

Efficient Physically-based Retexturing in Image-space

QINGQIN HUA, Saarland University, Germany and Style3D, China

PASCAL GRITTMANN, Saarland University, Germany

JUNQIU ZHU, Shandong University, China

LIKANG LUO, Style3D, China

PHILIPP SLUSALLEK, Saarland University, Germany and DFKI, Germany

CHI WANG*, Style3D, China

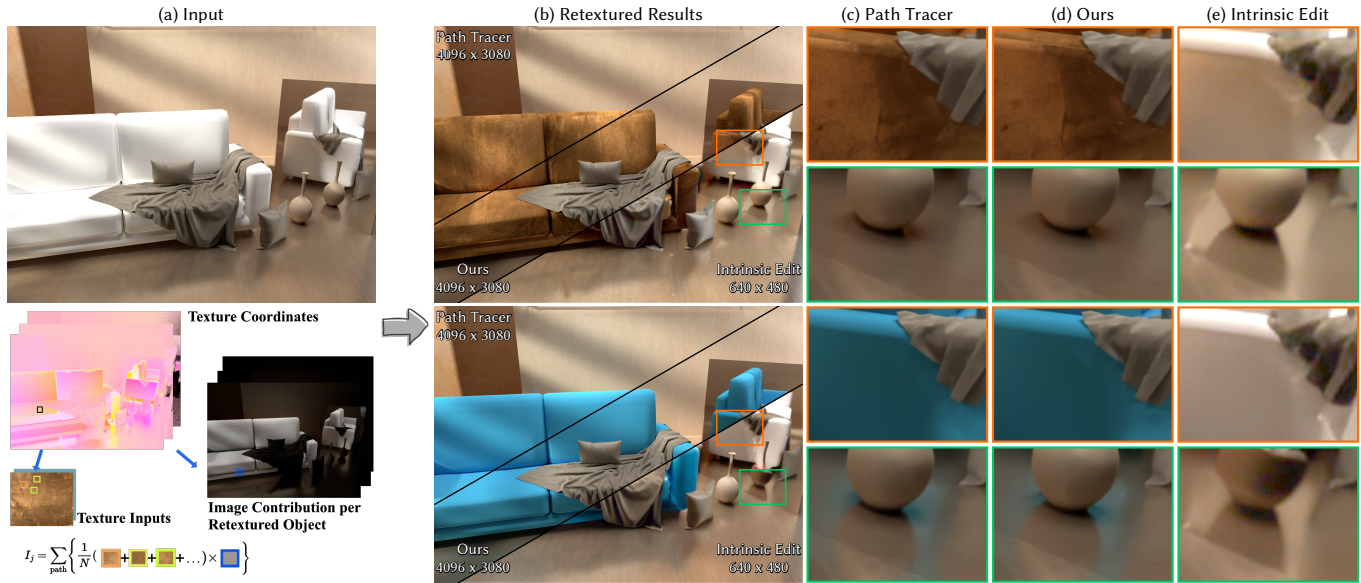


Fig. 1. Starting from a path-traced image containing a sofa as the *hero* object to be retextured (a), our method (d) efficiently generates multiple retextured results using on-the-fly texture sampling and precomputed AOVs efficiently (11s for a 4K image, 3s for a 1080p image on the CPU). Compared to prior image-based editing work [Lyu et al. 2025] (e), ours accurately captures complex lighting effects and changes in indirect reflection due to retexturing, closely matching the ground-truth path-traced results (c).

Texture plays a key role in the visual appearance of 3D objects in rendered images. In production, artists often spend a significant amount of time iterating on textures and re-rendering scenes to achieve the desired look. This workflow is generally time-consuming, and the cost increases with detailed geometry, challenging lighting, and material properties. While image-space

editing techniques offer an alternative, editing on complex global illumination and fine-grained appearance details during retexturing are highly challenging, and an accurate and efficient solution is yet to be found.

We propose a method that can efficiently render many versions of the same image with different textures applied to an object. We avoid expensive re-tracing of light paths by aggregating appropriate auxiliary information (e.g., radiance, UV coordinates) during a single initial rendering pass. Retexturing can then be done as an image-based post-process for arbitrarily many new textures. Our method generates photorealistic results with complex indirect illumination across a wide range of scenes within a few seconds, even at 4K resolution, and naturally extends to video sequences.

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

Additional Key Words and Phrases: global illumination, image-based rendering, texture mapping

ACM Reference Format:

Qingqin Hua, Pascal Grittmann, Junqiu Zhu, Likang Luo, Philipp Slusallek, and Chi Wang. 2026. Efficient Physically-based Retexturing in Image-space. In *SIGGRAPH Asia 2026 Conference Papers (SA Conference Papers '26)*, December 01–04, 2026, Kuala Lumpur, Malaysia. ACM, New York, NY, USA, 11 pages. <https://doi.org/10.1145/3829340.3842305>

*Project lead and corresponding author.

Authors' Contact Information: Qingqin Hua, Saarland University, Saarbrücken, Germany and Style3D, Hangzhou, China, hua@cg.uni-saarland.de; Pascal Grittmann, Saarland University, Saarbrücken, Germany, grittmann@cg.uni-saarland.de; Junqiu Zhu, Shandong University, Jinan, China, zhujunqiu@mail.sdu.edu.cn; Likang Luo, Style3D, Hangzhou, China, style_luo@163.com; Philipp Slusallek, Saarland University, Saarbrücken, Germany and DFKI, Saarbrücken, Germany, philipp.slusallek@dfki.de; Chi Wang, Style3D, Hangzhou, China, wangchiemail@gmail.com.



This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License.

SA Conference Papers '26, Kuala Lumpur, Malaysia

© 2026 Copyright held by the owner/author(s).

ACM ISBN 979-8-4007-2842-6/2026/12

<https://doi.org/10.1145/3829340.3842305>

1 Introduction

Texture is a vital ingredient to achieve the desired appearance of an object in computer graphics. Hence, a common task is to alter or replace textures – i.e., to *retexture* objects. In production, objects with multiple texture variants require a fast preview of each variant on rendered images with minimal noise while maintaining physically plausible highlights, reflections, and indirect illumination.

But doing so *efficiently* remains an open problem. A major challenge arises because the texture of an object not only alters that object’s direct visual appearance but also affects the indirect reflected colors throughout the scene. These effects are often complex and difficult to predict, but crucial to the overall visual quality.

A straightforward solution is to re-render the scene from scratch. However, rendering even a single image can take minutes to hours when difficult-to-sample light paths are involved. Prior work has explored re-rendering acceleration [Rousselle et al. 2016; Xu et al. 2024; Yan et al. 2024], but the computational cost is still limited by scene complexity. Alternatively, real-time global illumination methods (e.g., [Majercik et al. 2019; Müller et al. 2021]) or denoising can be used to curb the rendering time. These, however, struggle with complex light transport and can introduce objectionable artifacts.

As an alternative, image-based editing workflows allow users to manipulate textures in 2D image space using traditional editing tools. Recent works strive to make this process less tedious via image-based generative AI [Cheng et al. 2024; Garifullin et al. 2025; Cheng et al. 2025]. Those, however, lack fine-grained control. Zeng et al. [2024]; Lyu et al. [2025] improve controllability by decomposing images into “intrinsic channels” (such as albedo or roughness), editing these channels manually, and then re-composing the image. Asides from significant computational cost, a common limitation of all these methods is that their bias – the systematic deviation from ground-truth renderings – is impossible to predict and unbounded. In practice, that means such methods can sometimes work nicely while failing miserably on other cases.

Production renderers output a rich set of arbitrary output variables (AOVs) [Arnold 2025; Renderman 2025] that can be edited and composited in image space to achieve the desired appearance. However, typical AOVs are primarily designed for artistic control rather than physical correctness. To our knowledge, obtaining the correct AOVs to facilitate physically-based retexturing of full global illumination is an unexplored problem.

