Computational Visual Media
https://doi.org/10.26599/CVM.2025.9450489 Vol. 11, No. 4, August 2025, 745–761

Research Article

TexPro: Text-guided PBR texturing with procedural material modeling

Ziqiang Dang[1,*], Wenqi Dong[1,*], Zesong Yang[1], Bangbang Yang[2], Liang Li[3], Yuewen Ma[2], and Zhaopeng Cui[1] (✉)

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 physically-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 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 materials; multi-modal large language model (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 varied lighting conditions, we need to generate object material that includes a set of texture maps for diffuse and specular reflections, glossiness, surface roughness, etc. Thus, traditional mesh texturing typically involves substantial manual effort. Recent advances in 2D image generation have spurred the development of automated mesh texturing methods [1, 2], but they normally generate RGB textures with baked lighting, which limits their applicability 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 physically-based rendering, relighting, and additional benefits inherent to procedural materials, such as resolution independence (i.e., the ability to 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 useful

* Ziqiang Dang and Wenqi Dong contributed equally to this work.

1 State Key Lab of CAD&CG, Zhejiang University, Hangzhou 310058, China. E-mail: Z. Dang, ZiqDang@zju.edu.cn; W. Dong, dongwenqi@zju.edu.cn; Z. Yang, zesongyang0@zju.edu.cn; Z. Cui, zhpcui@zju.edu.cn (✉).

2 Pico, ByteDance Inc., Shanghai 200235, China. E-mail: B. Yang, ybbbbt@gmail.com; Y. Ma, mayuewen@bytedance.com.

3 School of Media Engineering, Communication University of Zhejiang, Hangzhou 310018, China. E-mail: liliang@cuz.edu.cn.

Manuscript received: 2025-02-13; accepted: 2025-04-18

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

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, our approach first generates multi-view reference images given the input textual prompt by employing the latest text-to-image models [3, 4], and then obtains diverse texture maps through rendering-based optimization with recent differentiable procedural materials [5, 6]. 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.

First, despite using additional rendered depth or normal maps as conditions [4], the generated images from the text-to-image model (e.g., stable diffusion [3]) often lack accurate alignment 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 [7], to extract the masks of corresponding objects in generated images, which will be further smoothed and integrated with the rendered masks to obtain the final aligned masks. Additionally, we design an adaptive camera sampling method that ensures each material part exceeds a specified pixel count threshold and is assigned a unique material 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 challenge initial material classification and matching for classical methods [8, 9]. 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 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 optimization. To make the optimization tractable, we adopt a plain 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 enables physically-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, Matcher [7], 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 the existing state of the art, and maintains the capabilities of relighting.

2 Related work

2.1 Texture generation

Texture maps are essential for 3D geometric models. They involve mapping pre-defined 2D planar images onto 3D models, endowing 3D models with vital color information. Traditional texture creation involves cumbersome manual drawing, assembling repetitive patterns, or stitching multi-view images [10, 11]. With the development of increasingly high-quality 3D datasets and advances in text-to-image generative techniques [3, 4, 12], learning-based approaches have been proposed to generate high-quality textures [1, 2, 13–15]. Text2Tex [1] 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 [16–18]. Similarly, TEXTure [2] also applies an iterative scheme to paint 3D models, while presenting a trimap representation and a novel elaborate diffusion sampling process to tackle inconsistency 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 the Janus (multi-faced) problem [19] and lack realism.

2.2 Procedural material modeling

Procedural material has been the standard method of material modeling in industry. Materials are represented as node graphs denoted with simple image processing operations which are combined to produce real-world spatially varying BRDF [20] material maps. Given user-specified images or text prompts, many approaches [5, 6, 21–23] have been proposed to recover the parameters of a predefined procedural material graph in agreement with the input. Moreover, Hu et al. [24] moved beyond parameter regression, utilizing the diffusion model to generate graph structures from text or image conditions. It is worth noting that MATch [5] proposes a differentiable procedural material method that translates procedural graphs into differentiable node graphs, enabling single-shot high-quality procedural material capture.

2.3 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 several specifically differentiable renderers have been developed and widely used in the community, like Redner [25], Mitsuba 2/3 [26, 27], Nvdiffrast [28], and PSDR-CUDA [29], the aim being to address challenges arising from the non-differentiable terms in the rendering integral. We make use of differentiable rendering to backpropagate the gradient from pixel space to the parameters of procedural materials, thereby optimizing the materials via gradient descent.

3 Method

3.1 Aims and approach

Most 3D meshes in online resource repositories like BlenderKit or Sketchfab, as well as meshes from existing 3D datasets [30–34], have been 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 of a complete mesh can be achieved through shape analysis methods [35–37]. Thus, given textual prompts, we aim to generate appropriate materials

