neural radiance field NeRF [40] for stylization [11, 34, 64]. Yet, these works focus on stylization rather than localization. Large 2D models have been used for analytical tasks in 3D such as localization and segmentation [1, 2, 10, 17, 27, 28, 54, 57, 67], however, none of these works produce textures. Furthermore, only [1, 2, 10, 67] aim to produce a tight localization on meshes and we find that these approaches still produce relatively smooth localization regions that cannot capture the high frequency details needed for sharp local edits.

Text-driven generation and editing. Existing works have leveraged pre-trained 2D models to generate 3D representations that adhere to a text prompt [4, 12, 16, 25, 29, 39, 41, 63]. Many recent methods [9, 26, 32, 44, 50, 53, 53, 58, 66] use score distillation [44, 58] from 2D models to generate both geometry and styles from scratch, while other works optimize the texture of an existing, fixed geometry [8, 38, 39, 47]. Other work aims to generate 3D representations from images [14, 35, 36, 45].

Existing text-to-3D generative methods [38, 44, 58, 59] can be used to perform global edits [18, 48, 67]. However, since these approaches do not have explicit edit localizations, they struggle to perform highly specific local edits without changing other components of the 3D representation's appearance. Different from our objective, these works aim to generate or globally manipulate existing 3D representations, while our work focuses on local editing.

Text-driven local editing. Many approaches can perform global 3D edits and progress has been made on local editing in images and videos [5, 7, 13, 20]. Yet, few works have addressed the task of precise, local editing for 3D representations. Local editing is challenging since, in addition to synthesizing the edit, methods need to localize the edit region. FocalDreamer [31] obtains precise user defined edit regions at the cost of requiring additional, tedious user input compared to strictly text-driven approaches. Vox-E [49] (operating on voxel representations) and DreamEditor [67] (operating on NeRFs) both use attention maps to localize an edit

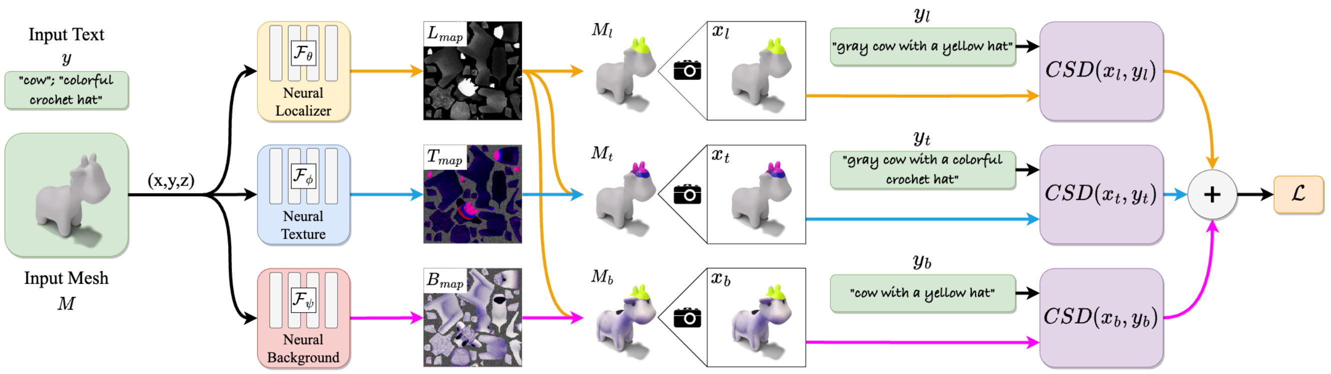

Figure 3. **Overview of 3D Paintbrush**. Each point on the surface of the mesh is passed into three different branches to produce a localization probability, texture map, and background map. We texture three different variants of the same mesh with the localization, texture, and background maps and render them from the same viewpoint. Each image along with the corresponding text condition is used to compute the CSD loss.

region and thus the localization has no visual meaning in isolation. Our approach imposes a visual loss on our localizations in order to enforce sharp boundaries that are tightly coupled with our texture edits. Additionally, since existing purely text-driven local editing approaches only work on voxels and NeRFs, our approach is the first to enable text-driven local editing on meshes.

High resolution text-to-3D. Several works have explored techniques to increase the resolution for text-to-3D. Many recent works apply SDS to latent diffusion models [32, 38, 58, 59, 66]. Recent works backpropagate the gradient through the encoder to get gradients in higher resolution 512x512 RGB space [32, 59, 66]. Other works use timestep annealing to give less noisy supervision towards the end of the optimization, thus increasing the detail of the generations [24, 59]. HiFA [66] proposes denoising over multiple successive timesteps each iteration to provide better gradients and achieve high fidelity appearance. While all of these approaches have shown impressive improvements to the resolution of SDS supervision, SDS only utilizes the base stage (not super-resolution stages). Thus, these proposed improvements are orthogonal to ours and can be incorporated at the super-resolution stages using CSD as well.

3. Method

We show an overview of our method in Fig. 3. The inputs to our system are a mesh M and a text description y of the desired local edit. Our system produces a local texture on the mesh M that adheres to the text prompt y. To supervise our optimization, we use score distillation with a pretrained text-to-image diffusion model. However, local editing requires higher detail than standard generation due to the small size and granularity of the desired edits. In order to further improve the detail of our localization and texture, we introduce Cascaded Score Distillation (CSD), a technique that distills scores at multiple resolutions of the 2D cascaded model. This approach enables leveraging all stages of a cascaded model and provides control over both the detail and global understanding of the supervision.

3.1. Local Neural Texturing

3D Paintbrush represents local textures as neural texture maps over the surface of a mesh M defined by vertices $V \in \mathbb{R}^{n\times3}$ and faces $F \in \{1, ..., n\}^{m\times3}$. Extracting an explicit texture map from our neural textures is trivial, making our representation compatible with existing graphics pipelines. Furthermore, using texture maps enables producing high resolution textures (i.e. sub-triangle values) without a computationally expensive high resolution mesh. A straight-forward approach of directly optimizing texture values results in texture maps with artifacts and noise (see supplemental material). To mitigate this, we leverage the smoothness of neural networks [46]. However, a straight-forward application of an MLP to a 2D texture map ($(u, v) \to (r, g, b)$) is inherently invalid at the texture seams (*e.g*. erroneous interpolations at boundaries), which may lead to texture discontinuities on the rendered mesh.

We instead formulate our MLPs to operate on 3D coordinates leading to predictions in 3D that are inherently smooth and without any seam discontinuities. To do so, we invert the UV mapping $\psi(x, y, z) = (u, v)$ to get a map $\psi^{-1}(u, v) = (x, y, z)$ from 2D texels to 3D coordinates on the surface of the mesh. We optimize our MLPs with the 3D coordinates obtained from the 2D texel centers. We employ two primary networks, one for localization and one for texturing. Our neural *localization* MLP is a function $\mathcal{F}_\theta$ that maps a 3D coordinate $\mathbf{x} = (x, y, z)$ to a probability p (which we map back to a 2D localization map). Similarly, our neural *texture* MLP is a function $\mathcal{F}_\phi$ that takes in a 3D coordinate and outputs an RGB value (which we map back to a 2D texture image). Our architecture first passes the 3D

