TexPro: Text-guided PBR Texturing with Procedural Material Modeling

Ziqiang Dang*
Zhejiang University
China
ZiqDang@zju.edu.cn

Wenqi Dong*
Zhejiang University
China
wqdong98@gmail.com

Zesong Yang
Zhejiang University
China
zesongyang0@zju.edu.cn

Bangbang Yang
ByteDance Inc
China
ybbbbt@gmail.com

Liang Li
Communication University of Zhejiang
China
liliang@cuz.edu.cn

Yuewen Ma
ByteDance Inc
China
mayuewen@bytedance.com

Zhaopeng Cui†
Zhejiang University
China
zhpcui@zju.edu.cn

Abstract

In this paper, we present TexPro, a novel method for high-fidelity material generation for input 3D meshes given text prompts. Unlike existing text-conditioned texture generation methods that typically generate RGB textures with baked lighting, TexPro is able to produce diverse texture maps via procedural material modeling, which enables physical-based rendering, relighting, and additional benefits inherent to procedural materials. Specifically, we first generate multi-view reference images given the input textual prompt by employing the latest text-to-image model. We then derive texture maps through a rendering-based optimization with recent differentiable procedural materials. To this end, we design several techniques to handle the misalignment between the generated multi-view images and 3D meshes, and introduce a novel material agent that enhances material classification and matching by exploring both part-level understanding and object-aware material reasoning. Experiments demonstrate the superiority of the proposed method over existing SOTAs and its capability of relighting.

Keywords: ***3D asset creation, text-guided texturing, procedural material, MLLM***

1. Introduction

3D mesh texturing is a critical process in the realm of 3D modeling and visualization, which holds significant importance across various applications such as AR/VR, gaming, filmmaking, *etc*. By transforming a basic geometric shape into a visually detailed and realistic object, mesh texturing not only improves the aesthetic appeal of the model but also greatly enhances immersion and realism in digital environments. To achieve photorealistic rendering that responds accurately to various lighting conditions, we need to generate object material that includes a set of texture maps like diffuse and specular reflections, glossiness, surface roughness, *etc*. As a result, traditional mesh texturing normally involves substantial manual effort. Recent advances in 2D image generation have spurred the development of automated mesh texturing methods [6, 33], but they normally generate RGB textures with baked lighting, which limits their application in downstream tasks.

In this paper, we propose a novel method for producing high-quality materials for 3D meshes based on user-provided text prompts as shown in Fig. 1. Unlike previous methods that directly generate pre-baked 2D texture maps, our approach is able to produce diverse texture maps via procedural material modeling, which enables physical-based rendering, relighting, and additional benefits inherent to procedural materials, such as resolution independence (*i.e.*, producing textures at any desired resolution), re-editability, efficient storage, and so on. The proposed system could simplify the workflow for amateur artists, eliminating the need to spend time on tedious parameter tuning for procedural materials. Moreover, the system's output can serve as a solid initialization, allowing artists to further adjust the parameters of nodes in the procedural material graph to achieve the desired detailed effects without starting from scratch.

Specifically, we first generate multi-view reference images given the input textual prompt by employing the latest text-to-image models [34, 43], and then obtain diverse texture maps through a rendering-based optimization with recent differentiable procedural materials [24, 35]. After the optimization, we can essentially obtain procedural materials for the mesh, thus supporting downstream applications. However, it is non-trivial to design such a system.

*Authors contributed equally.
†Corresponding author.

Figure 1. **Texturing Results.** We introduce TexPro, a text-driven PBR material generation method for photorealistic and relightable rendering.

First, despite using additional rendered depth or normal maps as conditions [43], the generated images from the text-to-image model (*e.g.*, Stable Diffusion [34]) often fail to align accurately with the meshes and may not maintain consistency across different views. To handle these misalignment challenges, as shown in Fig. 2, we propose to exploit an improved segmentation method Matcher [27] to extract the masks of corresponding objects in generated images, which will be further smoothed and integrated with the rendered masks to get the final aligned masks. Additionally, we design an adaptive camera sampling method that ensures each material part exceeds a specified pixel count threshold and assigns a unique material to each part to address the inconsistency across views.

Second, an initial estimation of material properties is requisite for differentiable procedural material rendering. However, the presence of baked lighting in generated images and the confusion of different materials with similar color distributions make initial material classification and matching challenging for classical methods [15, 31]. To overcome these challenges, we leverage the capabilities of Multi-modal Large Language Models (MLLMs) to create a novel material agent. Given the masked prompt image, the material agent is able to provide the possible material types in the order of likelihood for the corresponding object via the part-level understanding and object-aware material reasoning, which further facilitates the rendering-based optimization with procedural materials.

Additionally, to better recover the material properties, we also consider the environmental lighting during the optimization. To make the optimization tractable, we adopt a vanilla lighting setup to fit the lighting environment in the image. Given the object mesh, our method adaptively creates the floor and four walls in the scene of the differentiable renderer, and adaptively places initial area lights on each wall and above the object. Then we optimize the RGB intensity of these lights together with procedural materials.

Our contributions can be summarized as follows:

- We propose a novel framework that is able to create high-fidelity textures for input meshes given text prompts and enable physical-based rendering (PBR) and relighting.
- In order to handle the misalignment between the generated multi-view images and 3D meshes, we propose to combine the rendered mask with an advanced segmentation method, *i.e.*, Matcher [27], to achieve accurately aligned part-level masks, and design an adaptive camera sampling strategy to deal with inconsistency across views.
- We present a novel material agent for robust material classification and matching by exploring both part-level understanding and object-aware material reasoning.
- The experiments show that the proposed method outperforms existing SOTAs and maintains the capabilities of relighting.

