Procedural Material Generation with Reinforcement Learning

BEICHEN LI, MIT CSAIL, USA and Adobe Research, USA
YIWEI HU*, Adobe Research, USA
PAUL GUERRERO, Adobe Research, UK
MILOŠ HAŠAN, Adobe Research, USA
LIANG SHI, MIT CSAIL, USA
VALENTIN DESCHAINTRE, Adobe Research, UK
WOJCIECH MATUSIK, MIT CSAIL, USA

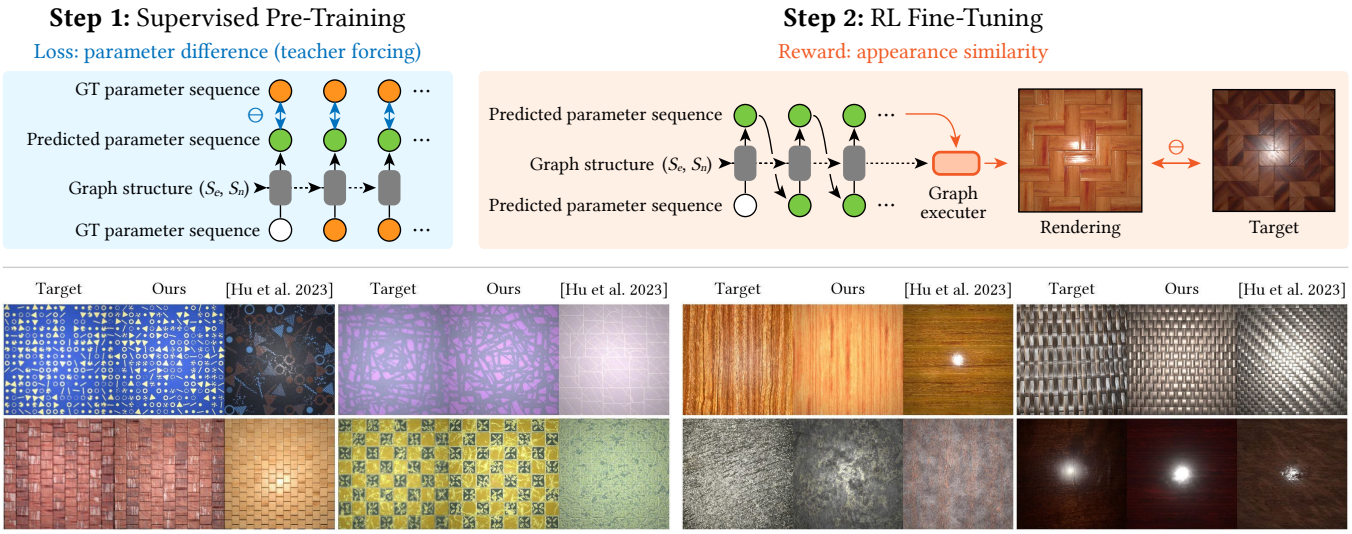

Fig. 1. Top: We present a reinforcement learning (RL) framework to improve the prediction accuracy of a parameter generator for procedural material graphs. We pre-train the parameter generator with synthetic data and fine-tune it on both synthetic and real images using a reward function based on appearance similarity. Bottom: We show rendered material appearances using parameters predicted by our RL-fine-tuned model and the supervised-learning-only model [Hu et al. 2023] without post-optimization (left: synthetic data; right: real data). Our model yields significantly better matches with the target images.

Modern 3D content creation heavily relies on procedural assets. In particular, procedural materials are ubiquitous in the industry, but their manipulation remains challenging. Previous work [Hu et al. 2023] conditionally generates procedural graphs that match a given input image. However, the parameter generation step limits how accurately the generated graph matches the input image, due to a reliance on supervision with scarcely available procedural data. We propose to improve parameter prediction accuracy for image-conditioned procedural material generation by leveraging reinforcement learning (RL) and present the first RL approach for procedural materials. RL circumvents the limited availability of procedural data, the domain gap between real and synthetic materials, and the need for end-to-end differentiable loss functions. Given a target image, we retrieve a procedural material and use an RL-trained transformer model to predict a set of parameters that reconstruct the target image as closely as possible. We show that using RL significantly improves parameter prediction to match a given target image compared to supervised methods on both synthetic and real target images.

CCS Concepts: • **Computing methodologies** → **Rendering**.

Additional Key Words and Phrases: Procedural materials, generative models, reinforcement learning

ACM Reference Format:
Beichen Li, Yiwei Hu, Paul Guerrero, Miloš Hašan, Liang Shi, Valentin Deschaintre, and Wojciech Matusik. 2024. Procedural Material Generation with Reinforcement Learning. *ACM Trans. Graph.* 43, 6, Article 280 (December 2024), 14 pages. https://doi.org/10.1145/3687979

*Primary mentor at Adobe

Authors' Contact Information: Beichen Li, beichen@mit.edu, MIT CSAIL, USA and Adobe Research, USA; Yiwei Hu, yiwhu@adobe.com, Adobe Research, USA; Paul Guerrero, guerrero@adobe.com, Adobe Research, UK; Miloš Hašan, mihasan@adobe.com, Adobe Research, USA; Liang Shi, liangs@mit.edu, MIT CSAIL, USA; Valentin Deschaintre, deschain@adobe.com, Adobe Research, UK; Wojciech Matusik, wojciech@csail.mit.edu, MIT CSAIL, USA.

ACM 1557-7368/2024/12-ART280
https://doi.org/10.1145/3687979

1 Introduction

Materials play an important role in virtual environment design. Modern 3D content creation workflows rely heavily on procedural

materials that define materials as outputs of procedures with large numbers of parameters and operations. Procedural material graphs provide particularly valuable advantages such as editability and unlimited resolution, but rely on significant expertise for creation and manipulation. Obtaining a desired material appearance may require careful adjustment of potentially hundreds of parameters.

As a result, some research has focused on automatically generating procedural materials. The problem of procedural material generation is a special case of program synthesis [Hu et al. 2023] as procedural materials are effectively functional programs. Given the extreme complexity of such programs, different approaches were proposed in recent years to either regress or optimize for the parameters of a given graph [Hu et al. 2019, 2022c; Li et al. 2023; Shi et al. 2020] or generate the graph alongside the parameters [Guerrero et al. 2022; Hu et al. 2023]. Nonetheless, regardless of how the graph structure is obtained, the estimated parameters play a key role in controlling the procedural graph's output texture, such as structure, color, etc. Existing parameter estimation methods, however, either rely on CNNs that need to be re-trained for each graph, or on a lengthy optimization requiring a few minutes for parameter fitting. On the other hand, graph generation approaches leverage transformer models [Vaswani et al. 2017] and a linearized representation of material graphs to generate sets of nodes, edges, and parameters, both in a conditional and unconditional setting [Guerrero et al. 2022; Hu et al. 2023]. However, to reach a given target appearance, these methods still rely on extensive random sampling and a costly post-optimization step as their parameter estimations are not accurate or robust enough. This amounts to several minutes of inference time, precluding their practical usage in responsive generative AI applications. We find the limitations of parameter generation in these methods partly due to the architecture used for conditioning, but mostly due to their supervised training approach. Supervised training is limited by the scarce availability of procedural material data and biases presented in the synthetic data. Only small sets of procedural materials are available for training, and there is a noticeable domain gap between these procedural material datasets and real materials. Additionally, supervised methods are trained with a parameter-space loss that does not necessarily correspond well to visual differences in material appearance.

To address these problems, we propose to leverage reinforcement learning (RL), presenting the first RL-based method to generate parameters for procedural materials that match the appearance given in a target image. Our use of RL is key to enabling two desirable properties: we can (1) directly maximize a reward function representing the visual similarity of procedural materials rather than their parameters, and (2) train on real material images, avoiding the scarcity and biases of procedural material datasets. Specifically, we follow a strategy inspired by RL from human feedback: first pre-train a parameter generator using supervised training, and then fine-tune it with RL. During RL training, we use the appearance similarity between the target image and the procedural material as the reward, allowing us to train on images of real materials directly.

Compared to the supervised-learning method [Hu et al. 2023] which relies on picking the best of 150 samples and a time-consuming post-optimization [Shi et al. 2020] to achieve a reasonable appearance match, we show that our RL-based training framework significantly improves the prediction accuracy of the parameter generator. Our method reaches a close appearance to the target image with much fewer samples and can even avoid the expensive post-optimization. We conduct several comparative evaluations to motivate the design choices in our two-stage (supervised and RL-fine-tuning) training pipeline. In summary, we enable faster, more accurate procedural material graph parameter generation through the following contributions:

- We propose the first RL training approach for procedural generative models, bridging the gap between generated tokens and image-space errors.
- Our setup is the first to introduce real material photographs as training data for procedural material generation.
- We show significantly better matching quality with substantially reduced inference time, unlocking the practical use-case of generative AI in procedural material modeling.

To facilitate reproducibility, we release the training and inference code on GitHub together with a parameter generator model pre-trained and fine-tuned on publicly available procedural materials.

2 Related Work

2.1 Material Generation and Acquisition

Related to our goal of conditional parameter generation for procedural material graphs are material generation and acquisition with analytical map representations. This has been a longstanding problem [Guarnera et al. 2016], which saw significant improvement leveraging CNNs for image to material acquisition with either flash [Deschaintre et al. 2018, 2019; Gao et al. 2019; Guo et al. 2021; Henzler et al. 2021; Vecchio et al. 2021; Ye et al. 2021; Zhou and Kalantari 2021] or environment-lit [Li et al. 2017; Martin et al. 2022] captured image(s). To compensate for the ill-posedness of material acquisition from few inputs, generative models were proposed for material generation and optimization for latent space control [Guo et al. 2020; Hu et al. 2022b; Vecchio et al. 2023a,b; Zhou et al. 2022, 2023]. While these approaches show impressive results, they target the creation of bitmap-based materials, which are memory-heavy and challenging to edit [Hu et al. 2022b].

In contrast, we target the generation of parameters for procedural materials, matching the overall desired appearance by leveraging recent progress in graph structure estimation and significantly improving parameters inference.

2.2 Procedural Material Generation and Optimization

Given how ubiquitous procedural materials are in the industry, their design has received significant interest in recent years. Various methods were proposed for material graph parameter estimation. In 2019, Hu et al. [2019] proposed to train a CNN per material graph, learning to regress the best parameters to match a photograph. As an alternative, Shi et al. [2020] made the graph execution partially differentiable, enabling gradient descent for parameter search. The set of differentiable nodes was later extended to pattern generators through differentiable proxies [Hu et al. 2022a], and later to support most operations typically used in material graphs [Li et al. 2023].