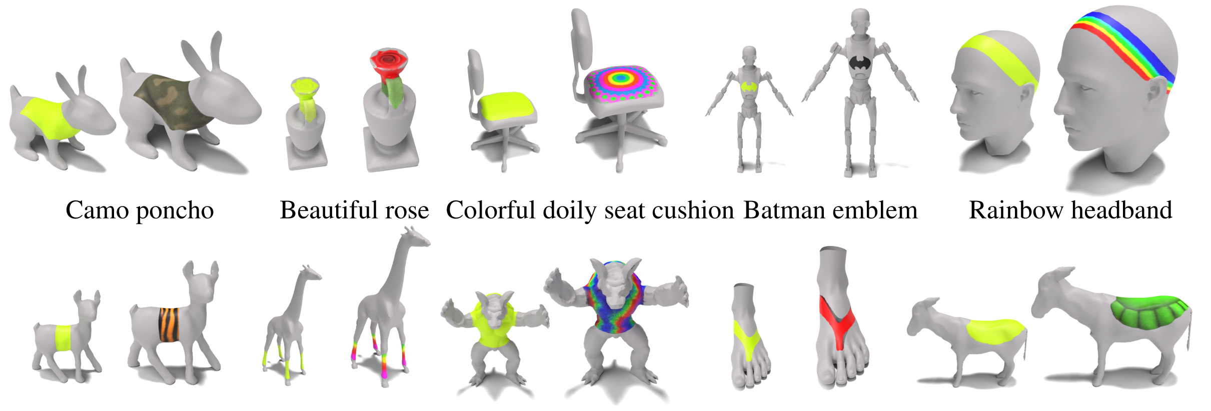

Figure 4. 3D Paintbrush produces highly detailed textures and localizations for a diverse range of meshes and prompts. Our method synthesizes meaningful local edits on shapes, demonstrating both global and local part-level understanding.

coordinates through positional encoding [52] before going through a 6-layer MLP. This formulation of using MLPs defined on the 3D surface leads to a neural texture which produces smoothly varying outputs in 3D, even though our 2D texture maps have discontinuities at the texture seams. The smoothness provided by the MLPs reduces artifacts, produces less noisy textures, and provides super resolution capabilities. Although we optimize our MLPs with 3D coordinates mapped from 2D texel centers, during inference, we may query the MLP for any value (*i.e.* sub-texels that enable super resolution texture maps even across seams).

3.2. Visual Guidance for Localized Textures

We guide our optimization using three distinct losses that encourage both the localization and texture towards visually desirable results. Each loss is visualized as a branch in Fig. 3 – top branch: localization loss, middle branch: local texture map loss, bottom branch: background loss.

Local texture map loss. First, we obtain our localization map $L_{map} \in [0,1]^{H\times W}$ from the neural localization MLP $L_{map} = \psi(\mathcal{F}_\theta(\mathbf{x}))$ and the texture map $T_{map} \in [0,1]^{H\times W\times 3}$ from the neural texture MLP $T_{map} = \psi(\mathcal{F}_\phi(\mathbf{x}))$. We use the localization L_{map} to mask the texture T_{map} to get a local texture map T'_{map} which only contains textures inside the localization region. We apply the masked texture T'_{map} to our mesh M to get a locally-textured mesh M_t and construct a local-texture text prompt y_t from the input text y (middle branch Fig. 3). We then supervise our optimization using a text-conditioned visual loss (cascaded score distillation, see Sec. 3.4) on M_t and y_t. By applying a visual loss to the localization-masked texture, we get informative and meaningful gradients for both our texture MLP and our localization MLP.

Localization loss. Using only the texture loss allows for trivial solutions where the mask contains a region that includes, but is much larger than, the desired localization region. To encourage the localization region to be meaningful, we employ a visual loss on the localization region in isolation (similar to 3D Highlighter [10]). Specifically, we blend a (yellow) color onto the mesh according to the localization map to get a localization-colored mesh M_l (top branch Fig. 3). From the text input y, we derive a target localization prompt y_l describing the localized region in the format used in 3D Highlighter [10]. We then use M_l and y_l as input to the text-conditioned visual loss. Using this loss significantly improves the detail and quality of the localization (see supplemental material).

Background loss. Using only the top two branches in Fig. 3 leads to broader localizations that incorporate superfluous elements characteristic of the input 3D model (*i.e.* a bill on a duck), in addition to the desired localization region (see supplemental material). To mitigate this, we learn a background texture $B_{map} \in [0,1]^{H\times W\times 3}$ that intentionally contains these characteristic elements of the input 3D shape in the inverse of the localization region $1 - L_{map}$ (the area outside the localization region). Specifically, we blend both the background texture B_{map} (using $1 - L_{map}$) and a yellow color (using L_{map}) to get a composited texture $B'_{map} = L_{map}(\text{YELLOW}) + (1 - L_{map})B_{map}$ (bottom branch in Fig. 3). We apply the composited texture B'_{map} to the mesh to get M_b and then supervise the background MLP using a visual loss conditioned on both M_b and a target text y_b (derived from y). The target text y_b describes the generic object class (*i.e.* 'cow' in Fig. 3) with a (yellow) colored localization region. See supplemental material for more details. The third loss directly encourages incorporating the superfluous elements in the background texture which *discourages* the localization region from incorporating such undesired elements (since L_{map} and $1 - L_{map}$ are inverse masks).

Key to our method is the simultaneous optimization of the localization map (that specifies the edit region) *and* the

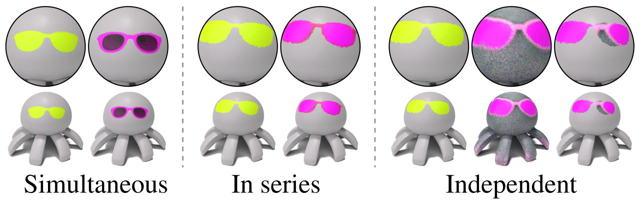

Figure 5. **Impact of simultaneous optimization**. Simultaneously optimizing the localization and texture (left) results in higher-detailed textures which effectively conform to the predicted localization. If we first optimize the localization, then optimize the texture within the localization region (middle), both the localization and texture are less detailed. Independent (right): if we optimize the localization independently (independent: left) and the texture independently (independent: middle), the texture does not align with the localization and thus the masked texture contains fringe artifacts (independent: right).

texture map that conforms to it. This approach improves the quality of both the localization and the stylization. The texture map drives the localization to become more detailed and intricate, while the localization explicitly masks the texture, ensuring a coherent local style which respects the localization boundary (see Fig. 5).

3.3. Score Distillation and Cascaded Diffusion

Score Distillation. To guide our local stylization, we leverage powerful pretrained text-to-image diffusion models. Existing approaches use these models in conjunction with Score Distillation Sampling (SDS) to supervise text-driven optimizations [44, 58]. For each iteration of an optimization of an image x that we want to supervise with diffusion model ϕ and text prompt y, SDS [44] proposes the following gradient:

$$\nabla_x \mathcal{L}_{SDS}(\phi, x, y) = w(t)(\epsilon_\phi(z_t, t, y) - \epsilon) \tag{1}$$

where timestep $t \sim \mathcal{U}(\{1, \dots, T\})$ is sampled uniformly and noise $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ is Gaussian. The noisy image z_t is obtained by applying a timestep-dependent scaling of ϵ to the image x. The weight $w(t)$ is a timestep-dependent weighting function and $\epsilon_\phi(z_t, t, y)$ is the noise predicted by the diffusion model conditioned on z_t, t, and y. Note that Eq. (1) omits the U-Net Jacobian term (not needed in practice [44]). This objective is similar to the objective used in diffusion model training, however, instead of optimizing the weights of the model, the gradient is applied to the image x.

Cascaded Diffusion. Text-to-image diffusion models often contain multiple cascaded stages at different resolutions in order to achieve high resolution outputs [21]. These cascaded diffusion models consist of a base stage ϕ^1 (stage 1) and some number of super-resolution stages $\phi^{i>1}$ (stages 2-N). The base stage is identical to a standard diffusion model, predicting noise $\epsilon_{\phi^1}(z_t^1, t, y)$ conditioned on noisy image z_t^1, timestep t, and text prompt y. However, the super-resolution stages are conditioned on two differently-noised images: one at the current resolution (z_t^i with timestep t and noise ϵ^i) and one at the lower resolution (z_s^{i-1} with timestep s and noise ϵ^{i-1}). The predicted noise for the super-resolution stage is given by $\epsilon_{\phi^i}(z_t^i, t, z_s^{i-1}, s, y)$. During inference, the lower resolution input image is obtained by adding noise to the output of the prior stage. However in training, both the high and low resolution images are obtained by sampling a single image from the training dataset and rescaling it to different resolutions.