2. Related Work

Texture generation. Texture maps are essential for 3D geometric models, which involve mapping from pre-defined 2D planar images onto 3D models, endowing 3D models with vital color information. The traditional texture creation involves cumbersome manual drawing, assembling repetitive patterns, or stitching multi-view images [3, 38]. With the development of increasingly high-quality 3D datasets and advancements in text-to-image generative techniques [32, 34, 43], learning-based approaches have been proposed to generate high-quality textures [6, 20, 28, 33, 40]. Text2Tex [6] incorporates a depth-aware image inpainting model to synthesize high-resolution partial textures progressively from multiple viewpoints, which has also been widely employed in text-to-scene tasks [9,14,42]. Similarly, TEXTure [33] also applies an iterative scheme to paint 3D models, while presenting a trimap representation and novel elaborated diffusion sampling process to tackle inconsistencies issues. Although existing methods are capable of generating high-quality textures, they are limited to generating only diffuse maps with baked lighting, lacking support for relighting. Additionally, the generated textures often suffer from Janus (multi-faced) problem [2] and lack realism.

Procedural material modeling and capture. Procedural material has been the standard of material modeling in industry. The materials are represented as node graphs denoted with simple image processing operations which are combined to produce real-world spatially varying BRDF [5] material maps. Given user-specified images or text prompts, many approaches [12, 16, 17, 24, 35] have been proposed to recover the parameters of a predefined procedural material graph to align with the input. Moreover, Hu *et al.* [18] moved beyond parameter regression, utilizing the diffusion model to generate graph structures from text or image conditions. It is worth noting that, MATch [35] proposes a differentiable procedural material method that translates the procedural graphs into differentiable node graphs, enabling single-shot high-quality procedural material capture.

Differentiable rendering. Differentiable rendering has consistently remained a significant research topic in computer graphics and vision. The gradient in rendering is required with respect to numerous factors, and specially several differentiable renderers widely used in community like Redner [25], Mitsuba 2/3 [19, 30], Nvdiffrast [23] and PSDR-CUDA [41] have been developed to address challenges arising from the non-differentiable terms in rendering integral. We leverage the differentiable rendering to backpropagate the gradient from pixel space to the parameters of procedural materials, thereby optimizing the materials via gradient descent.

3. Method

Most 3D meshes from online resource repositories like BlenderKit or Sketchfab, as well as meshes from existing 3D datasets [7, 8, 10, 26, 37], are segmented into different material parts by their authors during creation, *e.g.*, a chair consists of the frame, backrest and cushion. Moreover, part-level segmentation on a complete mesh can be achieved through shape analysis methods [13, 22, 39]. As a result, given textual prompts, we aim to generate appropriate materials for each material part of the input 3D mesh model and achieve photorealistic and relightable texturing.

As shown in Fig. 2, our method textures the mesh with PBR materials through three steps: multi-view reference images generation, materials initialization under material agent guidance, and materials optimization using differentiable rendering. In this section, we begin by introducing the procedural material and a differentiable version *DiffMat* in Sec. 3.1. We then provide the detailed descriptions of these three steps in Sec. 3.2, Sec. 3.3 and Sec. 3.4.

3.1. Preliminary

Procedural materials. Procedural material (*e.g.*, Substance materials [1]) is a standard way of modeling spatially-varying materials in the 3D design industry, which generates texture maps algorithmically. Different from the static, pre-drawn or pre-baked texture maps, procedural materials are represented as directed acyclic node graphs, primarily consisting of two types of nodes: generators and filters. Generator nodes create spatial textures from scratch, and filter nodes manipulate input textures using specific image processing operations. Specifically, each node has specific parameters, and changing these parameter values will produce different texture maps, including base color, roughness, normal and other maps. Procedural materials provide many beneficial properties, such as being resolution-independent, re-editability, the ability to be dynamically changed during runtime, more efficient in terms of storage space and memory usage and so on.

DiffMat. Shi *et al.* developed the *DiffMat* library [35], skillfully utilizing convolutional neural networks to implement the functionality of most filter nodes, thereby making procedural materials differentiable. Recently, *DiffMatV2* [24] was proposed to implement differentiable generator nodes and expand the scope of optimizable parameters of filter nodes. In this work, we utilize *DiffMatV2* to optimize the parameters θ in the node graphs.

3.2. Multi-view Reference Images Generation

Camera views selection. Since the input mesh is assumed to be segmented beforehand, the mask M_R for each material part in a certain camera view can be rendered directly through the renderer. We first adaptively select some cam-