These methods, however, require ground-truth material graphs to be provided for training/optimization. Alternatively, different methods were proposed to create variations of a flexible graph [Hu et al. 2022c] using scribble-driven material segmentation and mask synthesis [Baldi et al. 2023; Guehl et al. 2020]. We take inspiration from MatFormer [Guerrero et al. 2022], and more specifically, its conditional version [Hu et al. 2023]. MatFormer proposed to linearize material graphs and leverage three transformers to generate graph nodes, edges, and parameters given text or image prompts. We inherit a similar transformer framework, but improve the conditioning approach and, more importantly, the training framework, introducing Reinforcement learning (RL) to the field of procedural material generation. In particular, RL enables the training to take appearance-space loss, leading to significant improvement in parameter estimation.

2.3 Fine-Tuning with Reinforcement Learning

RL has been used in various applications to fine-tune models that were initially trained with supervision. Unlike supervised training that requires ground truth targets, RL only needs a score (reward), which does not have to be differentiable, to evaluate the quality of outputs. As a consequence, RL-based fine-tuning is less limited by the amount of data available or by a need for technically convenient loss functions. This was demonstrated famously with RL from human feedback in InstructGPT [Ouyang et al. 2022], the paper that laid the foundation for ChatGPT [OpenAI 2022]. Other large language models (LLMs) have followed a similar approach [Lee et al. 2023b; Touvron et al. 2023]. RL fine-tuning has also been applied to code generation [Duan et al. 2023; Le et al. 2022; Shojaee et al. 2023] using various scores based on the presence of compile errors or results from runtime tests.

RL fine-tuning in vision and graphics. Several tasks in vision and graphics have been shown to benefit from dropping the need for ground truth data or differentiable score functions. Early work used RL to train a simple network that can retouch a photo in a sequence of interpretable steps [Hu et al. 2018] using an adversarial loss as a score function. Multiple vision tasks have been shown to benefit from RL fine-tuning [Pinto et al. 2023] using quality metrics that are typically used for evaluation but not for supervised training due to their non-differentiability. Recently, there has been an interest in the RL fine-tuning of text-to-image diffusion models [Black et al. 2023; Fan et al. 2023; Lee et al. 2023a] using human feedback and other non-differentiable scores. In the 3D generation domain, this can be used to improve multi-view consistency [Xie et al. 2023].

RL for procedural content generation. Procedural content can benefit significantly from RL due to the limited availability of procedural data. Several RL-based methods have been proposed to train procedural generators for virtual environments targeted at education or games [Gisslén et al. 2021; Khalifa et al. 2020; López et al. 2020; Mohaghegh et al. 2023]. A different set of methods tackle the more challenging task of generating procedures for 2D or 3D shapes that reconstruct given target outputs [Laidlaw et al. 1986; Sharma et al. 2018] using RL, or propose alternatives to RL [Jones et al. 2022], but operate on relatively simplistic domains. We focus on industrial-grade procedural materials, which have a significantly more complex representation compared to the above works. Materials are represented as functional programs with a large number of operations, complex parameters, and strong dependencies between operations. This requires careful dataset construction to make the most of limited data, a good reward definition, a more advanced RL approach [Ramamurthy et al. 2022] that uses a value network for increased robustness, and capable generators specialized to procedural materials [Hu et al. 2023] for both the policy network and the value network.

3 Background

A procedural material is a functional program that takes a set of heterogeneous parameters as input and outputs SVBRDF material maps like albedo, normal, roughness, and metallicity. The program is represented as an acyclic computation graph $g = (\mathcal{N}, \mathcal{E})$ with *nodes* $\mathcal{N} = (n_1, n_2 \dots)$ and *edges* $\mathcal{E} = (e_1, e_2, \dots)$. Each node n_i is either a noise/pattern generator or an image processing function, and edges describe the dataflow of intermediate texture maps between these generators and functions. Each node implements an image processing function f_i that takes as input zero or more images I^*_{in} and processes them into one or more output images $I^+_{out} = f_i(I^*_{in}, P_i)$, where $P_i = (p^i_1, p^i_2, \dots)$ are the function parameters. Each edge e_i connects an output of a node to an input of another node. To generate complex material graphs, prior work [Guerrero et al. 2022; Hu et al. 2023] leverages transformers [Vaswani et al. 2017] to generate a linearized version of the graph. Specifically, the graph is produced in three sequences S_n, S_e, and S_p representing respectively nodes, edges, and parameters. These sequences are generated using three separate transformers p_θ, p_ϕ and p_ψ, either unconditionally from a distribution $p(g)$ [Guerrero et al. 2022] or conditioned on a text or image y, $p(g|y)$ [Hu et al. 2023]. For conditional generation, each transformer is conditioned as follows:

$$p(g|y) \coloneqq p_\theta(S_n|y)\; p_\phi(S_e|S_n, y)\; p_\psi(S_p|S_e, S_n, y), \tag{1}$$

where θ, ϕ and ψ are the trainable parameters of each transformer.

In this work, we focus on the parameter generator p_ψ. As a transformer network, p_ψ models the joint probability of the parameter sequence S_p as product of conditional probability for each token s_i of the sequence, conditioned on all previously generated tokens $s_{<i}$:

$$p_\psi(S_p|S_e, S_n, y) \coloneqq \prod_i p_\psi(s_i|s_{<i}, S_e, S_n, y). \tag{2}$$

In each step, the parameter estimator p_ψ outputs the probability distribution for the token s_i given the previous tokens, the graph structure, and the prompt. This probability distribution can be sampled to obtain the next token s_i.

To train the parameter generator, previous work relies on supervised training, where the ground truth token of each parameter is used to compute a parameter error. This approach has two main shortcomings: high-quality ground-truth data is scarce and expensive to create, and the parameter error does not always correlate well with the perceptual error of the resulting material. This frequently results in color or structural mismatch when trying to match a given target appearance. To mitigate the impact of this limitation. Hu et al. [2023] rely on an additional post-processing optimization

step [Shi et al. 2020], which further optimizes the continuous function parameters of the material for a better perceptual match. The optimization step is applied to the top 5 samples among 150 random graph samples to select the best result. However, each optimization process typically takes up to several minutes per graph sample, making it impractically expensive compared with the cost of transformer predictions. In our work, we aim to lift this fundamental efficiency bottleneck by improving the quality of the generated material graphs. Specifically, we focus on improving parameter generation $p_\psi(S_p|S_e, S_n, y)$ assuming node sequences S_n and edge sequences S_e are known.

4 Method

Our goal is to train a parameter generator p_ψ to predict a parameter sequence S_p that better matches a target appearance, given as a target image I, than previous approaches. As we identify the parameter-space loss as a main bottleneck in current methods, we use an appearance-space loss (reward) instead to measure the perceptual match of the resulting image to a target. Producing tokens representing a program using a transformer-based model, then executing the program and producing an image, is a fundamentally non-differentiable sequence of operations. Thus, our key motivation for using Reinforcement Learning (RL) is that it can handle any non-differentiable reward function.

Our training approach therefore breaks into two steps as shown in Fig. 1: (1) We first pre-train the parameter generator with supervised learning (Sec. 4.2) using augmented synthetic data (Sec.5.1) for which ground-truth is available. (2) We then leverage RL to further fine-tune the parameter generator using appearance similarity as the reward function (Sec. 4.3). The implementation details are provided in Sec. 4.4 and the supplementary material.

4.1 Parameter Generator

For the parameter generator p_ψ, we use the same transformer architecture as in previous work [Hu et al. 2023], but change the conditioning mechanism and reduce the network capacity. More specifically, p_ψ uses the CLIP embedding y of the target image I for conditioning. In previous work, the embedding y is added to the latent state vector. Instead, we leverage cross-attention from the latent state vector to the CLIP embedding as the conditioning mechanism. This allows the parameter generator to obtain more detailed information about the target image. Additionally, we reduce the number of parameters by setting the hidden state dimension, the number of layers, and the number of attention heads to 48/4/4 for the node encoder and 96/4/4 for the parameter decoder as we found that it helps prevent overfitting.

4.2 Supervised Pre-Training

While RL does not necessarily require supervised pre-training, our parameter prediction setting is complex with potentially over 1,000 steps and a large pool of possible parameter values to choose from in each step. Exploring this space from scratch with only the appearance as guidance would be challenging. Thus, we found it beneficial to pre-train our parameter generator using augmented synthetic material graph data (Sec. 5.1). We pre-train the parameter generator through teacher forcing using an Adam optimizer and a 1e-4 learning rate. We apply gradient norm clipping (1.0) and a cosine learning rate schedule with linear warm-up and train on 8 NVIDIA A10g GPUs for 10 epochs. Combining the architecture and conditioning modifications, we observe a small improvement in the validation loss for supervised training compared to previous work [Hu et al. 2023], where the average cross-entropy loss per token declines from 0.803 to 0.770. For a fair comparison, we evaluate [Hu et al. 2023] using the updated architecture and training method.

4.3 Reinforcement Learning by Program Execution

After pre-training, we fine-tune the parameter estimator p_ψ using RL based on an appearance-space reward. Inspired by the recent success of LLM fine-tuning from human feedback [Ouyang et al. 2022], we use Proximal Policy Optimization (PPO) [Schulman et al. 2017], where a *policy network* is trained to perform *actions* that maximize an accumulated per-action *reward*. In our setting the policy network is the parameter generator $p_\psi(s_i|s_{<i}, S_e, S_n, y)$ and an action is an evaluation of the generator, which predicts the probability distribution for the next token in the parameter sequence given all previous tokens. To simplify notations, we use "$\cdot$" in short for $(s_{<i}, S_e, S_n, y)$ in subsequent equations.

The goal of PPO is to find the network parameters ψ that maximize the *expected accumulated reward* $J(p_\psi)$:

$$J(p_\psi) \coloneqq \mathbb{E}_{S_p \sim p_\psi}\left[\sum_{i=1}^{N} \gamma^i R(s_i, \cdot)\right], \tag{3}$$

where $S_p = (s_1, \ldots, s_N)$ is the parameter sequence and $\gamma \in (0, 1)$ is a *discount factor* that emphasizes earlier parameter tokens. The step-wise reward $R(s_i, \cdot)$ is computed for each token prediction s_i and is based on the appearance similarity between the target image I and the material resulting from the predicted parameters. We define the reward in detail further below. It can be shown [Schulman et al. 2017] that the gradient of $J(p_\psi)$ w.r.t the generator parameters ψ (also known as the *policy gradient*) is:

$$\nabla_\psi J(p_\psi) = \mathbb{E}_{S_p \sim p_\psi}\left[\sum_{i=1}^{N} \gamma^i R(s_i, \cdot) \nabla_\psi \log p_\psi(s_i|\cdot)\right]. \tag{4}$$

The expected value is approximated by averaging over parameter sequences randomly sampled from our generator and the resulting gradient is used to update our parameter generator.

