Tactile DreamFusion: Exploiting Tactile Sensing for 3D Generation

Ruihan Gao[1] **Kangle Deng**[1] **Gengshan Yang**[1] **Wenzhen Yuan**[2] **Jun-Yan Zhu**[1]
[1]Carnegie Mellon University [2]University of Illinois Urbana-Champaign
https://ruihangao.github.io/TactileDreamFusion/

Abstract

3D generation methods have shown visually compelling results powered by diffusion image priors. However, they often fail to produce realistic geometric details, resulting in overly smooth surfaces or geometric details inaccurately baked in albedo maps. To address this, we introduce a new method that incorporates touch as an additional modality to improve the geometric details of generated 3D assets. We design a lightweight 3D texture field to synthesize visual and tactile textures, guided by 2D diffusion model priors on both visual and tactile domains. We condition the visual texture generation on high-resolution tactile normals and guide the patch-based tactile texture refinement with a customized TextureDreambooth. We further present a multi-part generation pipeline that enables us to synthesize different textures across various regions. To our knowledge, we are the first to leverage high-resolution tactile sensing to enhance geometric details for 3D generation tasks. We evaluate our method in both text-to-3D and image-to-3D settings. Our experiments demonstrate that our method provides customized and realistic fine geometric textures while maintaining accurate alignment between two modalities of vision and touch.

1 Introduction

Generating high-fidelity 3D assets is crucial for a wide range of applications, from content creation in gaming and VR/AR to developing realistic simulations for robotics. Recently, a surge in emerging methods has enabled the creation of 3D assets from a single image [1] or a short text prompt [2, 3]. Their success can be attributed to advances in generative models [4, 5] and neural rendering [6, 7], as well as the availability of extensive 2D and 3D datasets [8, 9].

Although existing methods can effectively capture the overall shape and visual appearance of objects, they often struggle with synthesizing fine-grained geometric details, such as stochastic patterns or bumps from the material, as shown on the left side of Figure 1. As a result, the final mesh output tends to be either overly smooth (e.g., the avocado in the top row) or has geometric details incorrectly baked into the albedo map (e.g., the beanie in the bottom row). But why is that?

We argue that there are two major bottlenecks. First, high-resolution geometric data are largely absent in current 2D and 3D datasets. For image datasets like LAION [8], obtaining geometric details is challenging due to the limited camera resolution in capturing real-world objects. In the case of 3D asset datasets like Objaverse [9], although the assets often include visual textures, they rarely feature high-resolution geometric textures. Second, it is difficult for humans to precisely describe fine geometric textures in natural language, making text-based applications even more challenging.

To address these issues, we propose exploring tactile sensing to capture high-resolution texture data for 3D generation. Given a text prompt or an input image, we first generate a base mesh with an albedo map as our 3D representation. We then capture detailed surface geometry of the target texture

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

 https://doi.org/10.52202/079017-0939

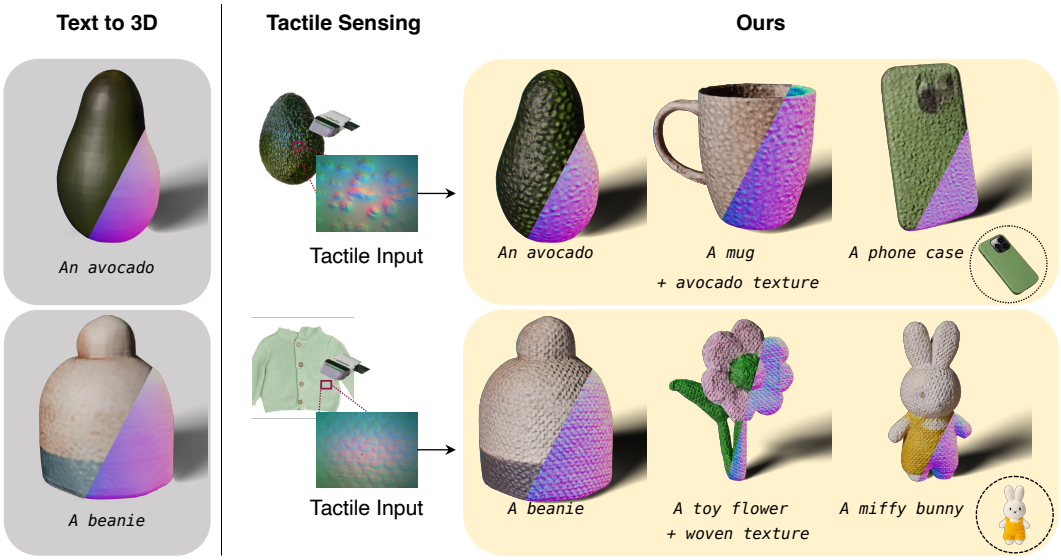


Figure 1: Our method leverages tactile sensing to improve existing 3D generation pipelines. *Left*: Given a text prompt, we first generate an image using SDXL [12] and then run Wonder3D [13] to generate mesh from the image. This process often results in a mesh with an overly smooth surface. *Right*: Our method takes a text prompt and several tactile patches and generates high-fidelity coherent *visual* and *tactile* textures that can be transferred to different meshes. Our method can easily adapt to image-to-3D tasks, as shown in the rightmost column, with the reference image's thumbnail displayed at the bottom right corner. Please visit our webpage for video results.

using GelSight [10, 11], a high-resolution tactile sensor. We then convert these tactile data into normal maps and train a TextureDreambooth on these normal maps. To refine the albedo map and ensure alignment between visual and tactile modalities, we learn a lightweight 3D texture field that co-optimizes the albedo and normal maps using 2D diffusion guidance across both domains.

Furthermore, we extend our approach to synthesize textures across multiple object regions by aggregating 2D diffusion-based part segmentation in a 3D label field.

Our experiments demonstrate that our method outperforms existing text-to-3D and image-to-3D methods in terms of qualitative comparison and user study. Our method ensures coherent high-resolution textures with precise alignment between appearance and geometry, as shown on the right side of Figure 1. Our code and datasets are available on our project website.

2 Related Work

3D generation. Following the success of text-to-image models [14, 15, 16, 17], we have witnessed a booming development of 3D generative models conditioned on text or images. Recent works have used 2D diffusion priors in 3D generation [2, 18, 3, 19, 20, 21, 22, 13, 23], with notable methods like DreamFusion [2] introducing Score Distillation Sampling (SDS) to optimize a 3D representation using gradients from 2D diffusion models. Follow-up works have further extended SDS optimization using multi-view diffusion models, improving both 3D generation and single-view reconstruction [1, 24, 25, 26, 27, 28, 13, 29, 30].

Another line of research [31, 32, 33, 34] trains large-scale transformers to generate 3D shapes in a feed-forward manner, requiring high-quality large-scale 3D asset datasets. While these models effectively capture global shapes, they often fail to accurately render fine-grained geometric details, which are either incorrectly baked in color or lost entirely.

Geometry representation in 3D generation. Researchers have explored various 3D representations for generation, such as voxel grids [35, 36, 37], point clouds [38, 39, 40], meshes [3, 41, 23, 42, 43, 44, 45, 19], neural radiance fields (NeRF) [46, 47, 2, 20, 21], neural surfaces [48, 22, 13, 25], and Gaussians [49, 50]. Among these, meshes support faster and more efficient rasterization rendering than volumetric rendering. Meshes also integrate seamlessly with graphics engines for downstream

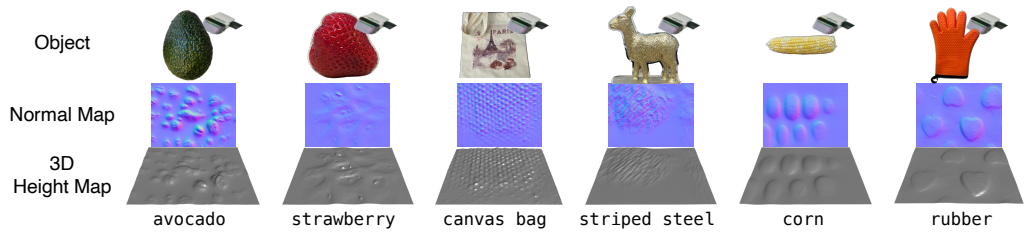


Figure 2: `TouchTexture` **dataset.** We collect tactile normal data from 18 daily objects featuring diverse tactile textures. To demonstrate the local geometric intricacies, we show the tactile normal map and a 3D height map for each object. Please refer to the supplement for the full set of our data.

applications. In contrast, volume representations require separate post-processing to extract meshes, often resulting in a loss of quality. Our approach uses meshes to represent coarse geometry while embedding local geometric details into a 3D texture field of tactile normal.

3D texture generation and transfer. Simultaneously generating geometry and texture often results in blurry textures due to surface smoothing or averaging across views. To address this, many approaches propose a subsequent stage that focuses on refining high-resolution textures [22, 13] or treats texture generation as a separate problem entirely [51, 52, 53, 54, 55, 56]. Additionally, various texture transfer methods enable the transfer of texture to new shapes using images [57, 58, 59, 60] or other material assets [61]. While most methods focus solely on visual appearance, our approach produces high-fidelity geometric details. Closely related to our method, NeRF-Texture [62] also transfers textures with geometric variations but requires 100-200 images of the same object. In addition, their method models the mesostructure with a signed distance and a coarse normal direction at each projected vertex. Hence, it can model sparse structures at a centimeter scale but not delicate patterns on the object's surface. In contrast, our method leverages high-resolution tactile sensing at the millimeter scale and provides accurate surface normal details, with richer details at fine scales.

Tactile sensing in 3D. Tactile data is useful in providing contact information and are widely used in 3D perception and robotics tasks. Early works start with reconstructing simple objects [63, 64, 65, 66], and then evolve to more complicated scenes with latest 3D neural representations [67, 68], as well as robotic in-hand reconstruction with multi-finger contacts [69, 70]. Vision-based tactile sensing [10, 71, 11], a particular sensing mechanism based on the photometric stereo principle, can provide high-resolution surface texture information like surface normals and thus can be useful for high-quality 3D synthesis and generation. Several recent works have also used it for 2D generation [72, 73, 74] and 3D scene reconstruction [75, 67, 76]. In contrast, to our knowledge, our work is the first to use tactile sensing for 3D generation.

