Material Palette: Extraction of Materials from a Single Image

Ivan Lopes[1] Fabio Pizzati[2] Raoul de Charette[1]
[1]Inria [2]University of Oxford

https://astra-vision.github.io/MaterialPalette

Figure 1. **Material Palette**. We introduce the task of material extraction from a single real-world image *without any prior knowledge*. Given an image as input (left and right), our method extracts Physically-Based Rendering (PBR) materials from input regions, which are either provided by a user (right) or output of a segmenter such as SAM [26] (left). The extracted Spatially Varying BRDF (SVBRDFs) encode material intrinsics (**A**lbedo**N**ormal**R**oughness). These can be reused for realistic material editing of 3D scenes (center).

Abstract

Physically-Based Rendering (PBR) is key to modeling the interaction between light and materials, and finds extensive applications across computer graphics domains. However, acquiring PBR materials is costly and requires special apparatus. In this paper, we propose a method to extract PBR materials from a single real-world image. We do so in two steps: first, we map regions of the image to material concept tokens using a diffusion model, allowing the sampling of texture images resembling each material in the scene. Second, we leverage a separate network to decompose the generated textures into spatially varying BRDFs (SVBRDFs), offering us readily usable materials for rendering applications. Our approach relies on existing synthetic material libraries with SVBRDF ground truth. It exploits a diffusion-generated RGB texture dataset to allow generalization to new samples using unsupervised domain adaptation (UDA). Our contributions are thoroughly evaluated on synthetic and real-world datasets. We further demonstrate the applicability of our method for editing 3D scenes with materials estimated from real photographs. Along with video, we share code and models as open-source on the project page: https://github.com/astra-vision/MaterialPalette

1. Introduction

Whether it is a soft blanket, a rugged carpet, or a crumbling stone, humans can identify materials from a photograph. Besides geometry understanding, this ability derives from our sensing of how light interacts with materials, allowing us to identify the substance at stake without even touching it. In sciences, this has pushed research in spectrophotometry [18] or light sensing [4], while in the arts Vermeers and Caravaggio, among others, have used this long standing observation to convey the feeling of materials in their paintings. Modern CG artists also deploy significant efforts to mimic realistic light-material interaction, through the design of Physically-Based Rendering (PBR) materials. While many libraries of material assets exist, no dataset can capture the true variety of real-world materials. What is more, capturing real-world materials is still a complex endeavor requiring special apparatus [2]. In many scenarios, however, one may wish to estimate a material from an RGB image, for example, to capture a unique marble stone during a trip or the fur of a wild animal from a souvenir photo.

Hence, we formulate the novel task of extracting PBR materials from a single real-world image, as shown in Fig. 1. Given a set of regions, our method solves this task by generating corresponding textures along with their Spatially Varying BRDFs (SVBRDFs) *without a priori knowledge* about the capturing viewpoint, scene geometry or lighting.

This sets our work aside from the literature. We coined our method Material Palette because, just like a painter would create their own color mix, one can create their palette of materials from their own photos (Fig. 1, left and right). Moreover, extracted materials are readily usable for CG applications such as 3D renderings (Fig. 1, middle).

There are major challenges in the estimation of PBR materials from just one RGB image, since single-view decomposition is highly ill-posed [9]. To address these hurdles, we rely on recent advances in text-to-image generation [39, 45, 47] to disentangle the specific material appearance from the scene geometry and imaging conditions, allowing us to generate close-up tileable RGB textures of the materials in the scene. We further extract the PBR intrinsics of these diffusion-generated images, with a domain adaptation strategy that benefits from a novel synthetic dataset. Experiments show that Material Palette outputs convincing results and performs better than baselines. The extracted materials closely resemble their real-world counterparts, which makes them usable for 3D scene editing.

We contribute in the following ways:

- We formulate the novel challenging task of material extraction from a single real-world image.
- We introduce Material Palette, a method to extract materials within an image, operating in either a user-assisted or fully automated mode (Sec. 3.4).
- We show how a finetuned text-to-image diffusion model can generate realistic tileable texture images (Sec. 3.2) suitable for SVBRDF estimation (Sec. 3.3).
- We provide a non-trivial evaluation pipeline to assess the quality of extracted PBR materials along with a novel prompt-generated dataset named TexSD. Experiments show our materials are close to those of real material datasets and readily usable for 3D editing.

2. Related works

To the best of our knowledge, we are the first to address end-to-end extraction of multiple materials from *single* real-world images, but we cover literature connected to our task.

Single-image intrinsics decomposition. Long after the pioneering work of [21], deep networks were leveraged for decomposition, exploiting their great pixel-wise estimation capabilities. Most early works focused on object-centric scenes with Lambertian assumption [55], user interaction [33], or scene layers [23]. To decompose in-the-wild objects, symmetry [60] or cross-instance [37] constraints are applied, while [24] requires the 3D mesh [24]. Holistic scene decomposition was addressed splitting albedo and shading [3, 30, 38], also with the support of image-to-image translation [34], or inverse rendering [53]. To account for spatially-varying lighting, some use mixture of illuminations or SVBRDF [3, 16, 31, 32, 66]. Notably, many of these works rely on estimated light sources and are designed for either indoor or outdoor scenes. Additionally, they capture the intrinsics of a scene image without distinguishing between the materials present. We instead wish to extract the intrinsics of dominant materials.

Material and texture extraction. Typical material capture requires expensive multi-view [2] or polarized [11] apparatus. Many use synthetic data to train single-view SVBRDF estimation networks [9], often coupled with additional single-view data [15, 36] or custom training strategies [10, 28, 56]. Importantly, all works mentioned *require orthogonal close-up views* of the materials which is impractical for real scenes. UMat [46] uses a single image acquired with a flatbed scanner. PhotoScene [63] is the closest to our work, but it requires CAD inputs and it is limited to a set of synthetic material graphs. Instead, we propose a single-image method targeting real-world materials. TexSynth [12] provides a guided texture editing method but does not model materials explicitly.

A connected field is texture extraction from real-world images. Note that while materials model light interaction, textures only describe the spatial arrangement of colors. A common strategy is to cluster the image textures and extend them to full resolution [29, 49] or apply dataset distillation [7]. While we inspire from texture extraction methods, our task differs drastically as we seek to estimate the full SVBRDF – not only the color.

Text-to-image generative models. Seminal works for text-to-image generations exploited conditional generative networks, allowing image synthesis in constrained scenarios only [62, 64, 67]. Instead, training on billions of samples has been proven effective in generalizing on a wide range of prompts [44, 45]. To this extent, diffusion models are exploited for their stability at scale [39, 45, 47, 51], although adversarial-based methods are also used [52]. Recently, MatFuse [58] and ControlMat [57] adopted diffusion processes for material generation. We get inspiration from them while avoiding long training times.

3. Material Palette

Our method extracts PBR materials from regions of a real-world image. Different from approaches relying on close-up captures [9, 36] or dedicated hardware [2], the problem is more challenging when presented with an in-the-wild image (Sec. 3.1) with unknown lighting and geometry. We build on advances in vision-language models to achieve this goal.

Given an input image $\mathcal{I}$ and some input regions $\{\mathcal{R}_1, \cdots, \mathcal{R}_N\}$, Material Palette extracts a set of corresponding materials SVBRDF $\{M_1, \cdots, M_N\}$. Fig. 2 illustrates the complete pipeline. For each region, we ex-

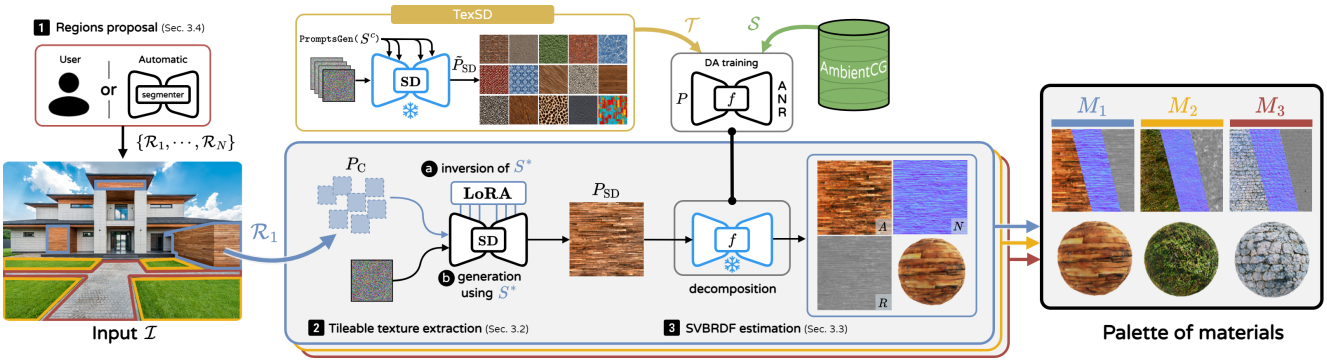


Figure 2. **Material Palette pipeline**. From a single image $\mathcal{I}$ (left) our method extracts the SVBRDF of dominant materials (right). Considering a set of regions $\{\mathcal{R}_1, \cdots, \mathcal{R}_N\}$ from a user or a segmenter 1 (Sec. 3.4), we process each region $\mathcal{R}_i$ separately following two steps. In 2 (Sec. 3.2), we finetune Stable Diffusion [47] on crops of the region P_C to learn a concept S^*, which is later used to generate a texture image P_{SD} resembling P_C. Then in 3 (Sec. 3.3), these patches are decomposed into SVBRDF intrinsics maps (**A**lbedo, **N**ormal, **R**oughness) using a multi-task network. Finally, the output is the palette of extracted materials $\{M_1, \cdots, M_N\}$ corresponding to input regions.

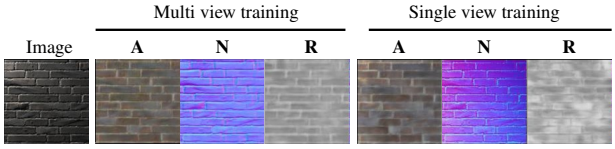


Figure 3. **Decomposition with known illumination.** Training on ACG [1] we note that even with known lighting, single-view leads to degenerated intrinsics (**A**,**N**,**R**) due to viewpoint ambiguities.

tract a texture approximating its material appearance using Stable Diffusion [47] (Sec. 3.2) 2 . Then, we rely on a domain-adaptive SVBRDF decomposition using our diffusion prompt-generated samples for generalizing to the extracted textures 3 (Sec. 3.3). While our pipeline can rely on user inputs to define the image regions, we can also query any off-the-shelf segmenter 1 (Sec. 3.4).

3.1. Problem statement

Considering a typical intrinsics decomposition, an image P with known illumination can be approximated as the result of a rendering operation $\tilde{\rho}(.)$ from SVBRDF maps $\tilde{M} = \{\tilde{A}, \tilde{N}, \tilde{R}\}$, being pixel-wise Albedo, Normals, and Roughness, respectively. This writes: $P = \tilde{\rho}(\tilde{M})$. Our goal is to learn the inverse rendering process with a neural network f to predict $\tilde{M}$ from P. In details:

$$f(P) = M = \{A, N, R\} \cong \tilde{M}, \tag{1}$$

where $\tilde{M}$ is the ground truth SVBRDF. A dataset with such labels is obtainable with expensive procedural generation of top-view materials or acquisitions in controlled scenarios [1]. We can train f, by rendering *multiple views with known illuminations* $\tilde{\rho}_{1\ldots n}$, and enforcing both a regression loss $\mathcal{L}_{\text{reg}}$ towards the ground truth maps and a multi-view rendering loss $\mathcal{L}_{\text{ren}}$ on the n renderings [19]:

$$\mathcal{L}_{\text{reg}} = ||M - \tilde{M}||_1, \quad \mathcal{L}_{\text{ren}} = \sum_{i=1}^{n} ||\tilde{\rho}_i(M) - \tilde{\rho}_i(\tilde{M})||_1 . \tag{2}$$

After training, f can be used to estimate M from unseen images with unknown SVBRDF (Fig. 3, multi-view). However, in our scenario we wish to estimate M from specific regions of an *in-the-wild* image with variable illumination and geometry, thus being shifted w.r.t. the training distribution of materials datasets. Besides, even *assuming known illumination*, a single view is ambiguous for intrinsics decomposition (Fig. 3, single-view).