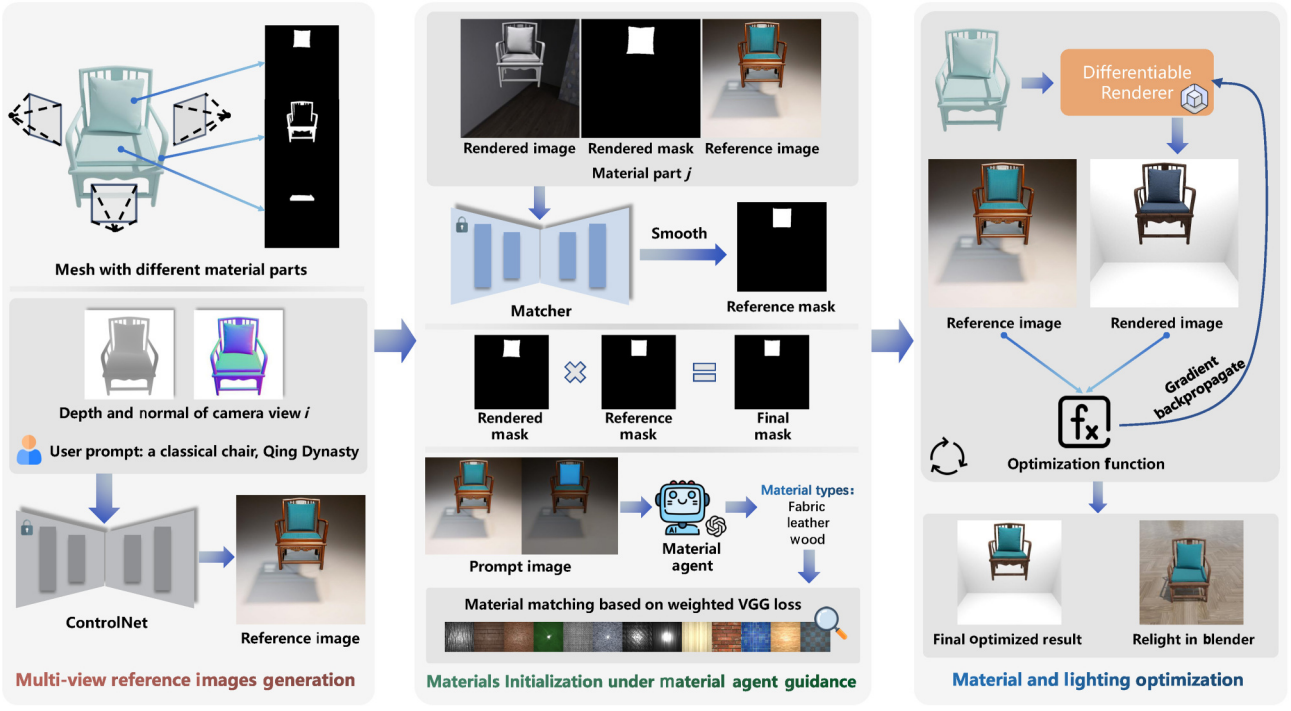


Fig. 2 Overview of TexPro.

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 the rest of this section, we first introduce procedural materials and a differentiable version *DiffMat* in Section 3.2. We then provide detailed descriptions of these three steps in Section 3.3, Section 3.4, and Section 3.5.

3.2 Preliminaries

Procedural materials. Procedural material (e.g., substance materials [38]) 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 certain 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 resolution-independence, re-editability, the ability to be dynamically changed during runtime, and greater efficiency in terms of storage and memory usage.

DiffMat. Shi et al. [5] developed the *DiffMat* library, skillfully utilizing convolutional neural networks to implement the functionality of most filter nodes, thereby making procedural materials differentiable. Recently, *DiffMatV2* [6] 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 of the node graphs.

3.3 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 camera 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 [3] is more realistic. The number of selected views is unconstrained. For more

details of the camera selection strategy, please refer to the Appendix.

Reference images generation. We use the differentiable renderer PSDR-CUDA [29] to render the normal maps and depth maps of the mesh under the selected camera views. Subsequently, we use stable diffusion and controlnet [4] to generate the reference image of the front view under the condition of the corresponding normal map. For other camera views, we also use the reference image of the 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 poor, the rendered depth maps are added as an extra condition.

3.4 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 classical material classification methods [8, 9] suffer from the baked lighting in generated images and confuse 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: for example, pillows should be assigned fabric or leather materials, not 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.

Mask 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 rendered images cannot be directly applied to the reference images. Therefore, we use Matcher [7] to get the mask M_I for 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 using white material under specific ambient lighting, the mask M_R of a material part on the rendered image, and

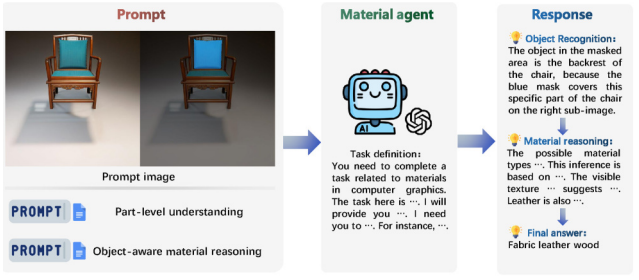


Fig. 3 Material agent. An example of a chair backrest is shown here. Providing both the image prompt and text prompt to our defined material agent allows it to predict all possible material categories in order of likelihood.

Fig. 4 Agent response examples. Left: prompt images provided to the material agent. Right: responses returned by the agent.

the reference image. 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 material part j under camera view i is obtained by $M_{ij} = M_R * M_I$. M_{ij} is 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 is derived from our carefully designed prompt template to guide the material agent. Note that the same prompt text is shared across all material parts, instead of using a separate one for each. 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 [39]. Subsequently, we employ various prompt techniques to construct the prompt for part-level understanding and object-aware material reasoning. Please refer to the Appendix for the complete prompt template.

Material agent part-level understanding. Similar to the idea of chain of thought [40], we first guide the agent to identify what object is in the marked area, instead of directly inferring the materials. This 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. Next, we ask the agent to combine pixel information and object information to reason about all possible material types and to predict results in the order of probability. We have divided the materials in the *DiffMat* library into 19 categories including asphalt, bricks, ceramics, 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 [39].

Material matching based on weighted VGG loss. Then, we design a multi-scale weighted material matching method based on VGG loss [41], considering the order of likelihood. 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_k^s on the rendered exemplars at different scales (128×128, 256×256, 512×512) for all materials of 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 $\mathcal{G}_j^0$ through:

$$\mathcal{G}_j^0 = \underset{\mathcal{G}}{\operatorname{argmin}}\, \alpha^{O(\mathcal{G})-1} \frac{1}{K} \sum_{s=1}^{3} \sum_{k=1}^{K} \sum_{l} \sum_{x} (\mathcal{F}_{\text{vgg}}^l(C_{ij}) - \mathcal{F}_{\text{vgg}}^l(M'_{ij} * E_k^s))^2 \quad (1)$$

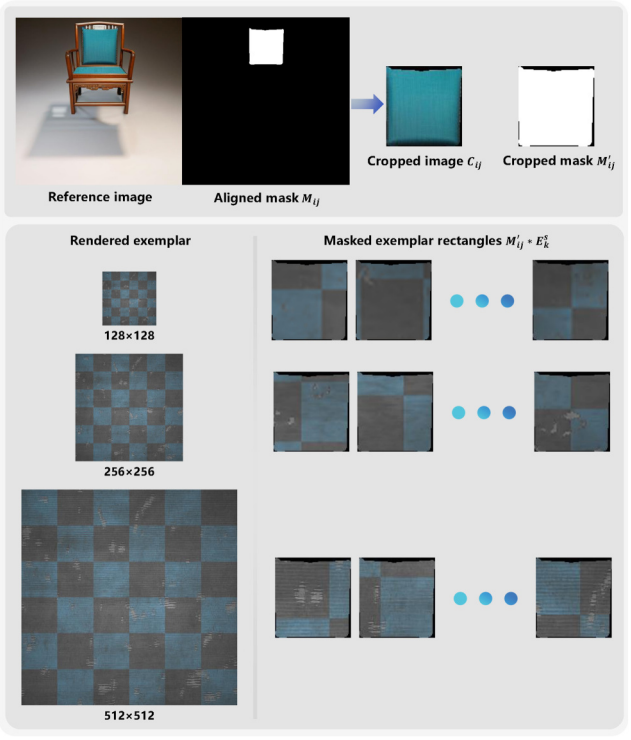


Fig. 5 Material matching example. Above: the cropped image C_{ij} and the cropped mask M'_{ij} obtained through the bounding box. Below: the masked exemplar rectangles $M'_{ij} * E_k^s$ randomly sampled on the rendered exemplars at different scales of a fabric material.

where $\mathcal{G}$ is a procedural material, $\alpha^{O(\mathcal{G})-1}$ is the weight related to the likelihood 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), $\mathcal{F}_{\text{vgg}}^l(\cdot)$ refers to the normalized VGG feature map extracted from layer l, $\sum_x$ denotes the sum over pixels x, and E_k^s is the sampled rectangle on the rendered exemplar of material $\mathcal{G}$ at scale s. To aid intuitive understanding of Eq. (1) we present an example in Fig. 5.