3 Tactile Data Acquisition

We use GelSight [10, 11] to acquire high-resolution geometric details by touching the object's surface. The raw sensor data are then pre-processed to obtain high-frequency signals, as shown in Figure 3.

GelSight tactile sensor. The GelSight sensor is a vision-based tactile sensor that uses photometric stereo to measure geometry at a high spatial resolution at the contact surface. It can capture the fine details on the surface, such as bumps on avocados' skin and patterns of crochet yarns. In our work, we use the GelSight mini with a sensing area of 21mm $\times$ 25mm ($h \times w$) and a pixel resolution of 240$\times$320, equivalent to about 85 micrometers per pixel. The sensor captures an RGB image, which can be mapped to the surface normals and then used to reconstruct a surface height map.

Tactile data pre-processing and contact mask. We manually press the GelSight sensor against the object's surface to obtain a tactile image over a small area. The derived height map contains both a low-frequency component, representing the surface's global shape, and high-frequency geometric textures. We first apply a high-pass filter to the height map to extract texture information and then center-crop the masked region to a square patch to remove any non-contact area. Finally, we convert the masked height map patch back to a normal map patch by computing its gradients.

Figure 2 shows our dataset `TouchTexture` collected from 18 daily objects with diverse tactile textures. Our dataset is available on our project website.

https://doi.org/10.52202/079017-0939

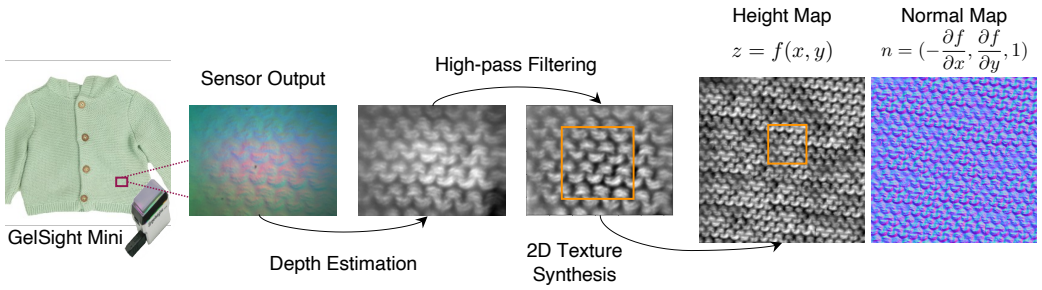


Figure 3: **Tactile data capture.** We collect one patch by pressing GelSight Mini on an object surface. We use Poisson integration to estimate the contact depth from the sensor output, apply high-pass filtering to extract the high-frequency texture information, and then run the 2D texture synthesis method of Image Quilting [77] to obtain an initial texture map. Finally, we convert the height map back to a normal map.

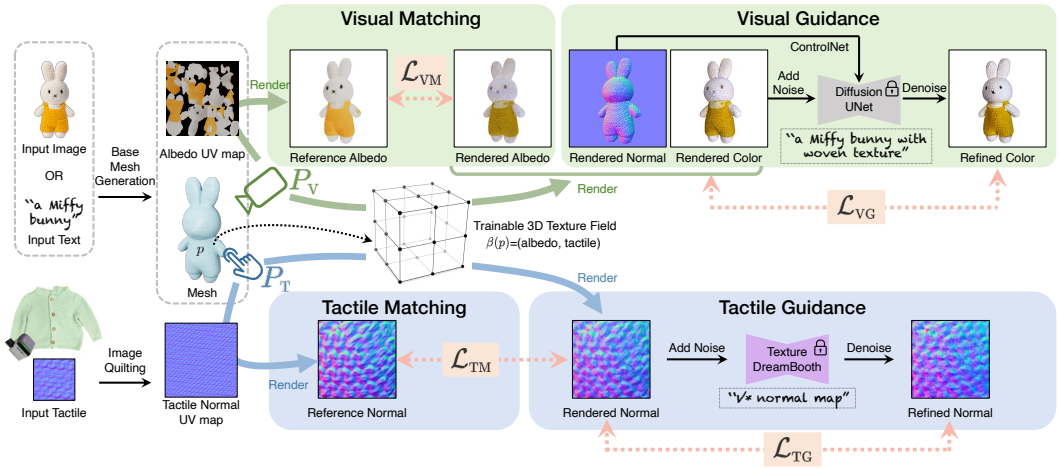


Figure 4: **Method overview.** Given an input image or a text prompt, our method generates a mesh with high-quality visual and normal texture. We first generate a base mesh with albedo texture using a text- or image-to-3D method. We use a 3D texture field with hash encoding to represent albedo and tactile normal textures and train it with loss functions on rendered images. To capture the scale differences between visual and tactile modalities, we sample distinct camera views, P_V for visual rendering and P_T for tactile rendering. For texture refinement, we train the texture field with a visual matching loss $\mathcal{L}_{VM}$, to ensure fidelity to the input mesh, and a visual guidance loss with normal-conditioned ControlNet, $\mathcal{L}_{VG}$, to enhance photorealism and cross-modal alignment. We further apply a tactile matching loss, $\mathcal{L}_{TM}$, and a tactile guidance loss, $\mathcal{L}_{TG}$, using a customized Texture Dreambooth, to achieve high-quality geometric details aligned with the distribution of tactile input V^* texture exemplars.

4 Method

Generating 3D assets with high-resolution geometric details is challenging due to the difficulty of acquiring low-level texture details from text prompts or a single image. Therefore, we incorporate an additional input modality, tactile normal maps, to enhance the geometric details in 3D asset generation. Figure 4 shows our overall pipeline. Our method takes an input image and a tactile normal patch as input and generates a 3D asset with a high-fidelity, color-aligned normal map. The input image can be a real image or generated by a text-to-image model such as SDXL [12]. Below, we first describe how we generate the base mesh in Section 4.1. We then introduce our core texture refinement algorithm in Section 4.2 and extend it to objects with multiple materials in Section 4.3.

4.1 Base Mesh Generation

Similar to prior works with stage-wise geometry sculpting and texture refinement [22, 49], we first generate a base mesh with albedo texture using a text- or image-to-3D method. We then unwrap a UV map of the exported mesh M and project the vertex albedo onto an albedo UV map.

A 2D texture transfer baseline. To incorporate the tactile information, one could synthesize a large 2D texture map with an exemplar of the preprocessed normal patch using 2D texture synthesis algorithms such as image quilting [77] and use it as a normal UV map of the object mesh. However,

https://doi.org/10.52202/079017-0939

this would result in inconsistencies between visual and tactile textures, especially when transition in visual color indicates geometric change, as shown in Figure 9.

4.2 Texture Refinement

To address this issue, we jointly optimize the albedo and the normal tactile texture to ensure alignment.

Texture representation. For ease of optimization, we represent the visual and tactile normal textures as a 3D texture field. In practice, we use a multi-resolution hash grid [78] $\beta(\cdot)$, whose input is a 3D spatial coordinate p on the mesh:

$$\beta(p) = (\mathbf{c}, \mathbf{n}_{\text{T}}), \tag{1}$$

where $\mathbf{c} \in \mathbb{R}^3$ is the albedo and $\mathbf{n}_{\text{T}} \in \mathbb{R}^3$ is the tactile normal defined in the tangent space on the mesh surface.

Given the base mesh M, the texture field $\beta(\cdot)$, a camera pose P, and a lighting condition L, we can render a color image $\hat{\mathbf{I}}_{\text{C}} \in \mathbb{R}^{H \times W \times 3}$ of the object using a differentiable rasterizer $\mathcal{R}$, nvdiffrast [79]:

$$\hat{\mathbf{I}}_{\text{C}}(P) = \mathcal{R}(M, \beta(\cdot), P, L), \tag{2}$$

where H and W represent image height and width, respectively.

Specifically, for each 3D point on the mesh, we composite the base normal $\mathbf{n}_{\text{B}}$ from the mesh geometry and the tactile normal $\mathbf{n}_{\text{T}}$ from the texture to get the shading normal $\mathbf{n}$:

$$\mathbf{n} = \mathbf{Q}_{\text{TBN}} \cdot \mathbf{n}_{\text{T}} = [\mathbf{t}, \mathbf{n}_{\text{B}} \times \mathbf{t}, \mathbf{n}_{\text{B}}] \cdot \mathbf{n}_{\text{T}}, \tag{3}$$

where $\mathbf{n}_{\text{B}}$ is the global geometric normal defined by the base mesh, $\mathbf{t}$ is the tangent vector, and $\mathbf{Q}_{\text{TBN}}$ is the Tangent-Bitangent-Normal (TBN) Matrix for every surface point. Given the calculated shading normal and a sampled camera pose P, we use a point light and a simple diffuse shading model to produce the color image $\hat{\mathbf{I}}_{\text{C}}(P)$ following prior works [2, 3]. We can also obtain $\hat{\mathbf{I}}_{\text{A}}(P)$, $\hat{\mathbf{I}}_{\text{T}}(P)$, and $\hat{\mathbf{I}}_{\text{N}}(P)$ by projecting the albedo, tactile normal, and shading normal onto a sample view P.

Learning 3D texture field. Given the scale discrepancy between vision and touch, we sample camera views differently for two modalities. To supervise visual renderings $\hat{\mathbf{I}}_{\text{C}}$ and $\hat{\mathbf{I}}_{\text{A}}$, we sample camera poses P_{V} orbiting the base mesh looking at the object center with perspective projection. To supervise tactile renderings $\hat{\mathbf{I}}_{\text{T}}$, we sample camera poses P_{T} close to mesh surfaces based on vertex normals to emulate the captured data using a real sensor. Specifically, we randomly sample a vertex, place a camera along this vertex's normal direction, and set the camera to look at the sampled vertex with orthographic projection.