Without any geometry or illumination priors, we tackle the problem per region by extracting tileable textures which are then decomposed into SVBRDF (Sec. 3.2) while accounting for the domain gap (Sec. 3.3).

3.2. Tileable texture extraction

Given an image $\mathcal{I}$ and a material region $\mathcal{R}$, a naive approach to disentangle the appearance from the scene geometry/lighting would be to classify the material in $\mathcal{R}$ and use its label to generate patches with a text-to-image network [47]. This would however fail to capture the fine-grained characteristics of the material in $\mathcal{R}$. Considering for example the picture of a rundown house in Fig. 1 (right), the label "brick" does not fully capture the intricate appearance of unaligned and weathered stones. Indeed, a mere classification of the material fails to encompass the complexity of the texture and its unique appearance.

We formulate the problem as texture extraction from a region $\mathcal{R}$, thus seeking to remove geometric distortion and lighting in $\mathcal{R}$ by generating a flat texture image which we further decompose. To do so, we build upon text-to-image models [47], exploiting their capabilities for disentangling

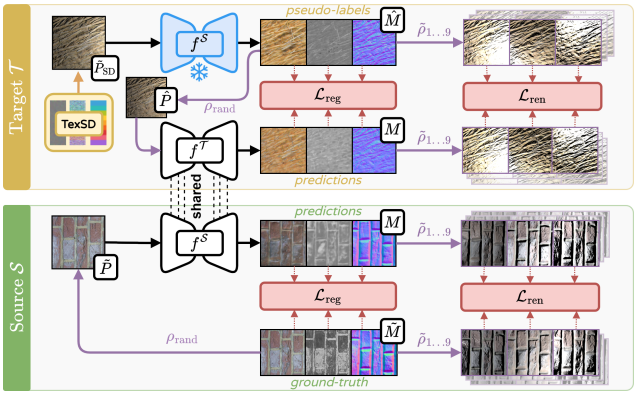


Figure 4. **SVBRDF Unsupervised Domain Adaptation.** We train a decomposition network f on labeled SVBRDF materials $\mathcal{S}$ and unlabeled target data $\mathcal{T}$ from our novel TexSD dataset. Ultimately, $\mathcal{T}$ reduces the domain gap between the SVBRDF dataset and the real domain, *i.e.*, patches generated from our extraction method (*cf.* Sec. 3.2). We enforce both regression and rendering losses on $\mathcal{S}$ and $\mathcal{T}$ using pseudo-maps $\hat{M}$ extracted by the source-only model $f^{\mathcal{S}}$ (top). The final adapted model is denoted $f^{\mathcal{T}}$.

semantics. Essentially, we finetune a text-to-image diffusion model [50] to encode the material depicted in $\mathcal{R}$ as a token. This allows us to generate a resembling tileable texture at any arbitrary resolution.

During finetuning (2a in Fig. 2), we extract crops P_{C} from $\mathcal{R}$, utilizing them to map the material to a concept token S^*. The aim is for S^* to accurately describe the appearance of the specific material in $\mathcal{R}$, more faithfully than when using a class name. In practice, we first finetune Stable Diffusion [47] and learn S^* [22, 50] using a single prompt template, `PromptTrain` = *"an object with S^* texture"*. Note, that the latter includes no information about the material class, removing needs for material labeling.

During inference (2b in Fig. 2), we rely on different prompts `PromptsGen` chosen to enforce a texture-like appearance on a planar surface, such as "*realistic S^* texture in top view*". The generative nature of the process lets us generate not only one but a set of multiple textures from S^*, all resembling the material in $\mathcal{R}$. We rely on minimum LPIPS [65] w.r.t. crops of $\mathcal{R}$ to select the ad-hoc texture P_{SD}. We later detail the effect of prompts on the acquisition of S^* and generation of textures (Sec. 4.5).

Additionally, we follow ControlMat [57] and adopt noise unrolling at inference to generate tileable textures. Different from ControlMat though, we are not conditioned on an input image at inference, but rather on S^* which we can leverage to generate textures at *any resolution*.

3.3. SVBRDF estimation

We now seek to decompose each generated RGB-only texture from P_{SD} into intrinsics $M = \{A, R, N\}$. From Sec. 3.1, a decomposition network f can be trained on a SVBRDF dataset and used on our generated textures P_{SD}. Although these are much closer, than $\mathcal{R}$, to actual renderings of SVBRDF datasets[1], f still suffers from a distribution shift. We address this problem as an unsupervised domain adaptation (UDA) $\mathcal{S} \mapsto \mathcal{T}$ where the source domain $\mathcal{S}$ consists of materials with SVBRDF labels, and the target domain $\mathcal{T}$ is composed of diffusion-generated RGB textures.

Source training. We train our source model $f^{\mathcal{S}}$ by enforcing a reconstruction loss on ground truth maps $\tilde{M}$ and a multi-view rendering loss with 9 lighting configurations[2], denoted $\rho_{1\ldots9}$. Explicitly, we optimize $f^{\mathcal{S}}$ by minimizing $\mathcal{L}_{\text{reg}}$ and $\mathcal{L}_{\text{ren}}$ defined in Eq. (2) with:

$$\lambda \mathcal{L}_{\text{reg}}(M, \tilde{M}) + \mathcal{L}_{\text{ren}}(M, \tilde{M}) \tag{3}$$

where $M = f^{\mathcal{S}}(\tilde{P})$ and $\tilde{P}$ the rendering of $\tilde{M}$ with a random lighting ρ_{rand}. Ground truth maps $\tilde{M} \in \mathcal{S}$ are obtained from any SVBRDF library such as ACG [1].

TexSD. To bridge the gap with SVBRDF libraries, we first generate training material textures in the target domain by prompting the text-to-image model with `PromptsGen`, *e.g.*, "*realistic S^c texture in top view*", replacing S^c with a class name. Notably, we *do not* rely on finetuning, but instead only exploit the text-to-image capabilities of large diffusion models and their innate knowledge of material classes. This allows us to construct a dataset, named TexSD, of 9,000 textures generated from a set of 130 classes derived from ACG and ChatGPT proposals [40]. A schematic view is in Fig. 2 (block 'TexSD'), and details are in the supplementary. Crucially, despite the generation of multiple images per class, these texture images are not multiple views of the same material instance. They are instead *single-view* variations within a class.

Adaptation. Equipped with TexSD as target domain $\mathcal{T}$, we overcome the ill-posed single-view training (*cf.* Fig. 3) drawing inspiration from pseudo-labels [27]. We extract *pseudo-decomposition maps* $\hat{M} = \{\hat{A}, \hat{N}, \hat{R}\}$ for all textures images $\tilde{P}_{\text{SD}} \in$ TexSD by processing them with our source model $f^{\mathcal{S}}$. This enables pseudo multi-view training on $\mathcal{T}$ with *only single view* images.

Hence, we follow Eq. (3) to train on $\mathcal{S}$ with ground truth $\tilde{M}$, then finetune concurrently on $\mathcal{S} + \mathcal{T}$, using pseudo-labels $\hat{M}$ for $\mathcal{T}$. An illustration of the training process is shown in Fig. 4. In both stages, SVBRDF inputs are rendered with random lighting conditions ρ_{rand} to encourage invariance and robustness. At inference, we use $f^{\mathcal{T}}$ to infer the SVBRDF of P_{SD}, leading to material maps M_{SD}.

[1] By construction, P_{SD} textures should be geometry- and lighting-free.

[2] Light configurations are defined as angles (α, ϕ) on the upper hemisphere, with α the light angle and ϕ the viewing angle. Following [9], we sample 6 symmetrical lighting/viewing angles to encourage specular and 3 uniformly sampled lighting/viewing angles to cover the parameter space.

3.4. Pipeline automation

Our pipeline is readily usable to extract materials from any region $\mathcal{R}$ of a real-world image. While 3D applications may benefit from user interaction to define $\mathcal{R}$, we also complement our pipeline with full automation.

To do so, we formalize the problem of defining regions $\{\mathcal{R}_1, \cdots, \mathcal{R}_N\}$ on real-world images as a 2D segmentation task. We integrate two segmentation models in our pipeline: the Segment Anything Model (SAM) [26], a large-scale instance segmentation model, and Materialistic [54], a material selection method. In Sec. 4.4, we show how all of these region proposals lead to accurate material extractions.

4. Experiments

We study the performance of Material Palette along four main axes: **i)** Measuring the quality of our generated textures with respect to textures scraping techniques (Sec. 4.2); **ii)** Quantifying our SVBRDF adaptation scheme on real material textures (Sec. 4.3); **iii)** Evaluating the quality of our extracted materials end-to-end (Sec. 4.4); **iv)** Through exhaustive rendering of 3D scenes with our materials. We also ablate our method in Sec. 4.5 and demonstrate the usage of our Material Palette for 3D editing in Sec. 4.6. Please refer to the demo video on the project page for better evaluation.

4.1. Experimental details

Networks. We use Stable Diffusion [47] v1.5 for texture extraction, training a LoRA [22] Dreambooth [50] for learning S^*. Optimization times take around 3-5 min per learned S^* on a Tesla V100-16GB. When learning S^*, `PromptTrain` is set to "*an object with S^* texture*" while for inference it is chosen randomly among `PromptsGen` (see Fig. 10). To ensure tileability and high resolution generation, we roll the latent tensor on both spatial axes by a random amount at every timestep of the diffusion process [57]. We apply Poisson solving [41] to remove seams remaining on the borders. We directly sample textures up to 1024px, while for higher resolutions we batch-decode the latent code and blend overlapping patches using a weighted average. For f, we use a multi-head CNN [35] with U-Net [48], ResNet-101 [20] backbone, and custom decoders with alternating `upsample-conv` layers.

Public datasets. We leverage three SVBRDF libraries (AmbientCG [1], PolyHaven [42], CGBookcase [8]) and one material segmentation dataset (OpenSurfaces [5]).

AmbientCG (ACG) contains 2000 high-resolution PBR materials obtained from real-world captures with special apparatus, procedural generation, or image approximation. It includes around 50 material classes. We use the high-resolution 2k textures from ACG to train all source models.

PolyHaven (PH) and ***CGBookcase*** (CGB) are smaller libraries composed of 320 materials each. We use them as evaluating sets to validate our adaptation method.

OpenSurfaces (OS) is an image dataset including dense material annotations. We use 14 overlapping classes with ACG and use a subset for end-to-end evaluation.

TexSD dataset. We introduce a new dataset used for adaptation (Sec. 3.3). It is obtained by prompting Stable Diffusion [47] with `PromptsGen` and 130 class names and totals 9,000 1K textures. Details in the supplementary.

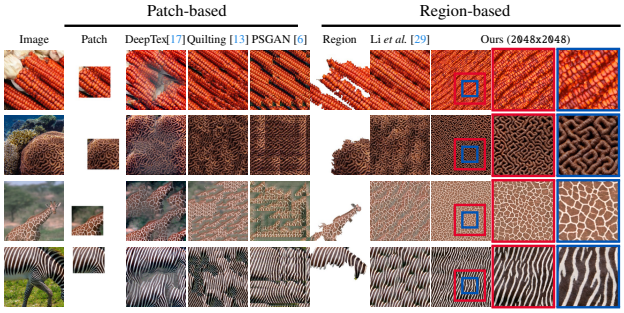


Figure 5. **Textures extraction.** We compare texture extracted from natural images with 4 patch-based or region-based baselines. Differently from baselines, our method is based on a learned concept S^* which, when used for generating samples, corrects artifacts, is not limited to a fixed resolution and is fully tileable, resulting in homogeneous textures. For ease of comparison, we show outputs at `2048x2048` along a **x2** and **x4** zoom. *Images are downscaled for visualization purposes.*

4.2. Texture extraction

We showcase our text-to-image texture extraction (Sec. 3.2) and existing techniques in Fig. 5. We compare qualitatively with [29] while also providing GAN-based baselines [6], methods inspired by style transfer [17] or image quilting [14]. For a fair comparison, we show outputs from our method using the same regions as [29].

