The Visual Computer
https://doi.org/10.1007/s00371-025-04096-0

RESEARCH

Interpretable procedural material graph generation via diffusion models from reference images

Xiaoyu Lv[1] · Zizhao Wu[1] · Jiamin Xu[2] · Xiaoling Gu[2] · Ming Zeng[3] · Weiwei Xu[4]

Accepted: 25 June 2025

Abstract

Procedural materials, generated through algorithmic processes, offer advantages such as resolution independence, editability, and real-time rendering capabilities. Despite these merits, constructing procedural material graphs remains a labor-intensive task. Recent advancements in generative neural networks, particularly diffusion models, have shown promise in automating this process. However, existing methods often struggle with issues related to generation quality, generalization, and interpretability. In this work, we introduce a novel approach for the interpretable generation of procedural material graphs from reference images using diffusion model. Our approach predicts individual nodes in reverse order, leveraging the generative capabilities of diffusion models to achieve significant improvements in generation quality, generalization, and interpretability. Specifically, we employ a two-stage framework: an adapter-based diffusion model predicts procedural nodes, forming an auxiliary graph, which is then refined using a DiffMat-based node parameter optimization method. To validate the effectiveness of our approach, we construct a fine-grained procedural material graph dataset containing extensive data and information defined at the node level. Our code and datasets are available at: https://github.com/InterS23/IPMGG.

Keywords Procedural material · Diffusion models · Texture generation · Procedural models

1 Introduction

Procedural materials are a type of material generated through algorithmic processes rather than traditional hand-drawn or scanned image textures. Compared to traditional pixel-based textures, procedural materials offer several key advantages, including resolution independence, which allows them to scale arbitrarily without loss of quality, as well as editability, enabling real-time adjustments to various material properties. As a result, they are widely used in the industry, such as gaming, film production, and product visualization [9]. Currently, procedural materials are usually represented as directed acyclic graphs (DAGs) with tools such as Blender or Adobe Substance 3D Designer (S3D) [2], where each node in the graph represents a specific operation, and the edges define the data flow between these operations. However, building a material graph from scratch has proven to be a time-consuming and labor-intensive task, requiring significant time and effort to define nodes and their parameters, even for professional users (Fig. 1).

To address this issue, researchers have begun exploring the problem of inverse procedural material modeling, where a procedural material graph—including procedural node and

✉ Zizhao Wu
wuzizhao@hdu.edu.cn

Xiaoyu Lv
lvxiaoyu@hdu.edu.cn

Jiamin Xu
superxjm@yeah.net

Xiaoling Gu
guxl@hdu.edu.cn

Ming Zeng
zengming@xmu.edu.cn

Weiwei Xu
xww@cad.zju.edu.cn

1 School of Digital Media Technology, Hangzhou Dianzi University, Hangzhou, China

2 School of Computer Science, Hangzhou Dianzi University, Hangzhou, China

3 School of Informatics, Xiamen University, Xiamen, China

4 School of Computer Science, Zhejiang University, Hangzhou, China

Published online: 17 July 2025

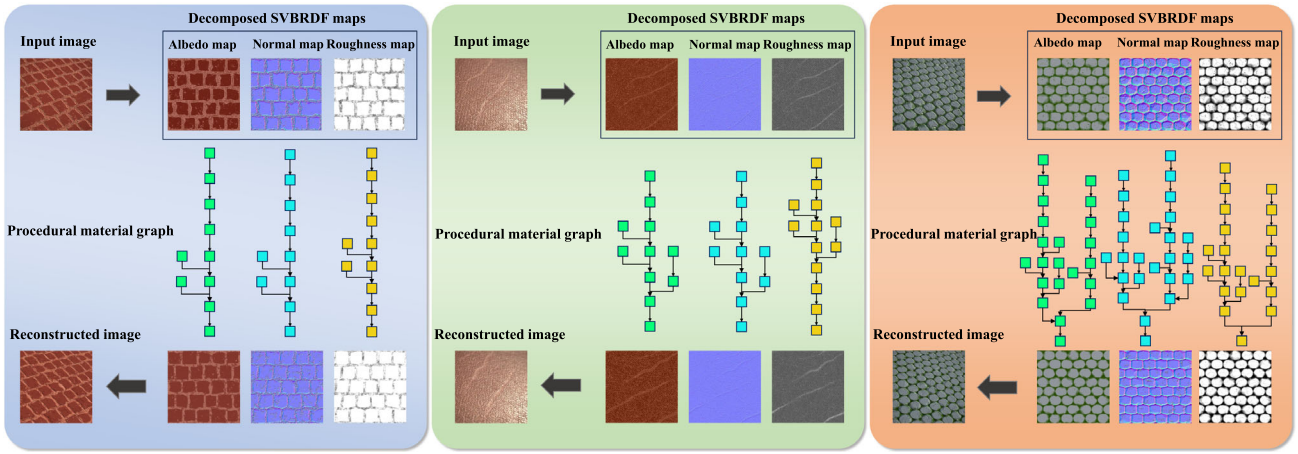


Fig. 1 The synthetic input texture and output of our method. The input to our model consists of SVBRDF maps decomposed from the input image. We then predict a procedural material graph for each map, where each node contains its operation type and parameter definitions, and each edge defines the computation flow between two nodes. Based on the predicted procedural material graph, our goal is to reconstruct the original input. In contrast to existing methods, where the entire graph is predicted at once, our approach predicts each node individually, which provides stronger interpretability

their connections—is generated from a given texture image. These methods are currently divided into two main categories: static graphs [10, 11, 16, 25] and dynamic graphs [6, 12, 13]. The former [10, 11, 16, 25] focuses on selecting appropriate graph structures from an existing dataset of graphs and determining node parameters through learning and optimization. The main drawback of this kind of approach lies in its lack of generalization, as it cannot generate new material graphs. Dynamic methods [6, 12, 13], on the other hand, effectively overcome this limitation. Among these methods, a notable approach [12] utilizes Transformer-based [26] techniques for procedural graph construction, enabling diverse graph generation and automatic graph completion based on multimodal conditional signals. However, we note that this approach relies solely on a single reference image to generate all nodes and, during the generation phase, nodes are generated based only on the node arrangement probability distribution, without considering the corresponding output images of intermediate nodes. While this method produces results that closely align with the input image, its generation process is a black-box procedure based on probabilistic predictions. As a result, this approach suffers from poor interpretability and generalization in node generation.