Standard SDS [44] only utilizes the first, low-resolution base stage, thus neglecting the full potential of the cascaded model. It is not immediately obvious how to formulate a score distillation technique for all stages of a cascaded diffusion model since super-resolution stages take multiple resolution inputs and, at inference, they require a fully denoised output from the prior stage [21]. We take inspiration from SDS and use the perspective of *diffusion training* as opposed to inference, and extend it to the training of cascaded diffusion models. To our knowledge, we are the first to consider score distillation using the cascaded super-resolution stages.

3.4. Cascaded Score Distillation

CSD overview. Our technique, referred to as Cascaded Score Distillation (CSD), simultaneously distills scores at multiple resolutions in a cascaded fashion (illustrated in Fig. 6). Since the stages of a cascaded diffusion model ϕ are trained entirely independently of one another, our insight is to formulate a distillation loss that incorporates gradients from all stages $(\phi^1, ..., \phi^N)$ simultaneously. We observe that different stages of the cascaded model provide different levels of granularity and global understanding (Fig. 7). Controlling the influence of each stage provides control over the details and the corresponding localization of the supervision (Fig. 8).

CSD Formalization. Consider a mesh M_θ with a neural texture parameterized by an MLP θ (This MLP could be either $\mathcal{F}_\theta$, $\mathcal{F}_\phi$, and $\mathcal{F}_\psi$ in Sec. 3.2). We first render M_θ at N different resolutions using a differentiable renderer g to get multiple images $g(M_\theta) = \mathbf{x} = \{x^1 \dots x^N\}$ such that x^i is the same resolution as stage ϕ^i. For the base stage ϕ^1, we perform standard SDS using Eq. (1) on x^1 and prompt y to get a gradient ∇_{x^1}. For all stages ϕ^i for $i > 1$, we sample two timesteps $t, s \sim \mathcal{U}(\{1, \dots, T\})$, noise $\epsilon^i \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ at the resolution of stage ϕ^i, and noise $\epsilon^{i-1} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ at the resolution of stage ϕ^{i-1}. Using timestep-dependent schedule coefficients α and σ, we compute a noisy image $z_t^i = \alpha_t x^i + \sigma_t \epsilon^i$ by applying a timestep-dependent scaling of ϵ^i to the image x^i. Similarly, we compute $z_s^{i-1} = \alpha_s x^{i-1} + \sigma_s \epsilon^{i-1}$ by applying a timestep-dependent scaling of ϵ^{i-1} to the image x^{i-1}. We then use

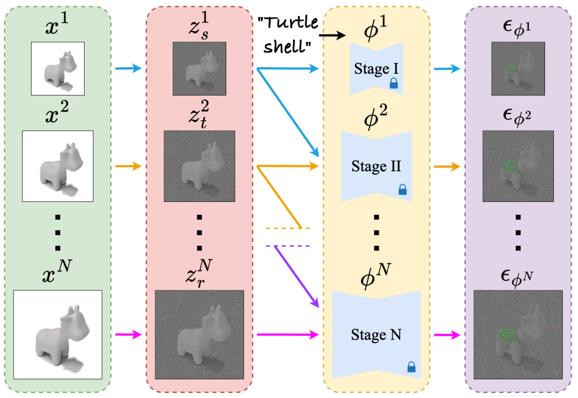

Figure 6. **Cascaded Score Distillation (CSD)**. We simultaneously distill scores across multiple stages of a cascaded diffusion model in order to leverage both the global awareness of the first stage and the higher level of detail contained in later stages. The difference between the predicted noise and sampled noise is the image gradient for each stage.

ϕ^i to predict noise $\epsilon_{\phi^i}(z_t^i, t, z_s^{i-1}, s, y)$ conditioned on the noisy images, timesteps, and text prompt. Our gradient ∇_{x^i} for stage ϕ^i for $i > 1$ is the difference between the predicted noise and the (higher-resolution) sampled noise ϵ^i, weighted by the timestep-dependent function $w(t)$:

$$\nabla_{x^i}\mathcal{L}_{CSD^i}(\phi^i, x^i, x^{i-1}, y) = w(t)(\epsilon_{\phi^i}(z_t^i, t, z_s^{i-1}, s, y) - \epsilon^i). \tag{2}$$

With all gradients $\nabla_{x^1}, \ldots, \nabla_{x^N}$ computed, we weight each gradient ∇_{x^i} with a user defined λ^i to provide control over the impact of the supervision from each stage of the cascaded model. Thus our full gradient with respect to any given neural texture θ can be described by:

$$\nabla_\theta\mathcal{L}_{CSD}(\boldsymbol{\phi}, \mathbf{x} = g(\theta), y) = \lambda^1\nabla_{x^1}\mathcal{L}_{SDS}(\phi^1, x^1, y)\frac{\partial x^1}{\partial\theta} + \sum_{i=2}^{N}\lambda^i\nabla_{x^i}\mathcal{L}_{CSD^i}(\phi^i, x^i, x^{i-1}, y)\frac{\partial x^i}{\partial\theta}. \tag{3}$$

Note that just as in SDS [44], we can avoid computing the U-Net Jacobian term $\frac{\partial\epsilon_\phi(z_t^i, t, z_s^{i-1}, s, y)}{z_t^i}$ (not shown in Eq. (3)) since each stage is entirely independent and our gradient is only with respect to the high-resolution image x^i. Thus, we directly apply $\lambda^i\nabla_{x^i}$ to the image x^i without having to compute the costly backpropagation through the U-Net. Using the gradient $\nabla_\theta\mathcal{L}_{CSD}(\boldsymbol{\phi}, \mathbf{x} = g(\theta), y)$, we update the weights of our MLP θ.

4. Experiments

We demonstrate the capabilities of 3D Paintbrush on a wide variety of meshes (from different sources [55, 56, 61, 65]) and prompts. We highlight key properties of our method such as localization precision and edit specificity. We then demonstrate the importance and capabilities of our CSD loss including its high resolution supervision and intuitive controls. Finally, we evaluate our system against other localization and editing baselines and ablate the key components of our method. In our experiments, we use DeepFloyd IF [3] for our cascaded model. Our unoptimized PyTorch [43] implementation takes 4 hours on a standard A40 GPU, typically achieving satisfactory results within 2 hours.

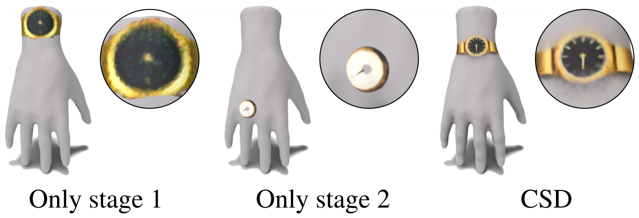

Figure 7. **Impact of cascaded stages**. Different stages of the cascaded model provide different levels of granularity and global understanding. Using only the (low resolution) stage 1 model gives a low-resolution result in the correct location. While the (high resolution) stage 2 model gives a high-resolution result, it is placed in the incorrect location. Our CSD simultaneously uses stage 1 and 2, resulting in a highly-detailed texture in the appropriate location.

4.1. Properties of 3D Paintbrush

3D Paintbrush generality. 3D Paintbrush is capable of producing highly detailed localizations and textures on a diverse collection of meshes and prompts (Fig. 4). Our method is not restricted to any category of meshes and we show results on organic and manufactured shapes. Furthermore, our local textures can be specified with open vocabulary text descriptions and are not limited to any predefined categories or constraints. This includes "out-of-domain" local textures such as the rainbow shinguards on a giraffe which are not naturally seen in the context of these objects, yet are precisely placed in semantically meaningful locations with highly detailed textures.

3D Paintbrush precision and composition. 3D Paintbrush produces precise localizations and highly-detailed textures that effectively adhere to these predicted localizations (see Fig. 2). The tight coupling between the localization and texture (see the gold chain necklace in Fig. 1) enables seamless composition of multiple local textures simultaneously on the same mesh without any layering artifacts. For example, the sharp localization boundary of the "Tie-dye apron" (in Fig. 2) allows us to composite this local texture on top of other textures without obstructing these textures in regions outside of the apron's boundary.