3.5 Material and lighting optimization

In this section, we detail our optimization process and the optimization function. 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. The intensity of these lights is included in the optimization. We set the light 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 U–V maps, we use "Smart UV projection" in Blender for each material part. Given the rendered images $R(\theta)_i$ and the reference images I_i, where i indexes the camera view, we calculate the loss in 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 [42] as the optimizer. The total optimization objective 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 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 of 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 $\mathcal{L}_1$ difference of the Gram matrix texture descriptor [43] T_g for each part. These four sub-objectives are defined as Eqs. (3)–(6):

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

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

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

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

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 downsized images, and M_{ij} is the mask for material part j.

4 Experiments

4.1 Experimental setup

Datasets. We performed experiments using the 3D-FUTURE [30], 3DCoMPaT [31], and Objaverse [33] datasets. For detailed datasets description, please refer to the Appendix.

Baselines. We compared our method with two state-of-the-art text-driven texture generation methods, TEXTure [2], and Text2Tex [1]. These two approaches utilize stable diffusion [3] 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 Section 4.3, we only present the results of our method. Please refer to our video in the Electronic Supplementary Material (ESM) for intuitive qualitative results.

4.2 Comparison to baselines

This section provides qualitative and quantitative comparisons with the baselines. Following TEXTure [2], we conducted a user study with the same configuration to quantitatively evaluate the quality of text-driven texture generation. Specifically, we prepared 10 test examples (shapes and text descriptions) for each method. We asked 30 participants to rate the overall quality and fidelity to the text, on a scale from 1 to 5. The results are shown in Table 1, indicating that our method achieves the best scores among all methods. In Fig. 6, we qualitatively compare the texturing results of different methods from both front and back viewpoints. Both TEXTure and Text2Tex produce 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

Table 1 User study results on overall quality and text fidelity with 30 respondents

Method	Overall quality (↑)	Text fidelity (↑)
Text2Tex [1]	3.44	3.95
TEXTure [2]	3.16	3.62
Ours	**4.64**	**4.53**

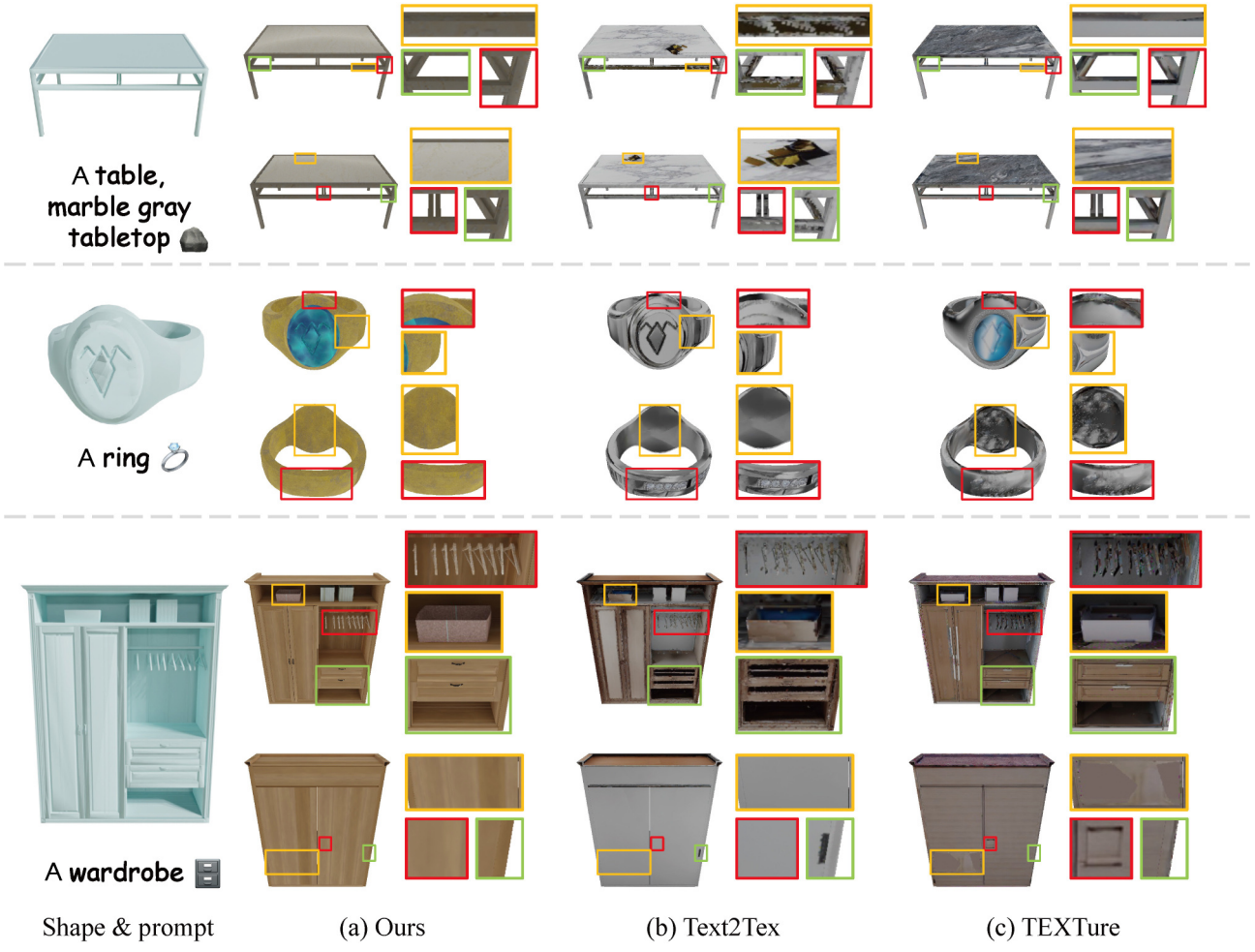


Fig. 6 Qualitative comparisons with Text2Tex and TEXTure from front and back views.

model utilized by these two methods can result in inconsistencies across different views, thus causing the Janus (multi-faced) problem. In contrast, our method can generate textures that maintain consistency across multiple views and produce photorealistic texturing results. 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.