Taken together, existing re-rendering solutions produce high-quality and physically accurate results, but are expensive and require access to the full 3D scene (potentially an immense amount of data). Existing image-based alternatives are either a tedious manual process, or lack controllability and exhibit unpredictable bias. The most promising solution seems via AOV-based compositing workflows. However, identifying and computing the correct AOVs and recombining them in a physically-valid manner is an open problem – that we tackle.

We propose a two-stage retexturing method that simultaneously ensures artistic flexibility and physically-based accuracy. Our method supports retexturing of any multiplicative factors, such as the *base color* or *albedo* parameters in common BSDF models. The scene is

rendered with an initial path-tracing pass, during which we compute image-space radiances associated with retextured objects and statistics of UV coordinates for different light paths to account for spatially varying indirect changes caused by retexturing. Retexturing is then performed entirely as a lightweight post-processing step, without re-invoking the renderer, and without any involvement of the original 3D assets. Hence, final images with many texture variations can be generated efficiently from shared precomputation.

Our method has the following advantages:

- *Predictable error*: We approximate the path integral to separate texture from illumination, akin to texture filtering methods. The resulting bias is a low-pass filtering effect that can, in theory, be computed mathematically.
- *Generality*: Our approach accurately captures appearance changes from simple direct illumination to complex indirect illumination. While we only support color changes, the method is applicable to a range of commonly-used material models, like “principled” BSDF models [Burley 2012] or fabric materials [Sadeghi et al. 2013].
- *Performance*: The method is fast and scales to high-resolution images and even video. After a single initial rendering pass, it can generate results for arbitrary texture variations on the fly. Unlike generative AI approaches, it requires neither heavy computational resources nor large training datasets.
- *Compatibility*: Our post-processing does not require access to the original 3D assets and works even when re-rendering is impossible, e.g., in compositing workflows, while preserving artistic flexibility and physical accuracy.

Fig. 1 shows a sample of our results, where retexturing is performed on a sofa. Unlike alternative image-space solutions, our method captures the reflection in the mirror and the subtle glossy reflections on the floor correctly. While our method easily processes 4K resolution images, we could only work with 640×480 images for the generative AI solution, due to GPU memory constraints.

2 Related work

Re-rendering. There is a sizable body of works aiming efficiently re-render scenes after changes to lighting, geometries, or materials. Retexturing is a subproblem of this overarching goal. Much of this work has happened in the context of real-time rendering, where subsequent frames render variations of the scene in the previous frame. Solutions include radiance caching [Silvennoinen and Lehtinen 2017; Müller et al. 2021; Majercik et al. 2019] or resampling [Bitterli et al. 2020; Lin* et al. 2022]. In the context of offline, high-quality rendering, solutions based on control variates have been explored [Xu et al. 2024; Günther and Grosch 2015; Rousselle et al. 2016]. Another opportunity arises if path guiding methods [Vorba et al. 2019; Jensen 1995; Vorba et al. 2014; Ruppert et al. 2020; Müller et al. 2017] are used: Their learned distributions can be reused for (minor) variations of the same scene. In the context of volumetric rendering, previous work has suggested to precompute some quantities to accelerate albedo changes of volumes [Hašan and Ramamoorthi 2013]. For granular media, Zhang and Zhao [2020] adjust, at render time, the albedo of precomputed “grain scattering distributions”. However, all these re-rendering options share a

common limitation: They require access to the full 3D scene data during the retexturing operation. Our goal is to instead perform retexturing in a purely image-based post-process, limiting rendering to the initial AOV computation. Thus, our retexturing can run on low-power devices and only little data needs to be shared.

Compositing. AOVs are widely used in rendering pipelines to decompose images into intermediate shading components. Modern renderers output a range of AOVs from geometry buffers to shader components such as diffuse, specular, and coat [Arnold 2025; Renderman 2025]. Desired AOVs can be specified via Light Path Expressions (LPEs) [Heckbert 1990], which allow artists to define and extract contributions from specific classes of light transport paths, or combinations of scattering events. Retexturing directly visible surfaces can be achieved by outputting AOVs with primary-hit UV coordinates, incident radiances, and individual BSDF components. However, generalizing to indirectly hit surfaces has remained challenging: changing the color of the surface can lead to noticeable changes in reflections at different positions in the scene. To account for such indirect changes, users still need to manipulate on a per-pixel basis on each AOV layer. Our method can be seamlessly integrated into the AOV-based composition workflow while supporting retexturing in full global illumination, including reflections, color bleeding, and complex effects like caustics.

Image-Based Editing and Generative Methods. Recently, generative models have become popular for image editing using text prompts [Labs et al. 2025; Rombach et al. 2021] or example-based guidance [Cheng et al. 2024; Garifullin et al. 2025]. These methods reduce user effort, but often lack precise controllability and do not guarantee physical or geometric consistency. Moreover, these approaches struggle to handle global illumination effects, such as complex reflections or color bleeding introduced by retexturing.

The works closest to ours, RGBX [Zeng et al. 2024] and Intrinsic Edit [Lyu et al. 2025], offer improved controllability by decomposing a synthetic image or photograph into several intrinsic channels. Users can manually edit these channels to control material properties or lighting conditions, and then combine them to generate the final image. However, errors in the estimated channels and out-of-distribution data can easily impact the final results.

3 Background

Materials. Surface material properties are described by a Bidirectional Scattering Distribution Function (BSDF) [Pharr et al. 2023]. The BSDF is a function $f(x, \omega_i, \omega_o)$ that describes the fraction of light scattered at the surface point x from direction ω_o to ω_i . In the following, we will use the notation $f(y \rightarrow x \rightarrow z)$ for scattering along a path of light, where the directions ω_o and ω_i are given by the surface points y and z . Renderers typically use analytical BSDF models that are mixtures of many components, carefully designed to achieve the desired appearance. Most BSDFs are effectively a sum

$$f(y \rightarrow x \rightarrow z) = \sum_k f_k(y \rightarrow x \rightarrow z) \quad (1)$$

of multiple components. For example, the Disney BSDF [Burley 2012] linearly blends diffuse, specular, metallic, sheen, clearcoat and

retroreflective components. The parameters (like the color) of such BSDFs are typically controlled by a texture.

Textures like the Disney BSDF’s “base color” impact the various components to different degrees. This makes retexturing a challenging operation: Naively multiplying a new color will generally not achieve consistent appearance.

Light transport equation. The value of each pixel in a rendered image is given by an integral over all possible paths of light that reach this pixel. This integral is known as the rendering equation or light transport equation (LTE) and can be formulated in various ways [Kajiya 1986; Veach 1997]. We utilize the path integral formulation [Pharr et al. 2023, Section 13.1].

In our notation a path $\bar{x}_n$ comprises n vertices x_i where x_0 is the camera. The amount of light, in terms of radiance, reaching the camera is given by the outgoing radiance at the directly visible point x_1 . This, in turn, is a sum over the contributions by all path lengths,

$$L(x_1 \rightarrow x_0) = \sum_{n=1}^{\infty} P(\bar{x}_n), \quad (2)$$

where $P(\bar{x}_n)$ is the integral over all paths of length n ,

$$P(\bar{x}_n) = \int_{A^{n-1}} L_e(x_n \rightarrow x_{n-1}) T(\bar{x}_n) dA(x_2 \dots x_n), \quad (3)$$

and the product of BSDFs and geometry terms,

$$T(\bar{x}_n) = \prod_{i=1}^{n-1} f(x_{i+1} \rightarrow x_i \rightarrow x_{i-1}) G(x_{i+1} \leftrightarrow x_i), \quad (4)$$

is called the throughput of a path.

Eq. (2) is usually computed by Monte Carlo integration – i.e., by sampling random paths – which is time-consuming. We show that retexturing can be formulated as a post-processing step applied to the light transport equation. This formulation allows efficient retexturing without the need for re-rendering.

4 Our approach

Our method consists of two stages:

- (1) an initial rendering pass that generates a set of AOVs, and
- (2) a post-processing stage to retexture in image-space.

We assume that the object(s) to be retextured are known in advance. We support retexturing multiplicative color parameters that modulate the BSDF (e.g., base color in the Disney principled BSDF or per-thread color albedo in [Sadeghi et al. 2013]). For conciseness, we refer to those parameters collectively as the *albedo* in the following.