To overcome these problems, we propose an interpretable method to predict the parameters of each procedural material node. Each node can be viewed as a function with two types of input parameters: image types and primitive data types such as integers and floats. We divide the prediction process into two stages because image parameters and non-image parameters belong to distinct domains, and current generative models struggle to manage these multi-type parameters effectively. Moreover, image-type parameters offer greater semantic information than other parameter types, making them our primary focus during prediction by leveraging the robust prior knowledge of stable diffusion (SD) models. For non-image parameters, we transform the prediction process into an optimization process to simplify the model.

Our approach sequentially predicts each node of procedural graph in reverse order, inspired by one of the existing manual workflows for constructing material graphs based on reference materials. Normally, when creating procedural materials from a reference image, people usually approach this process by dividing and conquering, that is, first extracting the spatially varying bi-directional reflectance distribution functions (SVBRDFs) of materials, such as normal maps, roughness maps, and albedo maps. Then, for each map, the individual material properties are separately predicted, where each property corresponds to a sequence of node operations. We design the node prediction process in reverse order, that is, predicting from the output nodes to the generator nodes. Compared to predicting from the generator nodes to the output nodes, this strategy significantly improves prediction tolerance and enhances the quality of generated images. Specifically, if we predict in a root-to-leaf order, an error in any intermediate node would propagate to its child nodes, causing the generated result to deviate completely from the given material, more details are in the supplemental materials.

In specific implementation, we propose a divide-and-conquer strategy consisting of two processes: a preprocessing process and a sampling process. In the preprocessing phase, we train a classifier on our constructed dataset to recognize node types and develop an SD-based adapter that can be used for input image prediction. In the sampling phase, given a ref-

erence image, we devise a two-stage method that operates by first predicting the type and input image of each generated node and then predicting the detailed parameters for each node. We illustrate several examples of the proposed method in Fig. 1.

To validate the effectiveness of our approach, we create a procedural material graph dataset containing 618,872 images across 78 categories. Extensive experiments demonstrate that our method achieves superior generation quality and generalization capability compared to previous approaches. Additionally, with semantic information available for each generated node, our method features a more interpretable generation process that enables a closer match to the synthetic input image.

In summary, our contributions are as follows:

- We propose a novel multi-stage framework that interpretably generates procedural material graphs, where each node's type and attributes are predicted in reverse order, with the corresponding reference image guiding the process at each step.
- We introduce an adapter-based diffusion model capable of predicting node attributes within a lightweight framework.
- We present a fine-grained procedural material graph dataset consisting of extensive data on individual nodes with various types and parameters.

2 Related work

In this section, we briefly review related work on inverse material modeling and diffusion models.

2.1 Inverse procedural material modeling

Inverse Procedural Material Capture aims to identify the most suitable graph from an existing dataset and then applies parameter prediction and optimization techniques to find the optimal parameters. Early approaches [5, 10] relied on hand-crafted graph templates and rules to capture procedural material details from examples. The success of these methods largely depends on template quality, and their application is often limited to specific material categories.

Later, Shi et al. [25] introduced the DiffMat library, which enabled differentiable procedural material nodes generation, facilitating gradient-based optimization of a wider range of continuous filter node parameters. Subsequently, Hu et al. [11] expanded the scope to include discrete parameters by training differentiable neural network approximations for generator nodes. More recently, Li et al. [16] introduced DiffMat v2, which supports a broader range of node types and allows for optimizing both continuous and discrete parameters. They also developed a three-stage optimization process to further refine the parameter optimization. However, these approaches are not generative methods and the effectiveness of the model largely depends on whether similar materials exist in the database.

Inverse Procedural Material Generation targets at generating complete procedural graphs from scratch based on reference material images. Hu et al. [13] proposed a method that hierarchically segments the input SVBRDF maps into sub-materials and their corresponding masks, constructing a binary tree of procedural materials. However, this approach is limited to specific material types and requires manual segmentation by the user. Guerrero et al. [6] introduced a transformer-based generative method, enabling the unconditional generation of procedural graphs. In a subsequent work, Hu et al. [12] incorporated a CLIP encoder into the transformer architecture and introduced a novel sampling strategy to expand the dataset, allowing procedural material generation to be performed unconditionally or conditioned on images, text prompts, or partial graphs. However, Transformer-based methods [6, 12] predict tokens based on the graph's arrangement order, without considering individual nodes' semantic information. As a result, while it can generate materials resembling the style of the input image prompts, the generated graph lacks a clear semantic association between each node and its parameters, leading to limited interpretability and generalization. Furthermore, Li et al. [15] proposed a reinforcement learning-based approach, while Li et al. [17] introduced a pretrained vision-language model (VLM), which outperforms previous methods on both synthetic and real-world examples.

In contrast, our method explicitly models each node, allowing it to generate material graphs with high interpretability and generalization.

2.2 Diffusion model

Diffusion models [7], as a novel generative paradigm, have achieved remarkable success in image generation. Numerous diffusion methods have been explored for text-to-image (T2I) generation tasks, such as Stable Diffusion [22], Imagen [23], and DALL-E [21]. Due to the high parameter requirements of diffusion models, full-parameter fine-tuning [20] for downstream tasks incurs significant training costs. Recently, some researchers have attempted efficient parameter tuning based on Adapters [8] or LoRAs [29], which adds an extra module to the frozen pretrained model.