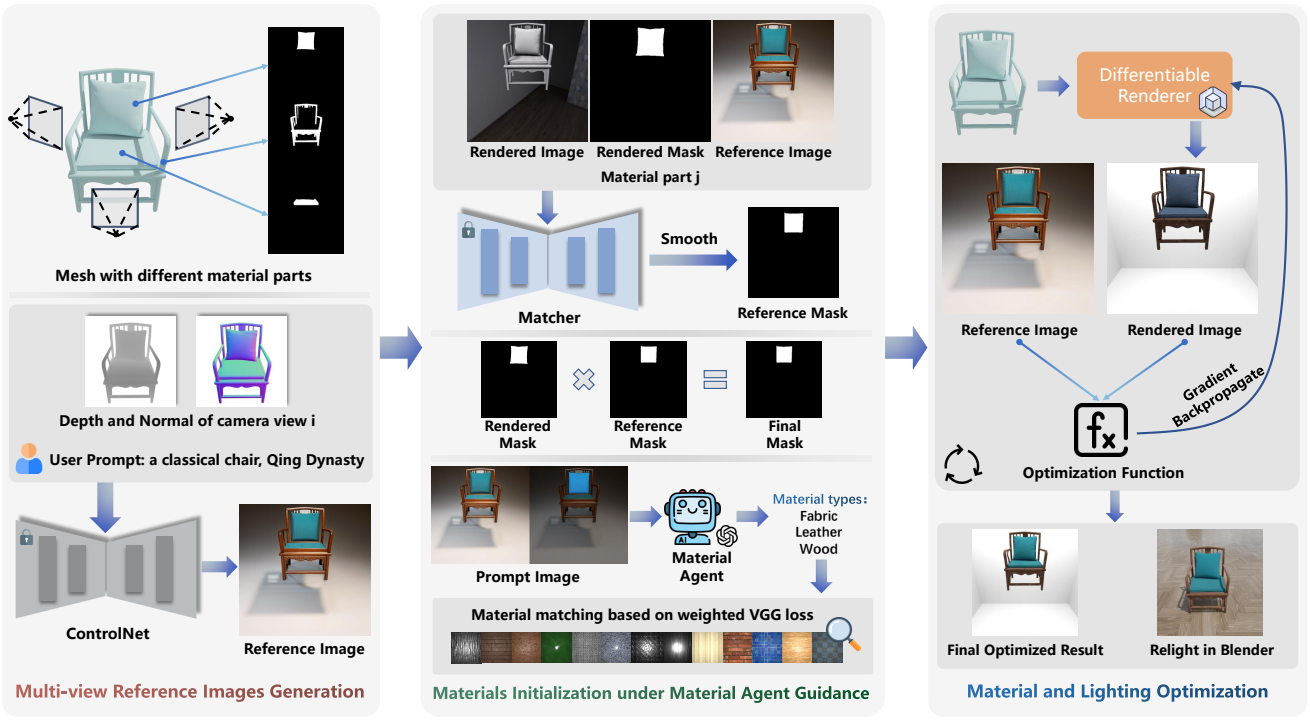


Figure 2. **Overview of TexPro.**

era views to ensure that the number of pixels for each material part exceeds a certain threshold. Note that the front view has to be selected because the reference image of this view generated by Stable Diffusion [34] is more realistic. The number of selected views is unconstrained. For more details about the camera selection strategy, please refer to our supplementary document.

Reference images generation. We use the differentiable renderer PSDR-CUDA [41] to render the normal maps and depth maps of the mesh under the selected camera views. Subsequently, we use Stable Diffusion and Controlnet [43] to generate the reference image of front view under the condition of the corresponding normal map. For other camera views, we also use the reference image of front view as an additional condition to generate reference images. Nonetheless, there may still be slight differences between images of different views. To deal with this problem, we select only one reference image for each material part according to the number of pixels in each view. Moreover, if the image quality is not good, the rendered depth maps will be added as an extra condition.

3.3. Materials Initialization with Agent Guidance

Since an initial estimation of material properties is requisite for differentiable procedural material rendering, we adopt a two-step approach to select the initial procedural material: first by classification, then by feature matching. However, we find that the classical material classification methods [15, 31] suffer from the baked lighting in generated images and confusion of different materials with similar color distributions. Thus, we design a novel material agent using GPT-4V for accurate material initialization via part-level understanding and object-aware material reasoning, *e.g.*, pillows should be assigned fabric or leather materials instead of metal or wood. Specifically, as shown in Fig. 3, given the masked prompt image and the text prompts, the material agent will provide all possible material types in the order of likelihood. Additionally, Fig. 4 presents some examples of agent response and Fig. 10 illustrates the complete classification process and results for a bed mesh.

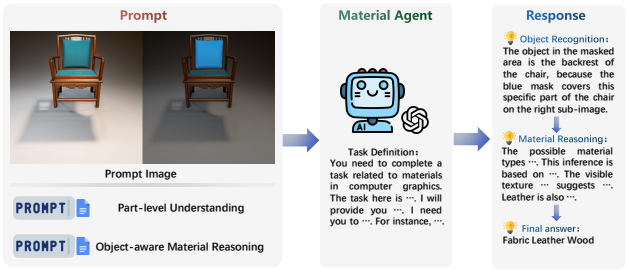


Figure 3. **Material Agent.** An example of a chair backrest is shown here. By providing both the image prompt and text prompt to our defined material agent, it can predict all possible material categories in order of likelihood.

Masks segmentation and alignment. Since the object geometry in the images generated by Stable Diffusion is not well aligned with the rendered images, the mask M_R on the

Figure 4. **Agent Response Examples.** On the left are the prompt images provided to the material agent, and on the right are the responses returned by the agent.

rendered images can not be directly applied to the reference images. Therefore, we use Matcher [27] to get the mask M_I on the reference images. Specifically, as shown in Fig. 2, for camera view i, we provide three inputs to Matcher: the rendering of the object textured by white material under specific ambient lighting, the mask M_R of a material part on the rendered image, and the reference image. The Matcher then determines the corresponding mask M_I of the material part on the reference image. Next, we use median filtering, morphological operations, and region area filtering to smooth M_I obtained by Matcher. The final mask M_{ij} of the material part j under camera view i is obtained by $M_{ij} = M_R * M_I$. M_{ij} will be used on both rendered images and reference images.

Material agent prompt design. For each material part, we blend its mask M_{ij} with the reference image, and then concatenate it horizontally with the reference image to create the prompt image. Given the name of the mesh (*e.g.*, chair), a specific prompt text can be derived from our carefully designed prompt template to guide the material agent. Note that all material parts share the same prompt text, rather than deriving a prompt for every material part. Our template starts by using the GPT-4V system prompt obtained through reverse prompt engineering to enhance performance. Then, we provide a detailed definition of the task along with an example for in-context learning [29]. Subsequently, we employ various prompt techniques to construct the prompt for part-level understanding and object-aware material reasoning. Please refer to our supplementary for the complete prompt template.

Material agent part-level understanding. Similar to the idea of chain of thought [4], we first guide the agent to identify what object is in the marked area, instead of directly inferring the materials. This way allows object information to be taken into account during material reasoning, enabling full use of the extensive world knowledge stored in LLMs. In this part of the prompt, we first explain the prompt image and ask the agent to zoom in on the corresponding area for better recognition. Additionally, we highlight some considerations, such as occlusion issues, and then ask the agent to perform object recognition step-by-step.

Material agent object-aware material reasoning. In this part, we ask the agent to combine pixel information and object information to reason about all possible material types and give prediction results in the order of possibility. We have divided the materials in the *DiffMat* library into 19 categories including asphalt, bricks, ceramic, concrete, fabric, *etc*., and require the agent to only choose from these types. We provide the detailed output format and an example for in-context learning [29].