In the initial rendering pass, we accumulate radiance and UV-coordinate statistics in AOVs. Essentially, we segment the path space based on the *number of times a path scattered from the retextured object*. For each scatter count – up to a maximum – we render a separate image. To replace the texture, we additionally track the mean and standard deviation of the UV coordinates – again for each scatter count – to later perform a filtered texture lookup.

In Section 4.1 we describe how we decompose BSDFs into “dependent” and “independent” terms rather than working on the BSDF components individually. In Section 4.2 we reformulate the path integral to separate texture from light transport. Finally, in Section 4.3

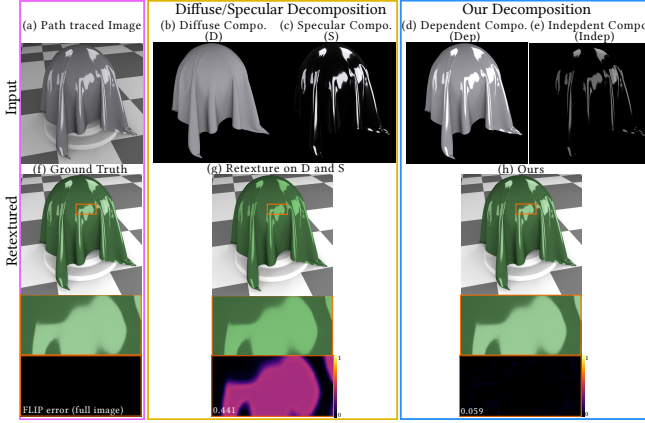


Fig. 2. Retexture a glossy fabric (a) with green color. BSDF is separated into different lobes, followed by retexturing on the precomputed contribution of each lobe. Retexturing on diffuse and specular components (b and c) results in overly green highlights (g). We decompose the BSDF into components that are dependent and independent of albedo (d and e), ours (h) offers the closest appearance to the ground-truth (f).

we show how we use statistics of UV coordinates to perform filtered texture lookup and thus avoid storing individual samples.

4.1 BSDF separation

The visual outcome of retexturing is fundamentally determined by the material’s BSDF behavior. For example, Fig. 2 illustrates a simple retexturing scenario under direct illumination: a glossy fabric with its original material (Fig. 2a) is re-rendered with a new texture, producing the target appearance in Fig. 2f (bottom left).

The BSDF can be divided into distinct lobes, such as a diffuse lobe (Fig. 2b) and a specular lobe (Fig. 2c). A baseline retexturing approach would be to precompute each lobe’s contribution and then scale it by the new texture, as shown in the center column of Fig. 2.

However, compared to a path-traced ground truth (Fig. 2f), such per-lobe retexturing yields a noticeable color shift (error image in 2g), because the specular component in this BSDF model (Disney BSDF) is only partially modulated by the albedo.

To address this issue, we propose a BSDF decomposition that explicitly separates texture-dependent and texture-independent components of the material’s appearance (illustrated in Fig. 2d and Fig. 2e). Given a texture $\rho(x)$ we group the BSDF components into “dependent” and “independent” ones to achieve a linear expression:

$$f(x, \omega_i, \omega_o) \approx \rho(x)f_{\text{dep}}(x, \omega_i, \omega_o) + f_{\text{indep}}(x, \omega_i, \omega_o). \quad (5)$$

We define f_{dep} as the component of the BSDF’s reflectance that depends on the input texture, i.e. the portion of the shading contribution directly scaled by the albedo. Conversely, f_{indep} denotes the component that is independent of the texture, representing the remainder of the BSDF not modulated by the albedo. f_{dep} includes the diffuse reflection and any specular term tinted by the albedo, while f_{indep} captures the untinted specular reflections and other effects unaffected by albedo. It is important to note that this decomposition is applied to the BSDF’s computed contribution after sampling; the

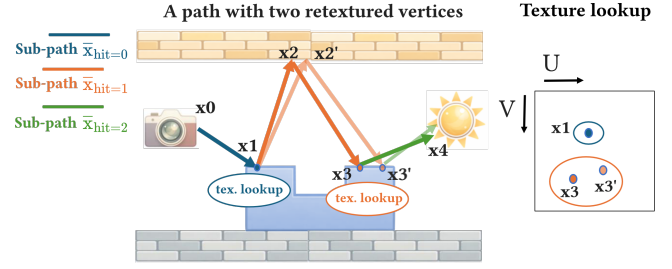


Fig. 3. A path of four edges interacts with an object to be retextured (marked in blue). The path is partitioned according to the number of times each sub-path intersects the retextured vertices. For each retextured vertex, we perform a filtered texture lookup using statistics of the UV samples.

underlying BSDF model and its importance sampling scheme remain unchanged, preserving the physical fidelity of the shading model. During BSDF evaluation, we compute the full BSDF value as well as the separated contribution according to the material parameters. Please refer to the supplemental document for an example of how a simplified Disney BSDF model can be separated.

When retexturing the material, we apply the new albedo only to the texture-dependent term f_{dep} , and leave f_{indep} unmodified (as shown in Fig. 2h). This simple yet effective strategy yields retextured results that closely match a path-traced rendering (Fig. 2f), significantly reducing color artifacts compared to naïve lobe-based scaling (Fig. 2g). Our method is general and extensible: for example, in a microcylinder-based cloth reflectance model [Sadeghi et al. 2013], we can extend our decomposition to support multiple f_{dep} where each represents the contributions along one fiber direction.

4.2 Path segmentation

A naïve approach to retexture could be to record the UV coordinate at the primary hit and scale the albedo accordingly. As illustrated in Fig. 4, simply scaling the albedo either with full global illumination (Fig. 4d) or with only direct illumination while leaving indirect contributions unmodified (Fig. 4e) leads to noticeable color shifts, particularly in regions where light bounces multiple times within the fabric weave. These failures arise because each path may encounter the retextured surface multiple times, with different UV coordinates at each hit. Our method (Fig. 4c) instead records the scattered radiance, segmented according to the number of times each path interacts with the retextured vertices:

$$L(x_1 \rightarrow x_0) = P(\bar{x}_1) + \sum_{h=0}^{\infty} P(\bar{x}_{\text{hit}=h}), \quad (6)$$

where $P(\bar{x}_1)$ is the emitted radiance, and $\bar{x}_{\text{hit}=h}$ denotes the set of paths (of any length) that intersect the retextured object exactly h times. We precompute each $P(\bar{x}_{\text{hit}=h})$ as a separate AOV image. Fig. 3 shows an illustration of our segmentation. This segmentation provides better scalability than considering paths up to a maximum total path length: We support arbitrarily many bounces of global illumination, but only a maximum number of bounces involving a retextured object.

The path throughput $\tilde{T}(\bar{x}_{\text{hit}=h})$ for each path set in Eq. (6), can be separated into path vertices that lie on retextured objects, denoted $i \in \mathcal{R}$, from those that do not:

$$\tilde{T}(\bar{x}_{\text{hit}=h}) = \underbrace{\prod_{i \in \mathcal{R}} (\rho(x_i) f_{\text{dep},i} + f_{\text{indep},i})}_{\text{retextured vertices}} G_i \times \underbrace{\prod_{i \notin \mathcal{R}} f_i G_i}_{\text{other vertices}} \quad (7)$$

At vertices in $\mathcal{R}$, we apply our BSDF decomposition Eq. (5). At all other vertices, we fall back to the original BSDF. We expand Eq. (7) to 3 types of throughput terms,

$$\tilde{T}_{\text{dep}}(\bar{x}_{\text{hit}=h}) = \prod_{i \in \mathcal{R}} \rho(x_i) f_{\text{dep},i} \prod_{i \notin \mathcal{R}} f_i G_i, \quad (8a)$$

$$\tilde{T}_{\text{indep}}(\bar{x}_{\text{hit}=h}) = \prod_{i \in \mathcal{R}} f_{\text{indep},i} \prod_{i \notin \mathcal{R}} f_i G_i, \quad (8b)$$

$$\tilde{T}_{\text{cross}}(\bar{x}_{\text{hit}=h}) = \sum_{\substack{S \subset \mathcal{R} \\ S \neq \emptyset, S \neq \mathcal{R}}} \left(\prod_{i \in S} \rho(x_i) f_{\text{dep},i} \prod_{j \in \mathcal{R} \setminus S} f_{\text{indep},j} \prod_{k \notin \mathcal{R}} f_k G_k \right). \quad (8c)$$

Intuitively,

- $\tilde{T}_{\text{dep}}(\bar{x}_{\text{hit}=h})$ denotes the portion of throughput where all retextured vertices contribute solely via their albedo-dependent term.
- $\tilde{T}_{\text{indep}}(\bar{x}_{\text{hit}=h})$ denotes the portion of throughput where all retextured vertices contribute solely via their albedo-independent term.
- $\tilde{T}_{\text{cross}}(\bar{x}_{\text{hit}=h})$ accounts for the remaining throughput component with mixed contributions, where some retextured vertices contribute via f_{dep} and others via f_{indep} .