We initialize the neural texture field of albedo with a reconstruction loss of rendered images:

$$\mathcal{L}_{\text{VM}} = \left|\left|\hat{\mathbf{I}}_{\text{A}}(P_{\text{V}}) - \mathbf{I}_{\text{A}}(P_{\text{V}})\right|\right|_2^2, \tag{4}$$

where $\mathbf{I}_{\text{A}}$ is rendered using the exported albedo UV map in Section 4.1 from the same camera P_{V}. Similarly, we initialize the tactile normal texture field with a **T**actile **M**atching (TM) loss:

$$\mathcal{L}_{\text{TM}} = 1 - \cos\left(\hat{\mathbf{I}}_{\text{T}}(P_{\text{T}}), \mathbf{I}_{\text{T}}(P_{\text{T}})\right), \tag{5}$$

where $\mathbf{I}_{\text{T}}$ is rendered using the tactile normal UV map from the image quilting results from a tactile camera pose P_{T}. Here, we use image quilting [77], a patch-based texture synthesis algorithm, to synthesize a reference texture map $\mathbf{I}_{\text{T}}$ that roughly matches the physical scale of real materials and use it for initialization.

To refine the visual texture, we use a diffusion guidance loss inspired by the multi-step denoising process in SDEdit [80], Instruct-NeRF2NeRF [81], and DreamGaussian [49]. Different from existing works, we compute the **V**isual **G**uidance loss $\mathcal{L}_{\text{VG}}$ based on a normal-conditioned ControlNet [82] to ensure the refined visual texture is consistent with the tactile normal. Given the rendered color image $\hat{\mathbf{I}}_{\text{C}}$ from P_{V}, we perturb it with random noise $\epsilon(t)$ to get a noisy image x_t and apply a multi-step denoising process $f_\phi(\cdot)$ using the 2D diffusion prior to obtaining a refined image:

$$\mathbf{I}_\phi(P_{\text{V}}) = f_\phi(x_t; t, y, \hat{\mathbf{I}}_{\text{N}}(P_{\text{V}})), \tag{6}$$

https://doi.org/10.52202/079017-0939

where y is the input text prompt, and f_ϕ is a normal-conditioned ControlNet with the Stable Diffusion backbone. We denoise the image from timestep t to a completely clean image $\mathbf{I}_\phi$, which is different from the typical single-step SDS loss in Dreamfusion [2]. The starting timestep $t \in (0, 1)$ gradually decreases from 0.5 to 0.3 as training iteration increases, balancing the noise reduction to enhance details without disrupting the original content. This refined image is then used to optimize the texture through the L1 and LPIPS [83] loss:

$$\mathcal{L}_{\text{VG}} = \left|\left|\hat{\mathbf{I}}_{\text{C}}(P_{\text{V}}) - \mathbf{I}_\phi(P_{\text{V}})\right|\right|_1 + \mathcal{L}_{\text{LPIPS}}\left(\hat{\mathbf{I}}_{\text{C}}(P_{\text{V}}), \mathbf{I}_\phi(P_{\text{V}})\right). \tag{7}$$

While refining the visual textures, we jointly optimize the tactile texture with a **T**actile **G**uidance loss $\mathcal{L}_{\text{TG}}$. Similar to $\mathcal{L}_{\text{VG}}$, we add random noise to a rendered tactile normal map $\mathbf{I}_{\text{T}}(P_{\text{T}})$, generated from a tactile camera pose P_{T}, and then apply a multi-step denoising process to obtain a refined normal image $\mathbf{I}_\psi$. To capture the distribution of tactile normals, we replace the standard diffusion prior with a *Texture DreamBooth*, a customized model created by fine-tuning the Stable Diffusion model $f_\psi(\cdot)$ for each tactile texture using a Low-Rank Adapter (LoRA) [84] with DreamBooth [85]. We leave the LoRAs training details in Appendix A.3. We use the output $\mathbf{I}_\psi$ to refine the tactile texture:

$$\mathcal{L}_{\text{TG}} = 1 - \cos\left(\hat{\mathbf{I}}_{\text{T}}(P_{\text{T}}), \mathbf{I}_\psi(P_{\text{T}})\right). \tag{8}$$

Optimization. The overall loss function is:

$$\mathcal{L} = \lambda_{\text{VM}}\mathcal{L}_{\text{VM}} + \lambda_{\text{TM}}\mathcal{L}_{\text{TM}} + \lambda_{\text{VG}}\mathcal{L}_{\text{VG}} + \lambda_{\text{TG}}\mathcal{L}_{\text{TG}}. \tag{9}$$

To initiate our texture grid, we start by only optimizing the visual matching loss $\mathcal{L}_{\text{VM}}$ and tactile matching loss $\mathcal{L}_{\text{TM}}$ for 150 iterations with $\lambda_{\text{VM}} = 500$ and $\lambda_{\text{TM}} = 1$. After that, we run optimization for another 50 iterations to refine the output guided by diffusion priors. We reduce λ_{TM} from 1 to 0.05, change $\mathcal{L}_{\text{VM}}$ from per-pixel error to mean-color error to allow more flexibility in texture refinement, and add the visual guidance loss $\mathcal{L}_{\text{VG}}$ and tactile guidance loss $\mathcal{L}_{\text{TG}}$ with $\lambda_{\text{VG}} = 5$ and $\lambda_{\text{TG}} = 0.05$.

4.3 Multi-Part Textures

Another advantage of our 3D texture field is its ability to easily define non-uniform textures in 3D, allowing different tactile textures to be assigned to distinct parts of an object. For instance, when generating a 3D asset of "a cactus in a pot", we can apply different textures to the cactus and the pot, by incorporating two tactile inputs along with a text prompt that specifies the texture assignment, e.g., "cactus with texture A, pot with texture B".

Part segmentation based on diffusion features. Our method can automatically segment object parts based on the text prompt without manual annotation. We leverage the internal attention maps of diffusion models to segment the rendered views $\mathbf{I}_{\text{A}}(P_{\text{V}})$ of the object. Specifically, we add a random noise of level t to $\mathbf{I}_{\text{A}}$ and apply one denoising step with the SD UNet. Inspired by DiffSeg [86] and other text-to-image methods [87, 88], we perform unsupervised part segmentation by aggregating and spatially clustering self-attention maps from multiple layers. To assign part labels, we aggregate cross-attention maps corresponding to different parts and match them with the unlabeled segmentation maps based on Kullback–Leibler (KL) divergence. This results in a list of labeled segmentation masks $\mathbf{M}^n(\mathbf{I}_{\text{A}}) \in \{0, 1\}^{H \times W}$, $n \in \{1, \ldots, N\}$, where n denotes one of the N object parts. For example, given the prompt "cactus in a pot," we aggregate cross-attention maps for "cactus" and "pot" from different layers, then assign each part segmented from the clustered self-attention maps to either cactus or pot according to respective cross-attention maps. More details are included in Appendix A.1.

3D part label field. Since the 2D part segmentations may contain noise and lack multi-view consistency, we integrate them from different viewpoints into a coherent 3D part label field. We achieve this by incorporating a part label $s \in \mathbb{R}^N$ into our texture field $\beta(\cdot)$. To supervise it, we render part label maps $\hat{S}(P_{\text{V}})$ from sampled camera poses P_{V} and compute the cross entropy loss:

$$\mathcal{L}_s = \mathcal{L}_{\text{CE}}\left(\hat{S}(P_{\text{V}}), S(\mathbf{I}_{\text{A}}(P_{\text{V}}))\right). \tag{10}$$

where $\hat{S}(P_{\text{V}})$ is the part logits and $S(\mathbf{I}_{\text{A}}(P_{\text{V}})) \in \{0, 1\}^{H \times W \times N}$ are labels concatenated from $\mathbf{M}^n$.

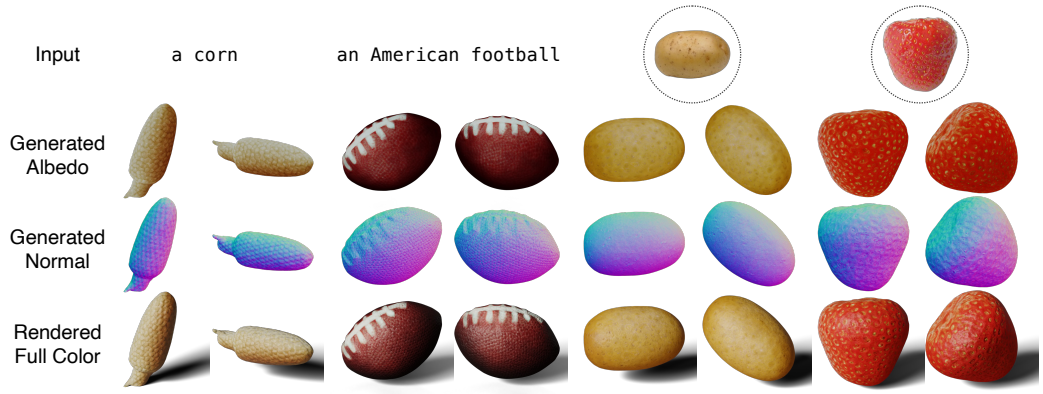


Figure 5: **3D generation with a single texture.** For each object, we show generated albedo (top), normal (middle), and full color (bottom) renderings from two viewpoints. Our method works for both text-to-3D (corn and football) and image-to-3D (potato and strawberry), generating realistic and coherent visual textures and geometric details. (We use roughness=0.5 when rendering color views in Blender for Figures 1, 5, 6, and 7.)

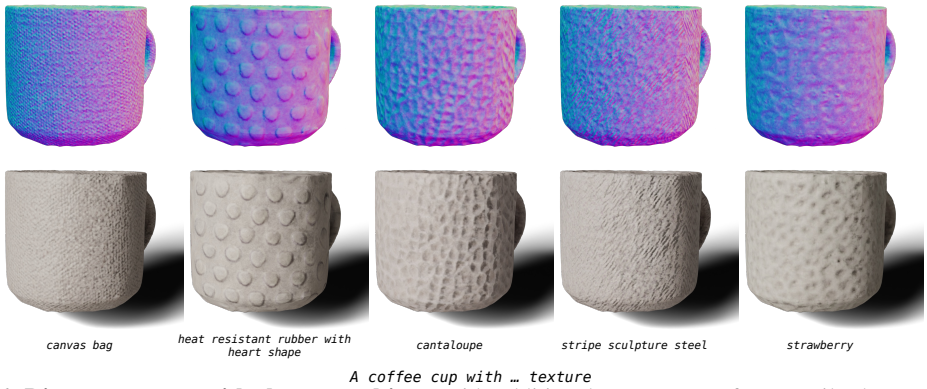


Figure 6: **Diverse textures with the same object.** With additional texture cues from tactile data, we can synthesize diverse textures with the same coarse shape for customized designs.