Material matching based on weighted VGG loss. Then we design a multi-scale weighted material matching method based on VGG loss [36], considering the order of possibility. Specifically, we first use the mask M_{ij} of material part j to obtain its bounding box, the cropped mask M'_{ij}, and the cropped image C_{ij} from the reference image I_i. Next, we randomly sample a certain number of rectangles E^s_k on the rendered exemplars at different scales (128×128, 256×256, 512×512) of all materials from the predicted possible types, according to the bounding box size. Then, we apply the cropped mask M'_{ij} to these sampled rectangles to obtain masked exemplar rectangles, thereby mitigating the effects of scale or spatial transformations on the texture patterns. Then we assign the initial material $\mathcal{G}^0_j$ through:

$$\mathcal{G}^0_j = \arg\min_{\mathcal{G}} \ \alpha^{O(\mathcal{G})-1} \frac{1}{K} \sum_{s=1}^{3} \sum_{k=1}^{K} \sum_{l} \sum_{x} (\mathcal{F}^l_{vgg}(C_{ij}) - \mathcal{F}^l_{vgg}(M'_{ij} * E^s_k))^2, \quad (1)$$

where $\mathcal{G}$ is a procedural material, $\alpha^{O(\mathcal{G})-1}$ is the weight related to the possibility order $O(\mathcal{G})$ of the type of material $\mathcal{G}$ ($O(\mathcal{G})$ starts from 1, α is set to $\frac{5}{3}$), K is the number of sampled rectangles at each scale, s refers to different scales ($s = 1$ is 128×128, $s = 2$ is 256×256, $s = 3$ is 512×512), $\mathcal{F}^l_{vgg}(\cdot)$ refers to the normalized VGG feature map extracted from layer l, $\sum_x$ denotes the sum over pixels x, and E^s_k is the sampled rectangle on the rendered exemplar of material $\mathcal{G}$ at scale s. To intuitively understand Eq. 1, we present an example in Fig. 5.

3.4. Material and Lighting Optimization

In this section, we focus on detailing our optimization process and the optimization function. Given the object mesh, our method adaptively create the floor and four walls in the scene of the differentiable renderer, and adaptively places initial area lights on each wall and above the object. The intensity of these lights is optimizable. We set the light

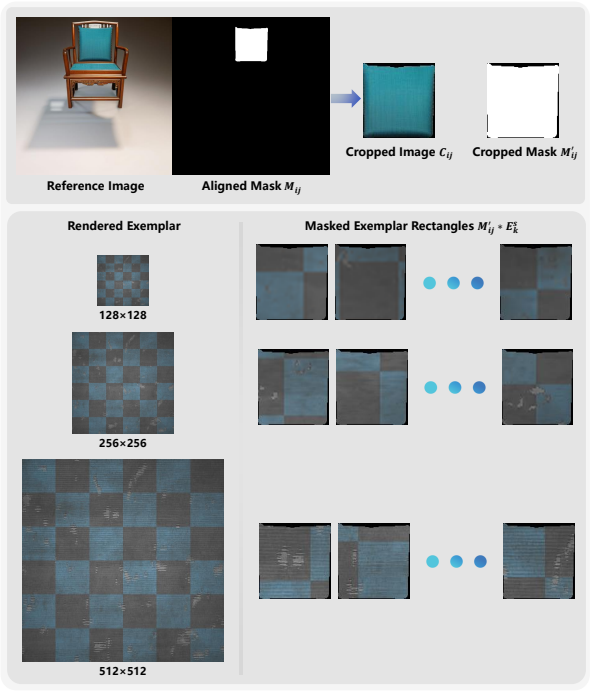


Figure 5. **Material Matching Example.** The upper part displays the cropped image C_{ij} and the cropped mask M'_{ij} obtained through the bounding box. The lower part depicts the masked exemplar rectangles $M'_{ij} * E^s_k$ randomly sampled on the rendered exemplars at different scales of a fabric material.

bounce count to 2 to enable global illumination. Our optimization process consists of two stages. In the first stage, we keep the lighting fixed and optimize only the parameters of the procedural materials. In the second stage, we jointly optimize the lighting and the procedural materials.

We denote the rendering function of the differentiable renderer PSDR-CUDA as $R(\cdot)$, and represent the parameters of all assigned procedural materials for the mesh as θ. To obtain the UV maps, we use "Smart UV projection" of Blender for each material part. Given the rendered images $R(\theta)_i$ and the reference images I_i, where i refers to the camera view index, we calculate the loss (Eq. 2) between them to backpropagate the gradient from the pixel level to the computational graph of procedural materials, thereby optimizing the parameters θ. We use Adam [21] as the optimizer, and the total optimization function is,

$$\mathcal{L}_{\text{total}} = \lambda_1 \mathcal{L}_{\text{resized}} + \lambda_2 \mathcal{L}_{\text{pixel}} + \lambda_3 \mathcal{L}_{\text{stat}} + \lambda_4 \mathcal{L}_{\text{gram}}, \quad (2)$$

where λ_{1-4} are the weights, $\mathcal{L}_{\text{resized}}$ is the $\mathcal{L}_1$ difference between the reference image and rendered image (downsized to $1/8$), $\mathcal{L}_{\text{pixel}}$ is the average $\mathcal{L}_1$ difference between the reference image and rendered image with each material part mask, $\mathcal{L}_{\text{stat}}$ is the mean absolute difference on the statistics (mean μ and variance σ^2) of the masked pixels of each material part between the two images, and $\mathcal{L}_{\text{gram}}$ is the average

Method	Overall Quality(↑)	Text Fidelity(↑)
Text2Tex [6]	3.44	3.95
TEXTure [33]	3.16	3.62
Ours	**4.64**	**4.53**

Table 1. User study results on Overall Quality and Text Fidelity with 30 respondents.