4.4 Ablation studies

Material agent for material classification. Through the material agent, we can achieve object-aware material classification with the order of likelihood. 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 voting 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 the agent and the lower part compares the material

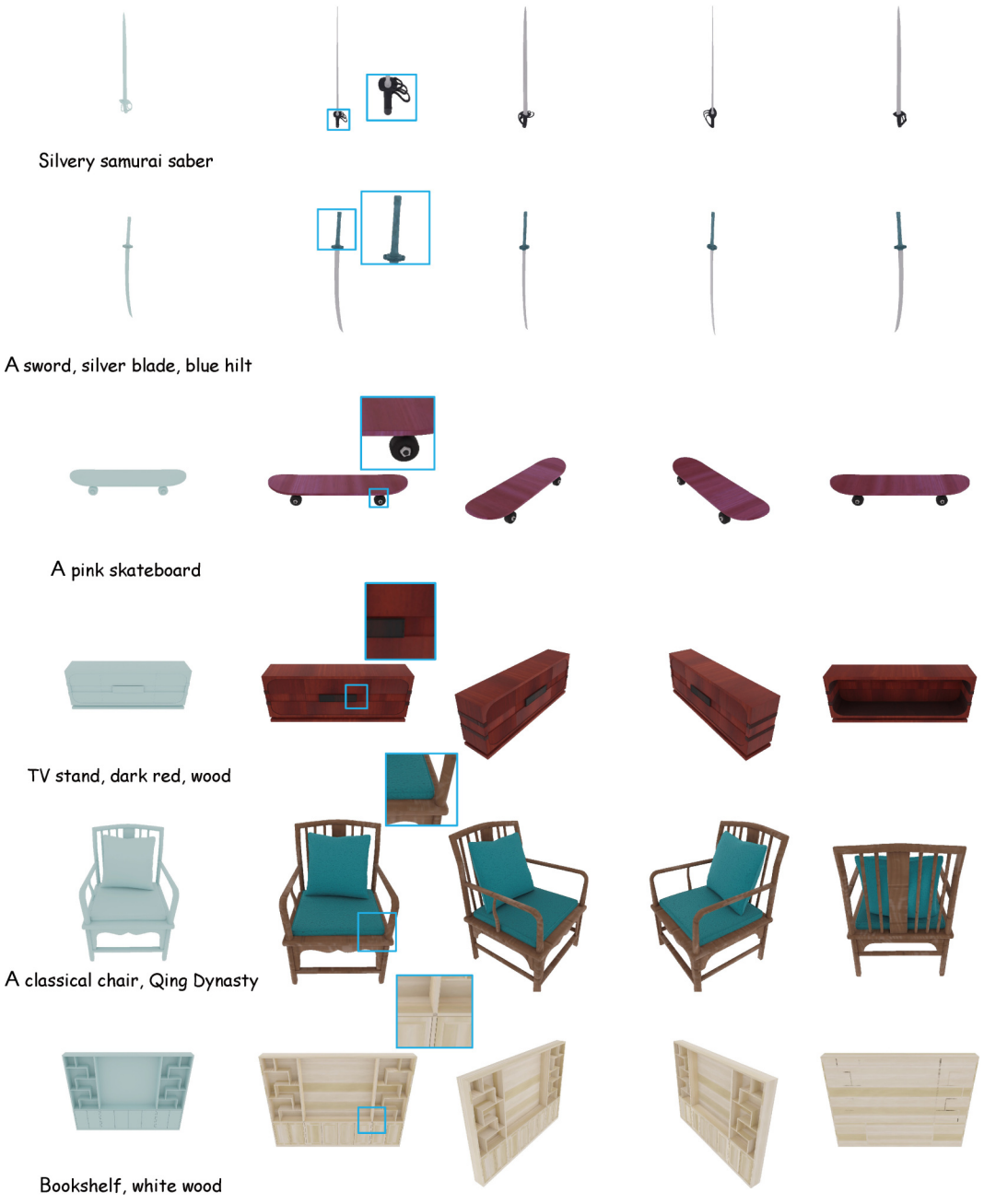


Fig. 7 More texturing results. For each sample, we present four different viewpoints.

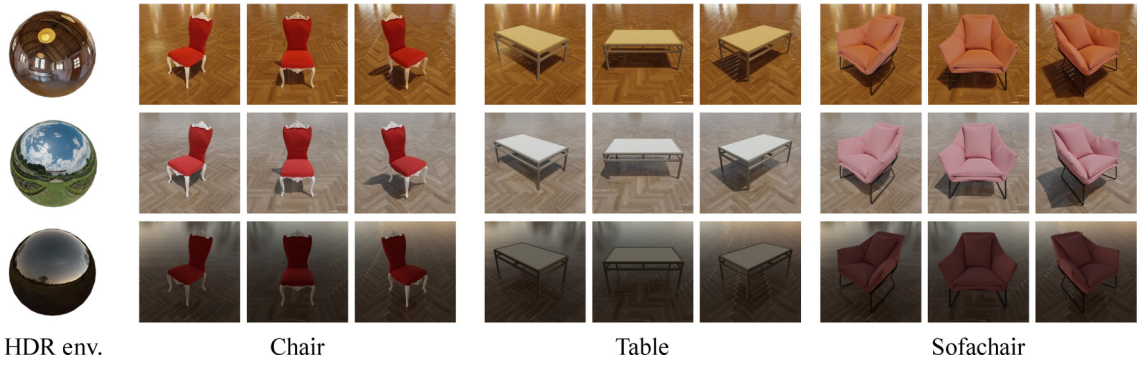


Fig. 8 Different ambient lighting. We present the relighting results for three meshes in Blender under three different ambient lighting scenarios, from three viewpoints.

Fig. 9 Different point light positions. The top row corresponds to a point light positioned northeast, and the bottom row corresponds to southeast, each shown from three viewpoints.

Fig. 10 Material classification.

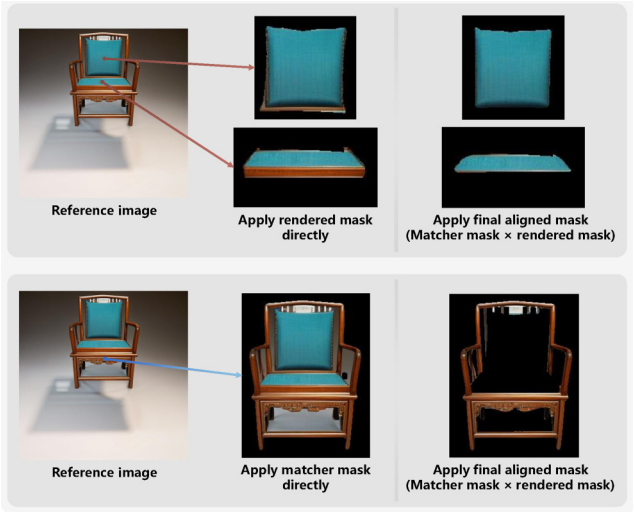


Fig. 11 Final aligned mask. We show the three material parts of a chair. We cannot directly apply the rendered mask or the Matcher mask to the reference image.

classification for different material parts.