Optimization with multi-part textures. To learn multi-part textures, we keep the same $\mathcal{L}_{\text{VM}}$ and $\mathcal{L}_{\text{VG}}$ loss functions while modifying $\mathcal{L}_{\text{TM}}$ and $\mathcal{L}_{\text{TG}}$ to incorporate part-specific tactile supervision:

$$\mathcal{L}_{\text{TM}} = \sum_{n=1}^{N} \left[1 - \cos\left(\hat{\mathbf{M}}^n \odot \hat{\mathbf{I}}_{\text{T}}, \hat{\mathbf{M}}^n \odot \mathbf{I}_{\text{T}}^n\right)\right], \tag{11}$$

$$\mathcal{L}_{\text{TG}} = \sum_{n=1}^{N} \left[1 - \cos\left(\hat{\mathbf{M}}^n \odot \hat{\mathbf{I}}_{\text{T}}, \hat{\mathbf{M}}^n \odot \mathbf{I}_{\psi}^n\right)\right], \tag{12}$$

where $\odot$ denotes the Hadamard product, $\mathbf{I}_{\text{T}}^n$ is the rendered reference view using the n-th part's tactile data, $\mathbf{I}_{\psi}^n$ is the refined tactile normal generated by Texture Dreambooth trained on the n-th part texture, and $\hat{\mathbf{M}}^n$ is the binary mask for the n-th part obtained from $\hat{S}$ rendered from our learned 3D label field. We omit P_{T} for clarity since all patches are rendered from the same tactile camera pose.

5 Experiment

We present comprehensive experiments to verify the efficacy of our method. We perform qualitative and quantitative comparisons with existing baselines and ablation studies on the major components.

Dataset. We have collected 18 diverse types of textures from daily objects in our `TouchTexture` dataset, five tactile patches per texture, and pair them with a short description. Please see Appendix A.2 for the full list. We randomly sample one patch to initialize the tactile UV map with the image quilting algorithm [77] and use five patches to train the customized Texture DreamBooth. For base mesh generation given a text or an image input, while any mesh generation method is applicable,

https://doi.org/10.52202/079017-0939

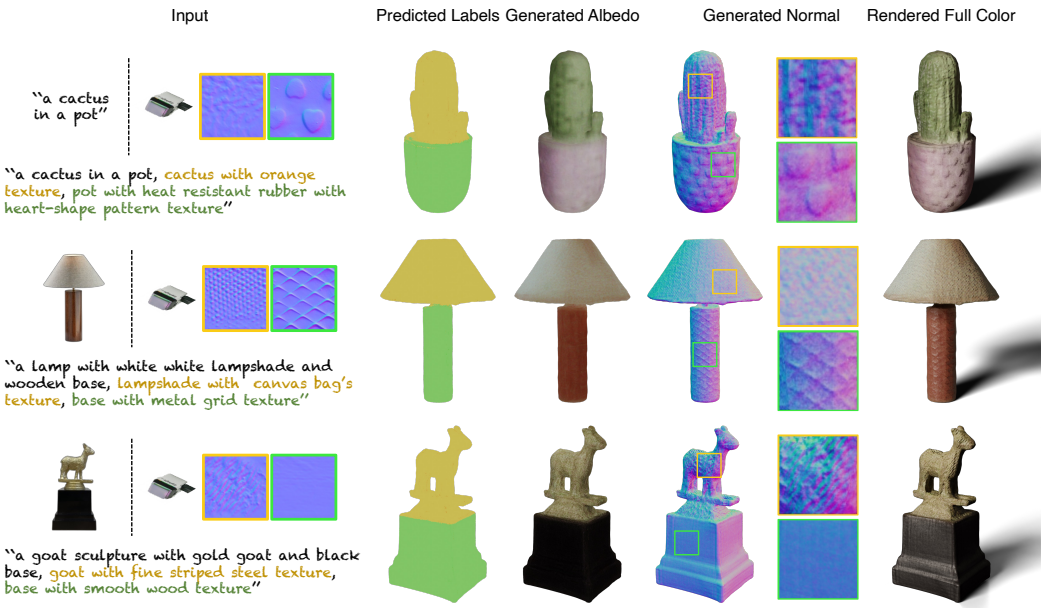


Figure 7: **Multi-part texture generation.** Our method allows users to specify an object (via text or image) and its two parts to assign different textures (color-coded text prompts correspond to text descriptions for two textures). We show paired results for the predicted label, albedo, normal, and full-color renderings. The zoom-in patches demonstrate the generated normal textures on different parts.

we use Wonder3D [13] in the main experiments but include results using InstantMesh [89] and RichDreamer [90] in Appendix A.4 as well. Specifically, Wonder3D takes an image as input and outputs a mesh with albedo stored as vertex colors. We then convert the vertex albedo into an albedo UV map as described in Section 4.1. During optimization in Section 4.2, we refine the albedo and tactile textures with the "textured prompts", i.e., *A [object name] with V* texture*, where *V** represents the text description corresponding to the selected tactile texture map.

3D generation with a single texture. Figure 5 shows our 3D generation results for a single texture, showing albedo, normal, and full-color rendering from two viewpoints for each object. Our method works for both text input (the corn and the football) and image input (the potato and the strawberry), generating realistic, coherent, and high-resolution visual and geometry details. Figure 6 shows the results of applying different textures to the same object (a coffee cup), with normal rendering on top and color rendering below. Our joint optimization produces coherent visual-tactile textures and smooth, natural shadows from geometric variations.

Multi-part texture generation. As introduced in Section 4.3, users can specify an object (via text or image) and assign different textures to two distinct parts. Figure 7 shows results for multi-part texture synthesis. The left column shows the text or image input as well as the tactile input. The image input can be either real or generated from a text prompt. The color-coded text prompts correspond to text descriptions for two textures. The right columns show the results of the albedo, normal, and full-color rendering. Our method effectively segments parts with our 3D label field. The joint optimization successfully applies the textures to each corresponding part designated by the user and generates an overall coherent visual appearance and tactile geometry.

Baselines. To our knowledge, this work is the first to leverage high-resolution tactile sensing to enhance geometric details for 3D generation tasks. Thus, we compare our method with existing text-to-3D and image-to-3D methods. For text-to-3D, we compare with DreamCraft3D [22] and use the same textured prompt, i.e., a prompt with a text description of the target tactile texture, as input. DreamCraft3D focuses on geometry sculpting using neural field and DMTet [91], followed by texture boosting through fine-tuning DreamBooth with augmented multi-view rendering, requiring 10x computational cost compared to our method. For image-to-3D, we compare with DreamGaussian [49] and Wonder3D [13]. DreamGaussian employs fast generation using 3D Gaussian primitives and refines an albedo map in Stage 2 with a multi-step denoising MSE loss. Wonder3D generates paired multi-view RGB and normal images, using a geometry fusion process to generate a 3D result. The input images remain the same as ours for these two baselines. We use the official implementations for all baselines.

https://doi.org/10.52202/079017-0939

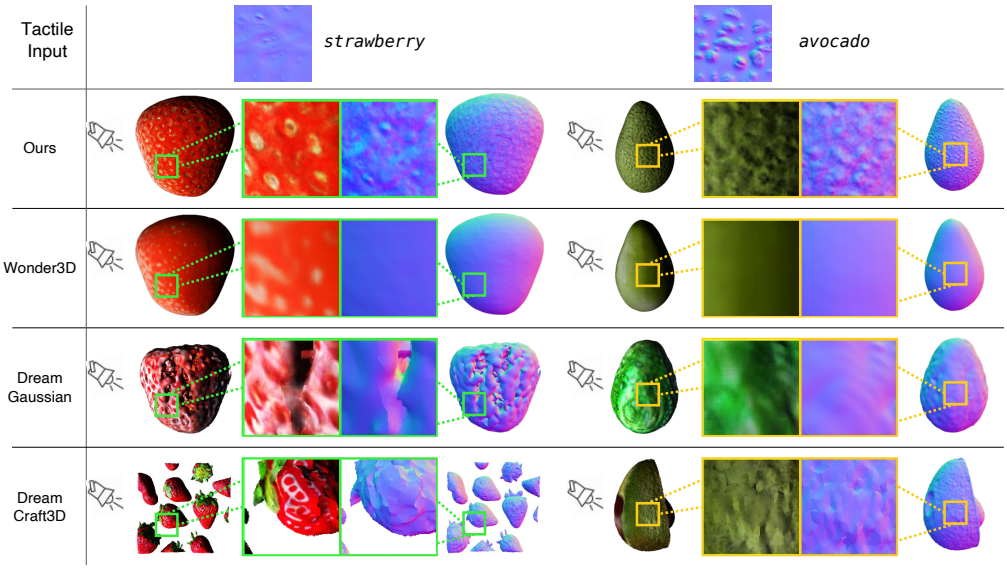


Figure 8: **Baseline comparison.** Compared to the SOTA image-to-3D (Wonder3D and DreamGaussian) and text-to-3D (DreamCraft3D) baselines, our method produces significantly more plausible low-level geometry. For a fair comparison, we use the same input image for the first three rows.

Table 1: **Human perceptual study.** For all paired comparisons, our method is preferred ($\geq 50\%$) over the baselines for both texture appearance and geometric details.

	Ours vs DreamGaussian	**Ours vs Wonder3D**	**Ours vs DreamCraft3D**
Texture	85.43 ± 4.76	86.36 ± 3.93	61.54 ± 4.43
Geometry	92.85 ± 3.47	88.07 ± 3.80	84.20 ± 4.17

Qualitative evaluation. Figure 8 shows qualitative results of our method compared against three baselines. For each example, we show color and normal rendering with zoomed-in patches at the same location for detailed comparison. Our method achieves higher visual fidelity, more realistic details in normal space, and better color-geometry alignment than the baselines. In particular, our textures exhibit sharper details than those of Wonder3D, which tend to be overly smooth. In contrast to DreamGaussian and DreamCraft3D, our generated geometry details are more realistic and align well with the color appearance. In the avocado example, DreamCraft3D suffers from the "Janus problem", generating a brown core on both sides.