Taken together, our generalized formulation is:

$$\tilde{P}(\bar{x}_{\text{hit}=h}) \approx \int_{A^{n-1}} \chi_{|\mathcal{R}|,h} \prod_{i \in \mathcal{R}} \rho(x_i) dA(x_2 \dots x_n) \quad (9a)$$

$$\cdot \int_{A^{n-1}} \chi_{|\mathcal{R}|,h} L_e \underbrace{\tilde{T}_{\text{dep}}(\bar{x}_{\text{hit}=h})}_{\text{albedo-dep. throughput}} dA(x_2 \dots x_n) \quad (9b)$$

$$+ \int_{A^{n-1}} \chi_{|\mathcal{R}|,h} L_e \underbrace{\tilde{T}_{\text{indep}}(\bar{x}_{\text{hit}=h})}_{\text{albedo-indep. throughput}} dA(x_2 \dots x_n) \quad (9c)$$

$$+ \int_{A^{n-1}} \chi_{|\mathcal{R}|,h} L_e \underbrace{\tilde{T}_{\text{cross}}(\bar{x}_{\text{hit}=h})}_{\text{cross-term throughput}} dA(x_2 \dots x_n), \quad (9d)$$

where $\chi_{|\mathcal{R}|,h}$ is defined as:

$$\chi_{|\mathcal{R}|,h} = \begin{cases} 1, & \text{if the number of retextured vertices } |\mathcal{R}| \text{ equals } h, \\ 0, & \text{otherwise.} \end{cases}$$

Note that this equation is only *approximately* equal to the correct light transport: Put in simplified notation, Eqs. (9a) and (9b) turn an integral of the form $\int \rho(x)g(x)dx$ into a product of integrals $\int \rho(x)dx \int g(x)dx$. Where $\rho(x)$ is the albedo and $g(x)$ is the remaining integrand. This is conceptually the same approximation applied by texture filtering methods. Effectively, the albedo will be low-pass filtered, which has little effect on locally smooth textures but can result in a loss of detail for high-frequency content. Note that we only integrate over the *visible part* of the texture at this

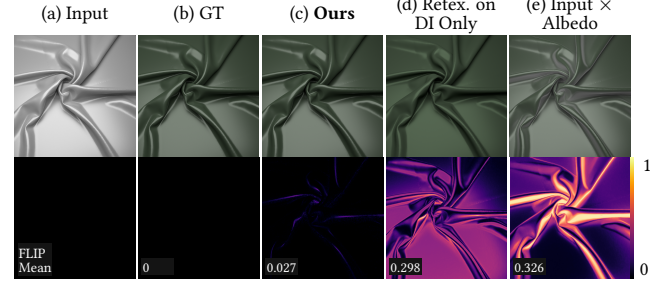


Fig. 4. Comparison of image-based retexturing results between our method (c) and two naïve approaches: retexturing of the direct illumination component (d) and retexturing on full GI scaled by albedo (e). Our path segmentation yields the lowest FLIP error. The highest error is caused by the approximated path throughput (approximated “cross” term) for paths with more than two retextured vertices.

depth, so significant smoothing occurs only for longer paths where it will usually be less visible.

Specifically, during post-processing, the albedo-dependent integral Eq. (9b) is scaled by the sampled texture along with its UV statistics (Section 4.3), and the albedo-independent integral Eq. (9c) is included as an additive term. The scattered radiance of path set $P(\bar{x}_{\text{hit}=0})$ remains unchanged and is simply added to the final image. The scattered radiance of cross terms Eq. (9d) is computed exactly for up to three retextured vertices and scaled accordingly. For paths with more retextured vertices, the cross terms are approximated by subtracting Eqs. (9b) and (9c) from the full radiance and multiplying the residual by the sampled texture. As illustrated in Fig. 4, the cross-term approximation introduces only a negligible bias. We include the derivation of how cross-term throughputs are computed in the supplemental.

4.3 Statistics of UV-coordinates

Retexturing requires us to know the texture coordinates (UV coordinates) at all retextured vertices along each path. Instead of storing an expensive point cloud of such coordinates, we assume that UV coordinates follow a normal distribution. During the initial rendering phase, we compute the mean and variance of that distribution and store it in an AOV.

For each retextured vertex, we consider N UV samples (see Fig. 3) $t_1, t_2, \dots, t_N$, where $t_i = (u_i, v_i)$ denotes the 2D UV coordinates, we estimate their mean $\bar{t} = \frac{1}{N} \sum_{i=1}^N t_i$, and standard deviation $\hat{\sigma} = \sqrt{\frac{1}{N-1} \sum_{i=1}^N (t_i - \bar{t})^2}$. We then use the per-pixel mean and standard deviation to construct two 1D Gaussian distributions over U and V .

Fig. 5 shows the effect of that on a simple Cornell Box, where a new texture is applied to the right wall. The texture is a step function, half red and half yellow. We visualize the ground-truth distribution for the pixel highlighted in green in the bottom row of Fig. 5. If only the mean UV coordinate is used, for each retextured vertex x_i in Eq. (9a), the texture lookup is done at $\bar{t}$, which essentially approximates Eq. (9a) using a single representative texel, leading to overly sharp transitions, most notably along the red-yellow boundary of the texture. Instead, we assume each UV sample follows a

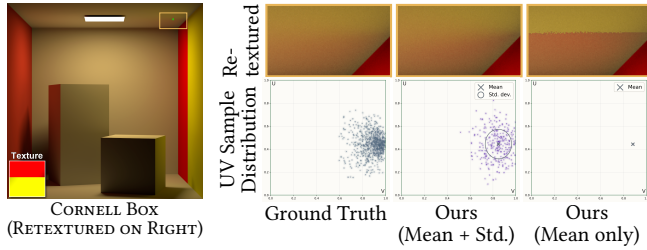


Fig. 5. The green right wall of a Cornell Box is retextured with a red-yellow step function texture. This introduces spatially-varying indirect reflections, caused by the different UV coordinates of each Monte Carlo sample. In the bottom row, we show the distribution of UV coordinates for a representative pixel (highlighted with a green dot). We accumulate statistics over these UV coordinates to reconstruct the distribution.

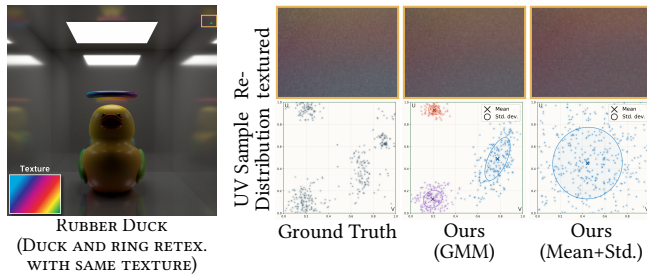


Fig. 6. We retexture the duck and ring, two disjoint meshes sharing the same texture (shown in the bottom left) but with separate UV parameterizations. A mixture of their colors is reflected on the glossy walls via indirect specular paths (top row). The bottom row shows UV coordinate distributions for a representative pixel (highlighted with a green dot), reconstructed using a Gaussian mixture model (GMM) and a single Gaussian.