$\mathcal{L}_1$ difference on the Gram matrix texture descriptor [11] T_g for each part. The four sub-optimization objectives are defined as follows:

$$\mathcal{L}_{\text{resized}} = \sum_{i=1}^{n} \mathcal{L}_1(R(\theta)'_i, I'_i), \quad (3)$$

$$\mathcal{L}_{\text{pixel}} = \frac{1}{\sum_{i=1}^{n} m_i} \sum_{i=1}^{n} \sum_{j=1}^{m_i} \mathcal{L}_1(M_{ij} * R(\theta)_i, M_{ij} * I_i), \quad (4)$$

$$\mathcal{L}_{\text{stat}} = \frac{1}{\sum_{i=1}^{n} m_i} \sum_{i=1}^{n} \sum_{j=1}^{m_i} |\mu(M_{ij} * R(\theta)_i) - \mu(M_{ij} * I_i)| + |\sigma^2(M_{ij} * R(\theta)_i) - \sigma^2(M_{ij} * I_i)|, \quad (5)$$

$$\mathcal{L}_{\text{gram}} = \frac{1}{\sum_{i=1}^{n} m_i} \sum_{i=1}^{n} \sum_{j=1}^{m_i} \mathcal{L}_1(T_g(M_{ij} * R(\theta)_i), T_g(M_{ij} * I_i)), \quad (6)$$

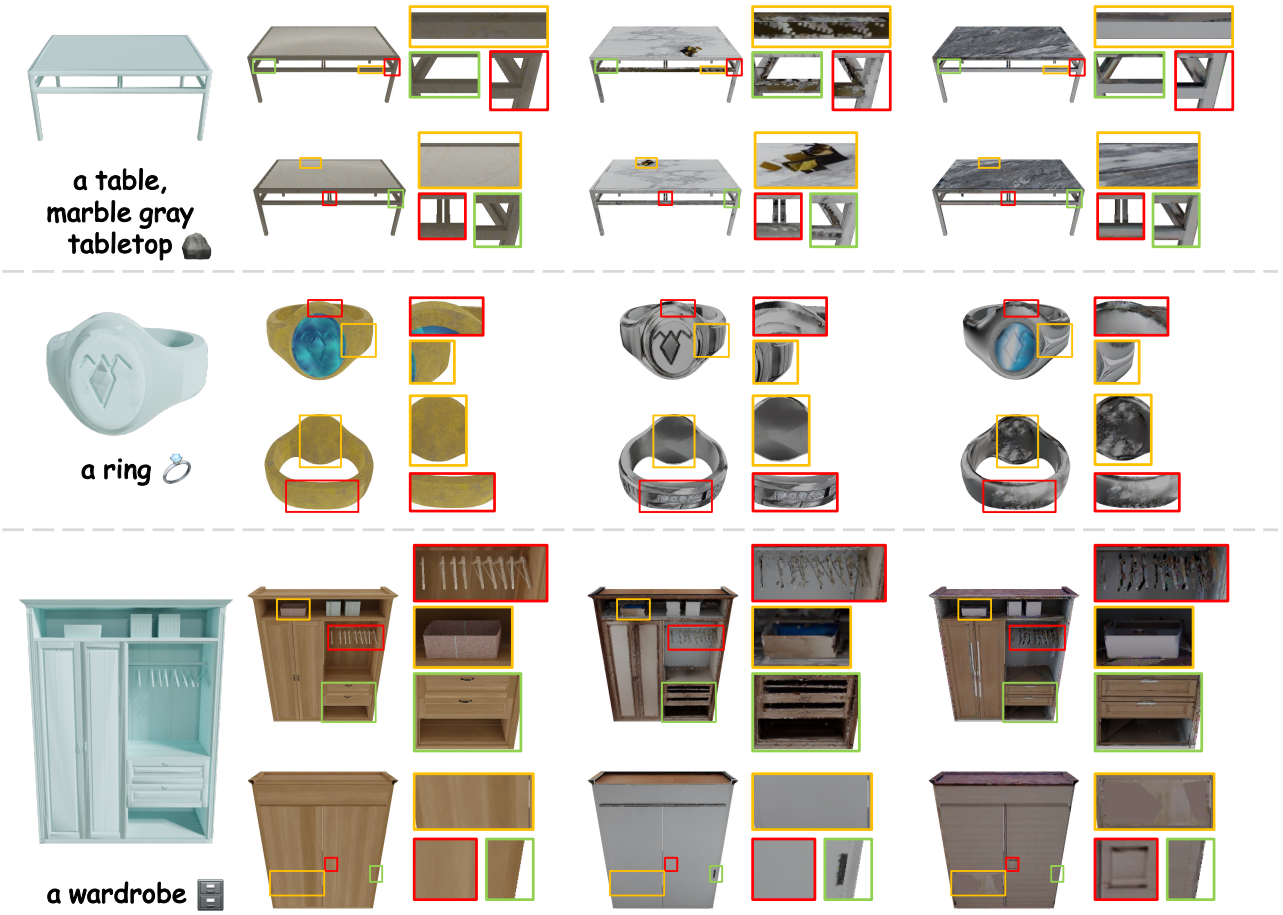
where i represents the camera view index, n represents the total number of camera views, j refers to the material part index, m_i refers to the total number of material parts under camera view i, $R(\theta)'_i$ and I'_i are the downsized images, and M_{ij} is the mask of the material part j.

4. Experiments

4.1. Experiment Setup

Datasets. We experimented our method on the 3D-FUTURE [10], 3DCoMPaT [26] and Objaverse [8] dataset. For detailed datasets description, please refer to our supplementary.

Baselines. We compare our method with two SOTA text-driven texture generation methods, TEXTure [33] and Text2Tex [6]. These two approaches utilize the Stable Diffusion [34] to generate 2D images, which are then projected onto the mesh. They generate complete textures through multiple iterations using an in-painting approach. However, they only produce diffuse textures with baked lighting. As a result, they cannot support relighting, so in the relighting experiment of Sec. 4.3, we only present the results of our method. **Please refer to our supplementary video for intuitive qualitative results.**



Shape & Prompt (a) Ours (b) Text2Tex (c) TEXTure

Figure 6. Qualitative comparisons with Text2Tex and TEXTure from front and back views.