Final aligned masks. For a material part, accurately obtaining its mask on the reference image is crucial for guiding the 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 obtain the final aligned mask by intersecting the Matcher [7] mask and rendered mask.

5 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.

6 Conclusions

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 physically-based rendering and relighting.

Additionally, our approach is based on procedural material modeling, which allows us to inherit many advantages of procedural materials. Currently, our method requires pre-segmented material parts which could be improved by integrating it with shape analysis methods [35–37] in the future.

Appendix

This appendix provides further details and results omitted from the main paper. Specifically, we provide additional implementation details in Section A.1 and experimental details in Section B.1. In Section C.1, we present more comparisons involving our method. Please refer to our video in the ESM for more vivid method explanations and qualitative results.

A.1 Implementation details

A.1.1 Camera views selection

Given two elevation angles and the distance to the object, we use spherical sampling to uniformly sample n camera coordinates for each elevation. We then construct the extrinsic camera parameters using the "Look At" approach. We calculate the number of pixels for each material part under each viewpoint through the rendered mask M_R. We select a sufficient number of cameras from these $2 \times n$ cameras to ensure that the pixel count for each part exceeds a threshold (set to 500). To deal with the slight differences between reference images from different views, we select only one reference image for each material part.

Specifically, the front view of the larger elevation will first be selected, because, using this view, the reference image generated by Stable Diffusion [3] will be more realistic and most material parts will exceed the pixel count threshold.

For material parts with a pixel count below the threshold under this front view, including those unseen parts, we then adaptively select k cameras based on their pixel counts. To balance the goals of minimizing the number of selected cameras and maximizing the pixel count for each material part, we adopt three steps to select the cameras for the remaining material parts. The first step is to select the "unavoidable" cameras. For each material part, we compare the maximum pixel count under the $2n-1$ viewpoints with the threshold. If the maximum value is less than the threshold, we select the camera corresponding to the maximum pixel count to ensure sufficient visibility for this material part. Through this step, we select n_1 viewpoints. Next, we select cameras for the remaining material parts. The second step is to select the minimal subset of cameras that satisfy the threshold condition. We mark the cameras for each material part where the pixel count exceeds the threshold, and then select the minimal subset of marked cameras to ensure that each material part meets the threshold condition. Through this step, we select n_2 viewpoints. The third step involves finding the intersection of the initially selected front view, the n_1 viewpoints selected in the first step, and the n_2 viewpoints selected in the second step, and using the resulting intersection as the final set of chosen cameras. Through this step, we can obtain the final k cameras. Note that during optimization, each material part will only use the viewpoint selected for it.

A.1.2 Smoothing matcher masks

Matcher [7] can segment anything by using an in-context example without training. As shown in Fig. 2 of the main paper, we use the rendering of the object textured by white material under specific ambient lighting and the mask M_R of a material part on this rendered image as the in-context example. Then, the Matcher will output the corresponding mask M_I for the reference image.

We experimented with different ambient lighting to illuminate the objects for the in-context example. In the end, we choose the indoor uniform white light for the ambient lighting, because the segmentation of Matcher will be more accurate.

The masks output by Matcher typically contain noise. Therefore, we subsequently smooth the masks through three image processing steps. We first apply median filtering to attempt to connect regions and remove some outliers. In the second step, we use the morphological erosion operation on the binary images (masks) to remove some outlier areas. Finally, for each Matcher mask M_I, we count the number of pixels $\{P_1, \ldots, P_i, \ldots\}$ in each connected region $\{R_1, \ldots, R_i, \ldots\}$, where P_i is the number of pixels of region R_i. We calculate the smallest number of pixels P_r for all connected regions in the rendered mask M_R. We then set the connected regions with a pixel count smaller than βP_r ($\beta = 0.5$) to False (i.e., Masked) for every region in $\{R_1, \ldots, R_i, \ldots\}$.

A.1.3 Material agent prompt design

Prompt image. We choose the reference image

generated by Stable Diffusion [3] to construct the prompt image rather than the rendered image of the object textured by white material under specific ambient lighting, because the former carries more information, leading to more accurate part-level understanding by MLLMs. To support part-level understanding, we tested four object labeling schemes: (i) obtaining contours of the material part through mask M_{ij}, then outlining the contours with red lines; (ii) building upon (i) by horizontally concatenating with the reference image; (iii) blending mask M_{ij} with the reference image, overlaying the mask onto the reference image; and (iv) building upon (iii) by horizontally concatenating with the reference image. Overall, we found that the fourth solution (i.e., the prompt image in Fig. 3) leads to more accurate object recognition results.

Text prompt template. Our prompt template is shown in Fig. A1. Given the name of the object (e.g., chair), a specific prompt text can be derived from the carefully designed template by replacing *[object_name]* in the template with the name of the object. Note that all material parts share the same prompt text, rather than deriving a prompt for every material part.

B.1 Experimental details

B.1.1 datasets

We evaluated our method on three datasets including 3D-FUTURE [30], 3DCoMPaT [31], and Objaverse [33] dataset.

3D-FUTURE [30] is a richly-annotated large-scale dataset of 3D furniture shapes in household scenarios. It contains 9992 unique industrial 3D CAD meshes of furniture with related attributes such as category, style, theme, etc., and high-resolution informative textures developed by professional designers.

3DCoMPaT [31] is a large-scale 3D dataset of more than 7.19 million rendered compositions of materials on parts of 7262 unique 3D models. 3DCoMPaT covers 43 shape categories, 235 unique part names, and 167 unique material classes that can be applied to parts of 3D objects. This dataset primarily focuses on stylizing 3D shapes at part-level with compatible materials. Therefore, each object in this dataset has material part segmentation annotation.

Objaverse 1.0 [33] is a large-scale corpus of high quality, richly annotated 3D objects, which contains over 800,000 3D assets designed by over 100,000 artists. Assets in Objaverse not only belong to varied categories like animals, humans, and vehicles, but also include interiors and exteriors of large spaces that can be used, e.g., to train embodied agents. Objaverse improves upon present day 3D repositories in terms of scale, number of categories, and in the visual diversity of instances within a category.