Gaussian distribution and perform Monte Carlo integration over this distribution, which produces results that more closely match the ground truth.

Fig. 6 shows a challenging case where two objects (duck and ring) share the same rainbow color texture, but each object has a different UV parameterization. The surrounding glossy walls show reflected colors from both objects. On those walls, light paths contributing to a single pixel map to widely separated UV regions, resulting in a multimodal distribution of UV coordinates. In the bottom row of Fig. 6, we again visualize the distribution of UV coordinates for a representative pixel highlighted in green. Fitting a single Gaussian from the sample mean and standard deviation is computationally cheap but fails to capture the multimodal distribution, leading to a less accurate result. The center column shows that a Gaussian mixture model (GMM) could be a well-suited alternative for such cases. However, for this experiment, we computed the mixture by storing individual UV samples, which is too expensive for practical use. Contriving an efficient online fitting scheme could be an interesting avenue for future work.

Table 1. Statistics of the speed-up (higher is better) in terms of equal-spp (256 spp) run-time cost of baseline and our method across 9 test scenes. Averaged across 10 runs at 1080 x 1080 image resolution. Baseline integrator includes path tracer and guided path tracer.

Speed-up vs. baseline	Our initial-pass	Our retexturing
AVERAGED	0.85×	179.76×
WORST	0.79×	106.78×
BEST	0.94×	249.45×

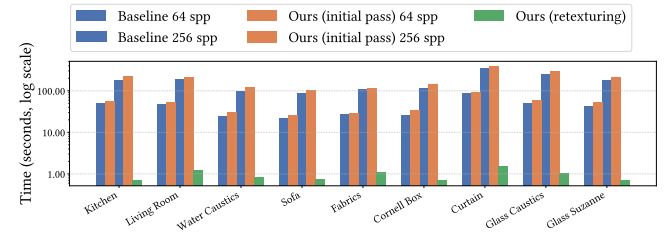


Fig. 7. We compare equal-spp (64 and 256 spp) rendering time across 9 test scenes rendered at 1080 x 1080. Baseline integrator includes path tracer and guided path tracer. Time is shown in logarithmic scale.

5 Results

5.1 Comparison with ground-truth path tracer

We compare our results with converged ground-truth path-traced images in Fig. 8. All experiments were run on an Intel Core i5-14600KF. Images are rendered at a resolution of 1080 x 1080 with a maximum path length of 10. Fig. 1 is with path length 20. The maximum retextured depth (maximum number of path segments that interact with texture) is set to 5. For each scene, we perform retexturing on the input image Fig. 8(a) and present two sets of retextured results. The retextured images are then denoised using [Áfra 2019]. We report errors using relative mean squared error (relMSE) and FLIP [Andersson et al. 2021] with respect to the converged ground truth. To reduce the impact of Monte Carlo variance on the error metrics, we evaluate the errors on denoised images for our results. Results including videos and non-denoised images can be found in the supplemental material.

The WATER CAUSTICS scene features expensive-to-render caustic patterns; integrating our method with path guiding [Herholz and Dittebrandt 2022] accelerates rendering and preserves fine caustics while maintaining the metallic appearance of objects under water.

The CURTAIN scene evaluates our method in an interior environment with richly textured curtains. Our results faithfully reproduce fine-grained curtain details, complex occlusions, and the corresponding changes in floor reflections. Similarly, the FABRICS scene demonstrates our method on advanced fiber-based materials [Sadeghi et al. 2013]; the cloth’s appearance is explained by separate lobes for specular reflection from individual fibers and diffuse reflection from the colored threads, among others.

The GLASS SUZANNE scene features a challenging case: a colored object inside a glass sphere is to be retextured. Light entering the glass sphere is refracted and hits the inner object. Some of the light

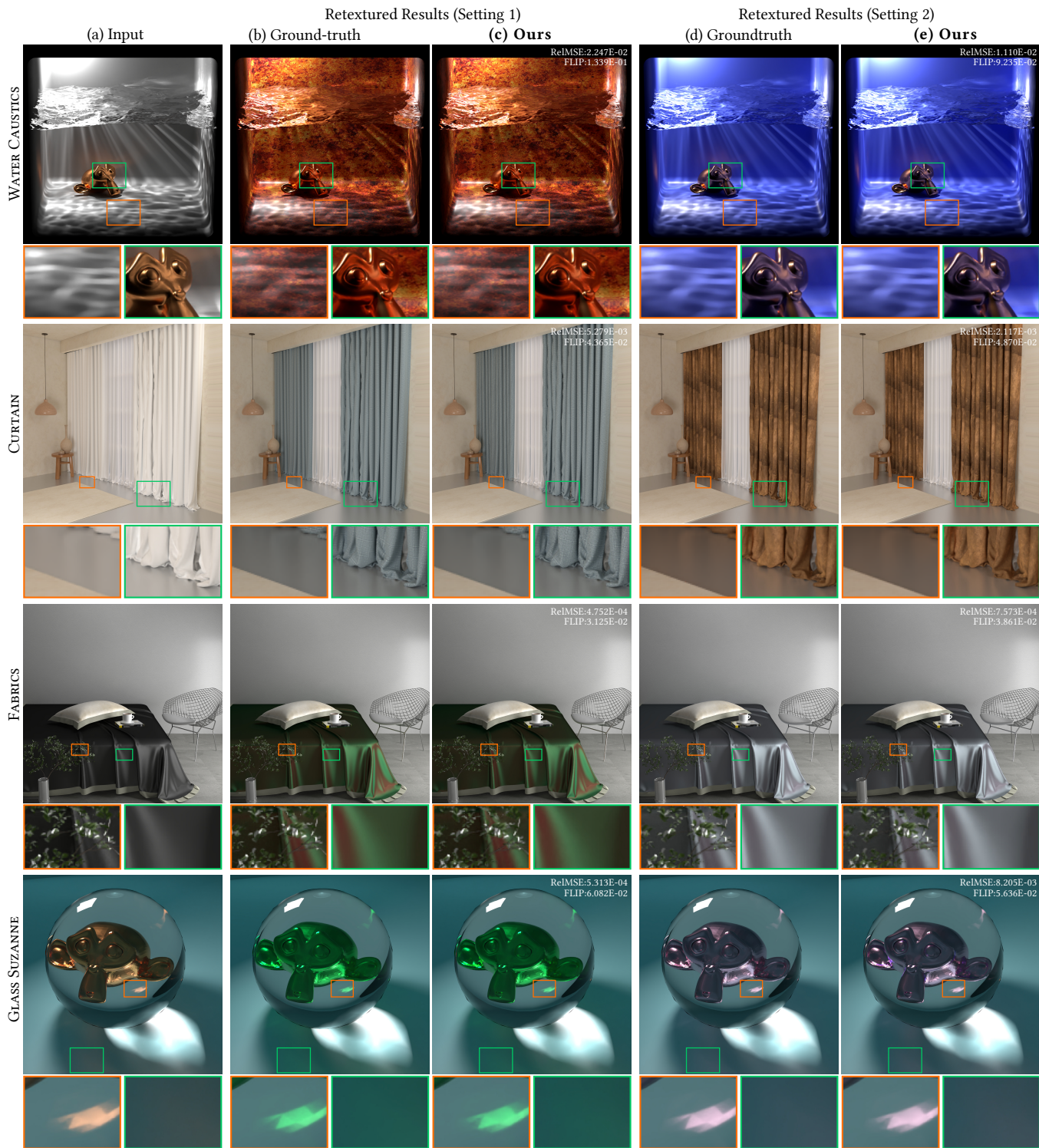


Fig. 8. Comparison of the converged ground-truth path tracer (b, d) and ours (c, e). Our physically-based retexturing successfully captures complex global illumination effects across a wide range of scenes and materials. The input image (a) is rendered with 2048 spp. To reduce the impact of Monte Carlo variance inherited from the input image, we evaluate the errors on denoised images for our results.