3D Paintbrush specificity and effectiveness. 3D Paintbrush produces accurate and high resolution local edits that closely adhere to the text-specification (see Fig. 10). Our

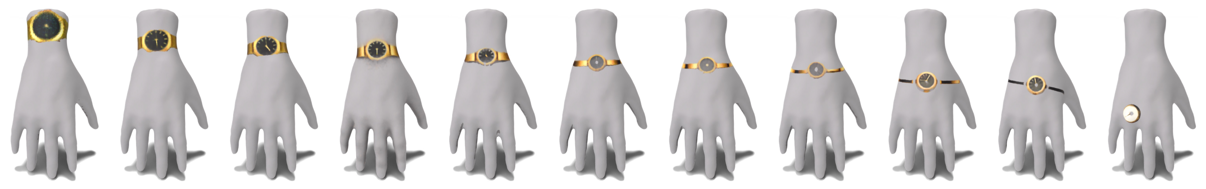

Figure 8. **Granular control with CSD**. Varying the weight between stage 1 and stage 2 results in control over the details and corresponding localization. Only using stage 1 (leftmost) is rather coarse; only using stage 2 (rightmost) is highly detailed with an incorrect localization. Increasing the stage 2 weight (moving left to right) progressively increases the detail and granularity of the supervision, enabling smooth and meaningful interpolation between stage 1 and 2.

method's fine-grained results contain intricate details (*i.e.* the badge on "Barcelona jersey") and reflect the subtle differences in the text prompts (*i.e.* the "cape" on the dog is more tapered than the boxier "poncho"). This specificity allows us to produce many diverse and distinct local styles. We show multiple local edits on the same mesh for multiple different meshes, demonstrating the effectiveness of our method on diverse prompts and meshes.

4.2. Importance of Cascaded Score Distillation

Impact and granular control of CSD. Our cascaded score distillation (CSD) simultaneously distills scores at multiple resolutions in a cascaded fashion. We observe that different stages of the cascaded diffusion model give different levels of granularity and global understanding (Fig. 7). Using only the (low resolution) stage 1 model is equivalent to SDS. Though SDS produces an accurate localization and coherent texture, the result is low-resolution (see Fig. 9). Conversely, using only the (high resolution) stage 2 model gives a high-resolution result, but often fails to properly localize the texture leading to undesirable results. Our CSD simultaneously combines the supervision from stages 1 and 2, resulting in a highly-detailed texture in the appropriate location. Increasing the stage 2 weight (moving left to right in Fig. 8) progressively increases the detail and granularity of the supervision, demonstrating smooth and intuitive interpolation between stage 1 and 2. In our experiments, we use a fixed weighting scheme, but this result demonstrates that our method works for a broad range of weights. Quantitative evidence supporting the importance of the CSD loss can bee seen in Tab. 1.

Localization Average Score ↑		SATR 1.89	3D Highlighter 2.03	**Ours 4.80**
Local Edits Average Score ↑	**Latent Paint 2.14**	**Vox-E 2.15**	**Ours (SDS) 4.06**	**Ours 4.88**

Table 1. **Quantitative evaluation.** We conduct a perceptual study where users evaluate our localizations and local edits compared to baseline methods (3D Highlighter [10], SATR [2], Latent Paint [38], Vox-E [49], and our method with standard SDS loss).

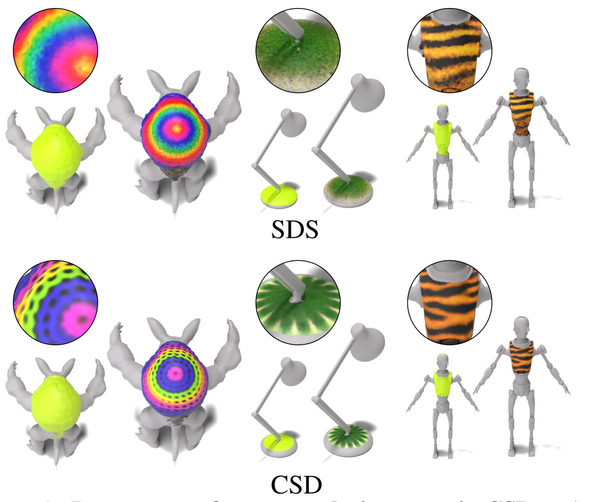

Figure 9. **Importance of super-resolution stage in CSD.** Using stage 1 only (equivalent to SDS) lacks fine-grained details. Incorporating the second super-resolution cascaded stage from our CSD increases the resolution and detail. Input text prompts (from left to right): Colorful crochet shell, Cactus base, Tiger stripe shirt.

4.3. Evaluation

Simultaneous localization and texture. We demonstrate the importance of simultaneously optimizing the localization region and texture in tandem in Fig. 5. We observe that simultaneous optimization results in highly detailed textures which effectively conform to the predicted localization regions (Fig. 5, left). Furthermore, the resulting localization region is sharp and intricate. Alternatively, we optimize the localization region first and use the predicted localization to learn a texture which is confined to the (precomputed) localization region (Fig. 5, middle). In this case, the texture is less detailed, and the localization region is less intricate. Finally, we can learn the texture and localization region independently (Fig. 5, independent). This results in a texture (Fig. 5 independent, middle) that is completely decoupled from the localization region (Fig. 5 independent, left). When masking the texture with the localization region, we observe a misaligned texture with fringe artifacts (Fig. 5 independent, right).

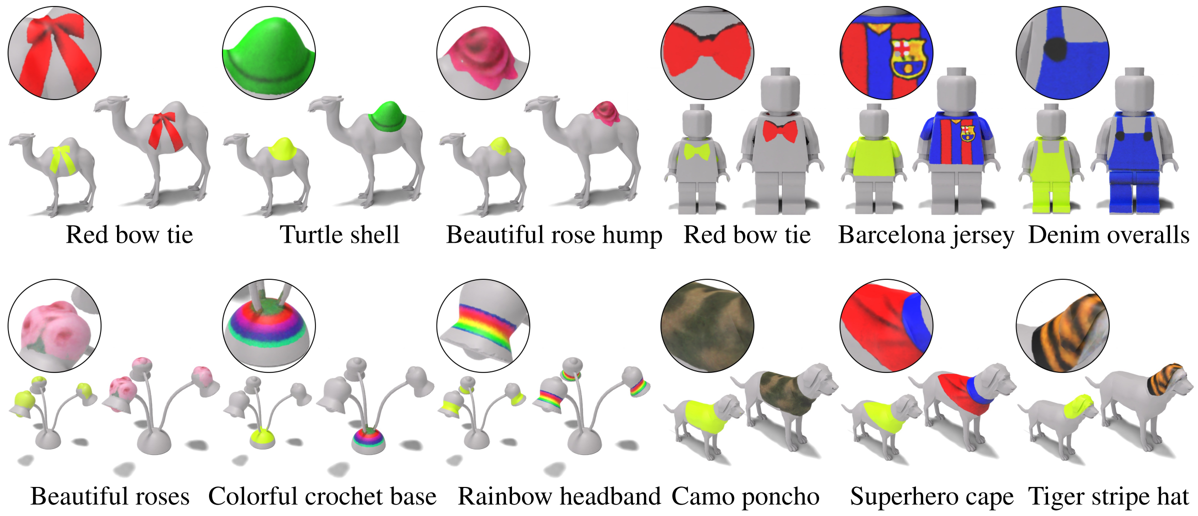

Figure 10. 3D Paintbrush is capable of producing a variety of local textures on the same mesh. Each result contains an accurate localization map (to specify the edit region) and a texture map that conforms to it.

Quantitative evaluation. 3D Paintbrush is the only method geared towards local editing that natively operates on meshes. We compare to the closest mesh-based methods which perform localization (3D Highlighter [10], SATR [2]) and texturing (Latent Paint [38]). We also compare to a voxel NeRF approach for local 3D editing (Vox-E [49]).

To evaluate our method against these baselines, we conduct a perceptual study where 39 users rate the effectiveness of each method for 9 different meshes (see Tab. 1). 3D Paintbrush consistently scores the highest for both localization and local editing, producing sharper localizations than 3D Highlighter and SATR and higher resolution textures than Latent Paint and Vox-E. Further quantitative evaluation using CLIP R-Precision and qualitative comparisons to these baselines are shown in the supplemental material.