For example, Zhang et al. [28] proposed ControlNet, which fine-tunes a pretrained diffusion model with a small number of parameters and introduces various types of image prompts, such as normal maps. Subsequently, works such as T2I-Adapter [19] and IP-Adapter [27] further improved the Adapter architecture, reducing the number of required

parameters while maintaining the fine-tuning performance. In this paper, we also trained an adapter on a pretrained diffusion model to achieve a lightweight procedural material attribute predictor for node-specific properties.

Diffusion models have also been utilized [3, 14] to solve the inverse problems, such as denoising, super-resolution [30], image restoration, etc. For example, Chung et al. [3] proposed the diffusion posterior sampling (DPS) algorithm, which leverages the prior knowledge learned by a pretrained diffusion model. By specifying the form of the operator in advance, the posterior distribution can be estimated without additional network training. In our work, the procedural material graph generation task can also be viewed as an inverse problem, where each node in the graph can be considered as a known operator, such as a linear or nonlinear noise operator, coordinate transformation, etc. However, because node functions involve not only a varying number of image-type inputs but also other types of parameters with unpredictable values, the operator parameters cannot be predefined in the DPS. Therefore, we opted to train the diffusion model to directly learn the inverse functions corresponding to the node functions from the dataset.

3 Method

Given a pixel-based reference material, our objective is to generate the corresponding procedural material graph. We adopt the directed acyclic graphs (DAGs) structure from Adobe S3D [2] as our procedural material graph representation, which is composed of nodes, edges, and parameters. In the graph, each node is parameterized, enabling fine-grained control over the material properties. The parameters defined on the node are divided into two categories: image types and primitive data types, e.g., integers and floats. At runtime, each node performs operations on the input image based on the operations and parameters defined for that node, producing an output image. This output is then used as the input for the subsequent node, continuing the process. Nodes are connected by edges, which define the flow of data throughout the DAG. In a typical procedural material graph, the direction of these edges flows from the generator node to the output nodes.

Figure 2 illustrates the pipeline of our framework, which is divided into two phases: the preprocessing phase and the sampling phase. In the preprocessing phase, we constructed a dataset for each node type and trained both a node category classifier and the corresponding adapters for each node type within the diffusion model. In the sampling phase, the prediction is conducted in two stages: in the first stage, we use an inverse rendering model [18, 24] to derive an SVBRDF texture map from the reference image. This is followed by a ViT-based encoder that predicts the node type. Using the predicted node type, the reference image, and adapter weights from the preprocessing phase, we generate the input image for the current node. In the next iteration, this predicted image serves as the reference image for the parent node, and the process continues iteratively in reverse order, starting from the output nodes and ending with the generator node. In the second stage, we use DiffMat v2 [16] to optimize the node parameters via end-to-end differentiable rendering. Starting from the generator node of the graph predicted in the first stage, each node is iteratively treated as the output node. Using its predicted image as the reference, we optimize the parameters of the current node and its ancestor nodes along the branch, repeating this process until we reach the original output nodes. In the end, we obtain a complete procedural material graph, containing the type and property information for all nodes.

In the following, we start with a systematic overview of the objectives and processes involved in our generation task, and then give a detailed explanation of the methods used for predicting node image-type parameters and optimizing other node parameters in Sect. 3.1. In Sect. 3.2, we provide an in-depth discussion of the approach and methodology used for constructing our dataset.

3.1 Node generation and parameter prediction

This subsection aims to predict the parameters from the corresponding node of a reference image, we frame the task as an inverse function problem, defined as $(x, R) = \mathcal{H}^{-1}(y)$, where $\mathcal{H}$ represents the node function, x is the image type parameters, R denotes the other primitive type parameters, and y is the output image. We solve the problem in the following order: First, we identify the node type $\mathcal{H}$ that was used to generate the input image y. Next, we predict the parameters x and R separately. Specifically, we observe that the output image, after applying the node operation, exhibits distinct classification features, allowing us to use a ViT classification model [4] to infer the node type. Since image parameters and other types of parameters belong to different domains, we enhance the accuracy of the inverse function prediction by separately estimating x and R. For x, we utilize the diffusion model's strength in function approximation and probabilistic modeling. For R, we reformulate its prediction as an optimization task, using DiffMat v2's differentiable node functions to achieve more precise parameter estimation.

3.1.1 Parameter prediction of image types

We employ an adapter-based approach for SD models to predict the image-type parameters for each node. Training an SD model to predict input images for all nodes simultaneously results in a significant drop in prediction quality. This is mainly due to the wide variation in the function spaces rep-

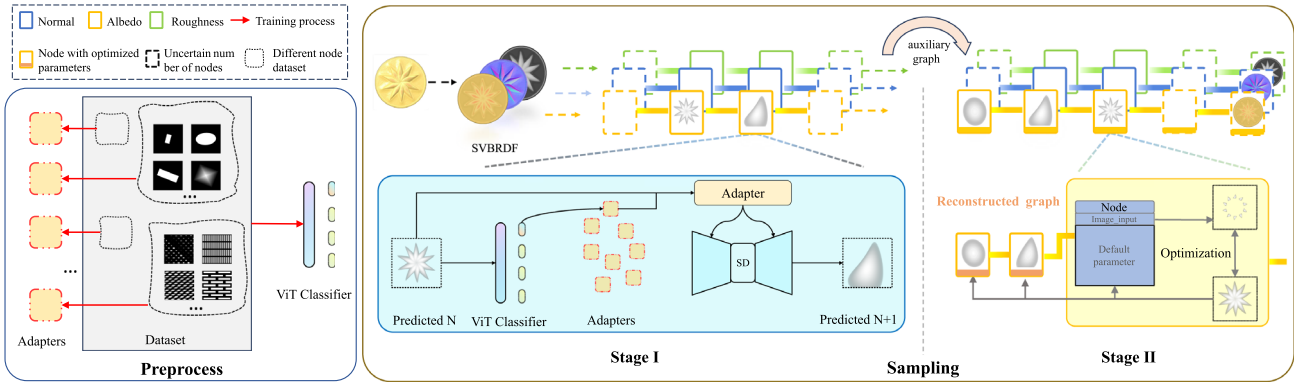


Fig. 2 The overall framework of our method, which consists of two phases. On the left, the preprocessing phase is depicted, where a classifier and adapter for individual nodes are trained based on the constructed dataset. On the right, the core algorithm of our approach is presented, which is divided into two stages: one for predicting the node types and the other for predicting the node parameters