Reward function. Our reward encourages similar appearances between the target image I and the material resulting from a parameter sequence S_p:

$$R_{\text{App}}(S_p, S_e, S_n, I) \coloneqq 1 - d(h(S_p, S_e, S_n), I), \tag{5}$$

where h is a non-differentiable executer of the graph defined by $h(S_p, S_e, S_n)$, and I is the target image. The function d defines an appearance difference between two images:

$$\begin{aligned} d(I_x, I_y) \coloneqq \lambda_0 ||GM(I_x) - GM(I_y)||_1 + \lambda_1 ||I_x^L - I_y^L||_1 \\ + \lambda_2 ||I_x^{\text{ab}} - I_y^{\text{ab}}||_1, \end{aligned} \tag{6}$$

where I_x and I_y are images and GM is an operator that computes the Gram matrices of extracted VGG features [Gatys et al. 2015]. I^L

and I^{ab} are the L* channel and a*b* channels of the image in the L*a*b* color space. λ_i $(i = 0, 1, 2)$ are adjustable weights to balance the contribution from color and texture statistics.

The appearance reward R_{App} is sparse in the sense that it is only available at the last step once the entire parameter sequence is generated. For all the steps, we follow [Wu et al. 2021] and impose a negative regularization reward $R_{\text{KL}}(s_i, \cdot)$ so that our generator p_ψ doesn't deviate excessively from the pre-trained generator p^*_ψ:

$$R_{\text{KL}}(s_i, \cdot) \coloneqq -\beta \text{KL}\left(p_\psi(s_i|\cdot) \parallel p^*_\psi(s_i|\cdot)\right), \tag{7}$$

where KL denotes the KL-divergence between parameter token probability distributions. The final step-wise reward is given by:

$$R(s_i, \cdot) = \begin{cases} R_{\text{App}}(s_i, \cdot) + R_{\text{KL}}(s_i, \cdot), & i = N; \\ R_{\text{KL}}(s_i, \cdot), & i < N. \end{cases} \tag{8}$$

Note that this reward only uses the appearance reward for the last token. Next, we describe how we propagate the reward of the last token to previous tokens to update p_ψ at earlier steps.

Advantage and value network. Previous work has shown [Schulman et al. 2016] that using the *advantage* of each step, i.e. how much better a given token is compared to alternatives that could have been taken in the same step, is preferable to the reward. In our case, the advantage also effectively propagates the appearance-based reward of the last token to all other tokens in the sequence. The advantage is derived as

$$A_i^{p_\psi}(s_i, \cdot) = R(s_i, \cdot) + \gamma V_{i+1}^{p_\psi} - V_i^{p_\psi}, \tag{9}$$

where $V_i^{p_\psi}$ is the *value* of the partial parameter sequence $s_{<i}$, defined as the expected accumulated reward (or the expected *return*) when completing the partial sequence $s_{<i}$ with our generator:

$$V_i^{p_\psi} \coloneqq \mathbb{E}_{s_{\geq i} \sim p_\psi}\left[\sum_{t=i}^{N} \gamma^{(t-i)} R(s_t, \cdot)\right], \tag{10}$$

where $s_{\geq i}$ denotes the sequence $(s_i, \ldots, s_N)$. The advantage $A_i^{p_\psi}(s_i, \cdot)$ from Eq. 9 then replaces the reward $R(s_i, \cdot)$ in policy gradient calculation (Eq. 4). Note how the advantage at a token s_i is based on estimated rewards for all future tokens, including the last token s_N which has our appearance-based reward.

Approximating the expectation in Eq. 10 with multiple samples is expensive since it is used in the inner loop of Eq. 4. Instead, the value is typically approximated with a *value network* v_ρ trained simultaneously with p_ψ. We follow standard practice and employ generalized advantage estimation (GAE) [Schulman et al. 2015] to approximate advantage scores, which provides a more effective trade-off between bias and variance than directly using Eq. 9.

PPO objective. The PPO algorithm maximizes an objective function to jointly train the policy network p_ψ and the value network v_ρ. The core intuition is to perform iterative policy updates in smaller steps so that the training process more likely converges to an optimal solution. We follow the canonical PPO objective function in [Schulman et al. 2017]:

$$L(\psi, \rho) = \hat{\mathbb{E}}_i\left[L_i^{\text{CLIP}}(\psi) - c_1 L_i^{\text{VF}}(\rho) + c_2 S_i(\psi)\right]. \tag{11}$$

The first component is the clipped surrogate objective function,

$$L_i^{\text{CLIP}}(\psi) = \min\left(r_i(\psi)\hat{A}_i, \text{clip}(r_i(\psi), 1-\epsilon, 1+\epsilon)\hat{A}_i\right), \tag{12}$$

$$r_i(\psi) = \frac{p_\psi(s_i|\cdot)}{p_{\psi_{\text{old}}}(s_i|\cdot)}, \tag{13}$$

which constrains policy changes by clipping the probability ratio $r_i(\psi)$ between the current policy p_ψ and the previous policy $p_{\psi_{\text{old}}}$. $\hat{A}_i$ is short for the empirical estimate of $A_i^{p_{\psi_{\text{old}}}}(s_i, \cdot)$ from GAE. Then, the second loss component supervises value predictions using expected returns.

$$L_i^{\text{VF}}(\rho) = \left(v_\rho(\cdot) - V_i^{\text{target}}\right)^2. \tag{14}$$

With GAE, the target value combines advantage and value estimates from the previous policy and value networks.

$$V_i^{\text{target}} = \hat{A}_i + v_{\rho_{\text{old}}}(\cdot). \tag{15}$$

Finally, the third loss component $S_i(\psi)$ is the entropy of the token prediction at s_i, which serves as a bonus to encourage sufficient exploration.

4.4 RL Implementation

We base our RL framework on the Reinforcement Learning for Language Models (RL4LMs) library [Ramamurthy et al. 2022] and adjust it to our application. We inherit the value network architecture from the parameter generator p_ψ and initialize it with the pre-trained parameters p^*_ψ except for the classification head, which is replaced by a randomly initialized linear regression layer. Each training epoch comprises the following steps:

- Sample batches of entire parameter token sequences, i.e., *trajectories*, using the current policy p_ψ and fill a replay buffer collecting all state-action transitions in the trajectories.
- Train the policy network p_ψ and the value network v_ρ by sampling mini-batches of trajectories from the replay buffer, computing PPO's objective function, and updating network weights via gradient backpropagation.
- Clear the replay buffer and continue.

This workflow is illustrated in Fig. 2. Additional implementation details, including hyperparameter values, are available in the supplementary document.

5 Dataset

We train our model with two datasets, a synthetic dataset (Sec. 5.1) used entirely for pre-training and partially for the RL fine-tuning step, and a real photograph dataset (Sec. 5.2) used only for the RL fine-tuning step.

5.1 Synthetic Dataset

We generate our synthetic dataset using the same 4,667 ground-truth procedural material graphs as previous work [Guerrero et al. 2022; Hu et al. 2023], but augment it to guarantee that our initial parameter generator has completely converged and cannot benefit from additional synthetic data in a supervised setting. In particular, we extensively sample parameter combinations for each graph, encouraging graphs with rare nodes to be sampled more in an attempt

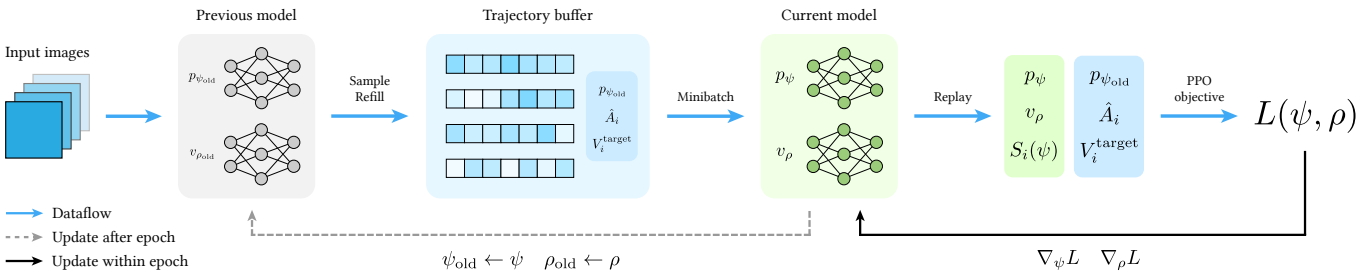

Fig. 2. Workflow of our RL implementation. Each epoch begins with the previous model randomly sampling parameter token sequences (trajectories) and refilling a trajectory buffer. We then extract mini-batches of trajectories from the buffer and replay them to obtain the latest next-token probability and value predictions. The predictions are used to compute the PPO objective and update the current model through gradient backpropagation. The current model replaces the previous model at the end of each epoch.

to better balance the dataset. With exhaustive sampling, we create 10 million image-to-graph pairs, maximizing the usage of the original procedural materials. We use 90% of this dataset for training and a 5%/5% split for validation and testing. The split is done at the graph structure level and ensures graphs seen during training cannot arise in the validation/test set. This dataset is principally used in the supervised pre-training of our policy network. We combine this synthetic dataset with a dataset of real photographs described below for RL fine-tuning. We found training with this mixed dataset performs well on both real and synthetic test images.

5.2 Real Photograph Dataset

As our RL approach uses an image-space reward function, which is estimated from the sampled trajectories yielded by the parameter generator, it does not require paired image-to-graph ground-truth data for training. We can therefore train our model using real photographs as image prompts. The real photograph dataset, PhotoMat, contains over 10,000 fronto-parallel flash-lit material surface images captured by smartphones [Zhou et al. 2023]. We divide this dataset into 90%/5%/5% splits for training, evaluation, and testing. Since we assume known node and edge sequences in our experiments, given a photograph, we retrieve the K-nearest-neighbor graphs from the rendered images in the synthetic dataset [Hu et al. 2023; Yan et al. 2023] using CLIP [Radford et al. 2021] embeddings. During training, we retrieve $K = 5$ graphs and randomly use one as S_n and S_e to sample the parameter sequence S_p. We create a combined dataset in which we draw samples with a chance of 50% from either synthetic dataset or real photograph dataset.

We fine-tune the parameter generator with RL using 8 GPUs for policy sampling and 96 CPU cores to evaluate the reward model. On an NVIDIA A100 40GB node, RL fine-tuning lasts for 800 epochs and finishes in one day.