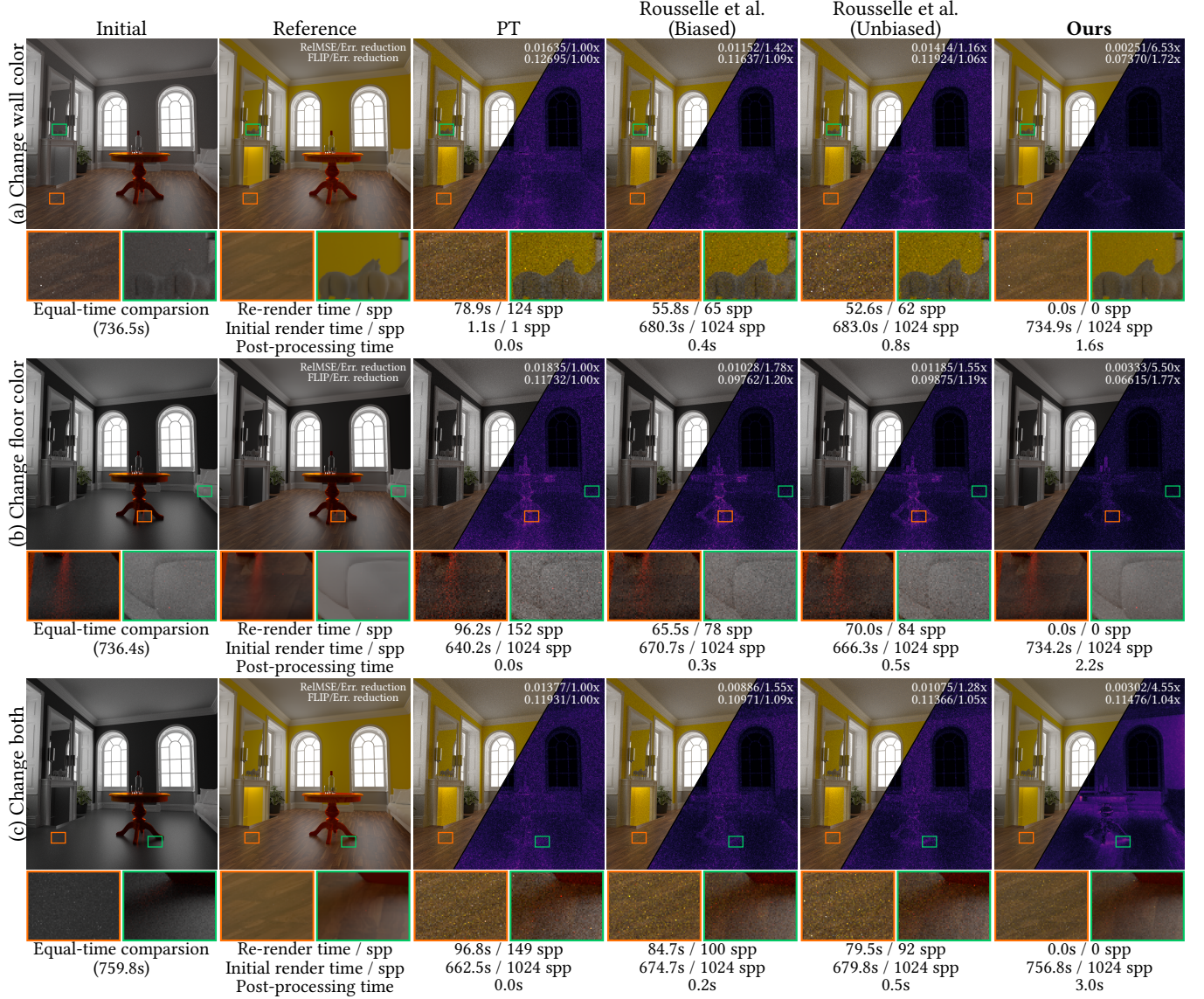


Fig. 9. We compare our method against [Rousselle et al. 2016] under an equal-time setting. All methods use 1024 samples for the initial rendering. The total time budget includes the time spent on initial rendering, re-rendering, and post-processing. The time for re-rendering is the remainder of the total equal-time budget after the initial rendering. Our method does not need to perform re-rendering for such tasks, but only a cheap post-processing.

then reflects off the inner surface and exits the sphere, causing color changes on the glass surface and the ground floor. Our method accurately reproduces these effects while retexturing the inner object.

Performance. We compare the runtime cost of our initial rendering pass and retexturing to a baseline path tracer in Fig. 7 and Table 1. Our additional AOVs introduce an average runtime overhead of about 15%, this cost is amortized by the much lower cost of post-processing retexturing, even at a low sample count of 64 spp. The runtime of our post-processing step at a resolution of 1080×1080 is approximately 1.373s, of which 0.301s is spent on reading the AOVs. The average AOV storage cost in EXR format

across the test scenes in Fig. 8 and Fig. 9, all rendered at 1080p resolution, is approximately 87.3 MB.

5.2 Comparison with re-rendering methods

We compare our method to the control-variate based re-rendering approach of Rousselle et al. [2016]. We have reimplemented their approach in our renderer and validated results against images from their paper. As far as we can tell, our implementation only deviates from the original method in that we use a simpler (and faster) Gaussian blur to filter variances rather than the NL-means filter [Rousselle et al. 2012] used in the original implementation. The

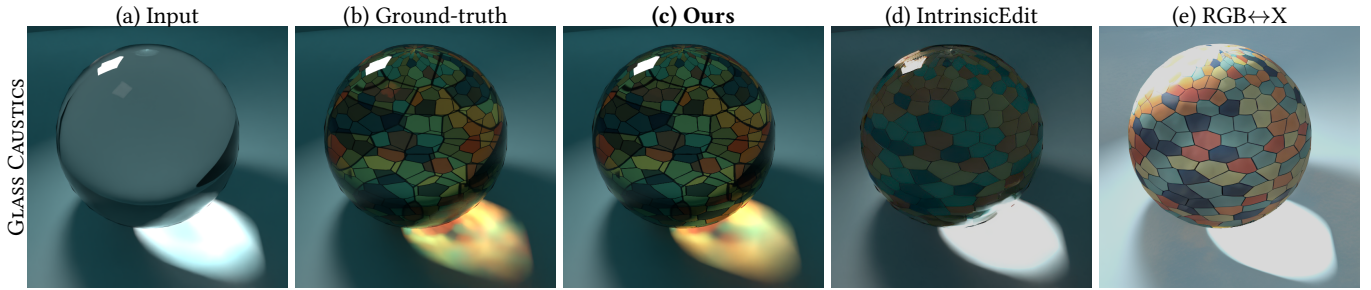


Fig. 10. We compare our results with RGB $\leftrightarrow$ X [Zeng et al. 2024] and Intrinsic Edit [Lyu et al. 2025]. We evaluate RGB $\leftrightarrow$ X [Zeng et al. 2024] and Intrinsic Edit [Lyu et al. 2025] using their official open-source implementations with default settings and ground-truth albedo channel. Our approach shows improved robustness across all scenes, more results are provided in the supplemental material.

NL-means filter from the original paper is more costly, but might yield better quality. In our test cases, we did not observe artifacts from the simpler Gaussian blur.

Similar to our method, Rousselle et al. [2016] start with an initial, high sample-count rendering. To achieve retexturing, they then essentially estimate the difference integral between this initial render and the desired target. This control variate approach introduces multiple sources of overhead: (1) materials need to be evaluated twice during re-rendering, (2) variances need to be estimated and filtered to weight the control variates, and (3) the retexturing pass needs to perform Monte Carlo path tracing. The supplemental document provides a comparison of the overhead between this control variate approach and our method. The method comes in two flavors: a cheaper biased solution (weights correlated with samples) and an unbiased one. We compare to both.

Fig. 9 shows an equal-time comparison for a *single* retexturing task. All methods use a 1024 spp initial render. For path tracing and Rousselle et al. [2016], the remaining time is invested into the (re-)rendering step. No denoising is performed.