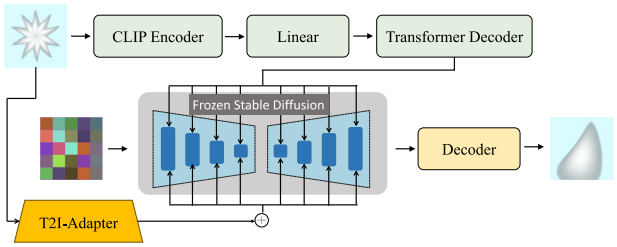


Fig. 3 The details of our adapter architecture built upon the pretrained SD model. A separate adapter is trained for each node type

resented by different nodes, making it difficult for the model to capture an overall probability distribution. Moreover, such a unified approach complicates training and reduces scalability. To address this issue, we train an adapter for each node type individually. This method preserves the original model's generative capabilities, mitigates interference between different functions, and allows for flexible node expansion.

Previously, adapters for SD models were primarily used to guide content in text-to-image models [19, 27, 28]. For instance, a normal map might be provided to control the surface normals of the generated image, with a text prompt guiding the image content. In our work, we repurpose adapters as inverse function solvers for node functions and eliminate the use of text guidance. Specifically, our goal is to enable the adapters to predict feature embeddings for each node's input image based on both the node type and its reference image. However, the pretrained SD model's weights are heavily tied to the text feature space. Therefore, it is essential to align the predicted image embeddings with the corresponding text embeddings. So, we designed a new adapter architecture to address these issues, as shown in Fig. 3. The reference image is first encoded by a CLIP Image Encoder to obtain an embedding aligned with the corresponding text prompt. This embedding is then mapped to the text-aligned feature space using a linear layer followed by a three-layer Transformer Decoder. Additionally, we incorporated the T2I-Adapter to enhance the model's sensitivity to high-frequency details in the reference image.

For certain special nodes, we adopted specific strategies. For generator nodes that do not take image inputs, such as shape node, perlin_noise node, etc., which are the generator nodes in procedural graphs, we directly optimize the node parameters starting from their initial values without going through adapter prediction. Some node functions, such as invert, Cartesian to polar, polar to Cartesian, etc., come with built-in inverse functions, so there is no need for a neural network to predict the input image. To further enhance the model's prediction accuracy, we applied specialized treatments to certain nodes. For example, for nodes with segmentation attributes, such as blend_copy_with_mask, we only predict the mask image, and then use the mask map to segment the foreground image and background image. Afterward, we apply the SD model's image completion capabilities to restore both images, as illustrated in Fig. 4. For RGB-type albedo maps, we set the node type as the dynamic gradient node implemented in [16]. During the initial sampling step, we first transform the RGB image into a grayscale map. Then, using the predefined grayscale-color mapping method, we extract the color gradient. The resulting grayscale map is then used for subsequent predictions.

3.1.2 Parameter prediction of primitive data types

We adopted the three-stage primitive type parameters optimization method proposed by [16]. The parameters can be classified into differentiable continuous parameters R_θ, non-differentiable continuous parameters R_ϱ, and non-differentiable discrete parameters R_ϕ. In the first stage, we use a gradient descent method to roughly optimize the differentiable con-

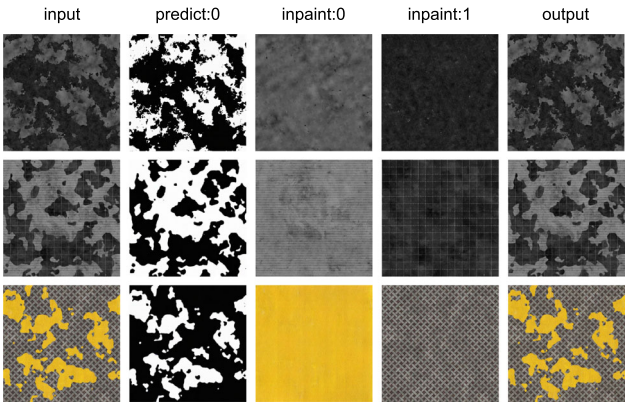


Fig. 4 Mask prediction and image inpainting results of our approach. The first column shows the input image, while the second column presents the mask predicted by our model. Using these inputs, our model generates the completed images, as seen in the third and fourth columns, attributed to the SD model's inpainting capability. Finally, the fifth column displays the reconstructed image produced by our model

tinuous parameters,

$$R_\theta \leftarrow \arg\min_{R_\theta} \|\mathcal{T}(\mathcal{H}(x, R)) - \mathcal{T}(y)\|, \tag{1}$$

where $\mathcal{T}$ is a texture descriptor operator that computes the Gram matrix of extracted VGG19 features. In the second stage, linear grid search is used to optimize the non-differentiable continuous parameters and non-differentiable discrete parameters. Linear grid search is an exhaustive search method that tries all possible combinations in the parameter space, with candidate values of each parameter are evenly distributed at certain intervals, aiming to maximize a predefined evaluation criterion.

$$\begin{aligned} R_\varrho, R_\phi \leftarrow \arg\min_{R_\varrho, R_\phi} & \|\mathcal{T}(\mathcal{H}(x, R)) - \mathcal{T}(y)\|_1 \\ + w_\mathcal{F} & \|\mathcal{F}(\mathcal{H}(x, R)) - \mathcal{F}(y)\|_1, \end{aligned} \tag{2}$$

where $\mathcal{F}$ is the 2D fast Fourier transform (FFT) operator to capture high-frequency structures or periodic textures in the material. Since the discrete parameters optimized in the second stage may substantially impact the node's final output, the third stage continues to refine the node parameters using Eq. 1.