6 Evaluation

We evaluate the RL-trained parameter transformer model p_ψ in terms of appearance similarity to a given target image after parameter prediction and program execution. We demonstrate performance improvement on both synthetic and real images (photographs) over the state-of-the-art method [Hu et al. 2023] on direct parameter prediction (Sec. 6.1, 6.2), with or without a state-of-the-art post-optimization method, MATch v2 [Li et al. 2023]. The graph structures (i.e., S_n and S_e) are assumed to be known for synthetic data and retrieved from our graph dataset for real data. To ensure a fair comparison, we retrain the [Hu et al. 2023] model using our augmented synthetic dataset, then evaluate both methods on the same ground-truth or retrieved graph structures. We further test RL fine-tuning using generated graph structures from [Hu et al. 2023] to evaluate the impact of graph structure retrieval (Sec. 6.3).

For the remaining subsections, we validate a few design choices of RL through ablation studies (Sec. 6.4), where we show evaluation results using only synthetic or real data for fine-tuning, without pre-training, or without the L*a*b* loss term. We also compare our method against alternative RL settings that use rewards instead of advantages or involve different reward combinations (Sec. 6.5). Finally, we analyze the trade-off between matching quality and time consumption for our method and discuss the implications for its deployment in real-time gen-AI applications (Sec. 6.6).

6.1 Qualitative Results

We qualitatively compare our method with [Hu et al. 2023] on both synthetic and real images, demonstrating a significant improvement in appearance matching. Each result figure comprises input images alongside material renderings using the predicted parameters from different methods.

Fig. 3 shows the performance of our model on the synthetic test set (Sec. 5.1). Given a synthetic image and its ground-truth graph structure, we sample 5 parameter sequences and take the best result in terms of appearance similarity. The rendered procedural graphs with sampled parameters from our RL-fine-tuned model match the target image better than [Hu et al. 2023] even without post-optimization or extensive sampling.

We further evaluate our approach to match real material photographs. This is particularly challenging for procedural material modeling as (1) the appearance distribution of real photos is more diverse than synthetic material renderings and (2) we do not have the ground-truth graph structure. Our RL training using real data partially resolves the first challenge and we find reasonably matching graph structures via retrieval from our material graph database. More specifically, we retrieve 5 graphs from the database based on the cosine similarity of CLIP image embeddings and sample 5

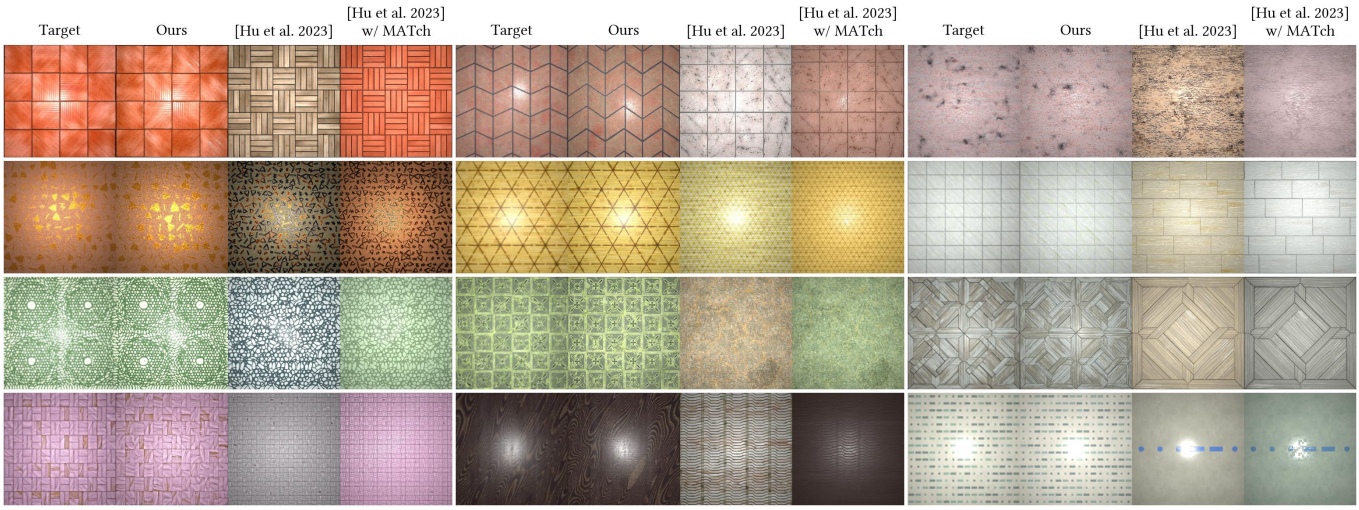

Fig. 3. Qualitative evaluation on the synthetic dataset. We evaluate the appearance similarity between the target images and the generated graphs. Our RL-fine-tuned model predicts more accurate parameters, resulting in better structure and color/roughness matching compared to [Hu et al. 2023]. Even with MATch post-optimization, the structure mismatching from [Hu et al. 2023]'s predictions cannot be fixed due to the non-differentiability of relevant parameters.

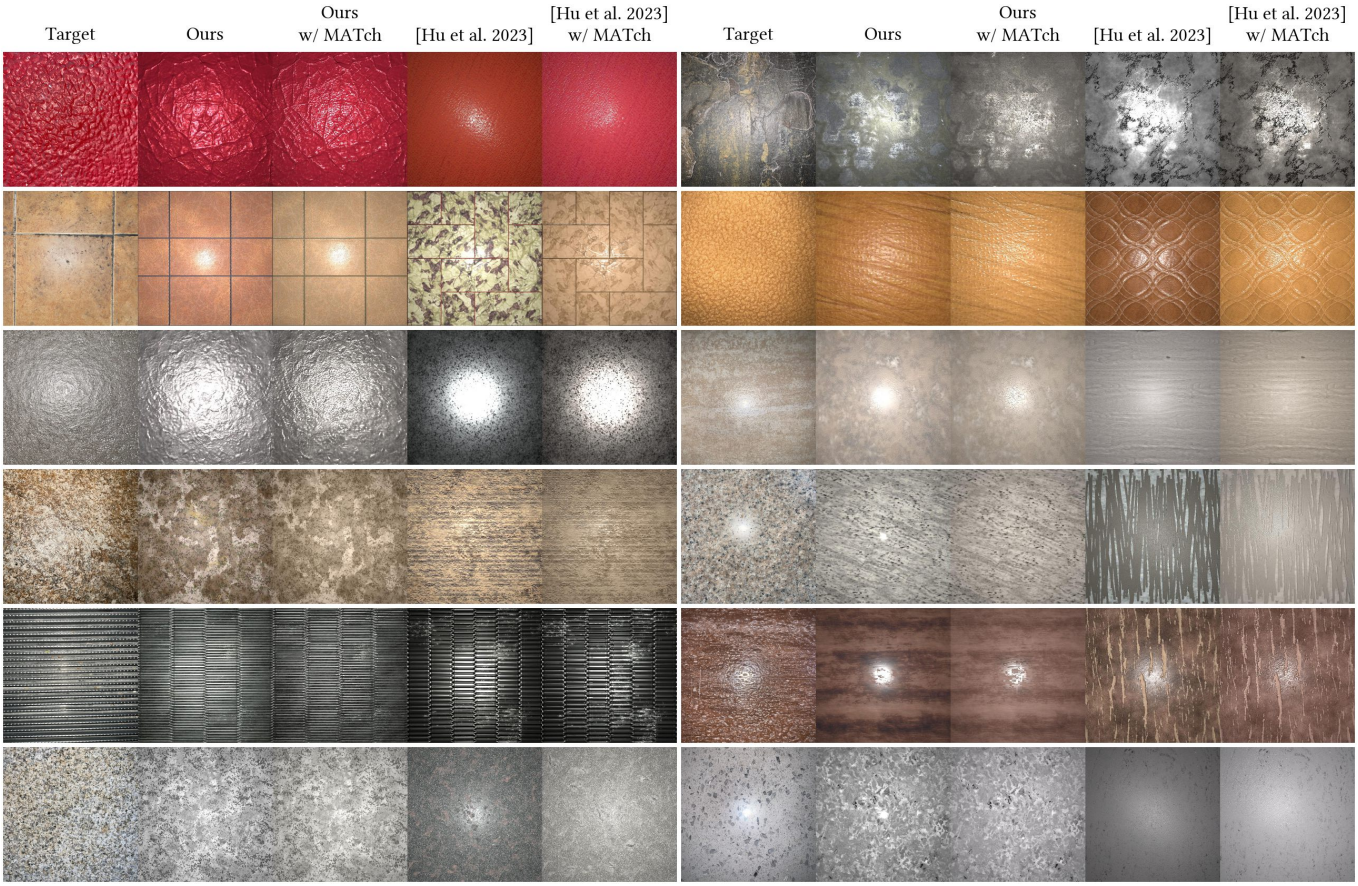

Fig. 4. Qualitative evaluation on the real photograph dataset (top-1 results). RL fine-tuning with real photographs reduces the domain gap between synthetic and real images. Thus, our predictions are much closer to the targets than [Hu et al. 2023] which is only pre-trained on synthetic data. Our model achieves decent appearance similarity even without MATch post-optimization, while [Hu et al. 2023] depends on MATch to improve matching quality.

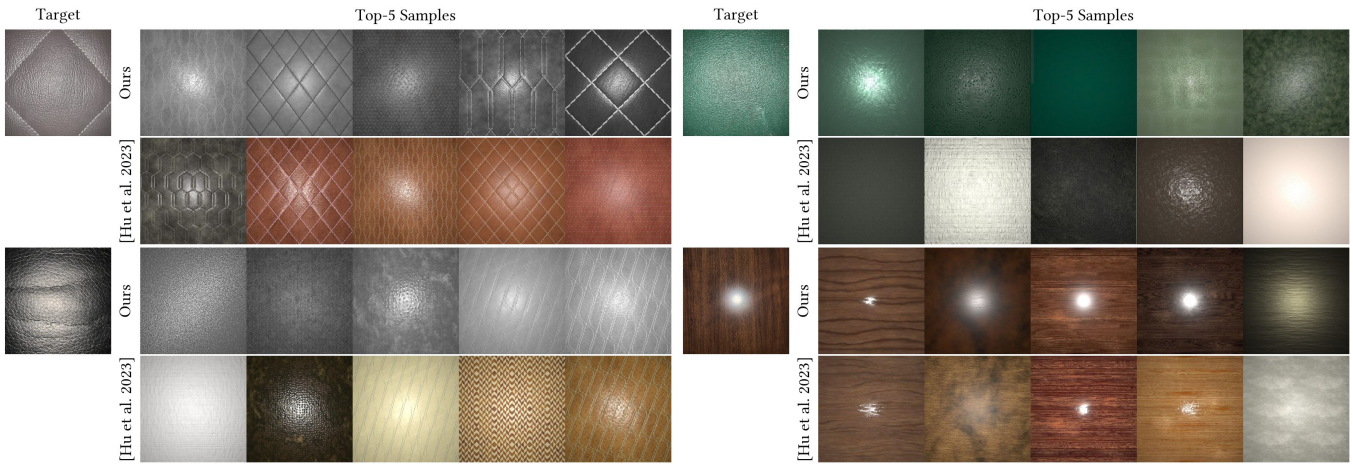

Fig. 5. Qualitative evaluation on the real photograph dataset (top-5 results). We present the top results generated by our RL model. All the samples match reasonably with the target photographs and are noticeably better than [Hu et al. 2023] on average. For the supervised-only model, the match quality decreases dramatically for lower-ranking samples.