B.1.2 Relighting in Blender

In the relighting experiments of the main paper (i.e., Section 4.3), we utilized the HDRI environment maps (copyright-free) from Poly Haven [44], a curated public asset library for visual effects artists and game designers. For the texture maps applied to the meshes in Blender, we extracted diffuse maps, normal maps, and roughness maps from the computational graphs of the optimized procedural materials, as all computational graphs can output at least these three types of maps. Moreover, for materials classified as metal, we also employed the metallic maps. Subsequently, we create a Blender Principled BSDF for each material part. The Principled BSDF in Blender is based on the OpenPBR surface shading model, and provides parameters compatible with similar PBR shaders found in other software, such as the Disney and Standard Surface models.

C.1 Extended experiments

In this section, we additionally compare our method with two other state of the art text-driven texture generation methods: Text2Mesh [14] and Paint-it [45]. Unlike diffusion-based methods, Text2Mesh directly optimizes textures and geometries via a CLIP-based optimization function. In addition, the mesh texture generated by Text2Mesh is saved as per-vertex color, rather than texture maps. Similar to Text2Tex [1] and TEXTure [2], we removed the geometry optimization and only optimized the textures for Text2Mesh. Paint-it [45] utilizes a neural network to represent the PBR texture maps of a mesh, and optimizes this specific neural network through a diffusion-based loss and differentiable rendering. For the texture maps generated by Paint-it [45], we constructed the PBR material nodes claimed by NVDiffRast [28] (i.e., the differentiable renderer used by Paint-it) in Blender to create the material, and then obtained the rendering results. As shown in Fig. A2, our method can produce more realistic and consistent textures. The texturing results generated by Text2Mesh lack texture information and show repetition because it

 # System Prompt
You are ChatGPT, a large language model trained by OpenAI, based on the GPT-4 architecture. Knowledge cutoff: 2024-xx.
Current date: 2024-xx-xx.

 # Task Definition
You need to complete a task related to materials in computer graphics. The task here is object recognition and reasoning about the possible material types of the object surface based on the object information you recognized.

I will provide you with a image of a [object_name]. This image is divided into two sub-images. The left one is the original image, and the right one uses blue color to mask the object.

I need you to identify what object the blue masked area corresponds to, and then use your identification results to infer what types of material the object surface may be.

For instance, I give you a image of a bed and the area masked in blue corresponds to the bed legs. You should identify them as bed legs, and then infer based on the bed legs that the surface material may be wood or metal.

 # Material Type Options
You can only choose the following 19 materials.
1. asphalt
2. bricks
3. ceramic
4. concrete
5. fabric
6. food
7. glass
8. jade and crystal
9. leather
10. marble and granite and travertine
11. metal
12. naturalGrounds
13. paper
14. plastic and rubber
15. road
16. roof
17. stone
18. weave
19. wood
Try to give all possible material categories.

 # Part-level Understanding Guidelines and Considerations
The right sub-image indicates the mask area, and then you could judge based on the left sub-image which is the original image.

You must zoom in the corresponding area in the left original image for better recognition.

IMPORTANT: There may be occlusion between objects in the image, please pay attention to the hierarchical relationship.

VERY IMPORTANT: For object recognition, you must make your judgment based on this principle: The mask area is all of the object and the area outside the mask does not belong to the object.

Let's think step by step for object recognition.

 # Material Reasoning Guidelines and Considerations
IMPORTANT: Please also combine image information when reasoning about materials, and sort the results according to likelihood.

Let's think step by step for material reasoning.

 # Output Format
To provide an answer, please provide a short analysis for object recognition and material reasoning.

The analysis should be very concise and accurate.

Then, in the last row, summarize your final decision by "<option for possible material 1> <option for possible material 2> <option for possible material 3> ...", and sorted by likelihood.

An example output looks like follows:
"
Analysis:
Object recognition: The object in the masked area is xxxx, because xxxx.

Material reasoning: The possible material types are xxxx, xxxx, ..., beacuse xxxx.

Final answer:
x x x (e.g., 10 / 3 6 / 4 5 6 7 / 1 2 9 12 13 14 17)
"

Fig. A1 Prompt template.