We adopt a node-by-node optimization strategy, in contrast to the approach by Li et al. [16], which optimizes all nodes of a procedural graph based on a single reference image. This often leads to cumulative errors that degrade the final output. Specifically, starting from the generator node of the predicted graph in the first stage of our sampling process, we iteratively treat each node as the output node. Using its predicted image as the reference, we apply the aforementioned three-phase method to optimize the parameters of the node and its ancestor nodes along the branch, continuing this process until the original output nodes are reached, as illustrated in Fig. 2 on the right side. This fine-grained approach significantly reduces error propagation, and Table 1 highlights the advantages of this detailed optimization process.

3.2 Procedural material graph dataset

We collected limited data of complete procedural material graphs from the material library [1] and analyzed the statistical characteristics of each node and parameters to build our dataset. Specifically, we analyzed some commonly used nodes implemented by DiffMat v2 in the library and obtained the probability distributions of each node's parameters, as well as the sequence distribution between nodes.

By analyzing the node sequence statistical characteristics, we can construct a dataset that reflects the probability distribution of node arrangements. This allows the classifier to learn the prioritization of different node functions, even when an image contains features from multiple nodes. Consequently, features that need to be constructed early in the procedural graph will not be misplaced during the reverse prediction process, which could otherwise result in the loss of other features. By analyzing the node parameters' statistical characteristics, we devised a sampling strategy to ensure that the generated parameters match the real data distribution. Then, based on prior knowledge, we carefully selected node parameters that produce output images with clear and distinct node-level semantic information.

We exclude cases where node parameters, such as setting the blur node's intensity parameters to zero, result in identical input and output images. These cases fail to reflect the node's characteristics. Additionally, we consider cases where different nodes, such as the tile generator and blend nodes, produce similar outputs under specific conditions, though their editing logic differs, e.g., random point distribution or structured combination. Therefore, we carefully compared such functionally similar nodes and excluded data that did not match the logical editing processes based on prior knowledge.

Some nodes have multiple image inputs, but not all of them need to be used. We set their usage frequency according to the probability distribution of the image parameters. For those with very low frequency, we decide not to generate them in future. Additionally, some nodes implement multiple function logics, controlled by certain parameters. However, if these function logics produce significantly different effects, treating them as a single node category would degrade the quality of the generated results. Therefore, we separate such parameters in the nodes, assigning each function logic to a distinct node category, e.g., splitting the blend node into blend_add, blend_multiply, etc.

Table 1 The quantitative evaluation of our approach

	w/o per-node optimization			Our method		
	SSIM↑	PSNR↑	LPIPS↓	SSIM↑	PSNR↑	LPIPS↓
Albedo	0.28	29.00	**0.26**	**0.45**	**30.01**	0.40
Roughness	0.44	30.41	**0.28**	**0.56**	**31.40**	0.35
Normal	0.20	29.29	0.26	**0.42**	**30.73**	**0.17**
PBR	0.17	28.20	0.36	**0.22**	**28.82**	**0.26**

Bold indicates the better performance between the two compared methods

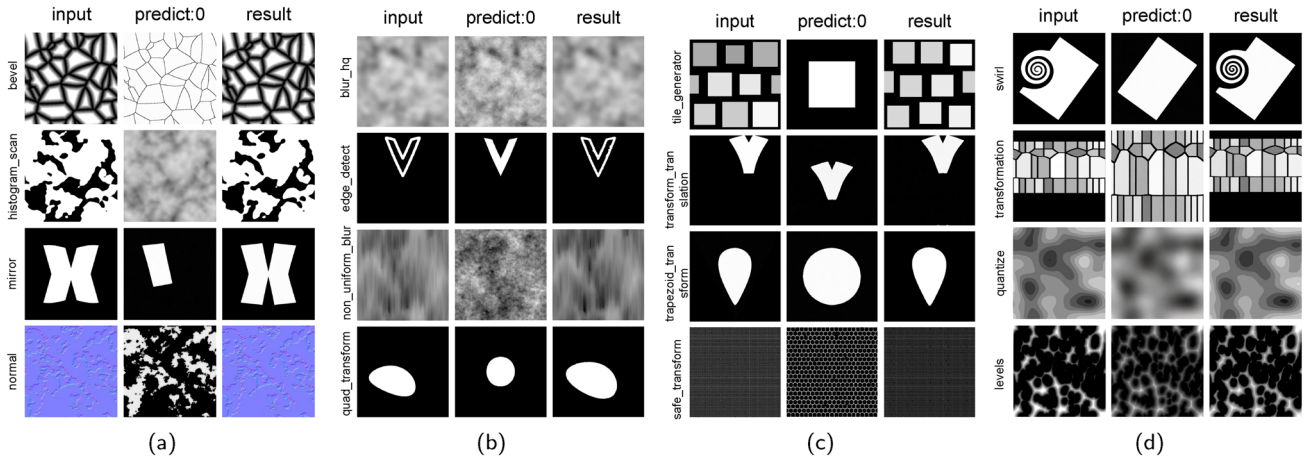


Fig. 5 The prediction results of our method for a single node. "predict:0" is the result from stage 1, generated by the SD model, while "result" is the output after parameter optimization in stage 2

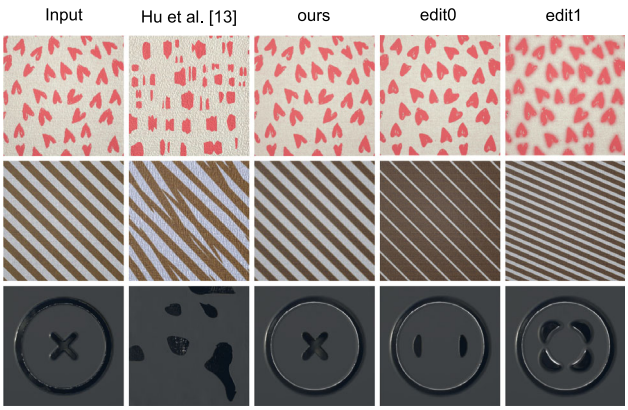


Fig. 6 The qualitative comparison between our method and Hu et al. [13]

The dataset includes images and their corresponding JSON files, with each JSON file containing the node type and input parameters for the image.