4.2. Comparison to Baselines

In this section, we show the qualitative and quantitative comparison results with baselines. Similar to TEXTure [33], we conducted a user study with the same configuration to quantitatively evaluate the quality of text-driven texture generation. Specifically, we prepare 10 testing examples (shapes and text descriptions) for each method. We ask 30 participants to rate the overall quality and the text fidelity, on a scale from 1 to 5. The results are shown in Table. 1, indicating that our method achieves the best score among all the methods. In Fig. 6, we qualitatively compare the texturing results of different methods from both front and back viewpoints. As shown in Fig. 6, both TEXTure and Text2Tex have some artifacts, as both methods require projecting the generated 2D images onto the mesh based on the inpainting approach, resulting in missing textures in some obscured areas. Moreover, due to the lack of multi-view datasets training, the Stable Diffusion model utilized by these two methods can result in inconsistencies across different views, thus causing the Janus (multi-faced) problem. In contrast, ours can generate textures that maintain consistency across multiple views and produce photorealistic texturing results. The more texturing results are shown in Fig. 7.

4.3. Relighting Results

Our method can generate various maps as defined in the procedural material computation graphs, including base color, normal, roughness, *etc*., which support physically based rendering, making the texturing results truly usable for downstream tasks. Thus, in this section, we present the relighting results of our method in the 3D design software Blender.

Different ambient lighting. We first use different ambient lighting scenarios (indoor warm light, daytime outdoor, nighttime outdoor) to relight our textured meshes in Fig. 8.

Different point light positions. We then keep the ambient lighting fixed, add a point light, and change its position (to the northeast and southeast, respectively) to relight the object, showcasing shadow and highlight details in Fig. 9.

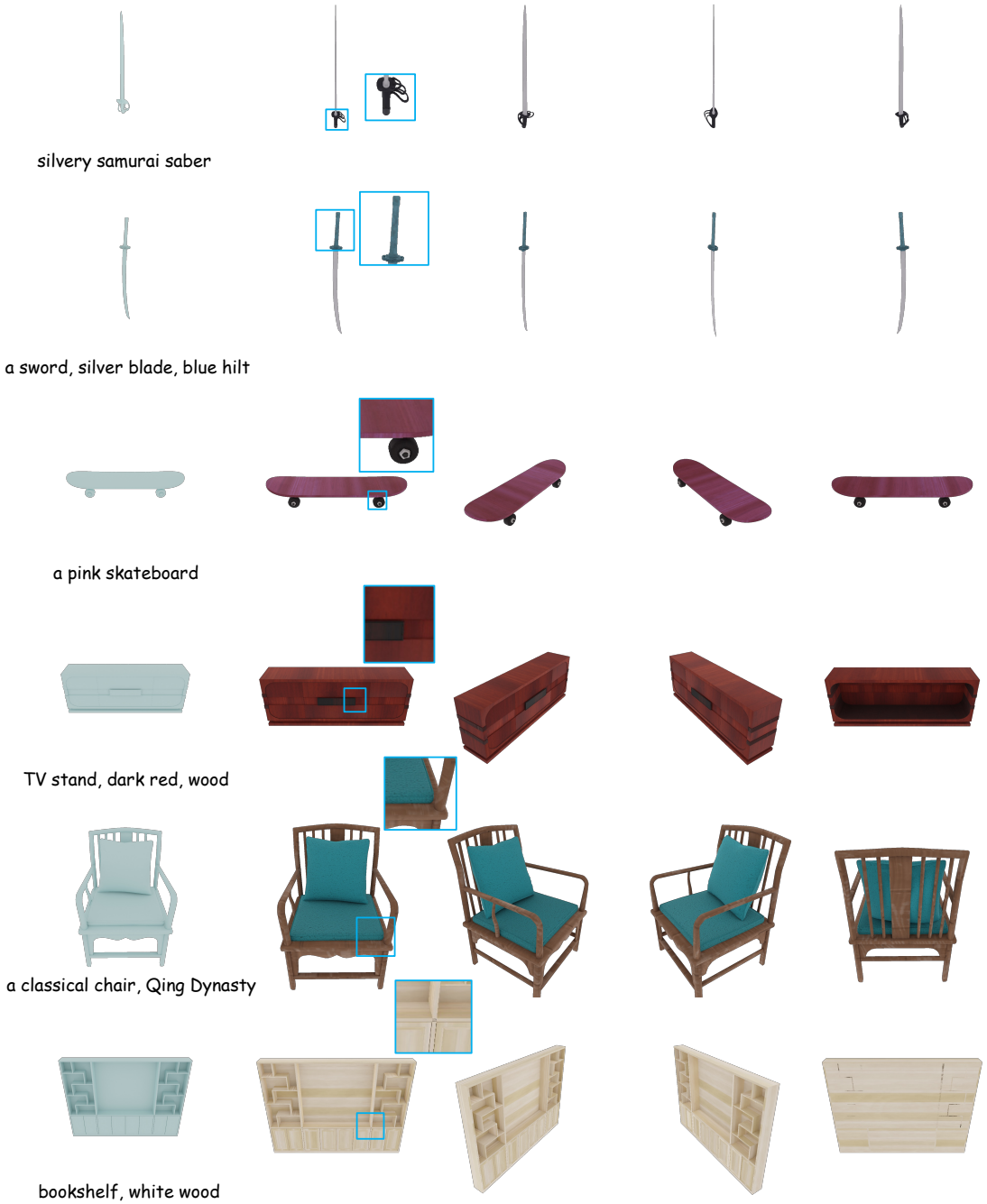


Figure 7. **More Texturing Results.** For each sample, we present four different viewpoints. We recommend zooming in to view the texturing results.