Quantitative evaluation. We perform a human perceptual study using Amazon Mechanical Turk (AMTurk). We set up a paired test, showing a reference prompt and two rendering results, one generated with our method and the other generated with one of the baselines. We conduct two separate surveys to evaluate the texture appearance and geometric details, respectively. For texture appearance, we render full-color RGB images and ask users "Which of the following views has more realistic textures?". For geometric details, we render shaded colorless images that only demonstrate the mesh geometry and ask users "Which of the following views has more realistic geometric details?". Please see Appendix A.5 for example screenshots of the paired rendering. Each user has two practice rounds followed by 20 test rounds to evaluate our method against DreamGaussian, Wonder3D, and DreamCraft3D. All samples are randomly selected and permuted, and we collect 1,000 responses.

Table 1 shows the mean and standard deviation of users' preference score; our method is preferred over all baselines. While DreamCraft3D has a close performance regarding texture appearance, it fails to obtain high-resolution geometric details that align well with color texture.

Ablation study. We perform an ablation study to evaluate key aspects of our method. We render both front and back views of a sampled object for each experiment. To illustrate details and cross-modal alignment, we present a global full-color view on the left, a global normal view on the right, and patch views of the full-color, albedo, and normal renderings in between. Figure 9 shows results of ablating diffusion-based guidance losses for texture refinement, specifically the visual guidance $\mathcal{L}_{\text{VG}}$ and tactile guidance $\mathcal{L}_{\text{TG}}$.

https://doi.org/10.52202/079017-0939

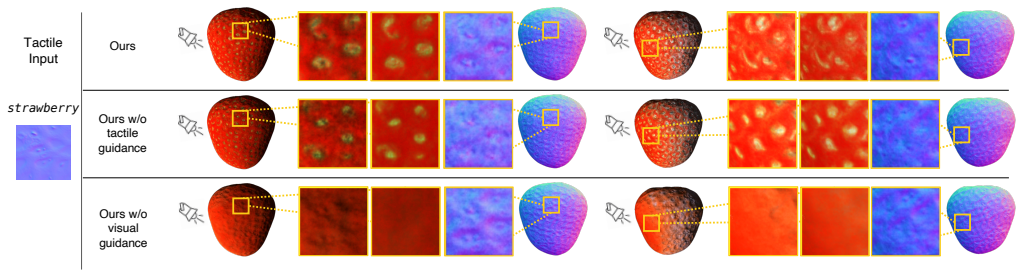


Figure 9: **Ablation study regarding texture refinement**. We ablate our method regarding the tactile guidance loss $\mathcal{L}_{\text{TG}}$ and visual guidance loss $\mathcal{L}_{\text{VG}}$. Removing $\mathcal{L}_{\text{TG}}$ results in fewer details of the generated tactile texture. Removing the visual guidance introduces misaligned visual and tactile normal textures. For example, the bumps in the normal map are misaligned with the locations of white seeds in the albedo rendering, as shown in the zoomed-in patches. Our method encourages the generated color to be consistent with the tactile normal using ControlNet-guided visual refinement loss while also enhancing the details in the tactile texture.

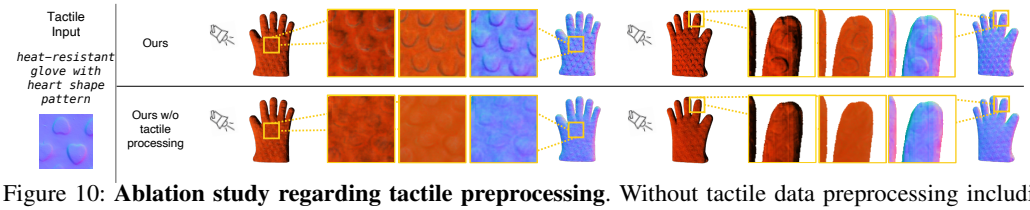


Figure 10: **Ablation study regarding tactile preprocessing**. Without tactile data preprocessing including high-pass filtering and contact area cropping, the generated geometric details tend to be flat and unrealistic.

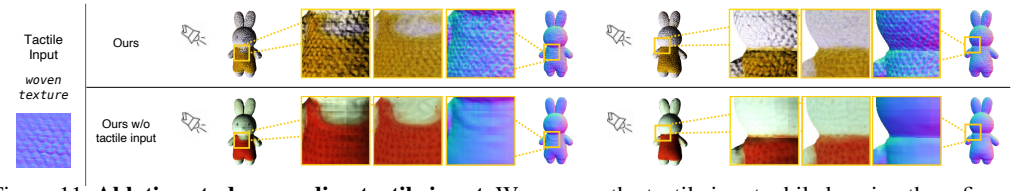


Figure 11: **Ablation study regarding tactile input**. We remove the tactile input while keeping the refinement loss. Without the tactile information, the generated 3D assets fail to capture fine-grained geometric details.

Omitting $\mathcal{L}_{\text{TG}}$ reduces tactile texture details, while omitting $\mathcal{L}_{\text{VG}}$ introduces misalignment between visual and tactile normal textures; for instance, bumps in the normal map do not match the white seeds in the albedo rendering. Our method ensures color consistency with tactile normals via ControlNet-guided visual refinement and enhances tactile texture details.

We also study the efficacy of tactile data processing by comparing our method with the texture map synthesized using the original tactile data without preprocessing stated in Section 3. Figure 10 shows that using the original data produces much more flattened textures since the low-frequency deformation of the gel pad due to uneven contact during the data collection would dominate the tactile signal and thus degrade the details of synthesized texture maps. Figure 11 shows the results of removing tactile input and only optimizing the albedo map with "textured prompts" using $\mathcal{L}_{\text{VM}}$ and $\mathcal{L}_{\text{VG}}$. Without tactile input, the output mesh is overly smooth, demonstrating the insufficiency of using text prompts only for fine geometry texture generation.

6 Discussion

Recent methods in 3D generation often struggle with unrealistic geometric details. To address this, we have introduced a novel approach that incorporates tactile information. Our method synthesizes both visual and tactile textures using a 3D texture field. Additionally, we have introduced a multi-part texturing pipeline for controllable region-wise texture generation. To our knowledge, this is the first use of high-resolution tactile sensing to improve 3D generation. Our method produces realistic, fine-grained geometric textures while maintaining accurate visual-tactile alignment.

Limitations. The quality of the generated coarse shape depends on the existing 3D generative models, which struggle with complex geometry. Additionally, slight seams may appear in our results due to UV unwrapping.

Acknowledgments and Disclosure of Funding

We thank Sheng-Yu Wang, Nupur Kumari, Gaurav Parmar, Hung-Jui Huang, and Maxwell Jones for their helpful comments and discussion. We are also grateful to Arpit Agrawal and Sean Liu for proofreading the draft. The project is partially supported by the Amazon Faculty Research Award, Cisco Research, and the Packard Fellowship. Kangle Deng is supported by the Microsoft Research PhD Fellowship. Ruihan Gao is supported by the A*STAR National Science Scholarship (Ph.D.).

References

[1] 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. In *IEEE International Conference on Computer Vision (ICCV)*, 2023.

[2] Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. In *International Conference on Learning Representations (ICLR)*, 2023.

[3] 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 *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[4] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2014.

[5] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathain, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning (ICML)*, 2015.

[6] 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 *European Conference on Computer Vision (ECCV)*, 2020.

[7] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics (TOG)*, 42(4):1–14, 2023.

[8] Christoph Schuhmann, Richard Vencu, Romain Beaumont, Robert Kaczmarczyk, Clayton Mullis, Aarush Katta, Theo Coombes, Jenia Jitsev, and Aran Komatsuzaki. Laion-400m: Open dataset of clip-filtered 400 million image-text pairs. *arXiv preprint arXiv:2111.02114*, 2021.

[9] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[10] Wenzhen Yuan, Siyuan Dong, and Edward H Adelson. Gelsight: High-resolution robot tactile sensors for estimating geometry and force. *Sensors*, 2017.

[11] Shaoxiong Wang, Yu She, Branden Romero, and Edward Adelson. Gelsight wedge: Measuring high-resolution 3d contact geometry with a compact robot finger. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[12] Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. Sdxl: Improving latent diffusion models for high-resolution image synthesis. In *International Conference on Learning Representations (ICLR)*, 2024.

[13] Xiaoxiao Long, Yuan-Chen Guo, Cheng Lin, Yuan Liu, Zhiyang Dou, Lingjie Liu, Yuexin Ma, Song-Hai Zhang, Marc Habermann, Christian Theobalt, et al. Wonder3d: Single image to 3d using cross-domain diffusion. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[14] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[15] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 1(2):3, 2022.

[16] Minguk Kang, Jun-Yan Zhu, Richard Zhang, Jaesik Park, Eli Shechtman, Sylvain Paris, and Taesung Park. Scaling up gans for text-to-image synthesis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[17] Jiahui Yu, Yuanzhong Xu, Jing Yu Koh, Thang Luong, Gunjan Baid, Zirui Wang, Vijay Vasudevan, Alexander Ku, Yinfei Yang, Burcu Karagol Ayan, et al. Scaling autoregressive models for content-rich text-to-image generation. In *International Conference on Machine Learning (ICML)*, 2022.

 https://doi.org/10.52202/079017-0939

[18] Haochen Wang, Xiaodan Du, Jiahao Li, Raymond A Yeh, and Greg Shakhnarovich. Score jacobian chaining: Lifting pretrained 2d diffusion models for 3d generation. In *IEEE International Conference on Computer Vision (ICCV)*, 2023.

[19] Rui Chen, Yongwei Chen, Ningxin Jiao, and Kui Jia. Fantasia3d: Disentangling geometry and appearance for high-quality text-to-3d content creation. In *IEEE International Conference on Computer Vision (ICCV)*, 2023.

[20] Gal Metzer, Elad Richardson, Or Patashnik, Raja Giryes, and Daniel Cohen-Or. Latent-nerf for shape-guided generation of 3d shapes and textures. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[21] 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. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2024.

[22] Jingxiang Sun, Bo Zhang, Ruizhi Shao, Lizhen Wang, Wen Liu, Zhenda Xie, and Yebin Liu. Dreamcraft3d: Hierarchical 3d generation with bootstrapped diffusion prior. In *International Conference on Learning Representations (ICLR)*, 2024.

[23] Oscar Michel, Roi Bar-On, Richard Liu, Sagie Benaim, and Rana Hanocka. Text2mesh: Text-driven neural stylization for meshes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[24] Peng Wang and Yichun Shi. Imagedream: Image-prompt multi-view diffusion for 3d generation. *arXiv preprint arXiv:2312.02201*, 2023.