Even though [29] outperforms older methods, it presents artifacts (last two rows) making images unsuitable for material extraction. Moreover, the extracted textures are non-tileable and entangle geometry and lighting. In particular, in the second row, the texture of [29] replicates the shading of the input coral image. Our method dramatically differs as it is able to map input images to plausible material textures, removing geometry and lighting while *remaining devoid of artifacts*. Importantly, we can generate any variations of *tileable* samples, at *any resolution*. Ultimately, this shows the inadequacy of prior extraction methods for extracting tileable high-resolution texture patches suitable for decomposition and rendering purposes.

4.3. SVBRDF decomposition

We now focus on our proposed UDA pipeline for decomposition (Sec. 3.3). Since our TexSD dataset does not come with associated SVBRDF ground truths, we rather eval-

	method	DA	MSE ($10e1$) ↓ A	N	R	SSIM ↑ A	N	R	Δ% ↑ ANR
PH ID	Deep Materials* [9]		0.264	0.380	0.453	0.379	0.235	0.358	
	$\mathcal{S}$ource-only		0.083	0.300	0.475	0.610	0.304	0.458	↴
	SurfaceNet [56]	✓	0.071	0.298	**0.427**	0.626	0.304	0.472	+5.18
	ours	✓	**0.069**	**0.291**	0.443	**0.630**	**0.309**	**0.476**	**+5.87**
CGB ID	Deep Materials* [9]		0.590	0.465	1.940	0.392	0.228	0.346	
	$\mathcal{S}$ource-only		0.098	0.221	**0.555**	0.662	0.437	0.476	↴
	SurfaceNet [56]	✓	0.101	0.230	0.615	0.657	0.445	0.471	-3.00
	ours	✓	**0.084**	**0.219**	0.588	**0.669**	**0.457**	**0.482**	**+2.62**
PH OOD	Deep Materials* [9]		0.264	0.380	0.453	0.379	0.235	0.358	
	$\mathcal{S}$ource-only		0.065	0.247	0.439	0.608	0.284	0.518	↴
	SurfaceNet [56]	✓	**0.053**	0.250	0.415	0.608	0.283	0.524	+3.94
	ours	✓	**0.053**	**0.246**	**0.409**	**0.618**	**0.288**	**0.531**	**+5.24**

* Note that Deep Materials [9] is trained on an in-house dataset.

Table 1. **DA evaluation.** Performances on In-Distribution (ID, top) and Out-of-Distribution (OOD, bottom). Δ measures the relative performance over {**A**,**N**,**R**} w.r.t. the ACG-only model. DA refers to methods using domain adaptation strategies. Our adaptation succeeds at preserving generalization abilities on OOD samples, while SurfaceNet [56] has slightly lower gains on OOD.

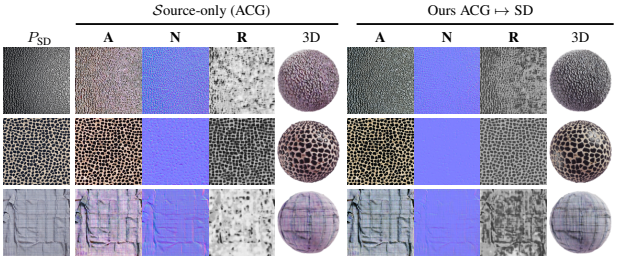


Figure 6. **Qualitative results.** Comparison between $\mathcal{S}$ource-only training (ACG) and our (ACG ↦ SD) adaptation on unseen SD samples. We notice that $\mathcal{S}$ource-only overestimates N and R and exhibits color shift in A. This results in a lower-quality 3D render.

uate on two additional $\mathcal{S}$↦$\mathcal{T}$ scenarios: ACG↦PH and ACG↦CGB. This allows us to measure the adaptation effectiveness on the target set $\mathcal{T}$ w.r.t. ground truth annotations. We report standard metrics: Mean Squared Error (MSE ↓) and Structural Similarity Index (SSIM ↑) [59]. For a refined comparison, we evaluate common classes in $\mathcal{S}$ and $\mathcal{T}$ as In-Distribution (ID), resulting in 53 and 58 materials for PH and CGB, respectively.

We propose three baselines. First, we evaluate Deep Materials [9] as an off-the-shelf decomposition network. We also compare with an ACG $\mathcal{S}$ource-only model, serving as *lower bound*. Then, we implement SurfaceNet [56] with our architecture and finetune the original ACG model for both SurfaceNet and ours. Results in Tab. 1 (top) suggest that we improve consistently the decomposition. Considering the richer PH material ontology, we also evaluate the 47 classes of materials Out-Of-Distribution (OOD). In Tab. 1 we obtain only a low-performance OOD drop (bottom) compared to ID (top), exhibiting better generalization than baselines.

	$\mathcal{R}$	LPIPS ↓ A	N	R
	upper bound	0.8288	0.5915	0.7255
Ours	OS masks [5]	**0.7959**	0.5730	0.7142
	SAM [26]	0.8048	0.5692	**0.7096**
	Materialistic [54]	0.8077	**0.5678**	0.7169
	lower bound	0.6789	0.4629	0.6843

	$\rho(M)$	CLIP Classif. ↑ top-1	top-5
	ACG	47.78	85.88
Ours	OS masks [5]	47.03	85.12
	SAM [26]	43.71	80.83
	Materialistic [54]	**50.89**	**86.82**

Table 2. **Resemblance to SVBRDF dataset.** We evaluate our extracted materials with various regions proposals w.r.t. materials from ACG (left). We also report zero-shot classification on ACG (right) which measures ability of CLIP to classify the class of the re-rendered material [43]. Both results demonstrate that our materials have coherent class-wise characteristics *without class annotations*, irrespective of the segmenter used.

Furthermore, we show in Fig. 6 a visual comparison of 'Ours ACG↦SD' vs 'source-only (ACG)', on TexSD *unseen* samples. It shows our adaptation better decomposes images, ultimately producing more realistic 3D renderings.

4.4. Material extraction

Given the lack of datasets combining real scenes with region-wise materials annotations, we highlight the complexity of evaluating all components of our method together.

Resemblance to SVBRDF datasets. We design experiments using OpenSurfaces (OS) and ACG, allowing us to understand if Material Palette preserves the expected characteristics of a material region. Considering that datasets class ontologies differ, we map OS and ACG classes to a common set of 14 materials C_m, detailed in the supplementary, in which OS classes are grouped following [61].

In a first experiment, we use OS ground truth, *i.e.* user-annotated material segmentation masks, as $\mathcal{R}$, and automatically extract the associated materials M_{SD} with Material Palette. Given that in the OS ground truth the material class c of $\mathcal{R}$ is known (but not the intrinsics), we compare the extracted M_{SD} with those of materials of the same class in ACG. We define an *upper bound* by evaluating the same extracted materials but on *all other materials* in ACG. Intuitively, if results are better than the upper bound, we correctly mapped the appearance of a particular material class to visual features specific to that class. In other words, a material labeled as 'brick' in a natural image should lead to extracted maps more similar to 'brick' samples in ACG, than to other classes such as 'wood'. In practice, the evaluation is conducted by sampling 100 $\{M_{\text{SD}}, \tilde{M}\}$ pairs and evaluating LPIPS [65] between them (lower is better), for each of the 14 material classes. Considering class $c \in C_m$, *Ours* will evaluate $\{M^c_{\text{SD}}, \tilde{M}^c\}$ pairs $\forall c \in C_m$, while the *upper bound* will have $\{M^c_{\text{SD}}, \tilde{M}^{\bar{c}}\}$ where $\bar{c}$ is a random $\bar{c} \neq c$. The reported LPIPS values are averaged over all classes. From Tab. 2 (left), 'Ours-OS Masks' improves performance over the upper bound, proving the effectiveness of our method.

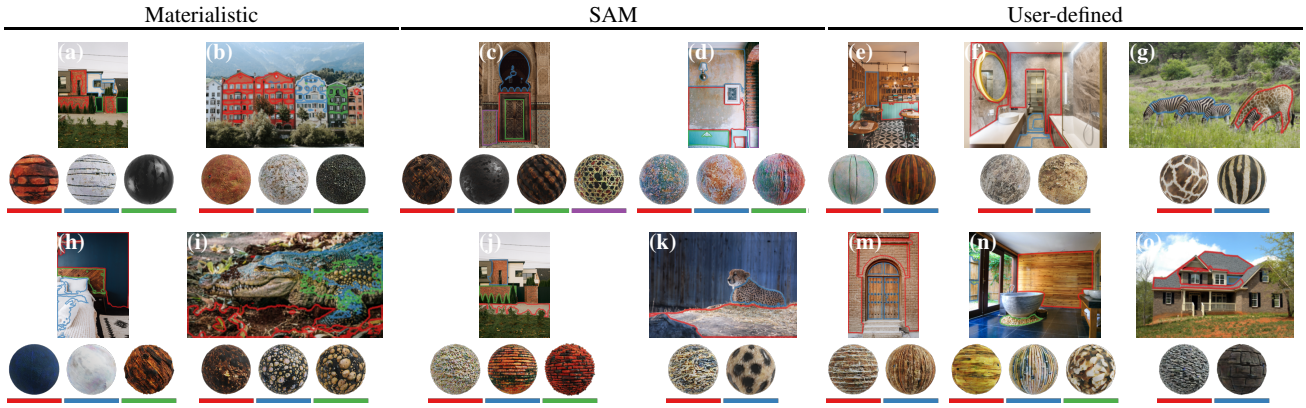
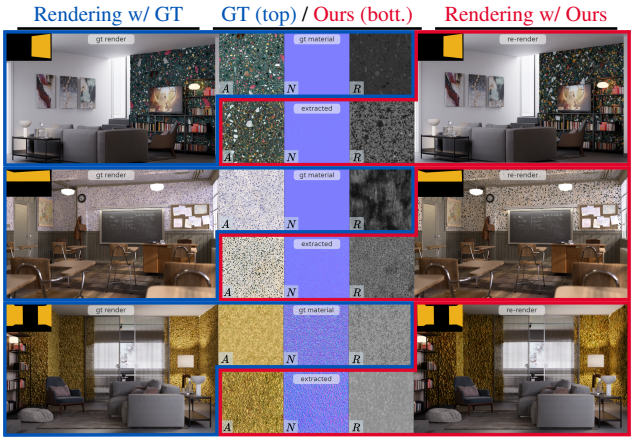


Figure 7. **Material palettes.** Results on images gathered on the internet with regions extracted using Materialistic, SAM, or user-defined.



P	LPIPS ↓ A	N	R
SD random	0.8163	0.6895	0.7441
ours (user)	**0.6839**	**0.6254**	**0.5951**
ACG random	0.6547	0.4508	0.5600

Figure 8. **End-to-end evaluation.** (top) 3D scenes are edited with ACG materials that we extract and re-render for direct comparison. (bottom) We quantify the extraction quality using LPIPS [65].

Additionally, we propose a *lower bound*, following our pipeline but using $\{M^c_{\text{ACG}}, \tilde{M}^c\}$ where M_{ACG} are estimated ACG maps from single-view samples of class c. We also evaluate the impact on material extraction of our automated pipeline using either SAM [26] or Materialistic [54] as region segmenters (Sec. 3.4). We highlight how in all setups we achieve comparable performance, always improving over the lower bound.

In a second experiment, we propose an evaluation with CLIP [43]. For each class $c \in C_m$, we generate textures using `PromptsGen` (*cf.* Sec. 4.5) and evaluate the CLIP ViT-B/32 zero-shot classification performance of all M_{SD}, rendered with random illumination. We do the same for all ground truth $\tilde{M}$ in ACG. In Tab. 2 (right) we report accuracies averaged over C_m, comparable performance suggest we are able to render materials similar to ACG.

Qualitative evaluation. In Fig. 7 we visualize web-scraped images and the materials extracted by Material Palette using regions from SAM [26], Materialistic [54] or a User input. Each material is rendered on a 3D sphere in Blender with a color below matching its region color. Given the task complexity, we emphasize the quality of the extraction for a wide variety of complex materials: bricks, tiles, skins, fur, *etc*. In particular, we highlight the quality of bricks in **(a)**[3] as well as in **(j)** and **(j)** with a different segmenter, and more astonishingly in the small roof region of **(o)**. In natural images, materials also match appearances such as crocodile skin **(i)**, or fur of jaguar **(k)**, giraffe **(g)** and zebra **(g)**. Other noticeable results are the complex mosaic pattern **(c)** or damaged wall **(d)** and **(d)**.