Limitations. We illustrate a limitation of our method in Fig. 11. In cases where the desired local texture has strong semantic connections to additional components, these auxiliary components can sometimes be included in the localization and local texture. For example, a "Pharaoh headdress" is closely associated with Egyptian necklaces and thus our method also localizes and styles this component as well. Our method also suffers from the Janus effect common to many text-to-3D methods that use 2D supervision.

5. Conclusion

We presented 3D Paintbrush, a technique that produces highly detailed texture maps on meshes which effectively adhere to a predicted localization region. Our system is capable of *hallucinating* non-obvious local textures on a wide variety of meshes (such as heart-shaped sunglasses on a cow). Our localizations are detailed and accurate, enabling seamless post-processing (such as compositing textures without unwanted fringe). We proposed cascaded score distillation, a technique capable of extracting supervision signals from multiple stages of a cascaded diffusion model. We observe that each stage controls different amounts of detail and global understanding. Further, varying the weights for each stage provides control over the resulting local textures. We show the effectiveness of CSD to locally texture meshes; yet, CSD is general and can be applied to other domains (such as images, videos, and alternative 3D representations). In the future, we are interested in extending localized editing to capabilities beyond texturing (such as deformations, normal maps, and more).

6. Acknowledgments

We thank the University of Chicago for providing the AI cluster resources, services, and the professional support of the technical staff. This work was also supported in part by gifts from Snap Research, Adobe Research, Google Research, BSF grant 2022363, and NSF grants 2304481 and 2241303. Finally, we would like to thank Brian Kim, Jack Zhang, Haochen Wang, and the members of 3DL and PALS for their thorough and insightful feedback on our work.

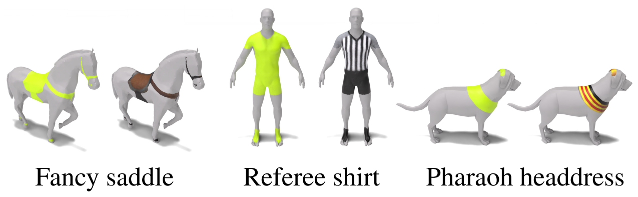

Figure 11. In cases where the desired localization carries a strong semantic context, elements from that context can also appear in the localization and style. For example, when adding a pharaoh headdress, 3D Paintbrush also adds an Egyptian necklace since they are commonly associated with pharaohs.

References

[1] Ahmed Abdelreheem, Abdelrahman Eldesokey, Maks Ovsjanikov, and Peter Wonka. Zero-shot 3d shape correspondence. In *SIGGRAPH Asia 2023 Conference Papers*, pages 1–11, 2023. 2

[2] Ahmed Abdelreheem, Ivan Skorokhodov, Maks Ovsjanikov, and Peter Wonka. Satr: Zero-shot semantic segmentation of 3d shapes. In *ICCV*, 2023. 2, 7, 8, 1

[3] Stability AI. Deepfloydif, 2023. 6

[4] Sudarshan Babu, Richard Liu, Avery Zhou, Michael Maire, Greg Shakhnarovich, and Rana Hanocka. Hyperfields: Towards zero-shot generation of nerfs from text. *arXiv preprint arXiv:2310.17075*, 2023. 2

[5] Omer Bar-Tal, Dolev Ofri-Amar, Rafail Fridman, Yoni Kasten, and Tali Dekel. Text2live: Text-driven layered image and video editing. In *European conference on computer vision*, pages 707–723. Springer, 2022. 2

[6] Alexey Bokhovkin, Shubham Tulsiani, and Angela Dai. Mesh2tex: Generating mesh textures from image queries. *arXiv preprint arXiv:2304.05868*, 2023. 2

[7] Tim Brooks, Aleksander Holynski, and Alexei A Efros. Instructpix2pix: Learning to follow image editing instructions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18392–18402, 2023. 2

[8] Dave Zhenyu Chen, Yawar Siddiqui, Hsin-Ying Lee, Sergey Tulyakov, and Matthias Nießner. Text2tex: Text-driven texture synthesis via diffusion models. In *ICCV*, 2023. 2

[9] Rui Chen, Yongwei Chen, Ningxin Jiao, and Kui Jia. Fantasia3d: Disentangling geometry and appearance for high-quality text-to-3d content creation. *arXiv preprint arXiv:2303.13873*, 2023. 2

[10] Dale Decatur, Itai Lang, and Rana Hanocka. 3d highlighter: Localizing regions on 3d shapes via text descriptions. In *CVPR*, 2023. 2, 4, 7, 8, 1, 3

[11] Zhiwen Fan, Yifan Jiang, Peihao Wang, Xinyu Gong, Dejia Xu, and Zhangyang Wang. Unified implicit neural stylization. In *European Conference on Computer Vision*, pages 636–654. Springer, 2022. 2

[12] Rao Fu, Xiao Zhan, Yiwen Chen, Daniel Ritchie, and Srinath Sridhar. Shapecrafter: A recursive text-conditioned 3d shape generation model. *Advances in Neural Information Processing Systems*, 35:8882–8895, 2022. 2

[13] 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. 2

[14] 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. In *Advances In Neural Information Processing Systems*, 2022. 2

[15] Lin Gao, Tong Wu, Yu-Jie Yuan, Ming-Xian Lin, Yu-Kun Lai, and Hao Zhang. Tm-net: Deep generative networks for textured meshes. *ACM Transactions on Graphics (TOG)*, 40 (6):1–15, 2021. 2

[16] William Gao, Noam Aigerman, Thibault Groueix, Vova Kim, and Rana Hanocka. Textdeformer: Geometry manipulation using text guidance. In *ACM SIGGRAPH 2023 Conference Proceedings*, pages 1–11, 2023. 2

[17] Huy Ha and Shuran Song. Semantic abstraction: Open-world 3D scene understanding from 2D vision-language models. In *Proceedings of the 2022 Conference on Robot Learning*, 2022. 2

[18] Ayaan Haque, Matthew Tancik, Alexei A Efros, Aleksander Holynski, and Angjoo Kanazawa. Instruct-nerf2nerf: Editing 3d scenes with instructions. *ICCV*, 2023. 2

[19] Amir Hertz, Rana Hanocka, Raja Giryes, and Daniel Cohen-Or. Deep geometric texture synthesis. *ACM Transactions on Graphics (TOG)*, 39(4):108–1, 2020. 2

[20] Amir Hertz, Ron Mokady, Jay Tenenbaum, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Prompt-to-prompt image editing with cross attention control. *arXiv preprint arXiv:2208.01626*, 2022. 2

[21] Jonathan Ho, Chitwan Saharia, William Chan, David J Fleet, Mohammad Norouzi, and Tim Salimans. Cascaded diffusion models for high fidelity image generation. *The Journal of Machine Learning Research*, 23(1):2249–2281, 2022. 2, 5

[22] Lukas Höllein, Justin Johnson, and Matthias Nießner. Stylemesh: Style transfer for indoor 3d scene reconstructions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6198–6208, 2022. 2

[23] Jingwei Huang, Justus Thies, Angela Dai, Abhijit Kundu, Chiyu Jiang, Leonidas J Guibas, Matthias Nießner, Thomas Funkhouser, et al. Adversarial texture optimization from rgb-d scans. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1559–1568, 2020. 2

[24] Yukun Huang, Jianan Wang, Yukai Shi, Xianbiao Qi, Zheng-Jun Zha, and Lei Zhang. Dreamtime: An improved optimization strategy for text-to-3d content creation. *arXiv preprint arXiv:2306.12422*, 2023. 3

[25] Ajay Jain, Ben Mildenhall, Jonathan T Barron, Pieter Abbeel, and Ben Poole. Zero-shot text-guided object generation with dream fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 867–876, 2022. 2, 1

[26] Oren Katzir, Or Patashnik, Daniel Cohen-Or, and Dani Lischinski. Noise-free score distillation, 2023. 2