[25] Yichun Shi, Peng Wang, Jianglong Ye, Mai Long, Kejie Li, and Xiao Yang. Mvdream: Multi-view diffusion for 3d generation. In *International Conference on Learning Representations (ICLR)*, 2024.

[26] Minghua Liu, Chao Xu, Haian Jin, Linghao Chen, Mukund Varma T, Zexiang Xu, and Hao Su. One-2-3-45: Any single image to 3d mesh in 45 seconds without per-shape optimization. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2024.

[27] Zhizhuo Zhou and Shubham Tulsiani. Sparsefusion: Distilling view-conditioned diffusion for 3d reconstruction. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[28] Minghua Liu, Ruoxi Shi, Linghao Chen, Zhuoyang Zhang, Chao Xu, Xinyue Wei, Hansheng Chen, Chong Zeng, Jiayuan Gu, and Hao Su. One-2-3-45++: Fast single image to 3d objects with consistent multi-view generation and 3d diffusion. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[29] Ruoxi Shi, Hansheng Chen, Zhuoyang Zhang, Minghua Liu, Chao Xu, Xinyue Wei, Linghao Chen, Chong Zeng, and Hao Su. Zero123++: a single image to consistent multi-view diffusion base model. *arXiv preprint arXiv:2310.15110*, 2023.

[30] Yuan Liu, Cheng Lin, Zijiao Zeng, Xiaoxiao Long, Lingjie Liu, Taku Komura, and Wenping Wang. Syncdreamer: Generating multiview-consistent images from a single-view image. In *International Conference on Learning Representations (ICLR)*, 2024.

[31] Yicong Hong, Kai Zhang, Jiuxiang Gu, Sai Bi, Yang Zhou, Difan Liu, Feng Liu, Kalyan Sunkavalli, Trung Bui, and Hao Tan. Lrm: Large reconstruction model for single image to 3d. In *International Conference on Learning Representations (ICLR)*, 2024.

[32] Jiahao Li, Hao Tan, Kai Zhang, Zexiang Xu, Fujun Luan, Yinghao Xu, Yicong Hong, Kalyan Sunkavalli, Greg Shakhnarovich, and Sai Bi. Instant3d: Fast text-to-3d with sparse-view generation and large reconstruction model. In *International Conference on Learning Representations (ICLR)*, 2024.

[33] Yinghao Xu, Hao Tan, Fujun Luan, Sai Bi, Peng Wang, Jiahao Li, Zifan Shi, Kalyan Sunkavalli, Gordon Wetzstein, Zexiang Xu, and Kai Zhang. Dmv3d: Denoising multi-view diffusion using 3d large reconstruction model. In *International Conference on Learning Representations (ICLR)*, 2024.

[34] Dmitry Tochilkin, David Pankratz, Zexiang Liu, Zixuan Huang, , Adam Letts, Yangguang Li, Ding Liang, Christian Laforte, Varun Jampani, and Yan-Pei Cao. Triposr: Fast 3d object reconstruction from a single image. *arXiv preprint arXiv:2403.02151*, 2024.

[35] Jun-Yan Zhu, Zhoutong Zhang, Chengkai Zhang, Jiajun Wu, Antonio Torralba, Josh Tenenbaum, and Bill Freeman. Visual object networks: Image generation with disentangled 3d representations. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2018.

[36] Xiaogang Wang, Marcelo H Ang, and Gim Hee Lee. Voxel-based network for shape completion by leveraging edge generation. In *IEEE International Conference on Computer Vision (ICCV)*, 2021.

[37] Pedro O O Pinheiro, Joshua Rackers, Joseph Kleinhenz, Michael Maser, Omar Mahmood, Andrew Watkins, Stephen Ra, Vishnu Sresht, and Saeed Saremi. 3d molecule generation by denoising voxel grids. 2024.

[38] Yongbin Sun, Yue Wang, Ziwei Liu, Joshua Siegel, and Sanjay Sarma. Pointgrow: Autoregressively learned point cloud generation with self-attention. In *IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2020.

[39] Zijie Wu, Yaonan Wang, Mingtao Feng, He Xie, and Ajmal Mian. Sketch and text guided diffusion model for colored point cloud generation. In *IEEE International Conference on Computer Vision (ICCV)*, 2023.

[40] Yoni Kasten, Ohad Rahamim, and Gal Chechik. Point cloud completion with pretrained text-to-image diffusion models. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2024.

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

[42] 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 *IEEE International Conference on Computer Vision (ICCV)*, 2023.

[43] Paul Henderson, Vagia Tsiminaki, and Christoph H Lampert. Leveraging 2d data to learn textured 3d mesh generation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[44] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2mesh: Generating 3d mesh models from single rgb images. In *European Conference on Computer Vision (ECCV)*, 2018.

[45] Chao Wen, Yinda Zhang, Zhuwen Li, and Yanwei Fu. Pixel2mesh++: Multi-view 3d mesh generation via deformation. In *IEEE International Conference on Computer Vision (ICCV)*, 2019.

[46] Can Wang, Menglei Chai, Mingming He, Dongdong Chen, and Jing Liao. Clip-nerf: Text-and-image driven manipulation of neural radiance fields. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[47] Amit Raj, Srinivas Kaza, Ben Poole, Michael Niemeyer, Nataniel Ruiz, Ben Mildenhall, Shiran Zada, Kfir Aberman, Michael Rubinstein, Jonathan Barron, et al. Dreambooth3d: Subject-driven text-to-3d generation. In *IEEE International Conference on Computer Vision (ICCV)*, 2023.

[48] Yen-Chi Cheng, Hsin-Ying Lee, Sergey Tulyakov, Alexander G Schwing, and Liang-Yan Gui. Sdfusion: Multimodal 3d shape completion, reconstruction, and generation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[49] Jiaxiang Tang, Jiawei Ren, Hang Zhou, Ziwei Liu, and Gang Zeng. Dreamgaussian: Generative gaussian splatting for efficient 3d content creation. In *International Conference on Learning Representations (ICLR)*, 2024.

[50] Jiaxiang Tang, Zhaoxi Chen, Xiaokang Chen, Tengfei Wang, Gang Zeng, and Ziwei Liu. Lgm: Large multi-view gaussian model for high-resolution 3d content creation. In *European Conference on Computer Vision (ECCV)*, 2024.

[51] An-Chieh Cheng, Xueting Li, Sifei Liu, and Xiaolong Wang. Tuvf: Learning generalizable texture uv radiance fields. In *International Conference on Learning Representations (ICLR)*, 2024.

[52] Fanbo Xiang, Zexiang Xu, Milos Hasan, Yannick Hold-Geoffroy, Kalyan Sunkavalli, and Hao Su. Neutex: Neural texture mapping for volumetric neural rendering. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[53] Dave Zhenyu Chen, Yawar Siddiqui, Hsin-Ying Lee, Sergey Tulyakov, and Matthias Nießner. Text2tex: Text-driven texture synthesis via diffusion models. In *IEEE International Conference on Computer Vision (ICCV)*, 2023.

[54] 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 (ECCV)*, 2022.

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

[56] Kangle Deng, Timothy Omernick, Alexander Weiss, Deva Ramanan, Jun-Yan Zhu, Tinghui Zhou, and Maneesh Agrawala. Flashtex: Fast relightable mesh texturing with lightcontrolnet. In *European Conference on Computer Vision (ECCV)*, 2024.

[57] Aymen Mir, Thiemo Alldieck, and Gerard Pons-Moll. Learning to transfer texture from clothing images to 3d humans. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[58] Tuanfeng Y Wang, Hao Su, Qixing Huang, Jingwei Huang, Leonidas J Guibas, and Niloy J Mitra. Unsupervised texture transfer from images to model collections. *ACM Transactions on Graphics (TOG)*, 2016.

[59] Anita Hu, Nishkrit Desai, Hassan Abu Alhaija, Seung Wook Kim, and Maria Shugrina. Diffusion texture painting. In *ACM SIGGRAPH*, 2024.

[60] Hendrik Baatz, Jonathan Granskog, Marios Papas, Fabrice Rousselle, and Jan Novák. Nerf-tex: Neural reflectance field textures. In *Eurographics Symposium on Rendering (EGSR)*, 2021.

[61] Keunhong Park, Konstantinos Rematas, Ali Farhadi, and Steven M. Seitz. Photoshape: Photorealistic materials for large-scale shape collections. *ACM Transactions on Graphics (TOG)*, 2018.

 https://doi.org/10.52202/079017-0939

[62] Yi-Hua Huang, Yan-Pei Cao, Yu-Kun Lai, Ying Shan, and Lin Gao. Nerf-texture: Texture synthesis with neural radiance fields. In *ACM SIGGRAPH*, 2023.

[63] Sudharshan Suresh, Zilin Si, Joshua G Mangelson, Wenzhen Yuan, and Michael Kaess. Shapemap 3-d: Efficient shape mapping through dense touch and vision. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2022.

[64] Mohamed Tahoun, Omar Tahri, Juan Antonio Corrales Ramón, and Youcef Mezouar. Visual-tactile fusion for 3d objects reconstruction from a single depth view and a single gripper touch for robotics tasks. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021.

[65] Edward Smith, David Meger, Luis Pineda, Roberto Calandra, Jitendra Malik, Adriana Romero Soriano, and Michal Drozdzal. Active 3d shape reconstruction from vision and touch. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2021.

[66] Shaoxiong Wang, Jiajun Wu, Xingyuan Sun, Wenzhen Yuan, William T Freeman, Joshua B Tenenbaum, and Edward H Adelson. 3d shape perception from monocular vision, touch, and shape priors. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018.

[67] Aiden Swann, Matthew Strong, Won Kyung Do, Gadiel Sznaier Camps, Mac Schwager, and Monroe Kennedy III. Touch-gs: Visual-tactile supervised 3d gaussian splatting. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024.

[68] Mauro Comi, Yijiong Lin, Alex Church, Alessio Tonioni, Laurence Aitchison, and Nathan F Lepora. Touchsdf: A deepsdf approach for 3d shape reconstruction using vision-based tactile sensing. In *Conference on Neural Information Processing Systems (NeurIPS) Workshop*, 2023.