End-to-end re-rendering. We also design a challenging end-to-end evaluation leveraging realistic 3D scenes. Through automatic editin we replace some 3D objects material with a PBR material of ACG, thus rendering a total of 174 images (2 scenes, 3 views) comprising 10 materials for each of the 16 dominant classes of ACG. As the latter comes with SVBRDF ground truths, we compare them with our Material Palette extractions on the rendered images with *ad-hoc* user-input regions. Visuals in Fig. 8 show the scenes rendered with GT materials, along with our extracted materials and re-renderings. Our materials capture the main characteristics though we observe some over/under saturation, highlighting the task complexity. Moreover, we provide quantification to compare LPIPS(↓) vs ground truth ACG for a 'SD random' generation *prompted with the true class* compared to a random ACG material of the true class. Notably, our materials are much closer to ACG than SD generation, demonstrating the benefit of our pipeline over text-to-image generation.

[3] Here, **(a)** refers to material of the red region from image **(a)** of Fig. 7.

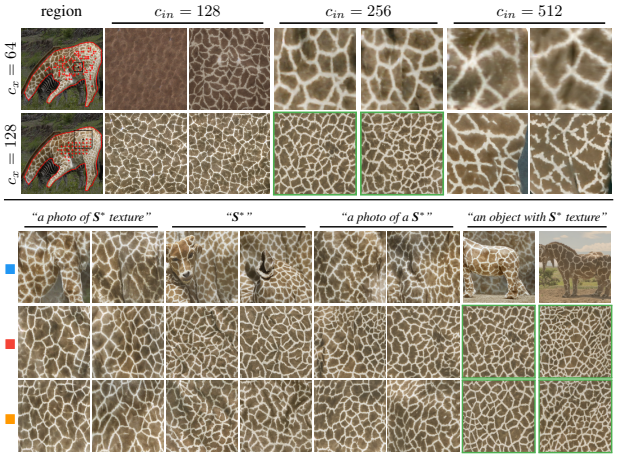


Figure 9. **Inversion ablation.** We show how crop and upsampling sizes affect SD generation results (top). Effects of `PromptTrain` for learning $\mathbf{S}^*$ are also given (bottom). For all four inversions, we provide generations using the templates ■, ■, and ■ (*cf.* Fig. 10). We highlight in green the parameters used.

Prompt templates	FID $\cdot 10^{-2} \downarrow$ A	N	R	KID $\cdot 10^{2} \downarrow$ A	N	R	Qualitative Paper	Pavement
■ *"a photo of a* $\boldsymbol{S}^c$*"*	1.89	1.74	1.83	2.60	6.52	3.75		
■ *"a* $\boldsymbol{S}^c$ *material"*	1.83	1.60	1.67	2.38	5.42	**2.74**		
■ *"a* $\boldsymbol{S}^c$ *texture"*	1.72	1.54	1.63	1.82	5.27	2.86		
■ *"realistic* $\boldsymbol{S}^c$ *texture in top view"*	**1.55**	**1.27**	1.59	**1.54**	**3.34**	3.32		
■ *"high resolution realistic* $\boldsymbol{S}^c$ *texture in top view"*	**1.55**	1.30	**1.55**	1.71	3.48	3.34		

Figure 10. **Prompt generation.** We compare prompt templates when generating using $\mathbf{S}^c$. With more detailed prompts and the word "texture", we can generate images similar to ACG.

Figure 11. **Scene editing results.** We edit objects (top-left insets) of 3D scenes using materials *extracted from real-world images*.

4.5. Ablations

Inversion. Considering we face objects of varying sizes, we ablate the input size used during inversion. Fig. 9 (top) shows the crop size c_x and training input size c_{in} (*i.e.*, upsampling size). Considering that crops have a much lower resolution than the pre-trained SD v1.5 inputs (512px), we make two choices: (i) extract crops with largest c_x possible within $\mathcal{R}$, and (ii) finetune SD at a lower resolution ($c_{in} = 256$). This minimizes the input distortion while retaining good generation at 512px and beyond.

Additionally, we evaluate Fig. 9 (bottom) the choice of `PromptTrain` when learning S^* by showing generations using three prompts from Fig. 10. We use *"an object with* $\boldsymbol{S}^*$ *texture"* as `PromptTrain` when learning the concept. This motivates our choices for learning S^* (Sec. 3.2).

Prompt engineering. We find that choosing the correct `PromptsGen` allows generating material images with the correct appearance. We ablate in Fig. 10 (left) different prompts by sampling 10 images per ACG class and processing them with our decomposition (Sec. 3.3). We then evaluate FID and KID against ACG annotations and rendered images. We find that the word "texture" improves synthesis over generic templates and removes additional context favoring top-view appearance. Furthermore, the text-to-image generation benefits from additional adjectives. Visual comparison of generated samples is in Fig. 10 (right).

4.6. 3D Scene editing.

We consider the extracted materials for scene editing applications. In Fig. 11, we present renderings of 3D scenes, replacing materials of objects (highlighted in insets) with ones extracted in *real-world* images with Material Palette. Note the realism of our jaguar (middle) and giraffe (right) sofas, or the bamboo wall (left).

5. Discussion

We introduced Material Palette, a comprehensive approach designed to extract tileable, high-resolution PBR materials from single real-world images. Although capable of extracting accurate materials, our method faces some unexpected limitations. For example, while prior methods may struggle to regress complex patterns we found it more challenging to capture simple uniform materials. In such cases, the concept collapses, leading to color artifacts, common in diffusion models. Another more predictable issue involves illumination ambiguities – particularly noticeable in shaded surfaces – causing inconsistent colors. Lastly, Material Palette is capable of making some geometric corrections, but cannot rectify slanted surfaces or account for strong distortion (perspective, lenses, depth of field). Addressing these shortcomings calls for further refinements. Material Palette shows very promising results on a newly introduced challenging task. We hope our work sparks interesting research in the same direction.

Acknowledgment. Research mainly funded by the French Agence Nationale de la Recherche (ANR), project SIGHT (ANR-20-CE23-0016). Fabio Pizzati was partially funded by KAUST (Grant DFR07910). Results obtained with GENCI-IDRIS (Grant 2023-AD011014389).

Material Palette: Extraction of Materials from a Single Image

Supplementary Material

In this supplementary material, we provide experimental details on Material Palette and baselines in Sec. A, and a description of our TexSD texture-generated dataset in Sec. B. Further, we report additional qualitative results in Sec. C and discuss limitations in Sec. D. Sources of our scenes and visuals are reported in Sec. E. We also illustrate the use of our extracted materials for 3D scene editing in the **supplementary video**.
Our code and dataset will be made public to foster research.

A. Experimental details

In this section, additional information is provided to ensure the reproducibility of our method and provide further insights. First, we highlight in Sec. A.1 details about the extraction of textures, how to ensure tileability, and also address issues with color shifts. We report decomposition details in Sec. A.2, and elaborate on the use of segmenters in Sec. A.3. Finally, Sec. A.4 contains evaluation details.

A.1. Tileable texture extraction

Crop selection. We extract crops from $\mathcal{R}$ at different resolutions $c_x = \{2^5, 2^6, 2^7, 2^8\}$ using a moving window with stride $c_x/5$. We retain crops of the largest scale that remain within the confines of $\mathcal{R}$, enabling us to select the optimal scale while accommodating the variable size of $\mathcal{R}$. This way, the maximum amount of information within each region is kept to learn the concept S^*.

Seamless generation. The Stable Diffusion checkpoint we use – version 1.5 [4] – is trained at 512px. Interestingly, we found that the VAE decoder can output at a resolution of 1024px without any noticeable performance drop. To ensure tileability, we use noise unrolling [57] and apply Poisson solving [41] to remove seams on the borders [5].

High-res generation. When generating textures larger than 1024×1024px, we stitch texture patches together. Although noise unrolling ensures the continuity of generated texture patches, seams still remain if patches are directly concatenated (Fig. 12, *left*). Similar to ControlMat [57], we decode overlapping 512×512 patches and apply a weighted average between them to remove visible seams from the resulting stitched texture (*center*). The patches overlap by 8 pixels on all sides ('blending' row). We present denoising times over 10 runs in Tab. 3 at 4 output resolutions (1K up to 8K) on a Tesla V100-16GB. Denoising times increase quadratically w.r.t. resolution size.

[4] https://huggingface.co/runwayml/stable-diffusion-v1-5
[5] https://github.com/bchao1/fast-poisson-image-editing

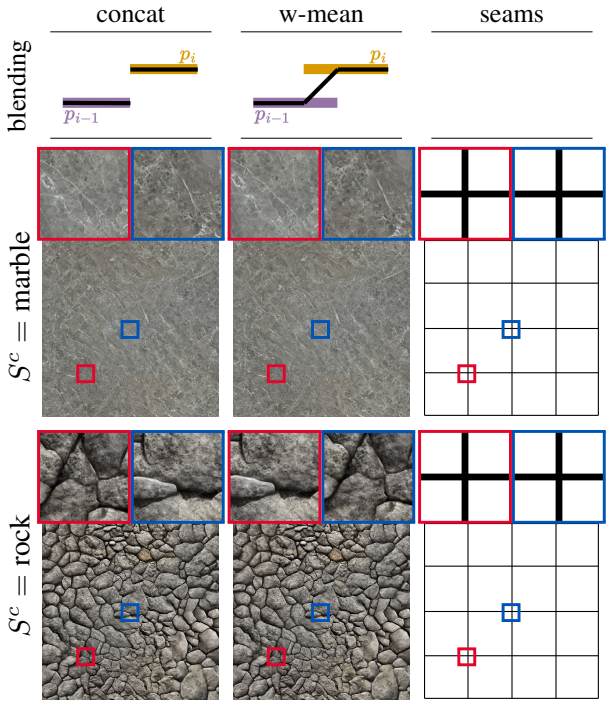


Figure 12. **Patched decoding.** Assembling patches decoded separately by SD. *left*: a direct concatenation of the patches leads to visible seams; *middle*: blending patches with a weighted average; *right* the location of the seams. The blending function shown *above* represents how two adjacent patches p_{i-1} and p_i are stitched together. We zoom in on two areas: red and blue.

Concept learning. To learn S^*, we fine-tune Stable Diffusion (v1.5 weights) [47] and run LoRA [22] Dreambooth [50] using PEFT [6]. We train for 800 steps with a batch size of 1, and a learning rate of $1e{-}4$. Input crops are resized to $c_{in} = 256$ (*cf.* Sec. 4.5) and we observed that augmentations are important to learn S^* without overfitting to the training views. Since textures are not all rotation invariant, we only apply random flip augmentations. Optimization times take no more than 5 minutes per S^* on a single Tesla V100-16GB. In Fig. 13, convergence times are ablated. With 400 steps our strategy yields good results.