[27] Justin Kerr, Chung Min Kim, Ken Goldberg, Angjoo Kanazawa, and Matthew Tancik. Lerf: Language embedded radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 19729–19739, 2023. 2

[28] Sosuke Kobayashi, Eiichi Matsumoto, and Vincent Sitzmann. Decomposing nerf for editing via feature field distillation. In *Advances in Neural Information Processing Systems*, 2022. 2

[29] Han-Hung Lee and Angel X Chang. Understanding pure clip guidance for voxel grid nerf models. *arXiv preprint arXiv:2209.15172*, 2022. 2

[30] Jiabao Lei, Yabin Zhang, Kui Jia, et al. Tango: Text-driven photorealistic and robust 3d stylization via lighting decomposition. *Advances in Neural Information Processing Systems*, 35:30923–30936, 2022. 2

[31] Yuhan Li, Yishun Dou, Yue Shi, Yu Lei, Xuanhong Chen, Yi Zhang, Peng Zhou, and Bingbing Ni. Focaldreamer: Text-driven 3d editing via focal-fusion assembly. *arXiv preprint arXiv:2308.10608*, 2023. 2

[32] Chen-Hsuan Lin, Jun Gao, Luming Tang, Towaki Takikawa, Xiaohui Zeng, Xun Huang, Karsten Kreis, Sanja Fidler, Ming-Yu Liu, and Tsung-Yi Lin. Magic3d: High-resolution text-to-3d content creation. In *CVPR*, 2023. 1, 2, 3

[33] Hsueh-Ti Derek Liu, Vladimir G. Kim, Siddhartha Chaudhuri, Noam Aigerman, and Alec Jacobson. Neural subdivision. *ACM Trans. Graph.*, 39(4), 2020. 2

[34] Kunhao Liu, Fangneng Zhan, Yiwen Chen, Jiahui Zhang, Yingchen Yu, Abdulmotaleb El Saddik, Shijian Lu, and Eric P Xing. Stylerf: Zero-shot 3d style transfer of neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8338–8348, 2023. 2

[35] Minghua Liu, Chao Xu, Haian Jin, Linghao Chen, Zexiang Xu, Hao Su, et al. One-2-3-45: Any single image to 3d mesh in 45 seconds without per-shape optimization. *arXiv preprint arXiv:2306.16928*, 2023. 2

[36] Ruoshi Liu, Rundi Wu, Basile Van Hoorick, Pavel Tokmakov, Sergey Zakharov, and Carl Vondrick. Zero-1-to-3: Zero-shot one image to 3d object, 2023. 2

[37] Yiwei Ma, Xiaoqing Zhang, Xiaoshuai Sun, Jiayi Ji, Haowei Wang, Guannan Jiang, Weilin Zhuang, and Rongrong Ji. X-mesh: Towards fast and accurate text-driven 3d stylization via dynamic textual guidance. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2749–2760, 2023. 2

[38] Gal Metzer, Elad Richardson, Or Patashnik, Raja Giryes, and Daniel Cohen-Or. Latent-nerf for shape-guided generation of 3d shapes and textures. In *CVPR*, 2023. 1, 2, 3, 7, 8

[39] Oscar Michel, Roi Bar-On, Richard Liu, Sagie Benaim, and Rana Hanocka. Text2mesh: Text-driven neural stylization for meshes. In *CVPR*, pages 13492–13502, 2022. 2

[40] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020. 2

[41] Nasir Mohammad Khalid, Tianhao Xie, Eugene Belilovsky, and Tiberiu Popa. Clip-mesh: Generating textured meshes from text using pretrained image-text models. In *SIGGRAPH Asia 2022 conference papers*, pages 1–8, 2022. 2, 1

[42] Michael Oechsle, Lars Mescheder, Michael Niemeyer, Thilo Strauss, and Andreas Geiger. Texture fields: Learning texture representations in function space. In *CVPR*, pages 4531–4540, 2019. 2

[43] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017. 6

[44] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. In *ICLR*, 2023. 2, 5, 6, 1

[45] Guocheng Qian, Jinjie Mai, Abdullah Hamdi, Jian Ren, Aliaksandr Siarohin, Bing Li, Hsin-Ying Lee, Ivan Skorokhodov, Peter Wonka, Sergey Tulyakov, et al. Magic123: One image to high-quality 3d object generation using both 2d and 3d diffusion priors. *arXiv preprint arXiv:2306.17843*, 2023. 2

[46] Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. In *International conference on machine learning*, 2019. 3

[47] Elad Richardson, Gal Metzer, Yuval Alaluf, Raja Giryes, and Daniel Cohen-Or. Texture: Text-guided texturing of 3d shapes. In *ACM TOG*, 2023. 2

[48] Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22500–22510, 2023. 1, 2

[49] Etai Sella, Gal Fiebelman, Peter Hedman, and Hadar Averbuch-Elor. Vox-e: Text-guided voxel editing of 3d objects. In *ICCV*, 2023. 1, 2, 7, 8

[50] Yichun Shi, Peng Wang, Jianglong Ye, Mai Long, Kejie Li, and Xiao Yang. Mvdream: Multi-view diffusion for 3d generation. *arXiv preprint arXiv:2308.16512*, 2023. 2

[51] Yawar Siddiqui, Justus Thies, Fangchang Ma, Qi Shan, Matthias Nießner, and Angela Dai. Texturify: Generating textures on 3d shape surfaces. In *European Conference on Computer Vision*, pages 72–88. Springer, 2022. 2

[52] Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. 2020. 4

[53] Christina Tsalicoglou, Fabian Manhardt, Alessio Tonioni, Michael Niemeyer, and Federico Tombari. Textmesh: Generation of realistic 3d meshes from text prompts. *arXiv preprint arXiv:2304.12439*, 2023. 1, 2

[54] Vadim Tschernezki, Iro Laina, Diane Larlus, and Andrea Vedaldi. Neural feature fusion fields: 3d distillation of self-supervised 2d image representations. In *2022 International Conference on 3D Vision (3DV)*, pages 443–453. IEEE, 2022. 2

[55] TurboSquid. Turbosquid 3d model repository, 2021. https://www.turbosquid.com/. 6

[56] Oliver van Kaick, Andrea Tagliasacchi, Oana Sidi, Hao Zhang, Daniel Cohen-Or, Lior Wolf, and Ghassan Hamarneh. Prior knowledge for part correspondence. *Computer Graphics Forum*, 30(2):553–562, 2011. 6

[57] Suhani Vora, Noha Radwan, Klaus Greff, Henning Meyer, Kyle Genova, Mehdi S. M. Sajjadi, Etienne Pot, Andrea Tagliasacchi, and Daniel Duckworth. Nesf: Neural semantic fields for generalizable semantic segmentation of 3d scenes, 2021. 2

[58] Haochen Wang, Xiaodan Du, Jiahao Li, Raymond A. Yeh, and Greg Shakhnarovich. Score jacobian chaining: Lifting pretrained 2d diffusion models for 3d generation. In *CVPR*, 2023. 1, 2, 3, 5

[59] Zhengyi Wang, Cheng Lu, Yikai Wang, Fan Bao, Chongxuan Li, Hang Su, and Jun Zhu. Prolificdreamer: High-fidelity and diverse text-to-3d generation with variational score distillation. *arXiv preprint arXiv:2305.16213*, 2023. 1, 2, 3

[60] Xingkui Wei, Zhengqing Chen, Yanwei Fu, Zhaopeng Cui, and Yinda Zhang. Deep hybrid self-prior for full 3d mesh generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5805–5814, 2021. 2

[61] 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. 6

[62] Kangxue Yin, Jun Gao, Maria Shugrina, Sameh Khamis, and Sanja Fidler. 3dstylenet: Creating 3d shapes with geometric and texture style variations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12456–12465, 2021. 2

[63] 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. 2

[64] Kai Zhang, Nick Kolkin, Sai Bi, Fujun Luan, Zexiang Xu, Eli Shechtman, and Noah Snavely. Arf: Artistic radiance fields. In *European Conference on Computer Vision*, pages 717–733. Springer, 2022. 2

[65] Qingnan Zhou and Alec Jacobson. Thingi10k: A dataset of 10,000 3d-printing models. *arXiv preprint arXiv:1605.04797*, 2016. 6

[66] Joseph Zhu and Peiye Zhuang. Hifa: High-fidelity text-to-3d with advanced diffusion guidance. *arXiv preprint arXiv:2305.18766*, 2023. 2, 3

[67] Jingyu Zhuang, Chen Wang, Lingjie Liu, Liang Lin, and Guanbin Li. Dreameditor: Text-driven 3d scene editing with neural fields. In *SIGGRAPH Asia*, 2023. 1, 2

Supplementary Material for 3D Paintbrush: Local Stylization of 3D Shapes with Cascaded Score Distillation

A. Additional Experiments

Variation. For a given mesh and prompt, there are often multiple valid interpretations for the local edit. With different seeds, our method produces various diverse, yet plausible, results (see Fig. 12) allowing users to choose the output that best matches their desired edit.