Table 1. Quantitative evaluation on the synthetic test set. For Reward, higher is better; for LPIPS, lower is better. Notice that our method, even without post-optimization, already outperforms previous work with MATch optimization, which verifies the high accuracy of our generated parameters.

Method	w/o MATch		w/ MATch	
	Reward↑	LPIPS↓	Reward↑	LPIPS↓
[Hu et al. 2023]	0.649	0.503	0.895	0.490
Ours	**0.904**	**0.242**	**0.928**	**0.347**

Table 2. Quantitative evaluation on the test set of real photographs. Our results are constantly superior to previous work. For the LPIPS metric, our results without optimization yield better performance than previous work with optimization.

Method	w/o MATch		w/ MATch	
	Reward↑	LPIPS↓	Reward↑	LPIPS↓
[Hu et al. 2023]	0.827	0.620	0.901	0.574
Ours	**0.847**	**0.571**	**0.912**	**0.555**

parameter sequences for each graph. We apply the same sampling strategy for Hu et al. [2023] against which we compare. For qualitative comparison, we report the top-1 generated results in Fig. 4 and provide the top-5 generated results in Fig. 5. The results from our RL-trained model (both top-1 and top-5) show a substantial improvement in appearance matching quality thanks to the reduced domain gap between training and testing images. Without post-optimization, previous work performs poorly on a limited sample budget (5 samples only) and non-top-ranked graph structures.

Despite predicting parameters for the same set of graph structures, our RL model predicts more accurate parameters for the essential noise/pattern generator nodes in these graphs, resulting in better structure matching for both synthetic and real images. While post-optimization can improve color or roughness matching for [Hu et al. 2023], mismatches in the material structure are much harder to remediate as most noise/pattern generator nodes are non-differentiable.

More visual comparisons can be found in the supplementary document.

6.2 Quantitative Evaluation

To showcase that the RL model performs constantly well on a wide range of image inputs, we report two metrics that measure the appearance similarity: 1) Reward as defined in Eq. 5, representing both texture statistics and color match; (2) LPIPS [Zhang et al. 2018], typically used as a perceptual distance metric for generic images.

Synthetic Data. We report the comparison on synthetic data in Table 1. The metrics are computed over 2,330 images from our synthetic test set, where we choose 10 random appearance variations for each graph structure. All results are generated from top-1 samples out of 5 sampled parameter sequences (same as Sec. 6.1). We note that the LPIPS score of our method becomes worse after MATch optimization as the reward function has a different landscape from LPIPS despite both being derived from VGG features. Still, our scores are constantly better than [Hu et al. 2023].

Real Data. We report the comparison on real data in Table 2. In this case, we use the entire test set of PhotoMat including 582 real images. We apply the same sampling strategy described in Sec. 6.1, i.e., 5 random parameter samples per graph, to 5 nearest neighbor graphs. The quality is measured using top-1 samples as in Fig. 4.

For both synthetic and real data, our RL-trained model outperforms [Hu et al. 2023] in both reward and LPIPS scores, although the advantage is more evident on synthetic data. Indeed, real material photos are significantly harder to match as they cover a wider appearance distribution than our synthetic material dataset. To a lesser extent, they exhibit rich variation in lighting and camera effects whereas our procedural material renderer adopts a fixed setup. Nonetheless, our results *without optimization* show similar or even better quality than [Hu et al. 2023] *with optimization*. This allows our approach to rely less on time-consuming post-optimization as the parameter predictions are more accurate.

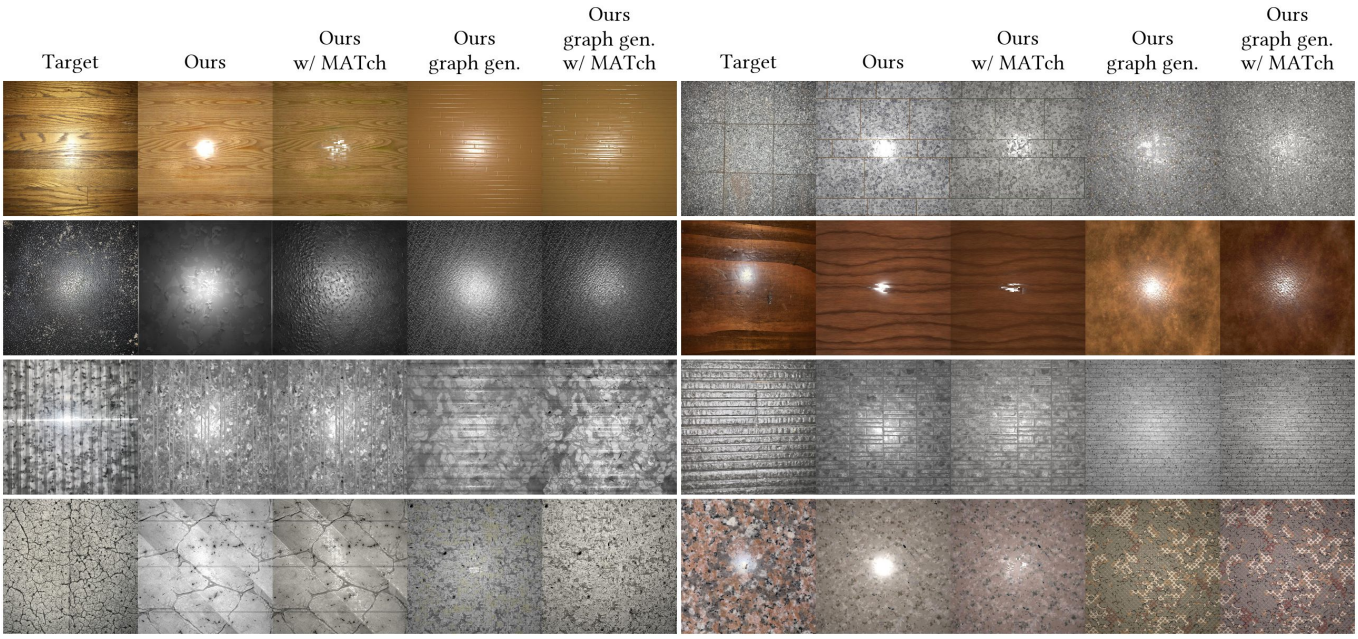

Fig. 6. Parameter prediction for the real photograph test set after fine-tuning the parameter generator using retrieved (ours) or predicted graph structures (ours graph gen.), with and without MATch post-optimization. In many cases (Row 1, Row 2 Column 2, and Row 4), the results from generated graphs show worse matching in material structure, especially after MATch optimization.

Table 3. Quantitative comparison between RL fine-tuning using graph structures retrieved from the synthetic dataset (ours) or predicted by the node and edge generators in [Hu et al. 2023] (ours w/ graph gen.). The metrics are computed for the real photograph test set.

Method	w/o MATch		w/ MATch	
	Reward↑	LPIPS↓	Reward↑	LPIPS↓
Ours w/ graph gen.	**0.852**	0.601	0.893	0.578
Ours	0.847	**0.571**	**0.912**	**0.555**

6.3 Graph Retrieval vs. Generation

We investigate whether incorporating more diverse graph structures during fine-tuning improves generalization to unseen real images. To test this idea, we pre-train the node and edge generators (p_θ, p_ϕ) in [Hu et al. 2023] with our augmented synthetic dataset and randomly sample S_n and S_e for real images during RL fine-tuning. p_θ and p_ϕ remain fixed in the process and only serve as sources of domain randomization for the parameter generator p_ψ. Thus, they do not contribute to the discounted reward in Eq. 3.

We compare our model with its variant trained on generated graph structures qualitatively and quantitatively. For each real image, we apply our model using the sampling method described in Sec. 6.1 and perform MATch optimization on the highest-ranked generated result. In contrast, we randomly generate both graph structures and parameters for the variant model under an identical budget of 25 samples. We report reward and LPIPS scores over the real photograph test set with and without MATch optimization in Table 3. Visual comparison results are included in Fig. 6.

We notice that domain randomization with pre-trained node and edge generators does not consistently improve the matching quality. The variant model shows worse metrics than our model except for a marginal lead in reward without MATch optimization. As also illustrated in Fig. 6, the generated graph structures are occasionally preferable to retrieved ones (e.g., more accurate beam patterns in Row 3) but inferior in other cases (e.g., Row 1 and Row 4). The suboptimal graph structure quality becomes more prominent after MATch optimization as it is non-trivial to correct structural mismatches with differentiable parameter optimization (see the discussion in Sec. 6.1). This contributes to a lower reward score with MATch than our model. Therefore, empirically, our graph retrieval strategy demonstrates comparable expressiveness to the state-of-the-art graph generation model without imposing additional restrictions.

6.4 Ablation Study

We conduct ablation studies to verify our choice of hybrid dataset composition, two-step training strategy, and L*a*b* color regularization in the reward function.

Choice of dataset. To validate the choice of a 50-50 mixed training dataset, we show the prediction accuracy on synthetic and real photograph test sets using RL-fine-tuned models trained on different dataset compositions (synthetic data only, real data only, mixed dataset) in Table 4. A visual comparison is provided in Fig. 7. We found that training with only synthetic data does not generalize well to real data (bottom table), highlighting the importance of introducing real data in the training through RL. Further, incorporating real data in training also helps improve parameter estimation for synthetic data (top table). We select the mixed dataset as our default option as it demonstrates better generalization across both synthetic and real test datasets.