[69] Edward Smith, Roberto Calandra, Adriana Romero, Georgia Gkioxari, David Meger, Jitendra Malik, and Michal Drozdzal. 3d shape reconstruction from vision and touch. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2020.

[70] Sudharshan Suresh, Haozhi Qi, Tingfan Wu, Taosha Fan, Luis Pineda, Mike Lambeta, Jitendra Malik, Mrinal Kalakrishnan, Roberto Calandra, Michael Kaess, et al. Neuralfeels with neural fields: Visuotactile perception for in-hand manipulation. *Science Robotics*, 9(96):eadl0628, 2024.

[71] Elliott Donlon, Siyuan Dong, Melody Liu, Jianhua Li, Edward Adelson, and Alberto Rodriguez. Gelslim: A high-resolution, compact, robust, and calibrated tactile-sensing finger. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018.

[72] Yunzhu Li, Jun-Yan Zhu, Russ Tedrake, and Antonio Torralba. Connecting touch and vision via cross-modal prediction. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[73] Ruihan Gao, Wenzhen Yuan, and Jun-Yan Zhu. Controllable visual-tactile synthesis. In *IEEE International Conference on Computer Vision (ICCV)*, 2023.

[74] Fengyu Yang, Chao Feng, Ziyang Chen, Hyoungseob Park, Daniel Wang, Yiming Dou, Ziyao Zeng, Xien Chen, Rit Gangopadhyay, Andrew Owens, et al. Binding touch to everything: Learning unified multimodal tactile representations. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[75] Yiming Dou, Fengyu Yang, Yi Liu, Antonio Loquercio, and Andrew Owens. Tactile-augmented radiance fields. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[76] Mauro Comi, Alessio Tonioni, Max Yang, Jonathan Tremblay, Valts Blukis, Yijiong Lin, Nathan F Lepora, and Laurence Aitchison. Snap-it, tap-it, splat-it: Tactile-informed 3d gaussian splatting for reconstructing challenging surfaces. *arXiv preprint arXiv:2403.20275*, 2024.

[77] Alexei A. Efros and William T. Freeman. Image quilting for texture synthesis and transfer. In *ACM SIGGRAPH*. Association for Computing Machinery, 2001.

[78] 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.

[79] Samuli Laine, Janne Hellsten, Tero Karras, Yeongho Seol, Jaakko Lehtinen, and Timo Aila. Modular primitives for high-performance differentiable rendering. *ACM Transactions on Graphics (TOG)*, 39(6), 2020.

[80] Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. Sdedit: Guided image synthesis and editing with stochastic differential equations. In *International Conference on Learning Representations (ICLR)*, 2022.

[81] Ayaan Haque, Matthew Tancik, Alexei A Efros, Aleksander Holynski, and Angjoo Kanazawa. Instruct-nerf2nerf: Editing 3d scenes with instructions. In *IEEE International Conference on Computer Vision (ICCV)*, 2023.

[82] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models, 2023.

[83] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.

[84] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2022.

[85] 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 *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[86] Junjiao Tian, Lavisha Aggarwal, Andrea Colaco, Zsolt Kira, and Mar Gonzalez-Franco. Diffuse, attend, and segment: Unsupervised zero-shot segmentation using stable diffusion. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[87] Songwei Ge, Taesung Park, Jun-Yan Zhu, and Jia-Bin Huang. Expressive text-to-image generation with rich text. In *IEEE International Conference on Computer Vision (ICCV)*, 2023.

[88] Or Patashnik, Daniel Garibi, Idan Azuri, Hadar Averbuch-Elor, and Daniel Cohen-Or. Localizing object-level shape variations with text-to-image diffusion models. In *IEEE International Conference on Computer Vision (ICCV)*, 2023.

[89] Jiale Xu, Weihao Cheng, Yiming Gao, Xintao Wang, Shenghua Gao, and Ying Shan. Instantmesh: Efficient 3d mesh generation from a single image with sparse-view large reconstruction models. *arXiv preprint arXiv:2404.07191*, 2024.

[90] Lingteng Qiu, Guanying Chen, Xiaodong Gu, Qi Zuo, Mutian Xu, Yushuang Wu, Weihao Yuan, Zilong Dong, Liefeng Bo, and Xiaoguang Han. Richdreamer: A generalizable normal-depth diffusion model for detail richness in text-to-3d. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

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

[92] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning (ICML)*, 2021.

 https://doi.org/10.52202/079017-0939

Appendix

A Implementation Details and Additional Results

Please check out our webpage for video results and more examples.

A.1 Diffusion-based Multi-Part Segmentation

In Section 4.3, we have mentioned using attention maps of the diffusion process to segment the input image based on text prompts. We provide more details here. Assume an input image x and its corresponding prompt $y = [y_1, y_2, \cdots, y_T]$ are given, where T is the number of tokens from the text encoder ($T = 77$ for CLIP text encoder [92]). Among these, $y_{p_1}, \cdots, y_{p_N}$ define N parts in the image x, where $p_1, \cdots, p_N$ indicate the indexes of the token. For example, with the prompt "a cactus in a pot", the second token y_2 ($p_1 = 2$) "cactus" and the fifth token y_5 ($p_2 = 5$) "pot" correspond to two parts ($N = 2$) of the object in the image, and our method outputs the segmentation mask for each part. Specifically, we first perturb x using random noise ϵ_t with a noise level of t, which is empirically set as 0.2. We then use the Stable Diffusion UNet to denoise x_t for one step. During the denoising process, we collect the cross-attention and self-attention probability maps for each of the 16 layers.

Segment and create masks using self-attention layers. Following DiffSeg [86], we aggregate the 16 self-attention probability maps to a single map $\mathcal{A}_f$ with the shape of $64 \times 64 \times 64 \times 64$, and run iterative attention merging [86] on upsampled $\mathcal{A}_f$ to obtain a preliminary cluster probability map $\tilde{\mathcal{A}}_f \in \mathbb{R}^{512\times512\times K}$, where each $\tilde{\mathcal{A}}_f[:,:,k], k = 1, \cdots, K$, is a probability map,

$$\sum_{i,j} \tilde{\mathcal{A}}_f[i,j,k] = 1, \quad k = 1, 2, \cdots, K. \tag{13}$$

We can then obtain a preliminary segmentation mask $\tilde{S}$,

$$\forall i, j, \qquad \tilde{S}[i,j] = \arg\max_k \tilde{\mathcal{A}}_f[i,j,k]. \tag{14}$$

Aggregate labels with cross-attention layers. Similarly, we aggregate the 16 cross-attention maps and up-sample them to $\mathcal{A}_c$ with the shape of $512 \times 512 \times 77$. To find the exact pixels corresponding to each part specified in the prompt input, we extract cross-attention maps associated with the N tokens $y_{p_1}, \cdots, y_{p_N}$, $\tilde{\mathcal{A}}_c \in \mathbb{R}^{512\times512\times N}$,

$$\forall i, j, n \qquad \tilde{\mathcal{A}}_c[i,j,n] = \frac{\mathcal{A}_c[i,j,p_n]}{\sum_{i',j'} \mathcal{A}_c[i',j',p_n]}. \tag{15}$$

Silimar to the self-attention probability maps $\tilde{\mathcal{A}}_f$, here we normalize the cross-attention maps over the spatial dimensions to ensure $\tilde{\mathcal{A}}_c[:,:,n]$ is a probability map for each n, where $\sum_{i,j} \tilde{\mathcal{A}}_c[i,j,n] = 1$.

Associate masks with labels. Note that $\tilde{S}$ contains K parts but in general $K \neq N$. Therefore, we need to assign each of the K segments to one of the N parts specified in the prompt.

We calculate a KL divergence matrix $D \in \mathbb{R}^{K\times N}$ between $\tilde{\mathcal{A}}_f$ and $\tilde{\mathcal{A}}_c$,

$$D[k,n] = D_{\text{KL}}\left(\tilde{\mathcal{A}}_f[k] \parallel \tilde{\mathcal{A}}_c[n]\right), \quad k = 1, 2, \cdots, K, \quad n = 1, 2, \cdots, N. \tag{16}$$

We merge the preliminary mask $\tilde{S}$ according to D to get the final segmentation S,

$$\forall i, j, \qquad S[i,j] = \arg\min_n D[\tilde{S}[i,j], n]. \tag{17}$$

An example of output masks is shown in Figure 12.

A.2 Dataset Details

Figure 13 shows additional textures included in our `TouchTexture` dataset, and Table 2 shows the list of text descriptions corresponding to each texture.

https://doi.org/10.52202/079017-0939

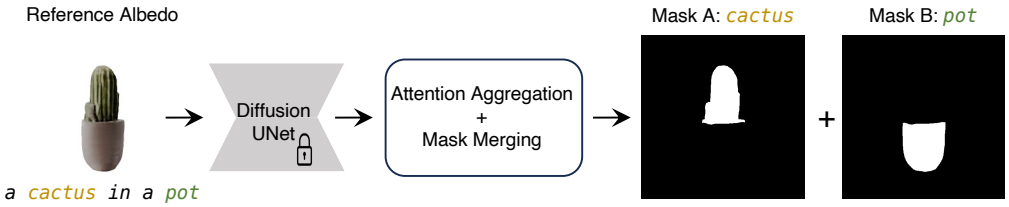


Figure 12: **Illustration of diffusion-based multi-parts segmentation.** At each iteration, we run a forward pass of the diffusion model for the target albedo image, aggregate its self-attention and cross-attention maps, and compute KL distance to merge the masks into N parts. We show an example of "a cactus in a pot", where we extract the masks corresponding to two tokens "cactus" and "pot", shown as *Mask A* and *Mask B*. The segmentation masks are then used to supervise the label field training to enable multi-part synthesis.