Qualitative comparisons. We show qualitative comparisons of our method to other localization and editing techniques. For our localizations, we compare to the mesh-native approaches 3D Highlighter [10] and SATR [2]. Our localizations are more accurate and more highly-detailed than 3D Highlighter and SATR (see Fig. 13). For example, the position and shape of our apron more closely adheres to that of an apron and our necklace contains the fine-grained chain links whereas both 3D Highlighter and SATR produce an overly smooth band. Since our method is the first approach to perform local edits on meshes, we evaluate our local editing capabilities by comparing to the mesh-native global editing method Latent Paint [38] and the voxel-based NeRF local editing method Vox-E [49]. In Fig. 14, we observe that our method produces local stylizations with higher resolution textures than Latent Paint [38] and Vox-E [49]. Furthermore, our texture edits are contained to a local semantic region corresponding to the target edit. In contrast, Vox-E often makes changes to parts of the texture that are unrelated to the target edit and Latent Paint consistently makes changes to the entire texture.

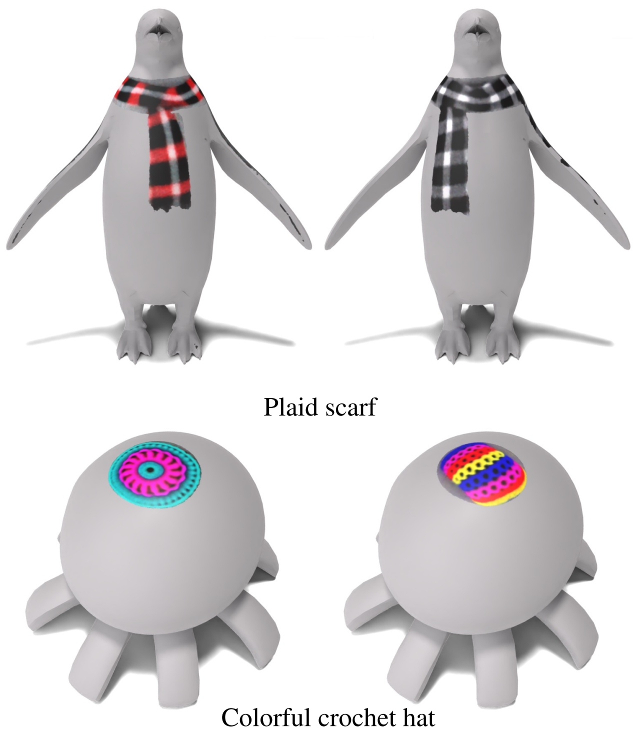

Figure 12. **Variation**. 3D Paintbrush can give multiple varied, yet plausible, outputs for a given local style depending on the input seed. We show results on two different seeds for each local style.

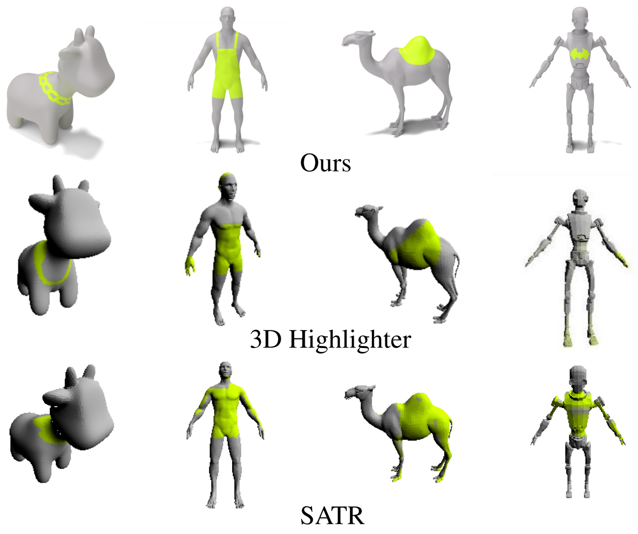

Figure 13. **Qualitative localization comparison**. Our localizations are more accurate and finer-grained than 3D Highlighter [10] and SATR [2]. Full input text prompts (from left to right): a 3D render of a gray cow with a yellow chain link necklace, a 3D render of a gray person with a yellow apron, a 3D render of a gray camel with a yellow turtle shell, a 3D render of a gray robot with a yellow batman chest emblem.

Method	Clip R-Precision ↑					
	CLIP B/32		CLIP B/16		CLIP L/14	
	Loc	Style	Loc	Style	Loc	Style
3DH	13.3	n/a	7.0	n/a	23.7	n/a
LatentPaint	n/a	20.0	n/a	20.0	n/a	20.0
Ours	**26.7**	**60.0**	**13.0**	**40.0**	**46.7**	**73.3**

Table 2. Quantitative evaluation. We compare our localizations to 3D Highlighter [10] (3DH) and our local textures to Latent Paint [38] and report CLIP R-Precision. Note, 3DH does not produce stylizations and Latent Paint does not produce localizations.

Quantitative evaluation with CLIP R-Precision. We compare our method to mesh-native baselines using CLIP R-Precision, an automated method for measuring alignment between generations and text prompts commonly used in text-to-3D [10, 25, 41, 44]. Given a set of text prompts

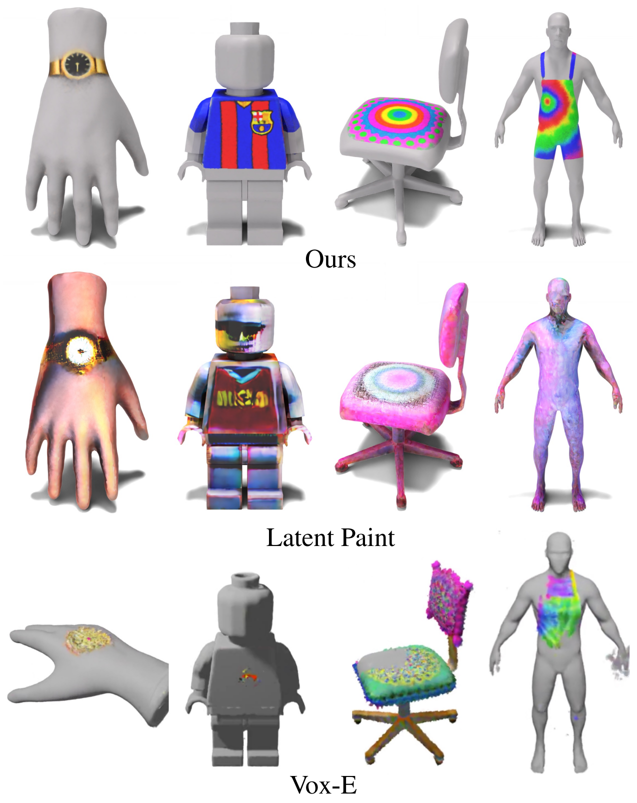

Figure 14. **Qualitative local stylization comparison**. Our 3D Paintbrush gives more highly detailed and local edits than Latent Paint [38] and Vox-E [49]. Full input text prompts (from left to right): a 3D render of a gray hand with a fancy gold watch, a 3D render of a gray Lego minifigure with a Barcelona jersey, a 3D render of a gray desk chair with a colorful doily seat cushion, a 3D render of a gray person with a tie-dye apron.

and corresponding generated textured meshes, this metric reports the percentage with which CLIP is able to retrieve the correct text prompt used to generate each given textured mesh. 3D Paintbrush consistently scores the highest for both localization and local texturing (Tab. 2). We give further details on CLIP R-Precision in Appendix B.