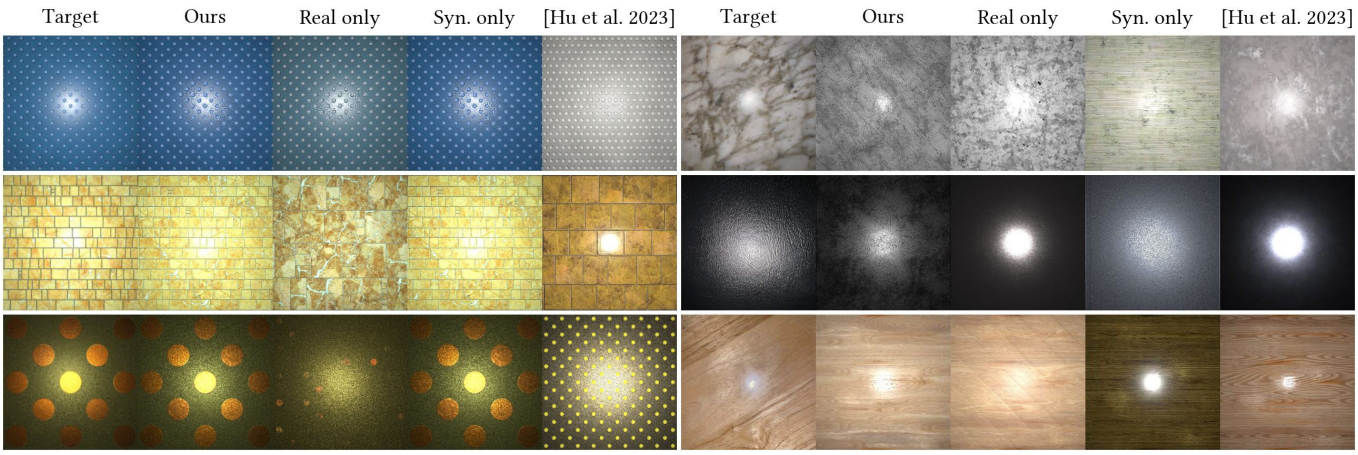

Fig. 7. Ablation study: choice of dataset. We experimented with RL fine-tuning on different dataset compositions including a mixed dataset (Ours), real photographs only (Real only), and synthetic data only (Syn. only). The examples in the left column are from the synthetic test set, while the ones in the right column are from the real photograph test set. The results from [Hu et al. 2023] (without post-optimization) are provided for reference. We notice that training only with synthetic data fails to produce good matching for real photographs, while interestingly, the model trained only on real data shows reasonable generalization to synthetic data. Our choice of hybrid dataset composition performs best on both test sets.

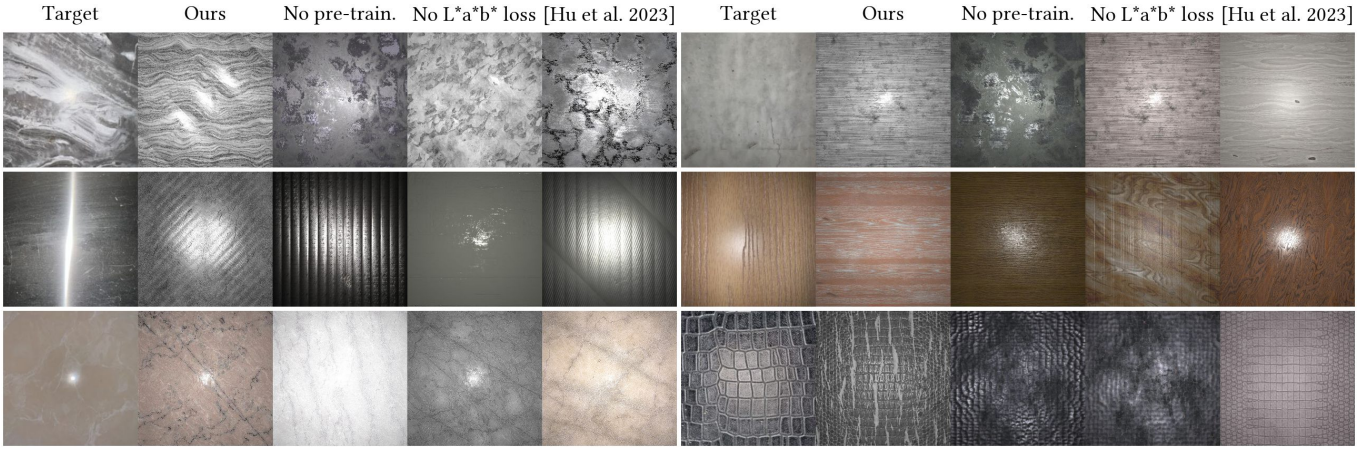

Fig. 8. Ablation study: pre-training and loss function. The appearance matching quality degrades without pre-training or the L*a*b* loss term, causing color or pattern mismatching. All input images are real photographs. The results from [Hu et al. 2023] (without post-optimization) are provided as a reference.

Table 4. Ablation study: choice of dataset. We test the prediction quality on the synthetic test set (top) and the real photograph test set (bottom) using RL models trained on different dataset compositions (synthetic data only, real data only, and our mixed dataset).

Synthetic test set	Reward↑	LPIPS↓
[Hu et al. 2023]	0.649	0.503
Ours w/ synthetic data only	0.897	0.249
Ours w/ real data only	0.854	0.311
Ours	**0.904**	**0.242**
Real photograph test set	Reward↑	LPIPS↓
[Hu et al. 2023]	0.827	0.620
Ours w/ synthetic data only	0.819	0.581
Ours w/ real data only	**0.850**	0.574
Ours	0.847	**0.571**

Table 5. Ablation study: pre-training and loss function. The metrics are collected from the real photograph test dataset. The matching quality worsens without pre-training or the L*a*b* component in the reward function.

Method	Reward↑	LPIPS↓
[Hu et al. 2023]	0.827	0.620
Ours w/o pre-training	0.828	0.580
Ours w/o L*a*b* loss	0.837	0.581
Ours	**0.847**	**0.571**

Pre-training and loss function. We evaluate how the pre-training step and the L*a*b* color term in the loss function affect RL performance. Table 5 and Fig. 8 show that removing these two components results in worse appearance matches with input images.

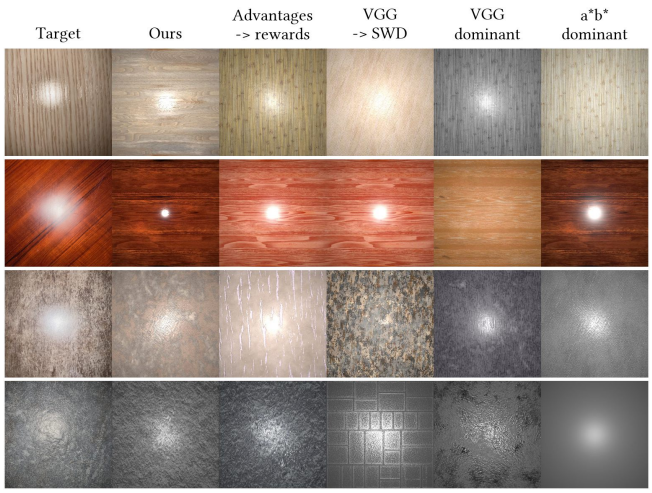

Fig. 9. We compare our method with several alternative RL settings including estimating returns using rewards instead of advantages (advantages -> rewards), replacing the VGG loss term with sliced Wasserstein distance (VGG -> SWD), or making the VGG or a*b* color term dominant in the reward function. All evaluation uses real test images. The other approaches are more prone to color and structure mismatches.

6.5 Alternative RL Settings

We inspect several technical details of the RL algorithm by validating our model against the following alternative RL settings:

- *Replacing advantages with rewards.* Estimate the expected return for each prediction step only using discounted rewards from the sampled trajectory instead of the advantage score (Eq. 9). The observed rewards also provide direct supervision for the value network.
- *Replacing VGG loss.* Measure differences in texture statistics using sliced Wasserstein distance (SWD) [Heitz et al. 2021] instead of the Gram matrices of VGG features.
- *Dominant VGG loss,* where we set $\lambda_0 = 20$, $\lambda_1 = 0.002$, and $\lambda_2 = 0.01$ in the reward function to enforce better matches of VGG features.
- *Dominant a*b* color loss,* which uses $\lambda_0 = 0.5$, $\lambda_1 = 0.006$, and $\lambda_2 = 0.06$ to strengthen color regularization.

Qualitative comparison results are shown in Fig. 9. Our method demonstrates a more robust matching quality on unseen real images than the alternative RL settings. Specifically, we find that overemphasizing the VGG reward term may lead to severe color mismatches, while excessively weighting the a*b* reward term may compromise the generated material structure.

6.6 Matching Quality and Time Cost

Our RL fine-tuning approach enables fast and accurate parameter prediction, promoting the practical use of generative AI for procedural material modeling. [Hu et al. 2023] achieves high-quality procedural material generation with 150 samples plus several minutes of post-optimization. This is typically too long for the deployment of state-of-the-art generative AI models [Sauer et al. 2023; Yin et al. 2024]. In contrast, we show in Fig. 10 that our method can generate materials of similar or superior quality to the previous work using significantly fewer samples. We also demonstrate superior generation quality under identical sampling budgets thanks to the high accuracy and low variance of the RL fine-tuned model. The sampling budget refers to the number of parameter sequences we sample. We provide additional examples in the supplementary material.

We analyze the trade-off between generation quality (reward) and time consumption over synthetic and real photograph datasets in Fig. 11. In both test cases, our model achieves a considerably higher reward within 1 second. The user may then opt to run post-optimization to further improve the quality. On the other hand, the supervised method requires more than 20× longer time or minute-long MATch optimization to reach the same reward level, making it less practical for interactive generative AI applications where faster response and higher throughput are generally preferred. Our model's faster inference is discussed in more detail in the supplementary material.

7 Limitations and Future Work

In Sec. 6.3, we conclude that our assumption of known or retrieved graph structures yields comparable expressiveness to those predicted using the state-of-the-art generative model [Hu et al. 2023]. However, mismatches may still occur if a material graph can not sufficiently represent the input texture. Fig. 12 shows instances where neither our approach nor [Hu et al. 2023] could match the target appearance with retrieved or generated graphs. Addressing these limitations requires improvements in material graph structure prediction techniques as the foundation for more effective parameter generator fine-tuning.

An attractive future work is to improve node and edge generation with RL. This is however more challenging than it may first appear [Black et al. 2023]. Indeed, the sample space of plausible graph structures is much broader, resulting in many possible local minima. A strong regularization metric is thus necessary to guide the generation of graph structures and ensure that the generated graph structures represent semantically meaningful material appearances.

Additionally, our RL-based approach opens up the possibility to apply advanced test-time search techniques guided by the learned value network [Silver et al. 2017], which is not possible for supervised training approaches. This may yield better parameter sequences than greedy search or stochastic sampling. Finally, supporting text prompts in addition to image conditioning would be valuable and likely straightforward using the text-to-image prior approach from Hu et al. [2023].