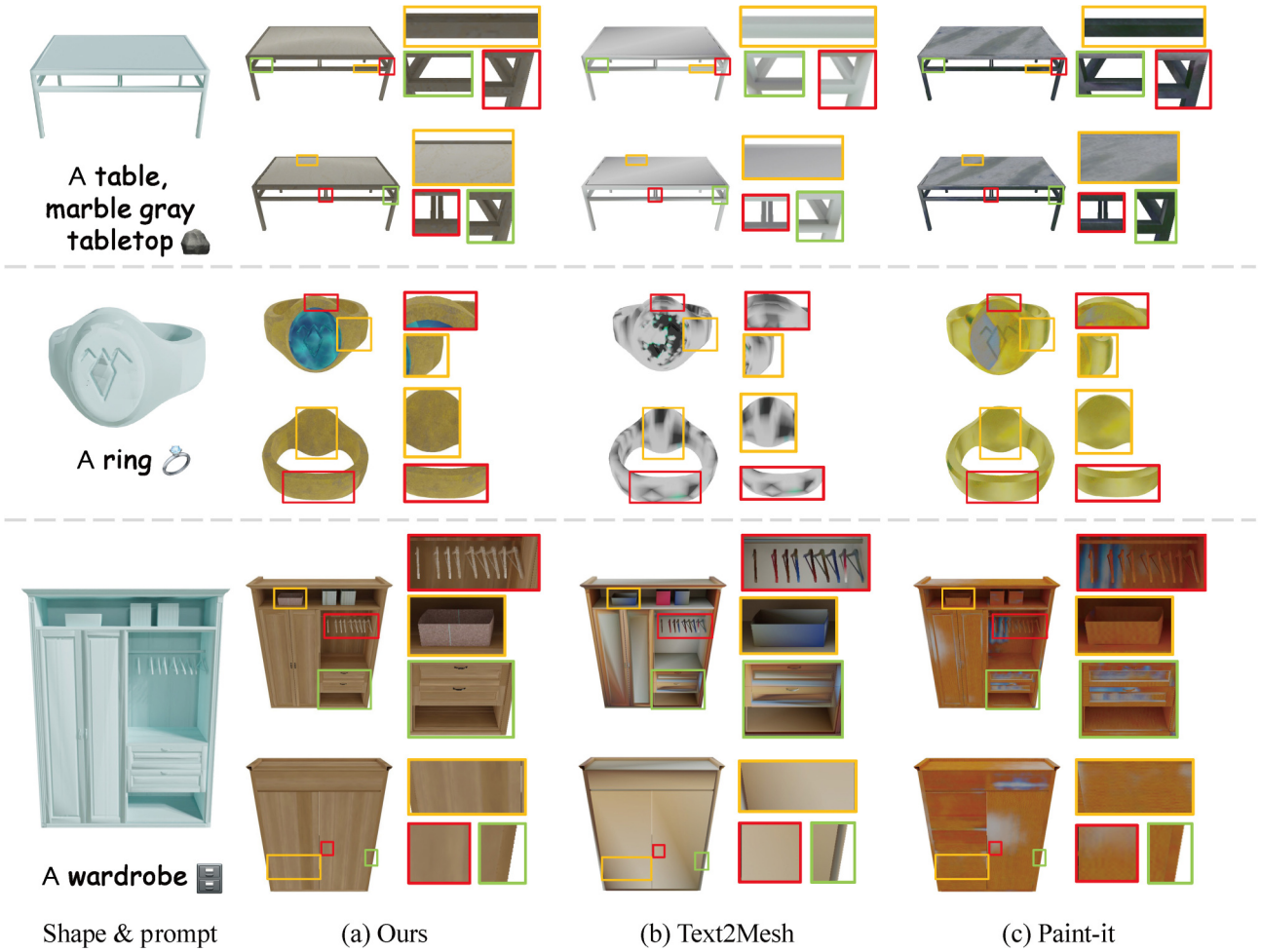


Fig. A2 Qualitative comparison to Text2Mesh and Paint-it. It can be seen that our approach can produce more realistic and consistent textures.

relies solely on CLIP guidance to optimize per-vertex colors, as also demonstrated in Text2Tex [1] and TEXTure [2]. Paint-it utilizes a randomly initialized U-Net network as a proxy for PBR texture maps and lacks material priors, leading to reduced realism in actual application scenarios (e.g., in Blender).

Availability of data and materials

The procedural materials utilized in this study can be downloaded from the official website of DiffMat [5, 6] at `https://github.com/mit-gfx/diffmat-legacy`.

Author contributions

Ziqiang Dang and Zhaopeng Cui conceived of the presented idea, Ziqiang Dang proposed and implemented the prototype system. Ziqiang Dang, Wenqi Dong, and Zesong Yang designed and performed the experiments. Ziqiang Dang and Zhaopeng Cui wrote the paper. Bangbang Yang and Liang Li contributed to the method design. Yuewen Ma and Zhaopeng Cui supervised the project. Ziqiang Dang created the demo video. All authors discussed the results and contributed to the final manuscript.

Acknowledgements

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

Declaration of competing interest

The authors have no competing interests to declare that are relevant to the content of this article.

Electronic Supplementary Material

Supplementary material (a video containing method explanations and qualitative experimental results) is available in the online version of this article at https://doi.org/10.26599/CVM.2025.9450489.

References

[1] Chen, D. Z.; Siddiqui, Y.; Lee, H. Y.; Tulyakov, S.; Nießner, M. Text2Tex: Text-driven texture synthesis via diffusion models. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, 18512–18522, 2023.

[2] Richardson, E.; Metzer, G.; Alaluf, Y.; Giryes, R.; Cohen-Or, D. TEXTure: Text-guided texturing of 3D shapes. In: Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference, Article No. 54, 2023.

[3] Rombach, R.; Blattmann, A.; Lorenz, D.; Esser, P.; Ommer, B. High-resolution image synthesis with latent diffusion models. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 10674–10685, 2022.

[4] Zhang, L.; Rao, A.; Agrawala, M. Adding conditional control to text-to-image diffusion models. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, 3813–3824, 2023.

[5] Shi, L.; Li, B.; Hašan, M.; Sunkavalli, K.; Matusik, W. Match: Differentiable material graphs for procedural material capture. *ACM Transactions on Graphics* Vol. 39, No. 6, Article No. 196, 2020.

[6] Li, B.; Shi, L.; Matusik, W. End-to-end procedural material capture with proxy-free mixed-integer optimization. *ACM Transactions on Graphics* Vol. 42, No. 4, Article No. 153, 2023.

[7] Liu, Y.; Zhu, M.; Li, H.; Chen, H.; Wang, X.; Shen, C. Matcher: Segment anything with one shot using all-purpose feature matching. *arXiv preprint* arXiv:2305.13310, 2023.

[8] Park, K.; Rematas, K.; Farhadi, A.; Seitz, S. M. PhotoShape: Photorealistic materials for large-scale shape collections. *ACM Transactions on Graphics* Vol 37, No. 6, Article No. 192, 2018.

[9] Hu, R.; Su, X.; Chen, X.; Van Kaick, O.; Huang, H. Photo-to-shape material transfer for diverse structures. *ACM Transactions on Graphics* Vol. 41, No. 4, Article No. 131, 2022.

[10] Waechter, M.; Moehrle, N.; Goesele, M. Let there be color! Large-scale texturing of 3D reconstructions. In: *Computer Vision – ECCV 2014. Lecture Notes in Computer Science, Vol. 8693.* Fleet, D.; Pajdla, T.; Schiele, B.; Tuytelaars, T. Eds. Springer Cham, 836–850, 2014.

[11] Bi, S.; Kalantari, N. K.; Ramamoorthi, R. Patch-based optimization for image-based texture mapping. *ACM Transactions on Graphics* Vol. 36, No. 4, Article No. 106, 2017.

[12] Ramesh, A.; Dhariwal, P.; Nichol, A.; Chu, C.; Chen, M. Hierarchical text-conditional image generation with clip latents. *arXiv preprint* arXiv:2204.06125, 2022.

[13] Mohammad Khalid, N.; Xie, T.; Belilovsky, E.; Popa, T. CLIP-Mesh: Generating textured meshes from text using pretrained image–text models. *arXiv preprint* arXiv:2203.13333, 2022.

[14] Michel, O.; Bar-On, R.; Liu, R.; Benaim, S.; Hanocka, R. Text2Mesh: Text-driven neural stylization for meshes. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 13482–13492, 2022.

[15] Yu, X.; Dai, P.; Li, W.; Ma, L.; Liu, Z.; Qi, X. Texture generation on 3D meshes with point-UV diffusion. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, 4183–4193, 2023.

[16] Fridman, R.; Abecasis, A.; Kasten, Y.; Dekel, T. SceneScape: Text-driven consistent scene generation. *arXiv preprint* arXiv:2302.01133, 2023.

[17] Höllein, L.; Cao, A.; Owens, A.; Johnson, J.; Nießner, M. Text2Room: Extracting textured 3D meshes from 2D text-to-image models. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, 7875–7886, 2023.

[18] Zhang, J.; Li, X.; Wan, Z.; Wang, C.; Liao, J. Text2NeRF: Text-driven 3D scene generation with neural radiance fields. *IEEE Transactions on Visualization and Computer Graphics* Vol. 30, No. 12, 7749–7762, 2024.

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

[20] Burley, B. Physically based shading at Disney. In: Proceedings of the ACM SIGGRAPH, 2012.

[21] Hu, Y.; Dorsey, J.; Rushmeier, H. A novel framework for inverse procedural texture modeling. *ACM Transactions on Graphics* Vol. 38, No. 6, Article No. 186, 2019.