We replicate edit tasks show-cased by Rousselle et al. [2016]: In the LIVING ROOM scene, we retexture the wall (Fig. 9a) floor (Fig. 9b) or both (Fig. 9c). Across all variants, our method achieves the lowest error retexturing in equal-time: The re-rendering step in Rousselle et al. [2016] introduces noise, while our method retains the low noise level of the initial render.

When changing both walls and floor using a different texture for each, the texture change alters the entire global illumination of the scene across multiple bounces within and between retextured objects. Then, the cross-terms in our segmentation (8) become more important – of which we only compute the first three. Thus, in Fig. 9 (c), the slight bias of our method becomes measurable, but the result is still of better visual quality than the re-rendering solution.

Note that re-rendering and our method are not direct alternatives but rather complementary: Our method can more efficiently alter color textures, especially if *multiple* variants are desired, and we do not require the 3D scene data to do that. One use-case is web/mobile applications with limited ray-tracing capability. By precomputing AOVs on high-end machines, our method enables interactive texture design and preview on low-end devices without re-invoking a 3D renderer. This reduces repeated transfer of large 3D assets, hardware

requirements, and the need for costly cloud rendering. Re-rendering is more costly but capable to also alter non-linear properties like surface roughness. Therefore, future work could investigate a potential combination, using our method for cheap color changes and the control variate approach to alter roughness or remove bias.

5.3 Comparison with diffusion-based image editing

We compare with the state-of-the-art diffusion-based method RGB $\leftrightarrow$ X [Zeng et al. 2024] and Intrinsic Edit [Lyu et al. 2025] for retexturing in Fig. 10. These methods allow fine-grained image editing via the albedo (and other intrinsic) channels.

We use the official implementations and pre-trained models; ground-truth albedo is used as input for fair comparison. Results from RGB $\leftrightarrow$ X and Intrinsic Edit are generated on an A100 GPU with 40GB memory. Due to the high computational cost of Intrinsic Edit, we use a rather small resolution: 640 x 480 for all methods.

The results in Fig. 10 show that while our method produces predictable retexturing results with minimal and controlled bias, this is not the case for diffusion-based methods. These methods introduce unpredictable changes to geometry, color, and luminance, and material appearance (the glass in GLASS CAUSTICS no longer resembles glass). Furthermore, they fail to capture complex indirect illumination, such as the textured caustics in GLASS CAUSTICS. More results are included in the supplemental material.

Overall, our method robustly offers higher image quality, requires fewer computational resources, and enables faster editing compared to 2D diffusion-based approaches. In contrast, Intrinsic Edit and RGB $\leftrightarrow$ X are better suited to general-purpose editing when 3D assets are unavailable.

6 Limitations and future work

While our method produces high-quality retexturing results, it is not fully unbiased: The UV filtering and the approximation of “cross” terms for paths with higher retexturing hit counts introduces errors that manifest as (barely visible) color shifts and blur. In the following, we analyze two representative cases that amplify the bias and discuss their visual impact. Then, we outline two promising directions for future work to broaden the scope of editable appearances.

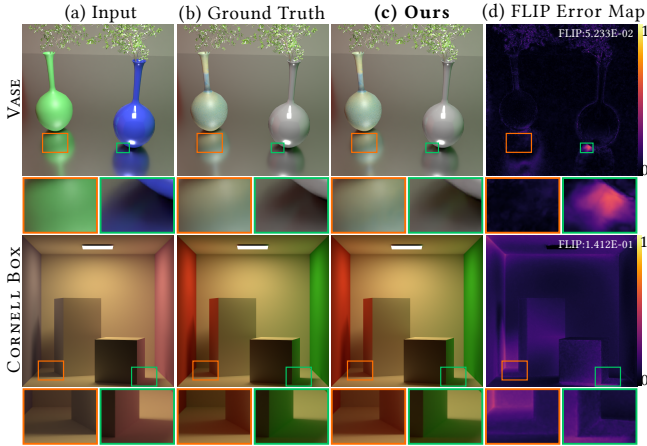


Fig. 11. Our method can handle indirectly hit surfaces even when retexturing with multiple textures. In CORNELL BOX the strong indirect illumination is dominated by saturated red and green colors, which leads to an increased color difference compared to the ground-truth. In contrast, our method performs robustly on the VASE scene, where reflections from indirect bounces mostly do not overlap within a pixel.

Multiple textures. Fig. 11 shows a limitation in retexturing with multiple textures. In CORNELL BOX, retexturing both the left and right wall causes strong and saturated colored reflections from both sides on each bounce. Accurately accounting for this would require tracking and weighting UVs separately by aggregating the statistics of sample counts. In the VASE scene, there is a small spot (green crop) where indirect reflections caused by two disconnected retextured regions overlap. There, our cross-term approximation is inaccurate and bias becomes visible.

UV distribution. Currently, we approximate the distribution of UV coordinates using two independent 1D Gaussian distributions. A potentially more accurate approach would be to model the UV distribution with a 2D Gaussian mixture model (GMM). Fitting a GMM during the initial rendering pass requires an adaptive fitting algorithm [Vorba et al. 2014; Herholz et al. 2018].

Complex appearance. Expensive material models like hair or fur shaders are likely to profit greatly from the cheap retexturing offered by our method. On the implementation side, however, a disadvantage of our method is that a manual segmentation of the material is required. This will be more challenging for complex materials.

Non-linear edits. A key benefit of our method is that no 3D scene data is required for retexturing. That, however, comes at a price: It is fundamentally impossible to retexture properties like roughness that alter path space distributions. The aforementioned combination with control variate solutions could be a possibility for future work to support such edits while retaining at least some of our benefits.

7 Conclusion

We present a physically-based retexturing method for fast and accurate retexturing, incorporating color bleeding and complex rendering effects. It runs efficiently even on 4K resolution images in just a few seconds.

Our approach employs a hybrid pipeline combining 3D rendering with image-based post-processing. After the initial rendering pass, no further re-rendering is required; instead, all physically-based editing is performed directly in image space. This allows our method to operate independently of expensive re-rendering and makes it compatible with traditional compositing workflows where re-rendering is no longer available in the composition stage. Moreover, our method can robustly handle commonly used multi-lobe material models, such as principled BSDF and specific fabric models.

Compared to traditional image-based methods, ours employs a physically-based approach, so artists don’t need to manually adjust colors on a per-pixel basis. Unlike other image-based generative AI methods, our approach can accurately handle both direct and indirect reflections across different scenes robustly, without requiring high-end hardware or large amounts of high-quality training data, even for high-resolution images. Furthermore, our method naturally generalizes to video sequences.

Acknowledgments

Junqiu Zhu was supported by the Overseas Excellent Young Scientists Fund of Shandong Province (Grant No. 2026HWYQ-011); Jing-Jin-Ji Regional Integrated Environmental Improvement-National Science and Technology Major Project (No. 2025ZD1202205); the National Natural Science Foundation of China (No.62502285).