8 Conclusion

In summary, we have introduced RL to improve the generation of high-quality procedural material graphs conditioned on images. Our approach leverages an image-space reward function to bridge the domain gap between parameter tokens and appearance similarity. This allows us to integrate real photographs into fine-tuning, enhancing generalization to unseen real images. We demonstrate substantial improvements in image matching quality compared to conventional supervised training methods, without relying on expensive post-optimization. Our RL fine-tuning workflow shows promise for a

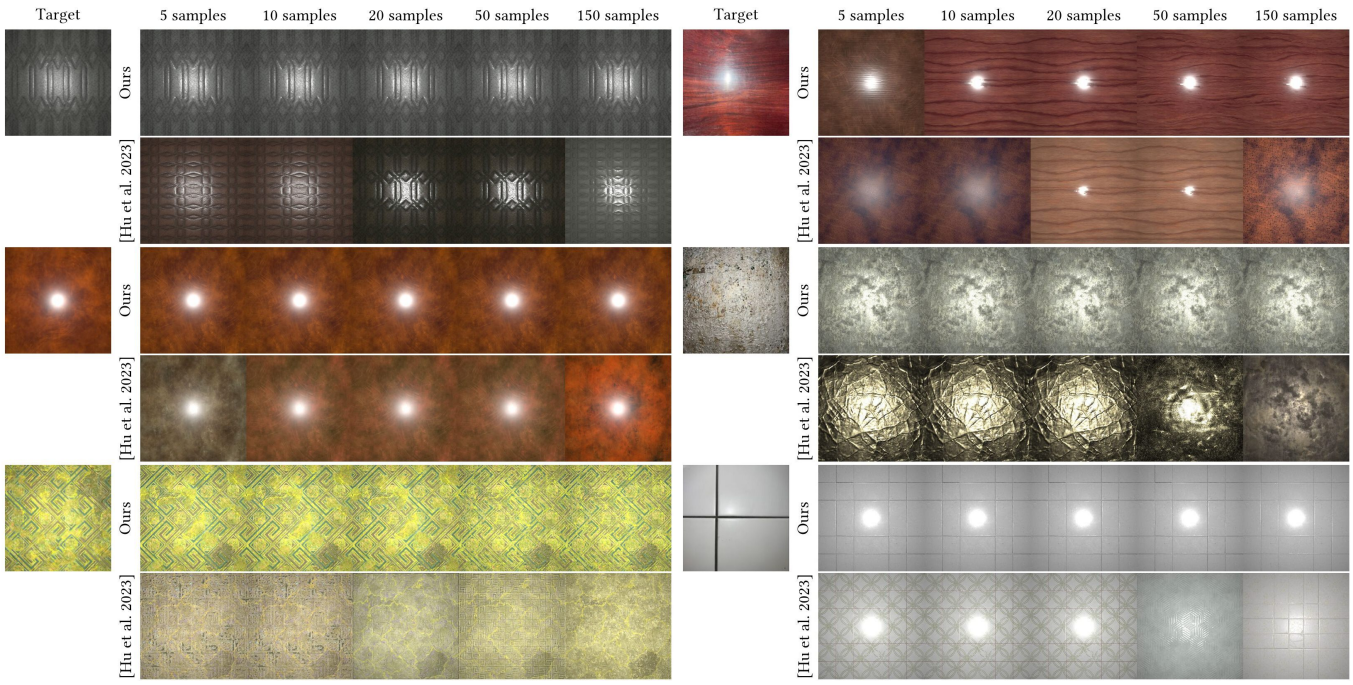

Fig. 10. Qualitative evaluation on synthetic (left column) and real photograph datasets (right column) with increasing random sampling budgets. Since RL fine-tuning effectively reduces the variance during inference, our model does not require extensive sampling to produce high-quality materials. Previous work usually needs to sample more graphs (150 samples in [Hu et al. 2023]) to refine the quality.

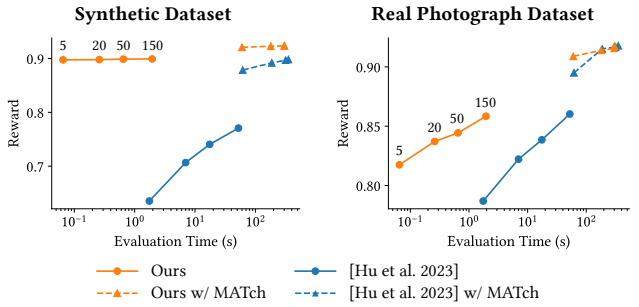

Fig. 11. We analyze the trade-off between matching quality (reward) and time consumption for both RL-trained and supervised models. We collect the total evaluation time, including parameter generation and post-optimization, at four sampling budgets: 5, 20, 50, and 150 samples (black text above line markers). The top-ranked samples used for post-optimization at these budgets are 1, 3, 5, and 5, respectively. Time consumption is measured on an NVIDIA RTX 3090 GPU at peak throughput. A longer evaluation time typically leads to better matching quality. Our model (solid orange lines) achieves higher reward at a much lower time cost. Conversely, the supervised model (solid blue lines) has to draw more samples or rely on expensive post-optimization to reach comparable quality, which is detrimental to evaluation time. The results illustrate that our model is fast and accurate, unlocking opportunities for practical deployment in generative AI applications.

wide range of generative models in procedural graphics, particularly in scenarios where (1) the parameter space has a weak correlation with the image space, or (2) the availability of ground-truth data is limited.

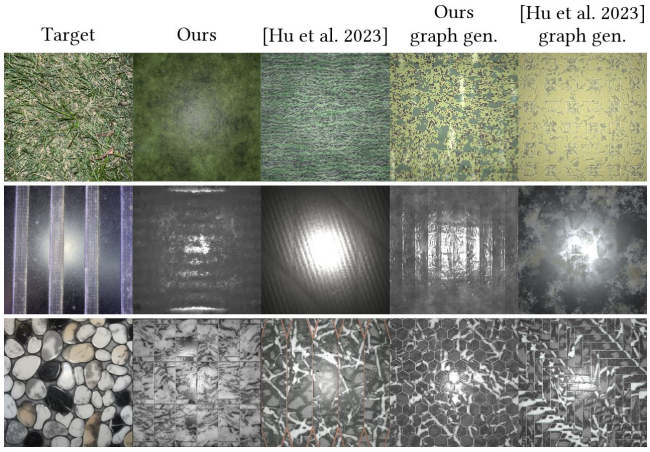

Fig. 12. In a few cases, both our RL-fine-tuned model and previous work struggle to match the input target closely due to imperfect retrieved graph structures (2nd and 3rd column). As in Sec. 6.3, we also experimented with generated graph structures (graph gen.) using node/edge generators trained from supervised learning [Hu et al. 2023] (4th and 5th column). They may provide slightly better matching results in some cases, but still fail to model important structural features in the input appearance.

References

Guillaume Baldi, Rémi Allègre, and Jean-Michel Dischler. 2023. Differentiable point process texture basis functions for inverse procedural modeling of cellular stochastic structures. *Computers & Graphics* 112 (2023), 116–131.

Kevin Black, Michael Janner, Yilun Du, Ilya Kostrikov, and Sergey Levine. 2023. Training diffusion models with reinforcement learning. *arXiv preprint arXiv:2305.13301* (2023).

Valentin Deschaintre, Miika Aittala, Frédo Durand, George Drettakis, and Adrien Bousseau. 2018. Single-Image SVBRDF Capture with a Rendering-Aware Deep Network. *ACM Trans. Graph.* 37, 4, Article 128 (Aug 2018), 15 pages.

Valentin Deschaintre, Miika Aittala, Frédo Durand, George Drettakis, and Adrien Bousseau. 2019. Flexible SVBRDF Capture with a Multi-Image Deep Network. *Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering)* 38, 4 (July 2019), 1–13.

Shukai Duan, Nikos Kanakaris, Xiongye Xiao, Heng Ping, Chenyu Zhou, Nesreen K Ahmed, Guixiang Ma, Mihai Capota, Theodore L Willke, Shahin Nazarian, et al. 2023. Leveraging Reinforcement Learning and Large Language Models for Code Optimization. *arXiv preprint arXiv:2312.05657* (2023).

Ying Fan, Olivia Watkins, Yuqing Du, Hao Liu, Moonkyung Ryu, Craig Boutilier, Pieter Abbeel, Mohammad Ghavamzadeh, Kangwook Lee, and Kimin Lee. 2023. Reinforcement Learning for Fine-tuning Text-to-Image Diffusion Models. In *Thirty-seventh Conference on Neural Information Processing Systems (NeurIPS) 2023*. Neural Information Processing Systems Foundation.

Duan Gao, Xiao Li, Yue Dong, Pieter Peers, Kun Xu, and Xin Tong. 2019. Deep Inverse Rendering for High-resolution SVBRDF Estimation from an Arbitrary Number of Images. *ACM Trans. Graph.* 38, 4, Article 134 (July 2019), 15 pages.

Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. 2015. A Neural Algorithm of Artistic Style. arXiv:1508.06576 [cs.CV]

Linus Gisslén, Andy Eakins, Camilo Gordillo, Joakim Bergdahl, and Konrad Tollmar. 2021. Adversarial reinforcement learning for procedural content generation. In *2021 IEEE Conference on Games (CoG)*. IEEE, 1–8.

Dar'ya Guarnera, Giuseppe Claudio Guarnera, Abhijeet Ghosh, Cornelia Denk, and Mashhuda Glencross. 2016. BRDF Representation and Acquisition. *Computer Graphics Forum* 35, 2 (2016), 625–650.

Pascal Guehl, Remi Allègre, Jean-Michel Dischler, Bedrich Benes, and Eric Galin. 2020. Semi-Procedural Textures Using Point Process Texture Basis Functions. *Computer Graphics Forum* 39, 4 (2020), 159–171. https://doi.org/10.1111/cgf.14061

Paul Guerrero, Milos Hasan, Kalyan Sunkavalli, Radomir Mech, Tamy Boubekeur, and Niloy Mitra. 2022. MatFormer: A Generative Model for Procedural Materials. *ACM Trans. Graph.* 41, 4, Article 46 (2022). https://doi.org/10.1145/3528223.3530173

Jie Guo, Shuichang Lai, Chengzhi Tao, Yuelong Cai, Lei Wang, Yanwen Guo, and Ling-Qi Yan. 2021. Highlight-Aware Two-Stream Network for Single-Image SVBRDF Acquisition. *ACM Trans. Graph.* 40, 4, Article 123 (July 2021), 14 pages.