4 Experiments

4.1 Implementation details

For the node classification task, we selected 45 node categories from the differentiable nodes implemented by [16]. Using the method described in Sect. 3.2, where node categories are divided into multiple subcategories based on their parameters, we expanded the 45 categories to 78, encompassing a total of 618,862 images. For the training of the adapters, we excluded categories from the previous dataset that do not require image inputs or have built-in inverse functions. Additionally, since some categories have multiple image inputs, we trained a separate adapter for each input index, resulting in 42 categories with 552,706 images. The performance metrics for all adapters are provided in the supplementary materials.

For the classifier, we use the pretrained SwinV2-base-window12to24-192to384 model and modified the final fully connected layer to output 78 classes with 618,862 images. The classifier was fine-tuned on 2 NVIDIA GeForce RTX 4090 GPUs with a batch size of 8, using AdamW optimizer with cosine warmup schedule and the learning rate is 1e−4. We trained for 10 epochs and employed Focal Loss with a gamma value of 2.0 as the loss function.

For the adapter, we use the pretrained SDXL-base−1.0 as the base model, which was trained on 552,706 images across 42 classes using 3 NVIDIA GeForce RTX 4090 GPUs with a

batch size of 8. The AdamW optimizer was utilized, incorporating a cosine annealing warmup strategy with the learning rate 1e-4. As the convergence speed varied among adapters, we set the number of training epochs to 100, corresponding to the slowest converging node. During training, we truncated the process based on each node's learning performance.

For the second stage of our method, we use a hybrid optimization approach. Differentiable parameters are optimized with Adam (learning rate: 5e−4), while non-differentiable parameters (e.g., integers) are tuned using linear grid search. This hybrid process iterates 8 times.

4.2 Qualitative results

We present the image prediction capabilities of adapters for each node category qualitatively in Fig. 5. The differences in metrics across various nodes reflect the varying levels of difficulty in predicting different nodes. It is noteworthy that the deviation between nodes of regular shape data types and noise data types affects the metrics differently. For example, in the case of noise-type images, the predicted values may differ significantly from the ground truth at the pixel level, but the overall structure and content of the image remain similar. Figure 5 illustrates the model's ability on individual nodes: by providing a reference image, followed by parameter prediction and optimization within our network, we can achieve good reconstruction results.

4.3 Comparison

Previous studies struggled to handle procedural materials with masks [6, 12]. Hu et al.[10] can handle this, but it requires additional segmentation models and manual annotation of the masked regions. Our method treats the original segmentation task as a node function, naturally incorporating it into our pipeline. It allows leveraging the SD model's inherent prior image knowledge to complete the missing areas, significantly improving generalization and robustness to noise. Figure 4 showcases our model's capability to predict mask regions and perform image completion for missing areas.

Our model has better generalization and editing capabilities. In contrast to [13], which struggles with patterns absent from their dataset and has its generation capabilities constrained by a Gaussian noise model, our approach effectively manages such cases, as shown in Fig. 6.

Compared to [6, 12], which generates procedural graphs from a single input in an autoregressive manner, limiting the ability to capture fine-grained node characteristics and restricting the diversity of generated images, our method predicts each node individually during the generation process. This allows us to generate a wider range of material images, not just starting from the final PBR-composed image, but from any node image in the procedural graph. More details are in the supplemental materials.

4.4 Procedural material editing

In practical applications, we frequently encounter scenarios requiring the modification of a specific attribute of a given reference material while keeping other attributes unchanged. Procedural materials are composed of a graph made up of various nodes, each with adjustable parameters, highlighting their inherent editability and flexibility. Once our model generates the procedural graph corresponding to the reference image, the material can be edited in a decoupled manner. As shown in Fig. 7, by simply modifying the parameters of certain nodes, corresponding editing operations can be completed.

4.5 Trade-off between fidelity and editability

In certain cases, while the nodes and images predicted in Stage I in Fig. 2 may be correct, nodes that depend on random number generation algorithms make it challenging for our optimization method to achieve pixel-level accuracy, preserving only stylistic similarity instead. This leads to a trade-off between fidelity and editability for such types of graphs. More details are in the supplemental materials.

5 Limitations and future work

Our model's generation capabilities are constrained by the number of nodes implemented in DiffMat v2, and unseen node features by our model may lead to generation errors, as shown in Fig. 8. Additionally, since each node is assigned a dedicated adapter, the space complexity increases linearly with the number of nodes, and the time complexity also rises due to the optimization required for each node and its parameters. Furthermore, our model does not handle exposed parameters in the method [2], restricting usability for general users and also the method is unable to connect edges to previously generated nodes in other branches. Consequently, the generated graph always has a tree-like topology.

To address these limitations, future work will focus on implementing more differentiable nodes and expanding the amount of real-world data available for each node. We will also refine the adapter architecture to improve the model's sensitivity to high-frequency details and its ability to handle noise. Further, we plan to design new loss functions for more accurate optimization. Beyond image-to-procedural graph conversion, future research will explore integrating language prompts, allowing the model to generate and edit procedural graphs from textual descriptions.