[22] Hu, Y.; Guerrero, P.; Hasan, M.; Rushmeier, H.; Deschaintre, V. Node graph optimization using differentiable proxies. In: Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference, Article No. 5, 2022.

[23] Guerrero, P.; Hasan, M.; Sunkavalli, K.; Měch, R.; Boubekeur, T.; Mitra, N. MatFormer: A generative model for procedural materials. *ACM Transactions on Graphics* Vol. 41, No. 4, Article No. 46, 2022.

[24] Hu, Y.; Guerrero, P.; Hasan, M.; Rushmeier, H.; Deschaintre, V. Generating procedural materials from text or image prompts. *arXiv preprint* arXiv: 2304.13172, 2023.

[25] Li, T. M.; Aittala, M.; Durand, F.; Lehtinen, J. Differentiable Monte Carlo ray tracing through edge sampling. *ACM Transactions on Graphics* Vol. 37, No. 6, Article No. 222, 2018.

[26] Nimier-David, M.; Vicini, D.; Zeltner, T.; Jakob, W. Mitsuba 2: A retargetable forward and inverse renderer. *ACM Transactions on Graphics* Vol. 38, No. 6, Article No. 203, 2019.

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

[28] Laine, S.; Hellsten, J.; Karras, T.; Seol, Y.; Lehtinen, J.; Aila T. Modular primitives for high-performance differentiable rendering. *ACM Transactions on Graphics* Vol. 39, No. 6, Article No. 194, 2020.

[29] Zhang, C.; Miller, B.; Yan, K.; Gkioulekas, I.; Zhao, S. Path-space differentiable rendering. *ACM Transactions on Graphics* Vol. 39, No. 4, Article No. 143, 2020.

[30] Fu, H.; Jia, R.; Gao, L.; Gong, M.; Zhao, B.; Maybank, S.; Tao, D. 3D-FUTURE: 3D furniture shape with TextURE. *International Journal of Computer Vision* Vol. 129, No. 12, 3313–3337, 2021.

[31] Li, Y.; Upadhyay, U.; Slim, H.; Abdelreheem, A.; Prajapati, A.; Pothigara, S.; Wonka, P.; Elhoseiny, M. 3D CoMPaT: Composition of materials on parts of 3D things. In: *Computer Vision – ECCV 2022. Lecture Notes in Computer Science, Vol. 13668.* Avidan, S.; Brostow, G.; Cissé, M.; Farinella, G. M.; Hassner, T. Eds. Springer Cham, 110–127, 2022.

[32] Slim, H.; Li, X.; Li, Y.; Ahmed, M.; Ayman, M.; Upadhyay, U.; Abdelreheem, A.; Prajapati, A.; Pothigara, S.; Wonka, P.; et al. 3DCoMPaT++: An improved large-scale 3D vision dataset for compositional recognition. *arXiv preprint* arXiv:2310. 18511, 2023.

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

[34] Deitke, M.; Liu, R.; Wallingford, M.; Ngo, H.; Michel, O.; Kusupati, A.; Fan, A.; Laforte, C.; Voleti, V.; Gadre, S. Y.; et al. Objaverse-XL: A universe of 10M+ 3D objects. *arXiv preprint* arXiv:2307.05663, 2023.

[35] Hanocka, R.; Hertz, A.; Fish, N.; Giryes, R.; Fleishman, S.; Cohen-Or, D. MeshCNN: A network with an edge. *ACM Transactions on Graphics* Vol. 38, No. 4, Article No. 90, 2019.

[36] Lahav, A.; Tal, A. MeshWalker: Deep mesh understanding by random walks. *ACM Transactions on Graphics* Vol. 39, No. 6, Article No. 263, 2020.

[37] Yin, Y.; Liu, Y.; Xiao, Y.; Cohen-Or, D.; Huang, J.; Chen, B. SAI3D: Segment any instance in 3D scenes. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 3292–3302, 2024.

[38] Adobe Substance Designer. 2024. Available at https://www.substance3d.com/

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

[40] Brown, T. B.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. Language models are few-shot learners. *arXiv preprint* arXiv:2005.14165, 2020.

[41] Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint* arXiv:1409.1556, 2014.

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

[43] Gatys, L. A.; Ecker, A. S.; Bethge, M. Image style transfer using convolutional neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2414–2423, 2016.

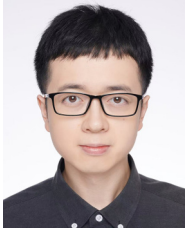
[44] Haven P. HDRIs. 2024. Available at https://polyhaven.com/hdris

[45] Youwang, K.; Oh, T. H.; Pons-Moll, G. Paint-it: Text-to-texture synthesis via deep convolutional texture map optimization and physically-based rendering. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 4347–4356, 2024.

Ziqiang Dang is a M.S. student at Zhejiang University, Hangzhou, China. He obtained his B.E. degree from Wuhan University, China, in 2022. His research interests include computer vision, computer graphics, 3D human, and AIGC.

Wenqi Dong received his B.S. degree from Shandong University, China, in 2021. He is currently pursuing his Ph.D. degree at Zhejiang University, advised by Zhaopeng Cui. His current research interests include 3D generative models and scene understanding.

Zesong Yang is a Ph.D. student at Zhejiang University. He obtained his B.E. degree from Southeast University, Nanjing, China in 2023. His research interests include computer vision, computer graphics, 3D reconstruction, and neural rendering.



Bangbang Yang is a researcher at PICO in ByteDance Inc. He obtained his Ph.D. degree from the ZJU3DV group of the State Key Lab of CAD&CG, Zhejiang University in 2022, advised by Prof. Guofeng Zhang and Prof. Zhaopeng Cui. His research interests include 3D computer vision and content generation, graphics, and mixed reality.

Liang Li is currently a full professor in the School of Media Engineering, the Communication University of Zhejiang, and the Key Lab of Film and TV Media Technology of Zhejiang Province, China. His research interests lie in augmented reality, virtual reality, services computing, computer vision, and 5G/B5G networks.

Yuewen Ma received his Ph.D. degree from Nanyang Technological University, Singapore, in 2013. His research interests focus on computer graphics and 3D vision, and he has published many papers in prestigious conferences and journals such as SIGGRAPH, TVCG, CVPR, ICCV, and T-PAMI. Currently, he leads the 3D reconstruction team at ByteDance PICO.

Zhaopeng Cui received his Ph.D. degree from Simon Fraser University in 2017. He has been a senior researcher at ETH Zurich. He is currently a research professor in the College of Computer Science, Zhejiang University. His research interests include 3D mapping and localization, 3D scene understanding, 3D generation, and 3D motion planning.

To submit a manuscript, please go to `https://jcvm.org`.