MLP and direct optimization ablation. We show an ablation of using MLPs by replacing them with direct optimization for the localization and texture maps (Fig. 15). Our full method (far left) using MLPs for both the localization and texture maps gives an accurate and contiguous localization with a clean, detailed local texture. Directly optimizing the texture map values instead of using an MLP (center left) results in a noisier, grainy texture. Using an MLP for the localization map is especially important since we want the localizations to be mostly contiguous and the smoothness of the MLPs gives us this. Directly optimizing the localization map values instead of using an MLP gives a non-contiguous, speckled localization. This localization detrimentally affects the texture as well, whether the texture is optimized either with an MLP (center right) or directly (far right).

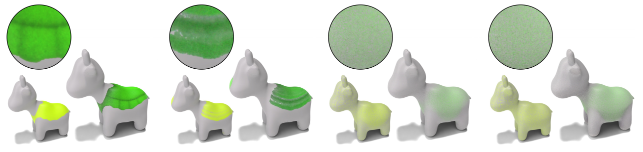

Figure 15. **MLP ablation**. We ablate the use of MLPs by using direct optimization for both the localization and texture maps. Using MLPs provides smooth localization and texture while using direct optimization leads to noisy and less coherent results.

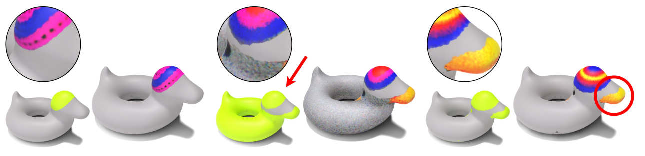

Figure 16. **Background loss ablation**. Our method (left) produces an accurate localization and texture that are highly detailed and tightly coupled. Removing the background MLP $\mathcal{F}_\psi$ and instead learning the background through the main texture map T_{map} leads to poor localization which also degrades the texture (middle). Removing the background loss completely leads to the incorporation of superfluous elements from the input model (*i.e.* a bill on a duck) into the localization (right).

Background loss ablation. We opt to learn the background texture in a separate map from the desired texture map (middle and bottom branches of Fig. 3) enabling our method to produce accurate localizations with tightly coupled and detailed textures (Fig. 16, left). If we instead remove the explicit background texture map (and MLP) and allow the background to be predicted using the same texture map as the main edit texture T_{map}, (by applying background loss to T_{map} masked with the inverse of the localization mask), then the localization and texture become misaligned (Fig. 16, middle). In this case, when the edit and background share the same texture map, if the localization region expands during training to include features that had been considered background, the edit texture may retain these features of the object (rather than expand the edit texture features to fill the localization) and the localization may continue to expand. If we instead remove the background loss entirely, this also produces undesirable results (Fig. 16, right). Specifically, we observe extra elements incorporated in additional localization regions that contain characteristics of the input shape (*e.g.* bill on a duck).

Localization loss ablation. We show an ablation on the localization loss (see Fig. 17). Using our full method, we obtain a precise localization that accurately depicts the local edit region "polo shirt" (left). Without our localization loss,

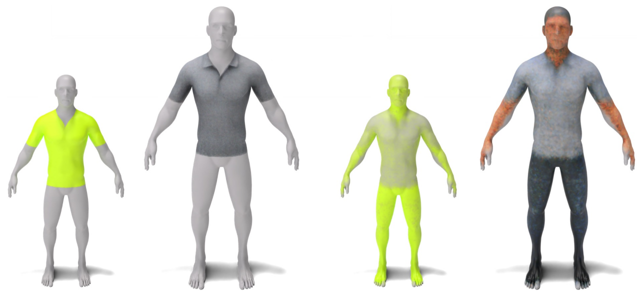

Figure 17. **Localization loss ablation**. We show an ablation of the localization loss using the local edit "polo shirt" on a person mesh. Without this loss, there is no explicit incentive for the localization to be visually meaningful in isolation which can lead to inaccurate localization regions (right).

the only supervision signal for the localization region comes from the style loss as that depends on the localization region to obtain the masked texture. From just this signal, there is no incentive for the localization to have visual meaning and thus we often end up with inaccurate localizations (right localization). These poor localizations can lead to worse textures as well (right texture).

B. Implementation and Further Details

MLP architecture. Our MLPs consist of a positional encoding layer followed by 6 linear layers. Both texture MLPs and the localization MLP take a dimension 3 input corresponding to a 3D coordinate. All linear layers have width 256 and each linear layer is followed by a ReLU activation and subsequently a layer norm. The localization MLP outputs a single probability between 0 and 1, while the texture MLPs output 3 values each also within the range 0 to 1 corresponding to RBG values.

Optimization and supervision details. We use PyTorch to implement our method. For our optimization, we use an Adam optimizer with a constant learning rate of $1e^{-4}$. We use the Huggingface Diffusers implementation of DeepFloyd IF for our diffusion supervision. For all experiments, we use a classifier free guidance weight of 20. We train for 4 hours on a single A40 GPU which typically equates to 4000 iterations.

Text prompt formulation. To create the text prompts, our method takes as input the object class (*i.e*. "cow"), the desired style term (*i.e*. "colorful crochet") and the desired local edit (*i.e*. "hat"). To create y_l, we formulate the prompt "a 3D render of a gray [object class] with a yellow [local edit]" or specifically, "a 3D render of a gray cow with a yellow hat." To get y_t, we use "a 3D render of a gray [object class] with a [style term] [local edit]" or "a 3D render of a gray cow with a colorful crochet hat." To obtain y_b, we use "a 3D render of a [object class] with a yellow [local edit]" or "a 3D render of a cow with a yellow hat."

View selection. At each iteration of our optimization we render our mesh from multiple views. To select these views, we randomly sample camera elevation, azimuth, and radius from predefined ranges that can be specified by the user. In all of our experiments, we render 4 views each iteration. For all experiments, we sampled azimuths ranging from 0 to 360 degrees. By default, we sample elevations from the range $0 - 150$ degrees, however for some meshes, we narrow this range to $0 - 100$ degrees. For all experiments, our camera radius is sampled uniformly from the range $1 - 1.5$.

Quantitative evaluation with Clip R-Precision details. To quantitatively evaluate our method, we use a CLIP R-Precision metric tailored to evaluating either localizations or local styles. We create 15 local edits and record the both the target localization y_l and target local style y_t for each edit as well as their corresponding generated localizations and local styles. To create our localizations for 3D Highlighter [10], we run 3D Highlighter on each of those 15 y_l's to get 15 localizations. To create our local styles for Latent Paint [38], we run Latent Paint on each of the 15 y_t's. To compute the CLIP R-Precision score for localizations for a given method, we do the following. For each localization result L_i of the 15 total from this method, we compute the CLIP similarity between renders of L_i and each of the 15 y_l's to get 15 similarity scores. If the y_l with the highest CLIP similarity to L_i is the y_l that was used to generate L_i then this L_i is awarded a score of 1, otherwise it is awarded a score of 0. To compute the CLIP R-Precision for this method, we take the average score of all 15 L_i's and multiply by 100 to get a percentage. The CLIP R-Precision for the local styles is computed similarly, except using local styles instead of localizations. Specifically to compute the CLIP R-Precision of the local style for a given method, we do the following. For each local style result S_i of the 15 total from this method, we compute the CLIP similarity between renders of S_i and each of the 15 y_t's to get 15 similarity scores. If the y_t with the highest similarity score to S_i is the y_t that was used to generate S_i then this S_i is awarded a score of 1, otherwise it is awarded a score of 0. To compute the CLIP R-Precision for this method, we take the average score of all 15 S_i's and multiple by 100 to get a percentage.