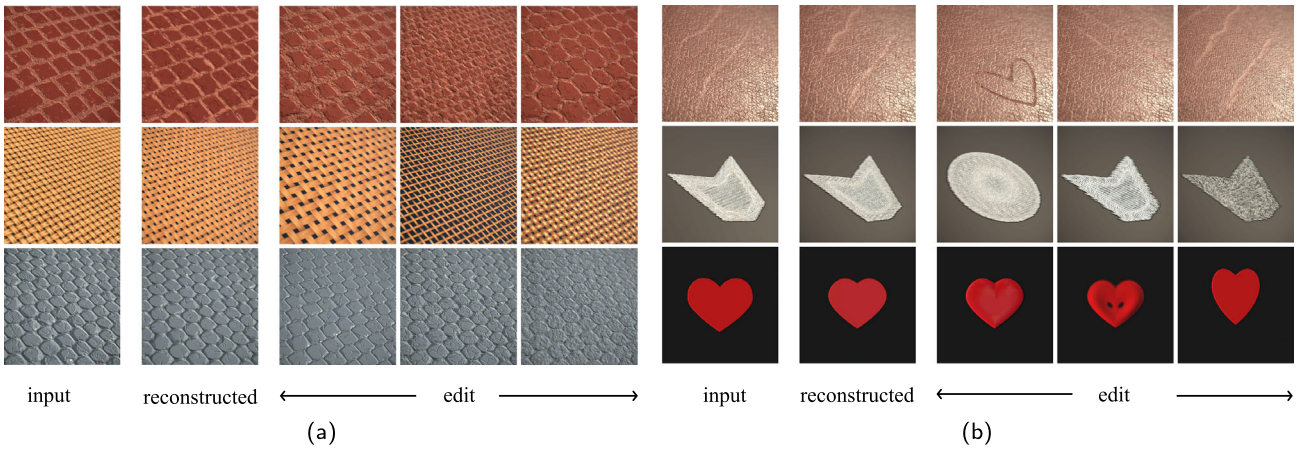


Fig. 7 Editing capability of our approach for procedural materials. For each image, the first column displays the synthetic image, and the second column shows the reconstructed image generated from the procedural material graph predicted by our model. The rest columns illustrate the effects of modifying the parameters of a specific node in the generated graph

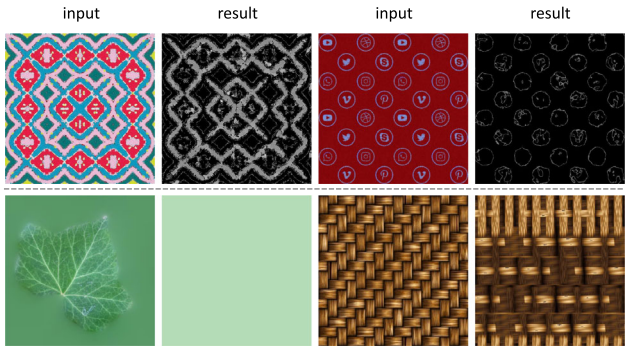


Fig. 8 Failure cases. The first row shows an erroneous intermediate node results generated during Stage 1. The second row presents the results after both Stage 1 and Stage 2

6 Conclusion

We introduce a novel framework for interpretable generation of procedural graphs from reference images using diffusion models. This framework models each procedural node, learning fine-grained semantic information at the node level. Our approach effectively generates procedural materials that are consistent with the semantic structure of the input image. By incorporating the image-level semantic information of each node, it offers enhanced interpretability during the generation process, thereby improving the generalization and generation capabilities of the results.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s00371-025-04096-0.

Author Contributions Xiaoyu Lv contributed to conceptualization, methodology, software, writing—original draft, and visualization. Zizhao Wu contributed to supervision, conceptualization, and writing—review and editing. Jiamin Xu, Xiaoling Gu, Ming Zeng, and Weiwei Xu contributed to methodology, resources, and software.

Data availability No datasets were generated or analyzed during the current study.

Declarations

Conflict of interest The authors declare no conflict of interest.

References

1. Adobe, Adobe substance 3d community assets. https://substance3d.adobe.com/community-assets (2024a)
2. Adobe, Substance designer. (2024b) https://www.substance3d.com
3. Chung, H., Kim, J., Mccann, M.T., Klasky, M.L., Ye, J.C.: Diffusion posterior sampling for general noisy inverse problems. (2022) arXiv:2209.14687
4. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N.: An image is worth 16x16 words: Transformers for image recognition at scale. in: International conference on learning representations, ICLR, OpenReview.net (2021)
5. Dumas, J., Lu, A., Lefebvre, S., Wu, J., Dick, C.: By-example synthesis of structurally sound patterns. ACM Trans. Graph. **34**, 1–12 (2015)
6. Guerrero, P., Hasan, M., Sunkavalli, K., Mech, R., Boubekeur, T., Mitra, N.J.: Matformer: a generative model for procedural materials. ACM Trans. Graph. **41**, 1–12 (2022)
7. Ho, J., Jain, A., Abbeel, P.: Denoising diffusion probabilistic models. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., Lin, H. (eds.) Advances in Neural Information Processing Systems. NeurIPS (2020)
8. Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., de Laroussilhe, Q., Gesmundo, A., Attariyan, M., Gelly, S.: Parameter-efficient transfer learning for NLP. In: Chaudhuri K, Salakhutdinov R (Eds.). In: Proceedings of the 36th international conference on machine learning, ICML, PMLR. pp. 2790–2799 (2019)