4.4. Ablation Studies

Material agent for material classification. Through material agent, we can achieve object-aware material classification with the order of possibility. Moreover, the number of predicted class labels is unconstrained. Here, we compare against a simple baseline. This baseline calculates the VGG loss for all materials in the library using Eq. 1 (with $\alpha = 1$), then selects the 10 materials with the lowest loss and uses a voting way to determine the most frequently occurring material type as the predicted type. We present the comparison results in Fig. 10, where the upper part shows part-level understanding by agent and the lower part shows the comparison of material classification for different material parts.

Final aligned masks. For a material part, accurately ob-

Figure 8. **Different Ambient Lighting.** We present the relighting results for three meshes in Blender under three different ambient lighting scenarios, from three viewpoints.

Figure 9. **Different Point Light Positions.** The top row corresponds to point light positioned northeast, and the bottom row corresponds to southeast from three viewpoints.

Figure 10. **Material Classification.**

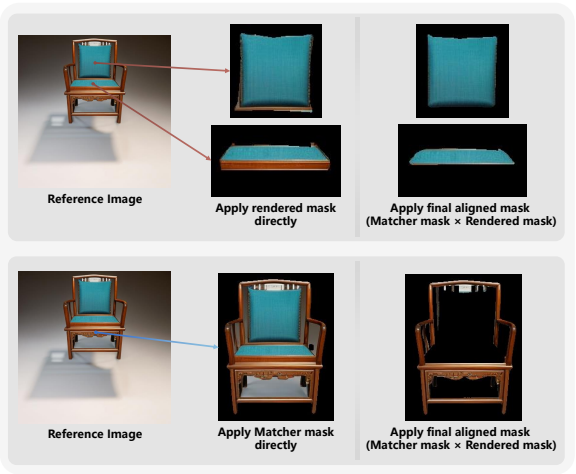


Figure 11. **Final Aligned Mask.** We show the three material parts of a chair. We can't directly apply the rendered mask or the Matcher mask to the reference image.

taining its mask on the reference image is crucial for guiding material agent and calculating the optimization function (Eq. 2). As shown in Fig. 11, since the reference image is not well aligned with the mesh geometry, the rendered mask cannot be directly applied to the reference image. Thus, we propose to obtain the final aligned mask by intersecting Matcher [27] mask and rendered mask.

5. Conclusion

We present a new method to generate high-fidelity materials for 3D shapes given textual prompts. Unlike previous texture generation methods that generate RGB textures containing baked lighting, our method can generate diverse texture maps that enable physical-based rendering and relighting. Additionally, our approach is based on procedural ma-

terial modeling, which allows us to inherit many advantages of procedural materials. Currently, our method requires presegemented material parts which can be improved by integrating with the shape analysis methods [13, 22, 39] in the future.

Limitation. Our approach also has some limitations. Since it optimizes the parameters of individual nodes within the procedural material graph, it is fundamentally constrained by the materials available in the procedural material library. As a result, it is currently incapable of generating complex textures, such as letters or characters. Nevertheless, if such textures already exist within the library, our method can still support their generation. A potential future direction could involve a two-stage generation process, where complex textures like characters are synthesized and overlaid on top of procedural materials, further expanding the expressiveness of our approach.

Acknowledgement

This research was partially supported by the the National Natural Science Foundation of China (No. 62441222), Information Technology Center and State Key Lab of CAD&CG, Zhejiang University. We also express our gratitude to all the anonymous reviewers for their professional and constructive comments.

References

[1] Adobe. Substance designer, 2024. https://www.substance3d.com/. 3

[2] M. Armandpour, H. Zheng, A. Sadeghian, A. Sadeghian, and M. Zhou. Re-imagine the negative prompt algorithm: Transform 2d diffusion into 3d, alleviate janus problem and beyond. *arXiv preprint arXiv:2304.04968*, 2023. 3

[3] S. Bi, N. K. Kalantari, and R. Ramamoorthi. Patch-based optimization for image-based texture mapping. *ACM Trans. Graph.*, 36(4):106–1, 2017. 3

[4] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020. 5

[5] B. Burley and W. D. A. Studios. Physically-based shading at disney. In *Acm Siggraph*, volume 2012, pages 1–7. vol. 2012, 2012. 3

[6] D. Z. Chen, Y. Siddiqui, H.-Y. Lee, S. Tulyakov, and M. Nießner. Text2tex: Text-driven texture synthesis via diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 18558–18568, 2023. 1, 3, 6

[7] M. Deitke, R. Liu, M. Wallingford, H. Ngo, O. Michel, A. Kusupati, A. Fan, C. Laforte, V. Voleti, S. Y. Gadre, E. VanderBilt, A. Kembhavi, C. Vondrick, G. Gkioxari, K. Ehsani, L. Schmidt, and A. Farhadi. Objaverse-XL: A universe of 10m+ 3d objects. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023. 3

[8] M. Deitke, D. Schwenk, J. Salvador, L. Weihs, O. Michel, E. VanderBilt, L. Schmidt, K. Ehsani, A. Kembhavi, and A. Farhadi. Objaverse: A universe of annotated 3d objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13142–13153, 2023. 3, 6

[9] R. Fridman, A. Abecasis, Y. Kasten, and T. Dekel. Scenescape: Text-driven consistent scene generation. *Advances in Neural Information Processing Systems*, 36, 2024. 3

[10] H. Fu, R. Jia, L. Gao, M. Gong, B. Zhao, S. Maybank, and D. Tao. 3d-future: 3d furniture shape with texture. *International Journal of Computer Vision*, pages 1–25, 2021. 3, 6

[11] L. A. Gatys, A. S. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2414–2423, 2016. 6

[12] P. Guerrero, M. Hasan, K. Sunkavalli, R. Mech, T. Boubekeur, and N. Mitra. Matformer: A generative model for procedural materials. *ACM Trans. Graph.*, 41(4), 2022. 3

[13] R. Hanocka, A. Hertz, N. Fish, R. Giryes, S. Fleishman, and D. Cohen-Or. Meshcnn: a network with an edge. *ACM Transactions on Graphics (ToG)*, 38(4):1–12, 2019. 3, 10

[14] L. Höllein, A. Cao, A. Owens, J. Johnson, and M. Nießner. Text2room: Extracting textured 3d meshes from 2d text-to-image models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7909–7920, 2023. 3