Color shift correction. We solve the common color shift of diffusion model outputs [57] by normalizing the generated patch with the per-channel statistics of $\mathcal{R}$. We ablate the importance of this operation in Fig. 14 showing results of a LoRA DreamBooth trained on crops extracted from $\mathcal{I}$ with and without color correction. In practice, we directly renormalize the generated samples with the mean and std

[6] https://github.com/huggingface/peft

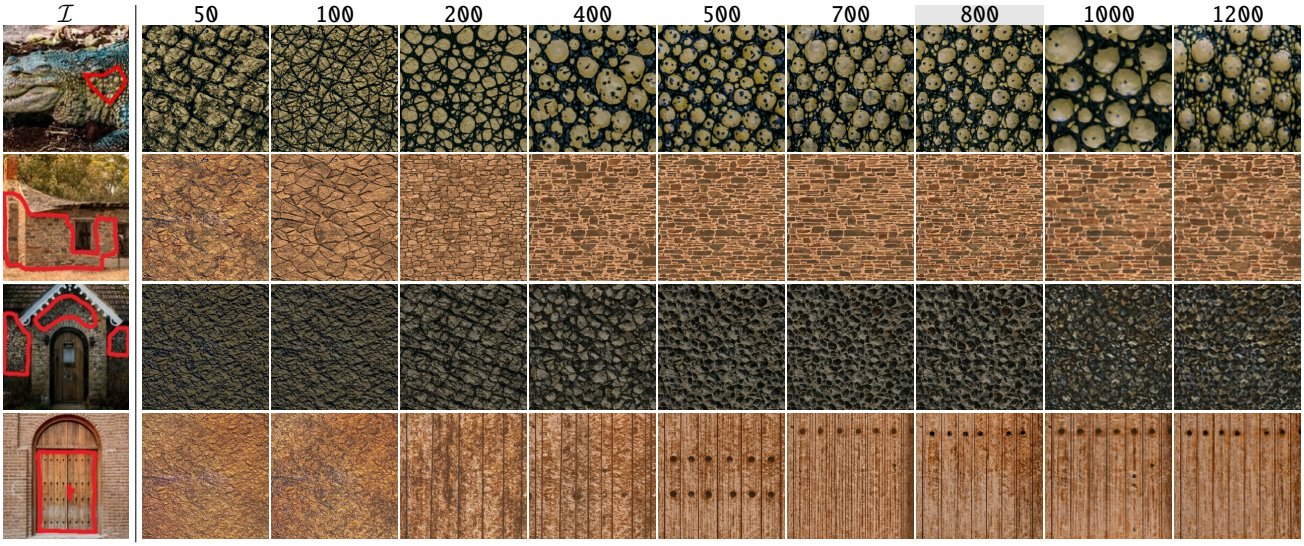


Figure 13. **Convergence.** Generations at different timesteps of the low-rank adaptation [22] of SD when learning S^* with Dreambooth [50]. Although in some cases, fewer timesteps are needed, we generally found that `800` steps work best to learn S^*.

	1K `1,024x1,024`	**2K** `2,048x2,048`	**4K** `4,096x4,096`	**8K** `8,192x8,192`
n	4	16	64	256
t	11s	43s	170s (≈3min)	685s (≈11min)

Table 3. **Denoising times.** All resolutions require 16GB. When dealing with resolutions larger than `1024px`, latent features and textures are processed in batches of 16 patches. For 50 denoising steps, $t \approx n * \texttt{2.7s}$, with n the total number of patches.

of $\mathcal{R}$. We highlight that the lack of renormalization on outputs (cols 2-3) leads to a noticeable color shift (darker tone). While applying this operation (cols 4-5) allows us to get a color that matches the input region $\mathcal{R}$.

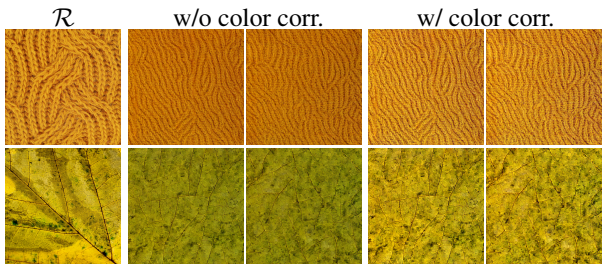


Figure 14. **Color correction.** We show the purpose of our color correction applied to generated textures. We show two generations per concept. The right-most outputs better match the color of $\mathcal{R}$.

A.2. Decomposition training

The decomposition network (*i.e.*, f) described in Sec. 4.1 is illustrated in Fig. 15. We train the source model $f^{\mathcal{S}}$ on $\mathcal{S}$ for 1,000 epochs and finetune on $\mathcal{S} + \mathcal{T}$ for 100 epochs. For augmentations, we apply random 512x512 crops, random horizontal/vertical flips as well as random 90° increment rotations. The decomposition maps A, N, and R are predicted by three separate decoders with $tanh$ activation functions, each outputting spatial 3-, 2-, and 1-channel maps, respectively. The z-axis of N is set to 1 and normalized. Finally A and R are mapped to $[0, 1]$. We apply the log on A and R, as well as all rendered views before computing the $L1$ losses to penalize larger errors. For source training, we use a batch size of 4, while when training on both source and target, we use batches of 2+2. Adam [25] is chosen as optimizer with a learning rate of $1e{-}4$ and λ is set to 0.1 [9].

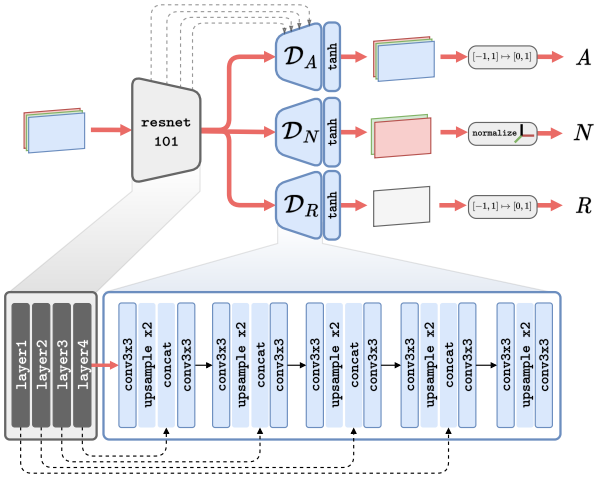


Figure 15. **Network architecture.** We use a Resnet-101 backbone with U-Net skip connections (*left*) to three separate custom decoders (*center*), one for each decomposed map. We apply different normalizations depending on the output type.

A.3. Segmenters

SAM. When using SAM [26], we retrieve the 16 largest masks from the image. We notice that SAM tends to output masks encompassing large portions of the image which are undesired for our task since they are likely to encompass more than one material. Instead, smaller masks are more prone to have a single material displayed. Thus, for overlapping masks with IoU > 0.5 we discard the *largest*.

Materialistic. Materialistic [54] requires an initial query point, so we select a random point within each of our user-defined regions and segment using the pre-trained weights.

User. Regions are defined by a single human annotator.

A.4. Evaluation

In order to use the annotation masks from the ACG dataset, in "Resemblance to SVBRDF dataset." evaluation (Tab. 2, left), we apply a class mapping between ACG and OS [61]. The mapping consists of **14** classes detailed in Tab. 4.

"brick" (**01**) → "brick"	"plastic" (**17**) → "plastic"
"cardboard" (**02**) → "cardboard"	"polished stone" (**18**) → "marble"/"granite"
"carpet" (**03**) → "carpet"	"stone" (**21**) → "paving stone"
"fabric" (**05**) → "fabric"	"tile" (**22**) → "tiles"
"laminate" (**11**) → "wood floor"	"wallpaper" (**23**) → "wallpaper"
"leather" (**12**) → "leather"	"wicker" (**24**) → "wicker"
"paper" (**16**) → "paper"	"wood" (**25**) → "wood"

Table 4. **Class mapping.** For evaluation purposes, we construct a mapping as the intersection of class sets from ACG and OS. Numbers correspond to label ids from OpenSurfaces [61].

Inference heuristics. For visualization *only*, we discard regions for which material images P_{SD} have a LPIPS > 0.65 w.r.t. P_C. This heuristic helps filter incorrect materials.

B. TexSD – Texture generated dataset

For generating our support dataset from Stable Diffusion, named TexSD, we compose an ontology (Tab. 5) of material classes starting from material categories in ACG (*top row*), complemented by classes obtained by providing ChatGPT (*bottom row*) the list of ACG classes and querying it to generate a list of "complementary classes" and "additional materials". We use SD [47] v1.5 and apply noise unrolling, outputting images directly at 1024px (*cf.* Sec. A). At inference, we use the same prompt templates as when conditioning with a S^*, except here a class name S^c is used. For instance, given S^c = pebbles, we can query SD with *"realistic pebbles texture in top view"* to generate a texture of pebbles. We recall the templates `PromptsGen`:

- *"realistic S^c texture in top view"* (■)
- *"high resolution realistic S^c texture in top view"* (■)
- *"top view realistic S^c texture"* (☒)
- *"top view realistic texture of S^c"* (☒)

We show some samples from TexSD in Fig. 16.

AmbientCG	*asphalt, bamboo, bark, bricks, candy, cardboard, carpet, chipboard, clay, concrete, cork, fabric, facade, foam, glazed terracotta, grass, gravel, ground, ice, ivory, lava, leather, marble, moss, paint, painted bricks, painted plaster, painted wood, paper, paving stones, planks, plaster, plastic, porcelain, road, rock, rocks, roofing tiles, shells, snow, sponge, tactile paving, terrazzo, tiles, wallpaper, wicker, wood, wood chips, wood floor, wood siding.*
ChatGPT	*acrylic, bamboo flooring, brocade, burlap, burnout velvet, canvas, carbon fiber, ceramic tiles, checkered, chiffon, cobblestone, concrete blocks, concrete pavers, corduroy, cotton, crocheted, crushed velvet, crystal, damask, denim, dupioni, embossed, felt, fiberboard, fur, gingham, giraffe fur, glass, granite, hammered velvet, hempcrete, herringbone, jacquard, knitted, lace, linen, linoleum, linoleum, mahogany wood, mosaic tiles, oak wood, octagonal paving, onyx, organza, particle board, pine wood, plaid, plywood, quartz top, quartzite top, rammed earth, reptile skin, resin, ribbed, rubber, rustic wood, sand, sandstone, sateen, satin, shantung, silk, silk velvet, slate, snake skin, straw bale, stucco, suede, taffeta, teak wood, terracotta, tiger fur, travertine, tweed, velvet, vinyl, vinyl, woodgrain, wool, woven, zebra fur.*

Table 5. TexSD **ontology.** Classes used to generate the dataset. We complement the classes existing in ACG using ChatGPT.

Figure 16. TexSD**.** The class names S^c are given left-to-right. **1st row:** *bambo*, *candy*, *brick wall*, *tiger fur*; **2nd row:** *zebra fur*, *corduroy*, *veneer*; **3rd row:** *fur*, *fabric*, *giraffe fur*, *granite*, *checkered pattern*; **4th row:** *onyx*, *gravel*, *leather*, *marble*; **5th row:** *marble*, *marble mosaic*, *oak wood*; **6th row:** *pebbles*, *painted wood*, *porcelain*, *quartzite top*, *wood floor*; **7th row:** *terrazzo*, *rustic wood*, *slate*, *rock*; **last row:** *damask*, *resin*, *fur*.

C. Qualitative results

In this section, we present additional qualitative results, for easing visualization of the high-quality extraction of textures and related materials.

Texture extraction from planar surfaces In this experiment, we further highlight possible applications of our texture extraction. Assuming an available top-view material picture, we can sample crops from the whole image (hence $\mathcal{R} = \mathcal{I}$) and generate a tileable texture with our proposed texture extraction pipeline. The resulting images in Fig. 17 are readily usable for any CG application.

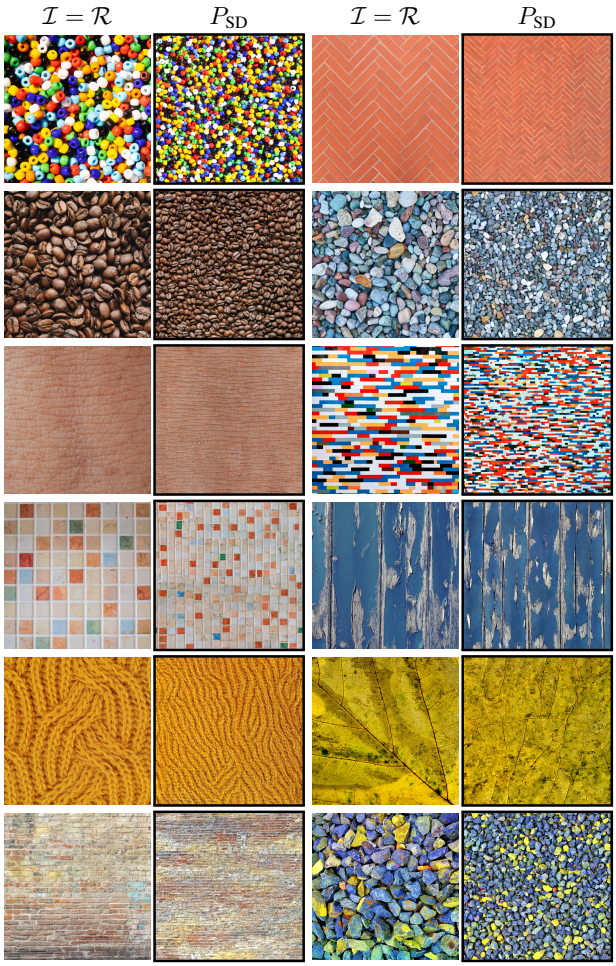


Figure 17. **Texture extraction.** Here, textures are extracted from planar objects that present interesting patterns. Overall S^* has the powerful ability to encapsulate the pattern. This conditioning allows generating tileable high-resolution textures (here `1024²`) which can be a valuable tool for CG artists.