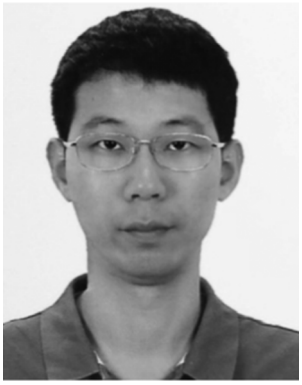
9. Hu, X., Yang, C., Fang, F., Huang, J., Li, P., ShengB, B., Lee, T.Y.: Msembgan: Multi-stitch embroidery synthesis via region-aware texture generation. IEEE Trans. Vis. Comput, Graph (2024)
10. Hu, Y., Dorsey, J., Rushmeier, H.: A novel framework for inverse procedural texture modeling. ACM Trans. Graph. **38**, 1–14 (2019)
11. Hu, Y., Guerrero, P., Hasan, M., Rushmeier, H., Deschaintre, V.: Node graph optimization using differentiable proxies. In: ACM SIGGRAPH 2022 conference proceedings, pp. 1–9 (2022a)
12. Hu, Y., Guerrero, P., Hasan, M., Rushmeier, H., Deschaintre, V.: Generating procedural materials from text or image prompts. In: ACM SIGGRAPH 2023 conference proceedings, pp. 1–11 (2023)
13. Hu, Y., He, C., Deschaintre, V., Dorsey, J., Rushmeier, H.: An inverse procedural modeling pipeline for svbrdf maps. ACM Trans. Graph. **41**, 1–17 (2022)
14. Kawar, B., Elad, M., Ermon, S., Song, J.: Denoising diffusion restoration models. Adv. Neural. Inf. Process. Syst. **35**, 23593–23606 (2022)
15. Li, B., Hu, Y., Guerrero, P., Hasan, M., Shi, L., Deschaintre, V., Matusik, W.: Procedural material generation with reinforcement learning. ACM Trans. Graph. **43**, 1–14 (2024)
16. Li, B., Shi, L., Matusik, W.: End-to-end procedural material capture with proxy-free mixed-integer optimization. ACM Trans. Graph. **42**, 1–15 (2023)
17. Li, B., Wu, R., Solar-Lezama, A., Zheng, C., Shi, L., Bickel, B., Matusik, W.: Vlmaterial: Procedural material generation with large vision-language models. arXiv preprint (2025) arXiv:2501.18623
18. Li, Z., Sunkavalli, K., Chandraker, M.: Materials for masses: Svbrdf acquisition with a single mobile phone image. In: Proceedings of the European conference on computer vision (ECCV), pp. 72–87 (2018)
19. Mou, C., Wang, X., Xie, L., Wu, Y., Zhang, J., Qi, Z., Shan, Y.: T2i-adapter: Learning adapters to dig out more controllable ability for text-to-image diffusion models. In: Proceedings of the AAAI conference on artificial intelligence, pp. 4296–4304 (2024)
20. Podell, D., English, Z., Lacey, K., Blattmann, A., Dockhorn, T., Müller, J., Penna, J., Rombach, R.: SDXL: improving latent diffusion models for high-resolution image synthesi. In: The Twelfth International conference on learning representations, ICLR, OpenReview.net (2024)
21. Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., Chen, M., Sutskever, I.: Zero-shot text-to-image generation. In: Meila, M., Zhang, T. (Eds.), Proceedings of the 38th international conference on machine learning, ICML, PMLR. pp. 8821–8831 (2021)
22. Rombach, R., Blattmann, A., Lorenz, D., Esser, P., Ommer, B.: High-resolution image synthesis with latent diffusion models. In: IEEE/CVF Conference on computer vision and pattern recognition, pp. 10674–10685. IEEE, CVPR (2022)
23. Saharia, C., Chan, W., Saxena, S., Li, L., Whang, J., Denton, E.L., Ghasemipour, S.K.S., Lopes, R.G., Ayan, B.K., Salimans, T., Ho, J., Fleet, D.J., Norouzi, M.: Photorealistic text-to-image diffusion models with deep language understanding. In: Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., Oh, A. (eds.) Advances in neural information processing systems. NeurIPS (2022)
24. Sartor, S., Peers, P.: Matfusion: a generative diffusion model for svbrdf capture. In: SIGGRAPH Asia 2023 conference papers, pp. 1–10 (2023)
25. Shi, L., Li, B., Hašan, M., Sunkavalli, K., Boubekeur, T., Mech, R., Matusik, W.: Match: differentiable material graphs for procedural material capture. ACM Trans. Graph. **39**, 1–15 (2020)
26. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: Advances in neural information processing systems. pp. 5998–6008 (2017)
27. Ye, H., Zhang, J., Liu, S., Han, X., Yang, W., 2023. Ip-adapter: text compatible image prompt adapter for text-to-image diffusion models. arXiv preprint arXiv:2308.06721
28. 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, pp. 3836–3847 (2023)
29. Zhang, M., Yang, J., Xian, Y., Li, W., Gu, J., Meng, W., Zhang, J., Zhang, X.: Ag-sdm: Aquascape generation based on stable diffusion model with low-rank adaptation. Comput. Anim. Virt. Worlds **35**, e2252 (2024)
30. Zhang, S., Deng, X., Shao, H., Jiang, Y.: Impres: implicit residual diffusion models for image super-resolution. Visual Computer , 1–11 (2024b)

Xiaoyu Lv is currently a postgraduate at the Faculty of Digital Media Technology, Hangzhou Dianzi University. His research interests include generative models and procedural materials.

Zizhao Wu is currently an Associate Professor with the Faculty of Digital Media Technology, Hangzhou Dianzi University. He received the PhD degree from the Department of Computer Science and Technology, Zhejiang University, in 2013. His main research interests include computer vision and computer graphics.

Jiamin Xu is currently a Lecturer with the School of Computer Science, Hangzhou Dianzi University. He received the PhD degree from the Department of Computer Science, Zhejiang University, in 2022. His main research interests include 3D reconstruction, neural rendering, and image-based rendering.

Xiaoling Gu received the PhD degree in computer science from Zhejiang University, Zhejiang, China, in 2017. She is currently an Associate Professor with the School of Computer Science and Technology, Hangzhou Dianzi University, Hangzhou, China. Her research interests include computer vision, machine learning, and fashion data analysis. She was also an invited reviewer or a program committee member for top conferences and prestigious journals.



Ming Zeng received the PhD degree from the State Key Laboratory of CAD & CG, Zhejiang University. He is currently an Associate Professor with the School of Informatics, Xiamen University. He was a visiting researcher with Visual Computing Group, Microsoft Research Asia, from 2009 to 2011 and also in 2017. His research interests include computer graphics and computer vision, especially in human-centered analysis, reconstruction, synthesis, and animation.

Weiwei Xu is currently a Full Professor with the School of Computer Science, Zhejiang University. His research interests include differentiable physics-based simulation, high-level vision, 3D reconstruction, and image-based rendering.