Yu Guo, Cameron Smith, Miloš Hašan, Kalyan Sunkavalli, and Shuang Zhao. 2020. MaterialGAN: Reflectance Capture Using a Generative SVBRDF Model. *ACM Trans. Graph.* 39, 6, Article 254 (Nov. 2020), 13 pages.

Eric Heitz, Kenneth Vanhoey, Thomas Chambon, and Laurent Belcour. 2021. A sliced wasserstein loss for neural texture synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 9412–9420.

Philipp Henzler, Valentin Deschaintre, Niloy J Mitra, and Tobias Ritschel. 2021. Generative Modelling of BRDF Textures from Flash Images. *ACM Trans Graph (Proc. SIGGRAPH Asia)* 40, 6 (2021).

Yiwei Hu, Julie Dorsey, and Holly Rushmeier. 2019. A Novel Framework for Inverse Procedural Texture Modeling. *ACM Trans. Graph.* 38, 6, Article 186 (Nov. 2019), 14 pages.

Yiwei Hu, Paul Guerrero, Milos Hasan, Holly Rushmeier, and Valentin Deschaintre. 2022a. Node Graph Optimization Using Differentiable Proxies. In *ACM SIGGRAPH 2022 Conference Proceedings* (Vancouver, BC, Canada). Article 5, 9 pages.

Yiwei Hu, Paul Guerrero, Milos Hasan, Holly Rushmeier, and Valentin Deschaintre. 2023. Generating Procedural Materials from Text or Image Prompts. In *ACM SIGGRAPH 2023 Conference Proceedings*.

Yiwei Hu, Miloš Hašan, Paul Guerrero, Holly Rushmeier, and Valentin Deschaintre. 2022b. Controlling Material Appearance by Examples. *Computer Graphics Forum* 41, 4 (2022), 117–128. https://doi.org/10.1111/cgf.14591

Yiwei Hu, Chengan He, Valentin Deschaintre, Julie Dorsey, and Holly Rushmeier. 2022c. An Inverse Procedural Modeling Pipeline for SVBRDF Maps. *ACM Trans. Graph.* 41, 2 (2022), 1–17.

Yuanming Hu, Hao He, Chenxi Xu, Baoyuan Wang, and Stephen Lin. 2018. Exposure: A white-box photo post-processing framework. *ACM Transactions on Graphics (TOG)* 37, 2 (2018), 1–17.

R. Kenny Jones, Homer Walke, and Daniel Ritchie. 2022. PLAD: Learning to Infer Shape Programs with Pseudo-Labels and Approximate Distributions. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2022).

Ahmed Khalifa, Philip Bontrager, Sam Earle, and Julian Togelius. 2020. Pcgrl: Procedural content generation via reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, Vol. 16. 95–101.

David H. Laidlaw, W. Benjamin Trumbore, and John F. Hughes. 1986. Constructive Solid Geometry for Polyhedral Objects. In *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '86)*. Association for Computing Machinery, New York, NY, USA, 161–170. https://doi.org/10.1145/15922.15904

Hung Le, Yue Wang, Akhilesh Deepak Gotmare, Silvio Savarese, and Steven Chu Hong Hoi. 2022. Coderl: Mastering code generation through pretrained models and deep reinforcement learning. *Advances in Neural Information Processing Systems* 35 (2022), 21314–21328.

Harrison Lee, Samrat Phatale, Hassan Mansoor, Kellie Lu, Thomas Mesnard, Colton Bishop, Victor Carbune, and Abhinav Rastogi. 2023b. Rlaif: Scaling reinforcement learning from human feedback with ai feedback. *arXiv preprint arXiv:2309.00267* (2023).

Kimin Lee, Hao Liu, Moonkyung Ryu, Olivia Watkins, Yuqing Du, Craig Boutilier, Pieter Abbeel, Mohammad Ghavamzadeh, and Shixiang Shane Gu. 2023a. Aligning text-to-image models using human feedback. *arXiv preprint arXiv:2302.12192* (2023).

Beichen Li, Liang Shi, and Wojciech Matusik. 2023. End-to-End Procedural Material Capture with Proxy-Free Mixed-Integer Optimization. *ACM Trans. Graph.* 42, 4, Article 153 (jul 2023), 15 pages. https://doi.org/10.1145/3592132

Xiao Li, Yue Dong, Pieter Peers, and Xin Tong. 2017. Modeling Surface Appearance from a Single Photograph using Self-augmented Convolutional Neural Networks. *ACM Trans. Graph.* 36, 4, Article 45 (2017), 11 pages.

Christian E López, James Cunningham, Omar Ashour, and Conrad S Tucker. 2020. Deep reinforcement learning for procedural content generation of 3d virtual environments. *Journal of Computing and Information Science in Engineering* 20, 5 (2020), 051005.

Rosalie Martin, Arthur Roullier, Romain Rouffet, Adrien Kaiser, and Tamy Boubekeur. 2022. MaterIA: Single Image High-Resolution Material Capture in the Wild. *Computer Graphics Forum* 41, 2 (2022), 163–177.

Sajad Mohaghegh, Mohammad Amin Ramezan Dehnavi, Golnoosh Abdollahinejad, and Matin Hashemi. 2023. PCGPT: Procedural Content Generation via Transformers. *arXiv preprint arXiv:2310.02405* (2023).

OpenAI. 2022. ChatGPT: Optimizing Language Models for Dialogue.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (Eds.), Vol. 35. Curran Associates, Inc., 27730–27744. https://proceedings.neurips.cc/paper_files/paper/2022/file/b1efde53be364a73914f58805a001731-Paper-Conference.pdf

André Susano Pinto, Alexander Kolesnikov, Yuge Shi, Lucas Beyer, and Xiaohua Zhai. 2023. Tuning Computer Vision Models with Task Rewards. In *Proceedings of the 40th International Conference on Machine Learning* (Honolulu, Hawaii, USA) *(ICML'23)*. JMLR.org, Article 1381, 11 pages.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*. PMLR, 8748–8763.

Rajkumar Ramamurthy, Prithviraj Ammanabrolu, Kianté Brantley, Jack Hessel, Rafet Sifa, Christian Bauckhage, Hannaneh Hajishirzi, and Yejin Choi. 2022. Is Reinforcement Learning (Not) for Natural Language Processing?: Benchmarks, Baselines, and Building Blocks for Natural Language Policy Optimization. *arXiv preprint arXiv:2210.01241*. https://arxiv.org/abs/2210.01241

Axel Sauer, Dominik Lorenz, Andreas Blattmann, and Robin Rombach. 2023. Adversarial Diffusion Distillation. arXiv:2311.17042 [cs.CV]

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. 2015. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438* (2015).

John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel. 2016. High-Dimensional Continuous Control Using Generalized Advantage Estimation. In *4th International Conference on Learning Representations, ICLR 2016*, Yoshua Bengio and Yann LeCun (Eds.). http://arxiv.org/abs/1506.02438

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

Gopal Sharma, Rishabh Goyal, Difan Liu, Evangelos Kalogerakis, and Subhransu Maji. 2018. CSGNet: Neural Shape Parser for Constructive Solid Geometry. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Liang Shi, Beichen Li, Miloš Hašan, Kalyan Sunkavalli, Tamy Boubekeur, Radomir Mech, and Wojciech Matusik. 2020. MATch: Differentiable Material Graphs for Procedural Material Capture. *ACM Trans. Graph.* 39, 6, Article 196 (Dec. 2020), 15 pages.

Parshin Shojaee, Aneesh Jain, Sindhu Tipirneni, and Chandan K Reddy. 2023. Execution-based code generation using deep reinforcement learning. *arXiv preprint arXiv:2301.13816* (2023).

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. 2017. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815* (2017).

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288* (2023).

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you

Need. In *Advances in Neural Information Processing Systems*, I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), Vol. 30. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

Giuseppe Vecchio, Rosalie Martin, Arthur Roullier, Adrien Kaiser, Romain Rouffet, Valentin Deschaintre, and Tamy Boubekeur. 2023a. ControlMat: Controlled Generative Approach to Material Capture. *arXiv preprint arXiv:2309.01700* (2023).

Giuseppe Vecchio, Simone Palazzo, and Concetto Spampinato. 2021. Surfacenet: Adversarial svbrdf estimation from a single image. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 12840–12848.

Giuseppe Vecchio, Renato Sortino, Simone Palazzo, and Concetto Spampinato. 2023b. MatFuse: Controllable Material Generation with Diffusion Models. *arXiv preprint arXiv:2308.11408* (2023).

Jeff Wu, Long Ouyang, Daniel M Ziegler, Nisan Stiennon, Ryan Lowe, Jan Leike, and Paul Christiano. 2021. Recursively summarizing books with human feedback. *arXiv preprint arXiv:2109.10862* (2021).

Desai Xie, Jiahao Li, Hao Tan, Xin Sun, Zhixin Shu, Yi Zhou, Sai Bi, Sören Pirk, and Arie E. Kaufman. 2023. Carve3D: Improving Multi-view Reconstruction Consistency for Diffusion Models with RL Finetuning. arXiv:2312.13980 [cs.CV]

K. Yan, F. Luan, M. Hašan, T. Groueix, V. Deschaintre, and S. Zhao. 2023. PSDR-Room: Single Photo to Scene using Differentiable Rendering. In *ACM SIGGRAPH Asia 2023 Conference Proceedings*.

Wenjie Ye, Yue Dong, Pieter Peers, and Baining Guo. 2021. Deep Reflectance Scanning: Recovering Spatially-varying Material Appearance from a Flash-lit Video Sequence. *Computer Graphics Forum* (2021). https://doi.org/10.1111/cgf.14387

Tianwei Yin, Michaël Gharbi, Richard Zhang, Eli Shechtman, Frédo Durand, William T Freeman, and Taesung Park. 2024. One-step Diffusion with Distribution Matching Distillation. In *CVPR*.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. 2018. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. In *CVPR*.

Xilong Zhou, Milos Hasan, Valentin Deschaintre, Paul Guerrero, Kalyan Sunkavalli, and Nima Khademi Kalantari. 2022. TileGen: Tileable, Controllable Material Generation and Capture. *CoRR* abs/2206.05649 (2022). https://doi.org/10.48550/arXiv.2206.05649 arXiv:2206.05649

Xilong Zhou, Miloš Hašan, Valentin Deschaintre, Paul Guerrero, Yannick Hold-Geoffroy, Kalyan Sunkavalli, and Nima Khademi Kalantari. 2023. PhotoMat: A Material Generator Learned from Single Flash Photos. In *SIGGRAPH 2023 Conference Papers*.

Xilong Zhou and Nima Khademi Kalantari. 2021. Adversarial Single-Image SVBRDF Estimation with Hybrid Training. *Computer Graphics Forum* 40, 2 (2021), 315–325.