Large textures extraction. In Fig. 18, we present high-resolution examples of extracted textures from real images. This shows further evidence of how Material Palette is able to correctly reproduce the learned texture and extend it to arbitrary resolutions. We highlight the high quality and the absence of visible seams even at 8K resolution.

Comparison with ACG renderings. We compare qualitatively materials in ACG and materials extracted by Material Palette in Figs. 19 and 20. We highlight how common materials emerge in both cases (*e.g.*, bricks, stones, etc.) while our extracted materials exhibit unique ones (*e.g.*, fur, fruits) challenging to extract from controlled scenarios.

Additional visualization. We also extend the visualization present in the main paper. In Fig. 21, we present web-collected images and corresponding material palettes (similar to main paper Fig. 7) but this time showing three samples per material. In Fig. 22, we propose additional results for the end-to-end estimation task (main paper, Fig. 8)

Blender rendering. All 3D renders are made in Blender, using the Principled BSDF shader with the predicted albedo, roughness, and normals map. To improve realism, we use displacement maps inferred from our surface normals using DeepBump [7]. The renderings are done using the rendering engine 'Cycles', with displacement as bumps. Some objects (*e.g.* floor) have a lower displacement scale than others to avoid irregular surfaces that would be unrealistic. Importantly, we do not adjust the shader parameters *per*-material to preserve the true appearance of our extracted materials.

[7] https://github.com/HugoTini/DeepBump

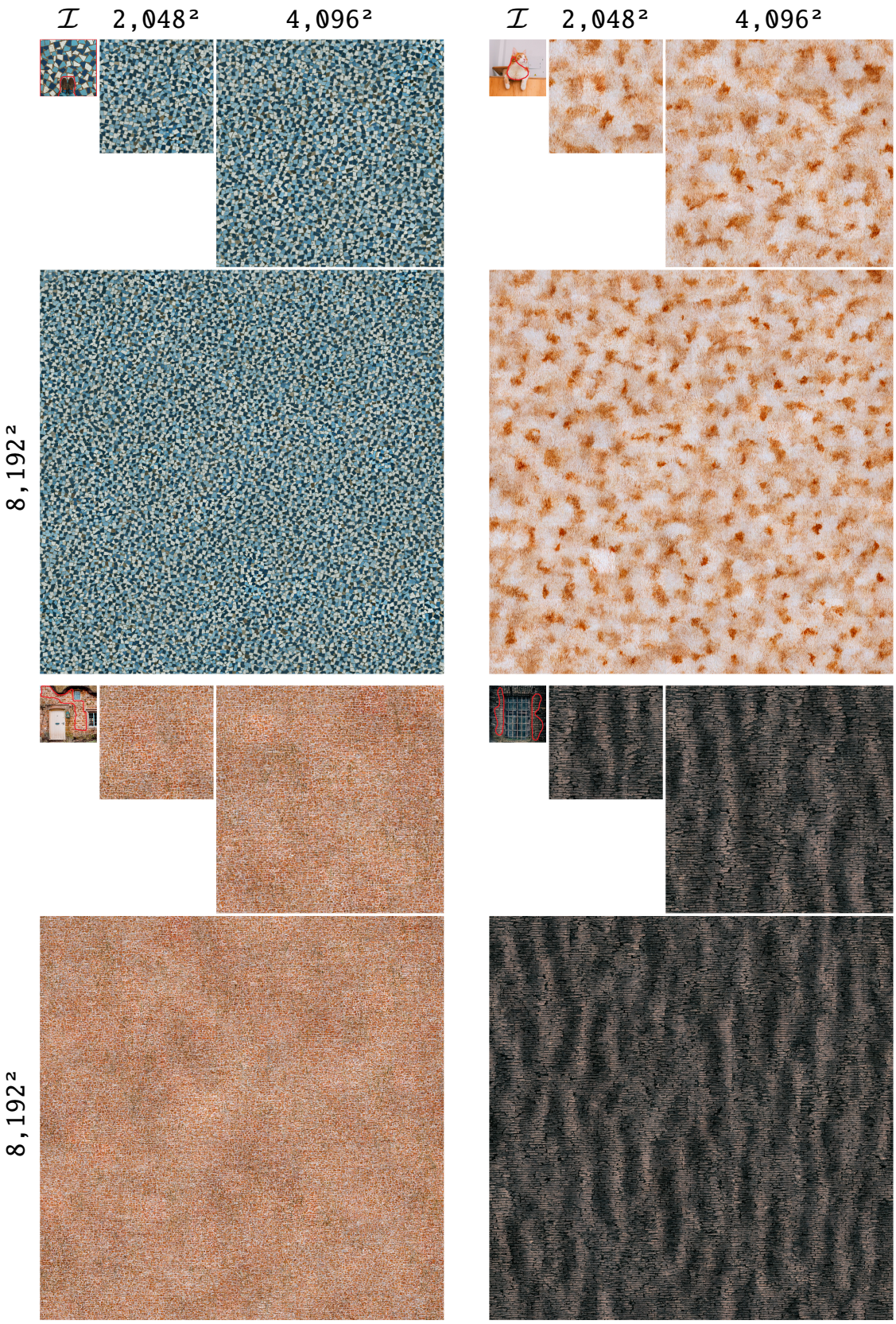


Figure 18. **High resolution outputs.** We provide generations for each image at three resolutions.

Figure 19. **Sphere renderings (a).** Showing spheres of materials from AmbientCG (*left*) and ones extracted from real-world images using our proposed pipeline, Material Palette (*right*). Renderings are done in Blender with a HDRI map.

Figure 20. **Sphere renderings (b).** Showing spheres of materials from AmbientCG (*left*) and ones extracted from real-world images using our proposed pipeline, Material Palette (*right*). Renderings are done in Blender with a HDRI map.

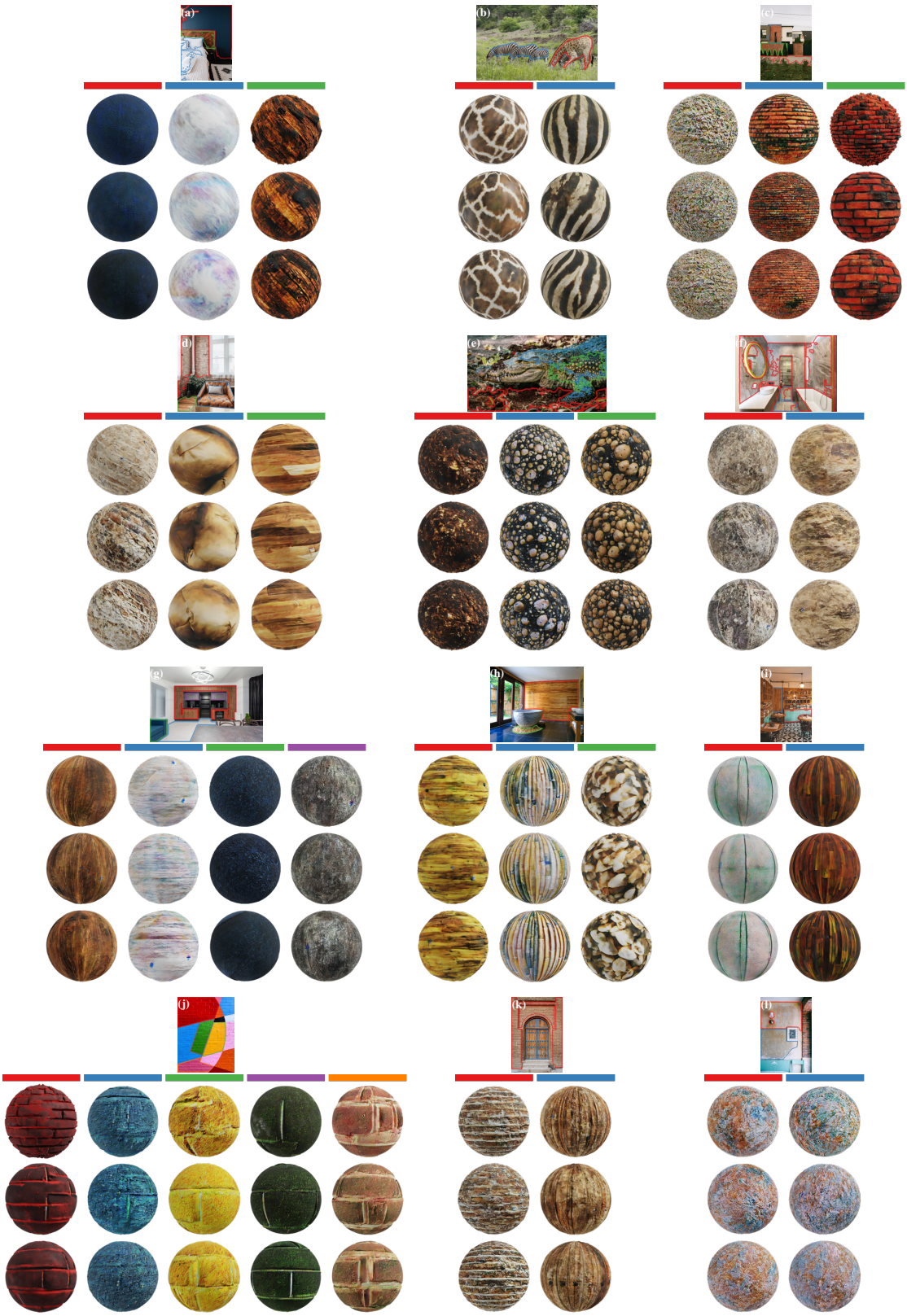


Figure 21. **Material palettes.** We present some results of our method Material Palette on images gathered on the web. Different from the palettes presented in the main paper, here we show three variations per extracted material.