[15] R. Hu, X. Su, X. Chen, O. Van Kaick, and H. Huang. Photo-to-shape material transfer for diverse structures. *ACM Trans. Graph.*, 41(4):131–1, 2022. 2, 4

[16] Y. Hu, J. Dorsey, and H. Rushmeier. A novel framework for inverse procedural texture modeling. *ACM Transactions on Graphics (ToG)*, 38(6):1–14, 2019. 3

[17] Y. Hu, P. Guerrero, M. Hasan, H. Rushmeier, and V. Deschaintre. Node graph optimization using differentiable proxies. In *ACM SIGGRAPH 2022 conference proceedings*, pages 1–9, 2022. 3

[18] Y. Hu, P. Guerrero, M. Hasan, H. Rushmeier, and V. Deschaintre. Generating procedural materials from text or image prompts. In *ACM SIGGRAPH 2023 Conference Proceedings*, pages 1–11, 2023. 3

[19] W. Jakob, S. Speierer, N. Roussel, M. Nimier-David, D. Vicini, T. Zeltner, B. Nicolet, M. Crespo, V. Leroy, and Z. Zhang. Mitsuba 3 renderer, 2022. https://mitsuba-renderer.org. 3

[20] N. M. Khalid, T. Xie, E. Belilovsky, and P. Tiberiu. Clip-mesh: Generating textured meshes from text using pretrained image-text models. *SIGGRAPH Asia 2022 Conference Papers*, December 2022. 3

[21] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 6

[22] A. Lahav and A. Tal. Meshwalker: Deep mesh understanding by random walks. *ACM Transactions on Graphics (TOG)*, 39(6):1–13, 2020. 3, 10

[23] S. Laine, J. Hellsten, T. Karras, Y. Seol, J. Lehtinen, and T. Aila. Modular primitives for high-performance differentiable rendering. *ACM Transactions on Graphics*, 39(6), 2020. 3

[24] B. Li, L. Shi, and W. Matusik. End-to-end procedural material capture with proxy-free mixed-integer optimization. *ACM Transactions on Graphics (TOG)*, 42(4):1–15, 2023. 1, 3

[25] T.-M. Li, M. Aittala, F. Durand, and J. Lehtinen. Differentiable monte carlo ray tracing through edge sampling. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*, 37(6):222:1–222:11, 2018. 3

[26] Y. Li, U. Upadhyay, H. Slim, A. Abdelreheem, A. Prajapati, S. Pothigara, P. Wonka, and M. Elhoseiny. 3dcompat: Composition of materials on parts of 3d things. In *17th European Conference on Computer Vision (ECCV)*, 2022. 3, 6

[27] Y. Liu, M. Zhu, H. Li, H. Chen, X. Wang, and C. Shen. Matcher: Segment anything with one shot using all-purpose feature matching. In *The Twelfth International Conference on Learning Representations*, 2024. 2, 5, 9

[28] O. Michel, R. Bar-On, R. Liu, S. Benaim, and R. Hanocka. Text2mesh: Text-driven neural stylization for meshes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13492–13502, June 2022. 3

[29] S. Min, X. Lyu, A. Holtzman, M. Artetxe, M. Lewis, H. Hajishirzi, and L. Zettlemoyer. Rethinking the role of demonstrations: What makes in-context learning work? *arXiv preprint arXiv:2202.12837*, 2022. 5

[30] M. Nimier-David, D. Vicini, T. Zeltner, and W. Jakob. Mitsuba 2: A retargetable forward and inverse renderer. *ACM Transactions on Graphics (TOG)*, 38(6):1–17, 2019. 3

[31] K. Park, K. Rematas, A. Farhadi, and S. M. Seitz. Photoshape: Photorealistic materials for large-scale shape collections. *ACM Trans. Graph.*, 37(6), Nov. 2018. 2, 4

[32] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 1(2):3, 2022. 3

[33] E. Richardson, G. Metzer, Y. Alaluf, R. Giryes, and D. Cohen-Or. Texture: Text-guided texturing of 3d shapes. In *ACM SIGGRAPH 2023 Conference Proceedings*, pages 1–11, 2023. 1, 3, 6, 7

[34] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022. 1, 2, 3, 4, 6

[35] L. Shi, B. Li, M. Hašan, K. Sunkavalli, T. Boubekeur, R. Mech, and W. Matusik. Match: Differentiable material graphs for procedural material capture. *ACM Transactions on Graphics (TOG)*, 39(6):1–15, 2020. 1, 3

[36] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015. 5

[37] H. Slim, X. Li, M. A. Yuchen Li, M. Ayman, U. U. A. Abdelreheem, S. P. Arpit Prajapati, P. Wonka, and M. Elhoseiny. 3DCoMPaT++: An improved large-scale 3d vision dataset for compositional recognition. In *arXiv*, 2023. 3

[38] M. Waechter, N. Moehrle, and M. Goesele. Let there be color! large-scale texturing of 3d reconstructions. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 836–850. Springer, 2014. 3

[39] Y. Yin, Y. Liu, Y. Xiao, D. Cohen-Or, J. Huang, and B. Chen. Sai3d: Segment any instance in 3d scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 3, 10

[40] X. Yu, P. Dai, W. Li, L. Ma, Z. Liu, and X. Qi. Texture generation on 3d meshes with point-uv diffusion. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4206–4216, 2023. 3

[41] C. Zhang, B. Miller, K. Yan, I. Gkioulekas, and S. Zhao. Path-space differentiable rendering. *ACM Trans. Graph.*, 39(4):143:1–143:19, 2020. 3, 4

[42] J. Zhang, X. Li, Z. Wan, C. Wang, and J. Liao. Text2nerf: Text-driven 3d scene generation with neural radiance fields. *IEEE Transactions on Visualization and Computer Graphics*, 2024. 3

[43] L. Zhang, A. Rao, and M. Agrawala. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3836–3847, 2023. 1, 2, 3, 4