References

- Attila T. Áfra. 2019. Intel® Open Image Denoise. <https://www.openimagedenoise.org/>.
- Pontus Andersson, Jim Nilsson, Peter Shirley, and Tomas Akenine-Möller. 2021. Visualizing Errors in Rendered High Dynamic Range Images. In *Eurographics Short Papers*. doi:10.2312/egs.20211015
- Arnold. 2025. AOVs for Image Compositing - Arnold for Maya. https://help.autodesk.com/view/ARNOL/ENU/?guid=arnold_for_maya_aovs_am_AOVs_for_Image_Compositing_html. Accessed: 2026-01-09.
- Benedikt Bitterli, Chris Wyman, Matt Pharr, Peter Shirley, Aaron Lefohn, and Wojciech Jarosz. 2020. Spatiotemporal reservoir resampling for real-time ray tracing with dynamic direct lighting. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 39, 4 (July 2020). doi:10/gg8xc7
- Brent Burley. 2012. Physically-Based Shading at Disney.
- Ta-Ying Cheng, Prafull Sharma, Mark Boss, and Varun Jampani. 2025. MARBLE: Material Recomposition and Blending in CLIP-Space. *CVPR* (2025).
- Ta-Ying Cheng, Prafull Sharma, Andrew Markham, Niki Trigoni, and Varun Jampani. 2024. ZeST: Zero-Shot Material Transfer from a Single Image. *arXiv preprint arXiv:2404.06425* (2024).
- Kamil Garifullin, Maxim Nikolaev, Andrey Kuznetsov, and Aibek Alanov. 2025. Material-Fusion: High-Quality, Zero-Shot, and Controllable Material Transfer with Diffusion Models. *arXiv preprint arXiv:2502.06606* (2025).
- Tobias Günther and Thorsten Grosch. 2015. Consistent Scene Editing by Progressive Difference Images. *Comput. Graph. Forum* 34, 4 (July 2015), 41–51.
- Milovs Hasan and Ravi Ramamoorthi. 2013. Interactive albedo editing in path-traced volumetric materials. *ACM Trans. Graph.* 32, 2, Article 11 (April 2013), 11 pages. doi:10.1145/2451236.2451237
- Paul S. Heckbert. 1990. Adaptive radiosity textures for bidirectional ray tracing. In *Proceedings of the 17th Annual Conference on Computer Graphics and Interactive Techniques* (Dallas, TX, USA) (SIGGRAPH ’90). Association for Computing Machinery, New York, NY, USA, 145–154. doi:10.1145/97879.97895
- Sebastian Herholz and Addis Dittbrandt. 2022. Intel® Open Path Guiding Library. <https://www.openpplg.org>.

- Sebastian Herholz, Oskar Elek, Jens Schindel, Jaroslav Krivánek, and Hendrik Lensch. 2018. A Unified Manifold Framework for Efficient BRDF Sampling based on Parametric Mixture Model. In *EGSR '18 EI&I (EGSR '18)*. Eurographics Association.
- Henrik Wann Jensen. 1995. Importance Driven Path Tracing using the Photon Map. In *Rendering Techniques*.
- James T. Kajiya. 1986. The Rendering Equation. *SIGGRAPH Comput. Graph.* 20, 4 (Aug. 1986), 143–150.
- Black Forest Labs, Stephen Batifol, Andreas Blattmann, Frederic Boesel, Saksham Consul, Cyril Diagne, Tim Dockhorn, Jack English, Zion English, Patrick Esser, Sumith Kulal, Kyle Lacey, Yam Levi, Cheng Li, Dominik Lorenz, Jonas Müller, Dustin Podell, Robin Rombach, Harry Saini, Axel Sauer, and Luke Smith. 2025. FLUX.1 Kontext: Flow Matching for In-Context Image Generation and Editing in Latent Space. arXiv:2506.15742 [cs.GR] <https://arxiv.org/abs/2506.15742>
- Daqi Lin*, Markus Kettunen*, Benedikt Bitterli, Jacopo Pantaleoni, Cem Yuksel, and Chris Wyman. 2022. Generalized Resampled Importance Sampling: Foundations of ReSTIR. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2022)* 41, 4, Article 75 (07 2022), 23 pages. doi:10.1145/3528223.3530158 (*Joint First Authors).
- Linjie Lyu, Valentin Deschaintre, Yannick Hold-Geoffroy, Miloš Hašan, Jae Shin Yoon, Thomas Leimkühler, Christian Theobalt, and Iliyan Georgiev. 2025. IntrinsicEdit: Precise generative image manipulation in intrinsic space. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 44, 4 (2025). doi:10.1145/3731173
- Zander Majercik, Jean-Philippe Guertin, Derek Nowrouzezahrai, and Morgan McGuire. 2019. Dynamic Diffuse Global Illumination with Ray-Traced Irradiance Fields. *Journal of Computer Graphics Techniques (JCGT)* 8, 2 (5 June 2019), 1–30. <http://jcgt.org/published/0008/02/01/>
- Thomas Müller, Markus H. Gross, and Jan Novák. 2017. Practical Path Guiding for Efficient Light-Transport Simulation. *Comput. Graph. Forum* 36 (2017), 91–100.
- Thomas Müller, Fabrice Rousselle, Jan Novák, and Alexander Keller. 2021. Real-time Neural Radiance Caching for Path Tracing. *ACM Trans. Graph.* 40, 4, Article 36 (Aug. 2021), 36:1–36:16 pages. doi:10.1145/3450626.3459812
- Matt Pharr, Wenzel Jakob, and Greg Humphreys. 2023. *Physically based rendering: From theory to implementation*. MIT Press. <https://pbr-book.org/4ed/>
- Renderman. 2025. Arbitrary Output Variables - RenderMan - RenderMan 27. <https://rmanwiki-27.pixar.com/space/REN27/542234274/Arbitrary+Output+Variables>. Accessed: 2026-01-13.
- Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2021. High-Resolution Image Synthesis with Latent Diffusion Models. arXiv:2112.10752 [cs.CV]
- Fabrice Rousselle, Wojciech Jarosz, and Jan Novák. 2016. Image-space Control Variates for Rendering. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)* 35, 6 (Dec. 2016), 169:1–169:12. doi:10/19cphw
- Fabrice Rousselle, Claude Knaus, and Matthias Zwicker. 2012. Adaptive rendering with non-local means filtering. *ACM Trans. Graph.* 31, 6, Article 195 (Nov. 2012), 11 pages. doi:10.1145/2366145.2366214
- Lukas Ruppert, Sebastian Herholz, and Hendrik PA Lensch. 2020. Robust fitting of parallax-aware mixtures for path guiding. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 147–1.
- Iman Sadeghi, Oleg Bisker, Joachim De Deken, and Henrik Wann Jensen. 2013. A practical microcylinder appearance model for cloth rendering. *ACM Trans. Graph.* 32, 2, Article 14 (April 2013), 12 pages. doi:10.1145/2451236.2451240
- Ari Silvennoinen and Jaakko Lehtinen. 2017. Real-time Global Illumination by Pre-computed Local Reconstruction from Sparse Radiance Probes. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)* 36, 6 (Nov. 2017), 230:1–230:13. doi:10.1145/3130800.3130852
- Eric Veach. 1997. *Robust Monte Carlo methods for light transport simulation*. Stanford University PhD thesis.
- Jiří Vorba, Johannes Hanika, Sebastian Herholz, Thomas Müller, Jaroslav Krivánek, and Alexander Keller. 2019. Path Guiding in Production. In *ACM SIGGRAPH 2019 Courses* (Los Angeles, California) (*SIGGRAPH '19*). ACM, New York, NY, USA, Article 18, 18:1–18:77 pages. doi:10.1145/3305366.3328091
- Jiří Vorba, Ondřej Karlík, Martin Šik, Tobias Ritschel, and Jaroslav Krivánek. 2014. On-line Learning of Parametric Mixture Models for Light Transport Simulation. *ACM Trans. Graph. (Proceedings of SIGGRAPH 2014)* 33, 4 (2014).
- Bing Xu, Tzu-Mao Li, Iliyan Georgiev, Trevor Hedstrom, and Ravi Ramamoorthi. 2024. Residual path integrals for re-rendering. *Computer Graphics Forum (Proceedings of EGSR)* 43, 4 (2024).
- K. Yan, V. Pegoraro, M. Droske, J. Vorba, and S. Zhao. 2024. Differentiating Variance for Variance-Aware Inverse Rendering. In *ACM SIGGRAPH Asia 2024 Conference Proceedings*.
- Zheng Zeng, Valentin Deschaintre, Iliyan Georgiev, Yannick Hold-Geoffroy, Yiwei Hu, Fujun Luan, Ling-Qi Yan, and Miloš Hašan. 2024. RGBX: Image decomposition and synthesis using material- and lighting-aware diffusion models. In *ACM SIGGRAPH 2024 Conference Papers* (Denver, CO, USA) (*SIGGRAPH '24*). Association for Computing Machinery, New York, NY, USA, Article 75, 11 pages. doi:10.1145/3641519.3657445
- Cheng Zhang and Shuang Zhao. 2020. Multi-Scale Appearance Modeling of Granular Materials with Continuously Varying Grain Properties. In *Eurographics Symposium on Rendering*. The Eurographics Association. doi:10.2312/sr.20201134