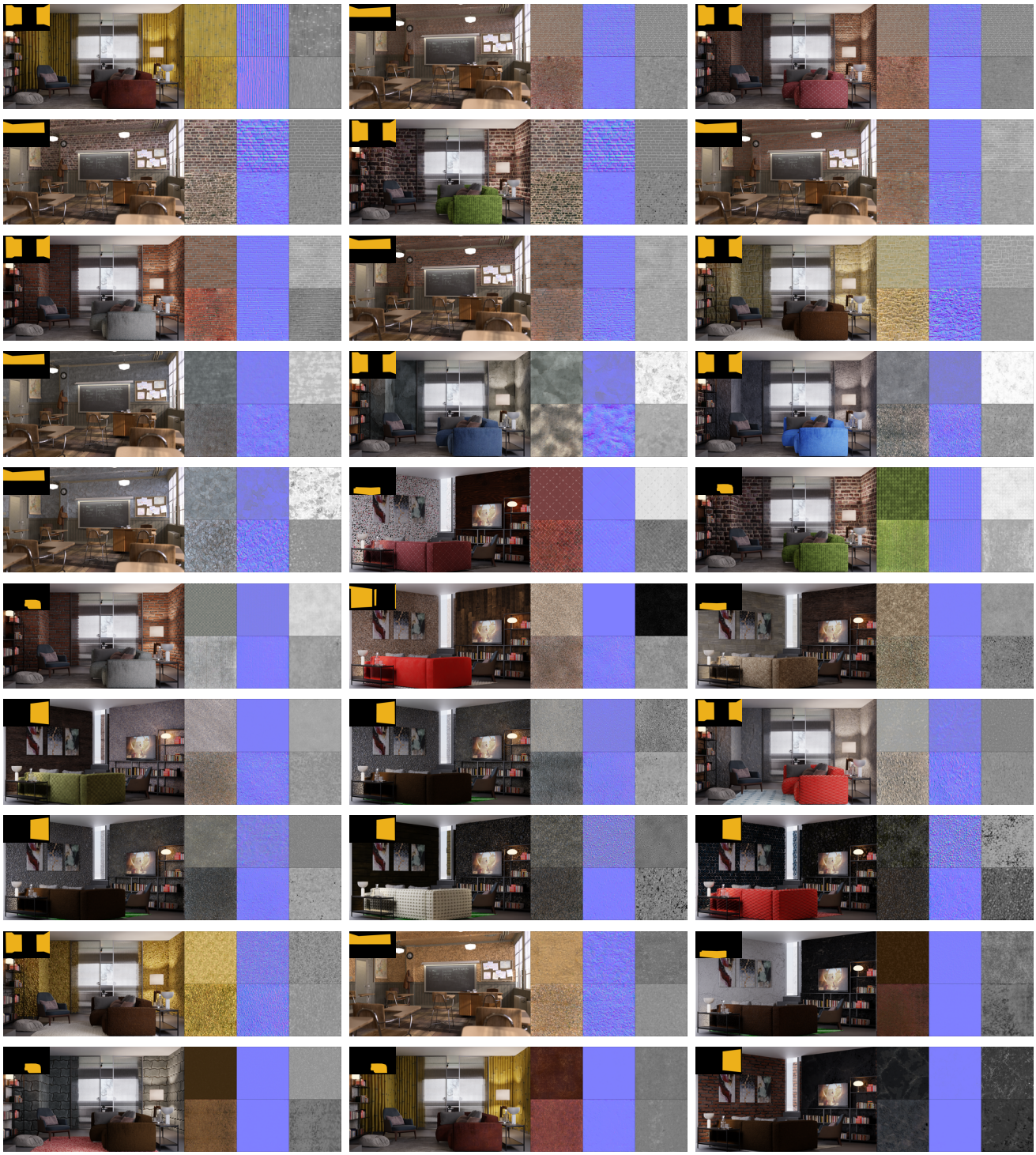


Figure 22. **End to end evaluation.** We edit 3D scenes with ACG materials rendering different scenes and viewpoints (each column, left). The edited regions are shown in the inset. Then, we use Material Palette to extract edited materials. This allows us to compare Material Palette extractions w.r.t. ACG ground truth (right).

D. Limitations

Material Palette lacks contextual information during the material decomposition stage and therefore may perform poorly with strong shading due to lighting or complex geometry. For complex materials, we show failure cases in Fig. 23. Overall, we notice that our pipeline may present incorrect S^* estimation for complex patterns with strong geometrical constraints (rows 1-2). Although our approach corrects geometry and homogenizes light information, errors may still be present with highly tilted surfaces or strong shadows (rows 3-4). Finally, the SD network tends to generate artifacts for highly saturated materials (last two rows). However, for uniform materials as in Fig. 24, we note we are still performing well even in complex lighting.

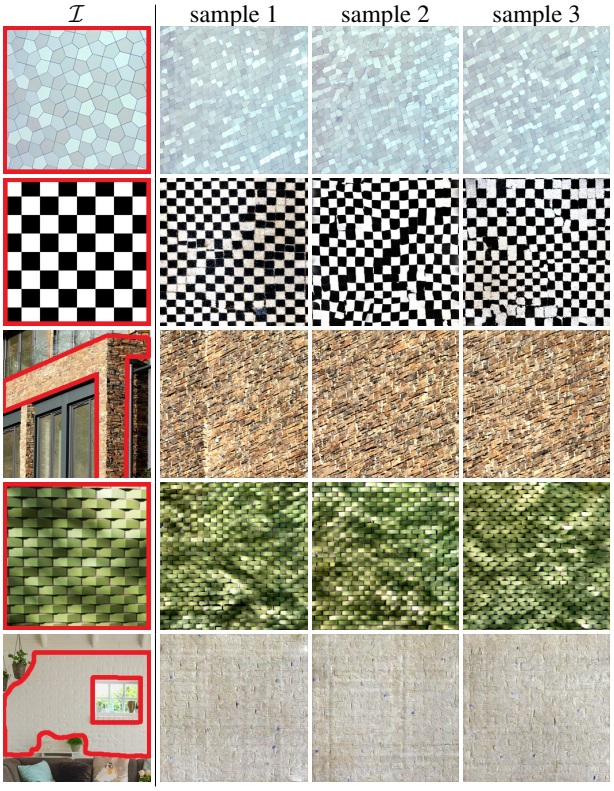


Figure 23. **Failure cases.** Complex patterns are difficult to extract (*first two rows*). Highly tilted surfaces create a bias in the optimization and are not properly rectified (*third row*). Shadows are entangled in the extraction (*fourth row*). We notice some weaknesses when trying to estimate from highly saturated objects (notice the blue spots *last row*).

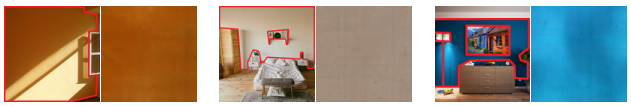


Figure 24. **Complex lighting.** Uniform material extraction in complex lighting

E. Acknowledgements

We provide the list of resources (images and 3D scenes) that may have been used during this project in Tab. 6.

name	author	resource link
private flat	Flavio Della Tommasa	https://blenderartists.org/t/private-flat/1294642
classroom	Christophe Seux	https://download.blender.org/demo/test/classroom.zip
brown wooden table	Skitterphoto	https://www.pexels.com/photo/brown-wooden-center-table-584399/
white couch table	QuarkStudio	https://www.pexels.com/photo/white-padded-couch-in-front-of-black-wooden-center-table-2506986/
wooden kitchen	Max Rahubovskiy	https://www.pexels.com/photo/contemporary-kitchen-and-dining-room-interior-with-shiny-chandelier-7045364/
empty black chair	Saviesa Home	https://www.pexels.com/photo/empty-two-black-chair-2089696/
mosque man	Mihai Vlasceanu	https://www.pexels.com/photo/mosque-door-surrounded-by-decorative-patterns-18538944/
fes door front	Gül Işık	https://www.pexels.com/photo/arabian-front-door-with-arch-and-mosaic-13794309/
door front	Faruk Tokluoğlu	https://www.pexels.com/photo/decorative-patterns-on-entrance-to-mosque-13811595/
innsbruck	Waldemar	https://www.pexels.com/photo/colourful-houses-in-innsbruck-austria-5052366/
four stools	Pixabay	https://www.pexels.com/photo/four-gray-bar-stools-in-front-of-kitchen-countertop-279648/
ivy brick	Boys In Bristol SmokZ	https://www.pexels.com/photo/ivy-growing-on-building-brick-wall-15869273/
fall modern	Tanya Pro	https://unsplash.com/photos/brown-and-white-concrete-building-oPyu636ASPw
brown brick house	pixabay	https://www.pexels.com/photo/brown-wall-paint-house-near-at-garden-209315/
hold interior home	Maria Orlova	https://www.pexels.com/photo/interior-of-cozy-room-in-old-house-4906484/
colored brick	Fatih	https://unsplash.com/photos/pink-and-yellow-wall-paint-kgqu_qs3B78
joint homes	Erol Ahmed	https://unsplash.com/photos/two-white-wooden-doors-with-grills-FTy5VSGIfiQ
cat	Camilo Ospina	https://www.pexels.com/photo/cat-on-furniture-in-room-18597909/
parrot	Dušan veverkolog	https://unsplash.com/photos/blue-and-yellow-parrot-perched-on-tree-during-daytime-2HgyU4YwraQ
crocodile	Philipp Deus	https://www.pexels.com/photo/a-crocodile-in-close-up-photography-3741492/
church	David Besh	https://www.pexels.com/photo/white-and-black-concrete-house-969260/
okapi	Magda Ehlers	https://www.pexels.com/photo/okapi-animal-in-zoo-12788062/
brick modern	Expect Best	https://www.pexels.com/photo/facade-of-modern-building-against-clear-sky-323781/
blue sofa	Max Rahubovskiy	https://www.pexels.com/photo/interior-of-spacious-living-room-with-minimalist-furniture-6492397/
coffee shop	Emre Can Acer	https://www.pexels.com/photo/three-brown-wooden-dinette-sets-2079448/
damaged bricks	Pixabay	https://www.pexels.com/photo/abandoned-ancient-antique-architecture-235986/
modern tub	Max Rahubovski	https://www.pexels.com/photo/bathroom-interior-with-sink-on-counter-near-mirror-and-bath-6444967/
mountain tub	Max Rahubovskiy	https://www.pexels.com/photo/interior-of-bathroom-with-mirror-above-sink-7061423/
merrigum house	wikimedia	https://commons.wikimedia.org/wiki/File:MerrigumHouse3.JPG
modern mansion	Max Rahubovskiy	https://www.pexels.com/photo/contemporary-villa-against-cloudy-blue-sky-7031595/
oldschool kitchen	Polina Kovaleva	https://www.pexels.com/photo/kitchen-room-with-ornamental-plants-5644353/
sofa space	Charlotte May	https://www.pexels.com/photo/sofa-with-cushions-in-cozy-apartment-5825404/
vintage living	Derick McKinney	https://unsplash.com/photos/grey-pillow-on-chair-Tc349sxFa2U
wooden door	Hisham Yahya	https://www.pexels.com/photo/wooden-door-of-a-bricked-wall-2909959/
wooden furniture	PNW Production	https://www.pexels.com/photo/brown-wooden-cabinet-beside-white-wall-8251248/
rundown home	Graham Meyer	https://unsplash.com/photos/brown-brick-house-near-green-trees-during-daytime-nV1phwYx3J0
stone house	Bernard Hermant	https://unsplash.com/photos/brown-and-white-concrete-house-near-green-grass-field-during-daytime-SLW8vO8Erwc
river stone	Pixabay	https://pixabay.com/photos/house-river-running-waterfall-2548478/
cottage	Alex Baber	https://unsplash.com/photos/closed-brown-house-door-tTKgcVWPdLM
ivy bricks	Liv Cashman	https://unsplash.com/photos/blue-wooden-door-on-brown-brick-house-oJAM9h0158c
green home	Binyamin Mellish	https://www.pexels.com/photo/lighted-beige-house-1396132/
wooden door	Tirachard Kumtanom	https://www.pexels.com/photo/photo-of-wooden-door-near-window-887822/
large garage	Curtis Adams	https://www.pexels.com/photo/photo-of-house-3555615/
marble bedroom	Max Rahubovskiy	https://www.pexels.com/photo/bedroom-interior-with-vases-with-branch-near-bed-and-mirror-6480209/
concrete wall	Henry Co.	https://www.pexels.com/photo/brown-metal-staircase-and-gray-painted-wall-1246078/
ancient wall	Pixabay	https://www.pexels.com/photo/ancient-architecture-brick-brick-wall-277544/
brick fireplace	Max Rahubovskiy	https://www.pexels.com/photo/brick-fireplace-in-a-living-room-7746461/
bricks paradise	Charles Parker	https://www.pexels.com/photo/weathered-brick-residential-building-on-city-street-5847374/
home couch	Max Rahubovskiy	https://www.pexels.com/photo/room-with-open-kitchen-near-sofa-6782429/
orange wool	Engin Akyurt	https://www.pexels.com/photo/brown-knitted-textile-1487703/
green tiles	Sutee Pheera	https://www.pexels.com/photo/green-woven-pavement-570047/
chevron bricks	Mitchell Luo	https://www.pexels.com/photo/close-up-shot-of-a-brick-wall-4115538/
yellow leaf	Hilary Halliwell	https://www.pexels.com/photo/close-up-photography-of-dried-leaf-612328/
blue polygons	Damir Mijailovic	https://www.pexels.com/photo/abstract-background-of-wall-with-geometric-ornament-3695238/
yellow facade	Athena	https://www.pexels.com/photo/yellow-and-black-pattern-2254103/
blue tiles	Natã Romualdo	https://www.pexels.com/photo/brown-low-top-sneakers-2904284/
human skin	Karolina Grabowska	https://www.pexels.com/photo/close-up-view-of-human-skin-4046567/
beads	Magda Ehlers	https://www.pexels.com/photo/assorted-color-beads-1331705/
colored stones	Engin Akyurt	https://www.pexels.com/photo/close-up-photo-of-assorted-rocks-3129641/
multicolored tiles	Monstera Production	https://www.pexels.com/photo/abstract-backdrop-of-multicolored-tiled-floor-7794425/
red planks	Magda Ehlers	https://www.pexels.com/photo/red-wooden-surface-960137/
legos	Omar Flores	https://unsplash.com/photos/blue-red-and-white-artwork-lQT_bOWtysE
checkered board	Wikimedia Commons	https://commons.wikimedia.org/wiki/Category:SVG_checkered_patterns
old wood	Suzy Hazelwood	https://www.pexels.com/photo/gray-wall-with-cracked-blue-paint-2096543/
coffee beans	Polina Tankilevitch	https://www.pexels.com/photo/brown-coffee-beans-4109743/
crumbling paint	Laura Tancredi	https://www.pexels.com/photo/shabby-brick-wall-with-crumbling-paint-7078669/
graffiti brick	ShonEjai	https://www.pexels.com/photo/closeup-photo-of-brown-brick-wall-1227511/
minerals	msvr	https://www.pexels.com/photo/gray-and-yellow-gravel-stones-997704/
white paint	Henry Co. Henry Co.	https://www.pexels.com/photo/white-painted-wall-1939485/

Table 6. **Resources.** List of images and 3D scenes used in this project. All properties have open licenses.