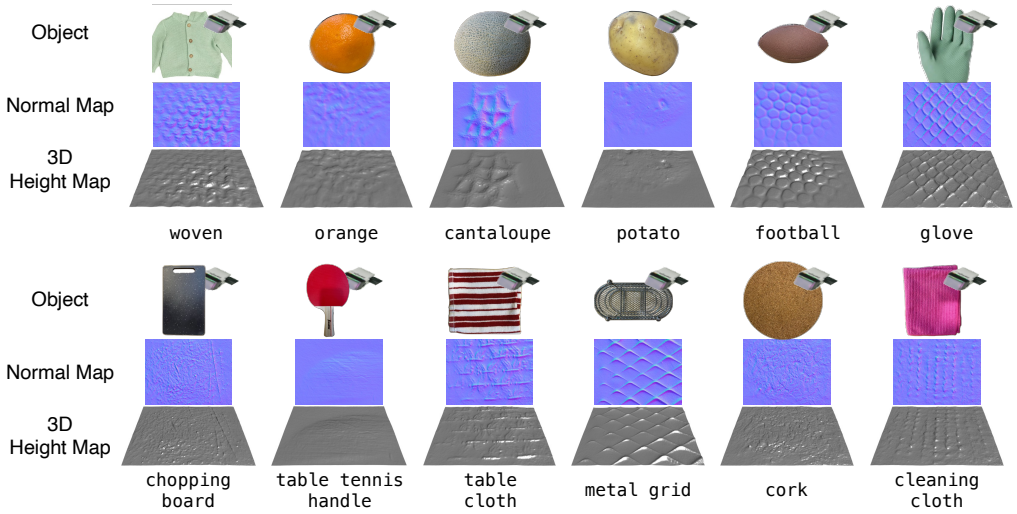


Figure 13: **Additional materials in tactile normal dataset** `TouchTexture`**.** We collect tactile normal data from 18 daily objects featuring diverse tactile textures. We show the tactile normal map and a 3D height map for each object. This library of tactile samples covers a wide range of diverse materials commonly found in daily life.

A.3 Training Details

Regarding the network, we use TCNN encoding with two base linear layers, which then branches out into three output layers, one linear layer followed by sigmoid activation for albedo output, one linear layer followed by $\tanh$ activation for tactile normal output, and one optional linear layer for label field. We follow DreamBooth [85] to train texture LoRAs with Stable Diffusion (SD) V1.4 for the tactile guidance loss. We find empirically that SD V1.4 works better for generating tactile normal maps. We use ControlNet (v1.1 - normalbae version) with SD V1.5 for the diffusion loss. We follow Wonder3D to generate the base mesh and train the texture field network using Adam optimizer with $lr = 0.01$. We train all models on A6000 GPUs and each experiment takes about 10 mins and 20G RAM to run.

A.4 Flexible Base Mesh Generation

Given a text or image input, our method seamlessly integrates with a wide range of mesh generation approaches to enhance the 3D output with refined details. Here, we presents results using RichDreamer [90] for base mesh generation in Figure 14a and results using InstantMesh [89] in Figure 14b. RichDreamer generates detailed and consistent 3D meshes from text prompts by incorporating depth and normal priors during pre-training, while InstantMesh employs a feed-forward image-to-3D approach with a transformer-based reconstruction model and FlexiCubes for efficient mesh production. Both methods provide solid base meshes, and our approach effectively adapts to each, refining the output meshes with intricate texture details for greater realism and fidelity.

https://doi.org/10.52202/079017-0939

Table 2: List of objects in TouchTexture and their text descriptions.

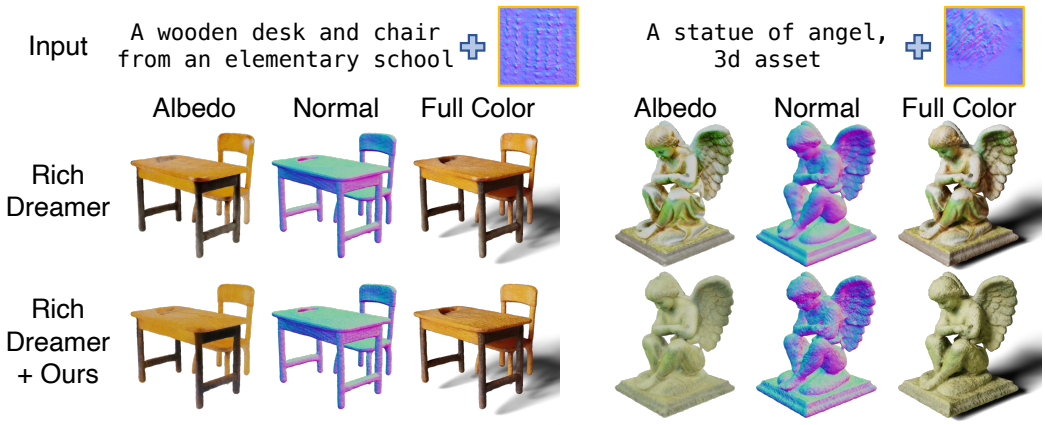
Object Name	Description
avocado	"avocado skin's "
strawberry	"strawberry"
canvas bag	"canvas bag"
striped steel	"fine striped steel"
corn	"corn"
rubber	"heat-resistant rubber with heart-shape pattern"
woven	"woven crochet"
orange	"orange"
cantaloupe	"cantaloupe skin's"
potato	"potato"
football	"football"
glove	"thin rubber with grid pattern"
chopping board	'cutting board"
table tennis handle	"smooth wood"
table cloth	"patterned table cloth"
metal grid	"metal grid"
cork	"cork mat"
cleaning cloth	"cleaning cloth's"

A.5 Human Perceptual Study Details

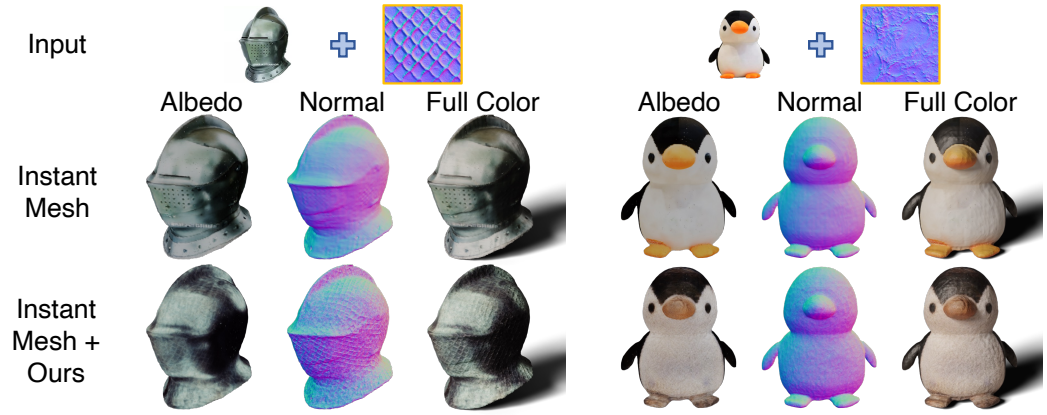
We perform a human perceptual study on Amazon Mechanical Turk (AMTurk). We set up a paired test, showing a reference prompt and two rendering results, one generated with our method and the other generated with one of the baselines. For each result, we include a single view of the entire object with a zoom-in patch in the red box to show the details. We use directional light shooting in from the left side for all mesh renderings. We give a text prompt, e.g., "an avocado", and ask the user to select the result that better matches the prompt. We render shaded colorless images for geometry evaluation as shown in Figure 15a and render full-color images for texture evaluation as shown in Figure 15b. The estimated time for each test is about 5 minutes, and we pay users a corresponding amount of compensation higher than the minimum wage in the country of the data collector. The online survey does not pose any explicit potential risks expected to be incurred by participants.

B Soceital Impacts

Our method of incorporating tactile information into 3D generation enhances the geometric details of the generated results. This advancement helps automatically create high-fidelity assets, which are highly applicable in game and film production. However, there is also a potential risk of misuse, as 3D assets could be used to generate fake content for misinformation. Despite this concern, we believe humans can currently distinguish our synthesized objects from real ones.

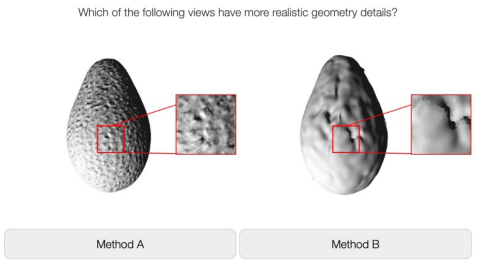


(a) We integrate RichDreamer, a purely text-to-3D baseline, into our method.

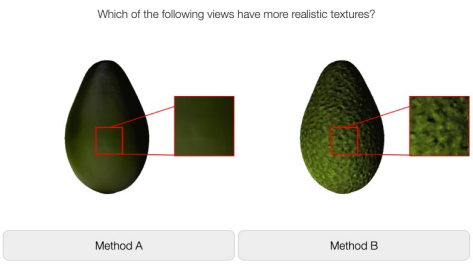


(b) We integrate InstantMesh, an image-to-3D baseline, into our method

Figure 14: Our method is flexible and compatible with diverse text-to-3D and image-to-3D pipelines.



(a) A sample test page for geometry evaluation.



(b) A sample test page for texture evaluation.

Figure 15: **AMTurk setup for evaluation**. For each paired result, we include a single view of the entire object with a zoom-in patch in the red box to show the details. For each result, we include a single view of the entire object with a zoom-in patch in the red box to show the details. We use directional light shooting in from the left side for all mesh renderings. We give a text prompt, e.g., "an avocado", and ask the user to select the result that better matches the prompt.

https://doi.org/10.52202/079017-0939

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Yes, we include the paper's contribution and scope in the abstract and introduction section.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Yes, we discuss the limitations of the work in the main paper.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

Justification: The paper does not include theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We describe the network architecture and implementation details in the paper. We will also release our code and dataset upon publication.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

https://doi.org/10.52202/079017-0939

Answer: [Yes]

Justification: We describe the implementation details in the paper to ensure reproducibility. The code and data will be released on our project website.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide essential experiment settings in the main paper and provide additional details and parameters in the supplementary document.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We provide error bars for all quantitative evaluations.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide the computational requirement, memory usage, and time of execution in the supplementary material.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Yes, our research conforms well with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Yes, we discuss societal impact in the supplementary material.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

https://doi.org/10.52202/079017-0939

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: We do not release data or models that have a high risk of misuse and thus pose no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We cite all the original paper/code package/dataset to our best efforts.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: The paper does not release new assets. The code and data will be released upon publication.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [Yes]

    Justification: We use AMTurk for user study and provide required details in the supplementary material.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [Yes]

    Justification: Yes, we provide related details in the supplementary material.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

https://doi.org/10.52202/079017-0939