References

[1] AmbientCG. Pbr repository. https://www.ambientcg.com, 2017. Accessed: 2023-04-01. 3, 4, 5

[2] Louis-Philippe Asselin, Denis Laurendeau, and Jean-François Lalonde. Deep svbrdf estimation on real materials. In *3DV*, 2020. 1, 2

[3] Jonathan T Barron and Jitendra Malik. Intrinsic scene properties from a single rgb-d image. In *CVPR*, 2013. 2

[4] Joseph R Bartels, Jian Wang, William Whittaker, Srinivasa G Narasimhan, et al. Agile depth sensing using triangulation light curtains. In *ICCV*, 2019. 1

[5] Sean Bell, Paul Upchurch, Noah Snavely, and Kavita Bala. OpenSurfaces: A richly annotated catalog of surface appearance. In *ACM TOG*, 2013. 5, 6

[6] Urs Bergmann, Nikolay Jetchev, and Roland Vollgraf. Learning texture manifolds with the periodic spatial gan. In *ICML*, 2017. 5

[7] George Cazenavette, Tongzhou Wang, Antonio Torralba, Alexei A Efros, and Jun-Yan Zhu. Wearable imagenet: Synthesizing tileable textures via dataset distillation. In *CVPR-W*, 2022. 2

[8] CGBookCase. Pbr repository. https://www.cgbookcase.com, 2019. Accessed: 2023-04-01. 5

[9] Valentin Deschaintre, Miika Aittala, Fredo Durand, George Drettakis, and Adrien Bousseau. Single-image svbrdf capture with a rendering-aware deep network. *ACM TOG*, 2018. 2, 4, 6, 10

[10] Valentin Deschaintre, George Drettakis, and Adrien Bousseau. Guided fine-tuning for large-scale material transfer. *Comput. Graph. Forum*, 2020. 2

[11] Valentin Deschaintre, Yiming Lin, and Abhijeet Ghosh. Deep polarization imaging for 3d shape and svbrdf acquisition. In *CVPR*, 2021. 2

[12] Olga Diamanti, Connelly Barnes, Sylvain Paris, Eli Shechtman, and Olga Sorkine-Hornung. Synthesis of complex image appearance from limited exemplars. *ACM TOG*, 2015. 2

[13] Alexei A Efros and William T Freeman. Image quilting for texture synthesis and transfer. In *ACM TOG*, 2001. 5

[14] Alexei A Efros and Thomas K Leung. Texture synthesis by non-parametric sampling. In *ICCV*, 1999. 5

[15] Duan Gao, Xiao Li, Yue Dong, Pieter Peers, Kun Xu, and Xin Tong. Deep inverse rendering for high-resolution svbrdf estimation from an arbitrary number of images. *ACM TOG*, 2019. 2

[16] Mathieu Garon, Kalyan Sunkavalli, Sunil Hadap, Nathan Carr, and Jean-Francois Lalonde. Fast spatially-varying indoor lighting estimation. In *CVPR*, 2019. 2

[17] Leon Gatys, Alexander S Ecker, and Matthias Bethge. Texture synthesis using convolutional neural networks. In *NeurIPS*, 2015. 5

[18] Thomas Germer, Joanne C Zwinkels, and Benjamin K Tsai. *Spectrophotometry: Accurate measurement of optical properties of materials*. Elsevier, 2014. 1

[19] Darya Guarnera, Giuseppe Claudio Guarnera, Abhijeet Ghosh, Cornelia Denk, and Mashhuda Glencross. Brdf representation and acquisition. In *Comput. Graph. Forum*, 2016. 3

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. 5

[21] Berthold KP Horn. Determining lightness from an image. *CGIP*, 1974. 2

[22] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *ICLR*, 2022. 4, 5, 9, 10

[23] Carlo Innamorati, Tobias Ritschel, Tim Weyrich, and Niloy J Mitra. Decomposing single images for layered photo retouching. In *Comput. Graph. Forum*, 2017. 2

[24] Alen Joy and Charalambos Poullis. Multi-view gradient consistency for svbrdf estimation of complex scenes under natural illumination. *arXiv*, 2022. 2

[25] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2017. 10

[26] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *ICCV*, 2023. 1, 5, 6, 7, 11

[27] Dong-Hyun Lee et al. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *ICML*, 2013. 4

[28] Xiao Li, Yue Dong, Pieter Peers, and Xin Tong. Modeling surface appearance from a single photograph using self-augmented convolutional neural networks. *ACM TOG*, 2017. 2

[29] Xueting Li, Xiaolong Wang, Ming-Hsuan Yang, Alexei A Efros, and Sifei Liu. Scraping textures from natural images for synthesis and editing. In *ECCV*, 2022. 2, 5

[30] Zhengqi Li and Noah Snavely. Cgintrinsics: Better intrinsic image decomposition through physically-based rendering. In *ECCV*, 2018. 2

[31] Zhengqin Li, Mohammad Shafiei, Ravi Ramamoorthi, Kalyan Sunkavalli, and Manmohan Chandraker. Inverse rendering for complex indoor scenes: Shape, spatially-varying lighting and svbrdf from a single image. In *CVPR*, 2020. 2

[32] Zhengqin Li, Jia Shi, Sai Bi, Rui Zhu, Kalyan Sunkavalli, Miloš Hašan, Zexiang Xu, Ravi Ramamoorthi, and Manmohan Chandraker. Physically-based editing of indoor scene lighting from a single image. In *ECCV*. Springer, 2022. 2

[33] Zicheng Liao, Kevin Karsch, Hongyi Zhang, and David Forsyth. An approximate shading model with detail decomposition for object relighting. *IJCV*, 2019. 2

[34] Yunfei Liu, Yu Li, Shaodi You, and Feng Lu. Unsupervised learning for intrinsic image decomposition from a single image. In *CVPR*, 2020. 2

[35] Ivan Lopes, Tuan-Hung Vu, and Raoul de Charette. Cross-task attention mechanism for dense multi-task learning. In *WACV*, 2023. 5

[36] Rosalie Martin, Arthur Roullier, Romain Rouffet, Adrien Kaiser, and Tamy Boubekeur. Materia: Single image high-resolution material capture in the wild. In *Comput. Graph. Forum*, 2022. 2

[37] Tom Monnier, Matthew Fisher, Alexei A Efros, and Mathieu Aubry. Share with thy neighbors: Single-view reconstruction by cross-instance consistency. In *ECCV*, 2022. 2

[38] Takuya Narihira, Michael Maire, and Stella X Yu. Direct intrinsics: Learning albedo-shading decomposition by convolutional regression. In *ICCV*, 2015. 2

[39] Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. Glide: Towards photorealistic image generation and editing with text-guided diffusion models. In *ICML*, 2022. 2

[40] OpenAI. Chatgpt. chat.openai.org, 2023. Accessed: 2023-05-20. 4

[41] Patrick Pérez, Michel Gangnet, and Andrew Blake. Poisson image editing. In *Seminal Graphics Papers: Pushing the Boundaries*. SIGGRAPH, 2023. 5, 9

[42] PolyHaven. Pbr repository. https://www.polyhaven.com, 2021. Accessed: 2023-04-01. 5

[43] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021. 6, 7

[44] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *ICML*, 2021. 2

[45] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv*, 2022. 2

[46] Carlos Rodriguez-Pardo, Henar Domínguez-Elvira, David Pascual-Hernández, and Elena Garces. Umat: Uncertainty-aware single image high resolution material capture. In *CVPR*, 2023. 2

[47] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022. 2, 3, 4, 5, 9, 11

[48] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015. 5

[49] Amir Rosenberger, Daniel Cohen-Or, and Dani Lischinski. Layered shape synthesis: automatic generation of control maps for non-stationary textures. *ACM TOG*, 2009. 2

[50] Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. In *CVPR*, 2023. 4, 5, 9, 10

[51] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. In *NeurIPS*, 2022. 2

[52] Axel Sauer, Tero Karras, Samuli Laine, Andreas Geiger, and Timo Aila. Stylegan-t: Unlocking the power of gans for fast large-scale text-to-image synthesis. In *ICML*, 2023. 2

[53] Soumyadip Sengupta, Jinwei Gu, Kihwan Kim, Guilin Liu, David W Jacobs, and Jan Kautz. Neural inverse rendering of an indoor scene from a single image. In *ICCV*, 2019. 2

[54] Prafull Sharma, Julien Philip, Michaël Gharbi, William T. Freeman, Fredo Durand, and Valentin Deschaintre. Materialistic: Selecting similar materials in images. In *ACM TOG*, 2023. 5, 6, 7, 11

[55] Yichuan Tang, Ruslan Salakhutdinov, and Geoffrey Hinton. Deep lambertian networks. *arXiv*, 2012. 2

[56] Giuseppe Vecchio, Simone Palazzo, and Concetto Spampinato. Surfacenet: Adversarial svbrdf estimation from a single image. In *ICCV*, 2021. 2, 6

[57] Giuseppe Vecchio, Rosalie Martin, Arthur Roullier, Adrien Kaiser, Romain Rouffet, Valentin Deschaintre, and Tamy Boubekeur. Controlmat: A controlled generative approach to material capture. *arXiv*, 2023. 2, 4, 5, 9

[58] Giuseppe Vecchio, Renato Sortino, Simone Palazzo, and Concetto Spampinato. Matfuse: Controllable material generation with diffusion models, 2023. 2

[59] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *T-IP*, 2004. 6

[60] Shangzhe Wu, Christian Rupprecht, and Andrea Vedaldi. Unsupervised learning of probably symmetric deformable 3d objects from images in the wild. In *CVPR*, 2020. 2

[61] Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified perceptual parsing for scene understanding. In *ECCV*, 2018. 6, 11

[62] Tao Xu, Pengchuan Zhang, Qiuyuan Huang, Han Zhang, Zhe Gan, Xiaolei Huang, and Xiaodong He. Attngan: Fine-grained text to image generation with attentional generative adversarial networks. In *CVPR*, 2018. 2

[63] Yu-Ying Yeh, Zhengqin Li, Yannick Hold-Geoffroy, Rui Zhu, Zexiang Xu, Miloš Hašan, Kalyan Sunkavalli, and Manmohan Chandraker. Photoscene: Photorealistic material and lighting transfer for indoor scenes. In *CVPR*, 2022. 2

[64] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris N Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *ICCV*, 2017. 2

[65] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. 4, 6, 7

[66] Hao Zhou, Xiang Yu, and David W Jacobs. Glosh: Global-local spherical harmonics for intrinsic image decomposition. In *ICCV*, 2019. 2

[67] Minfeng Zhu, Pingbo Pan, Wei Chen, and Yi Yang. Dm-gan: Dynamic memory generative adversarial networks for text-to-image synthesis. In *CVPR*, 2019. 2