Robust In-Engine Texture Optimization from Inconsistent Generative Targets

TAEJOON KIM, Electronics and Telecommunications Research Institute (ETRI), South Korea
SEUNG-UK YOON, Electronics and Telecommunications Research Institute (ETRI), South Korea
SEONG-JAE LIM, Electronics and Telecommunications Research Institute (ETRI), South Korea
BON-WOO HWANG, Electronics and Telecommunications Research Institute (ETRI), South Korea
KINAM KIM, Electronics and Telecommunications Research Institute (ETRI), South Korea
SEUNG WOOK LEE, Electronics and Telecommunications Research Institute (ETRI), South Korea

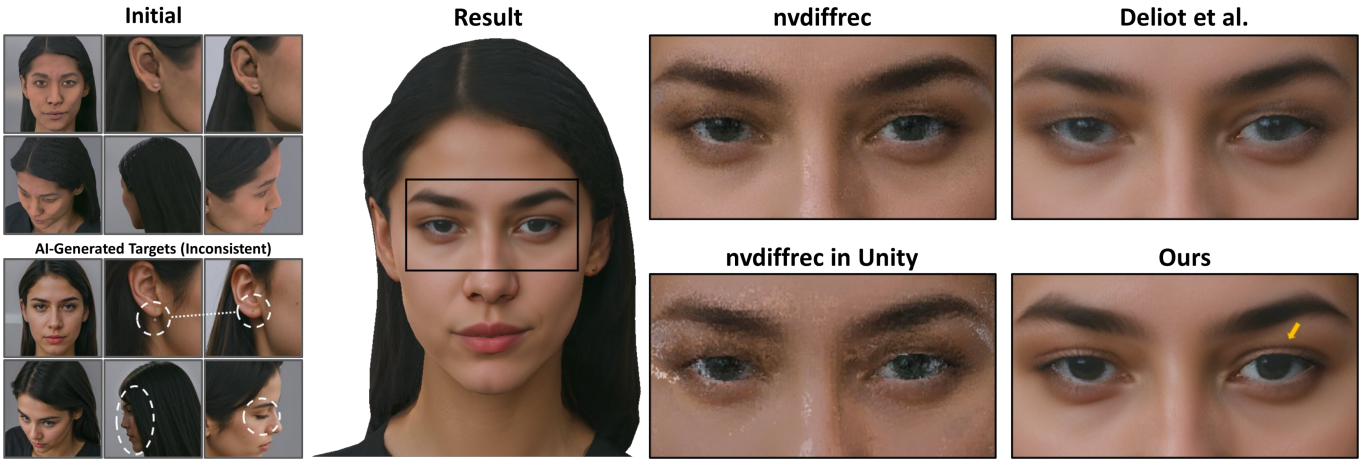


Fig. 1. Our method optimizes textures from inconsistent AI-generated target images, producing sharp results despite geometric misalignment and semantic hallucinations. Unlike existing methods that produce blurry or ghosting artifacts, our approach recovers high-fidelity details (e.g., the sharp creases of double eyelids) through joint uncertainty estimation and geometric alignment.

We propose a robust texture optimization framework that handles inconsistent AI-generated targets by operating directly within non-differentiable production pipelines. Standard optimization approaches struggle with inconsistent targets, often producing blurry textures and ghosting artifacts. Moreover, existing inverse rendering methods rely on differentiable renderers, causing a rendering gap when assets are deployed in production engines. To address the inconsistencies, we introduce a robust formulation that jointly optimizes a deformation grid and an uncertainty map. This formulation effectively decouples geometric misalignment from semantic hallucinations. To avoid the rendering gap, we leverage finite-difference gradient estimation to operate entirely within standard rasterizers. Extensive experiments demonstrate that our approach recovers sharp, high-fidelity textures from inconsistent AI-generated targets, achieving higher visual quality than existing methods. This makes our technique a practical tool for film production, gaming, and virtual reality, where flexibility and visual quality are paramount.

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

ACM Reference Format:
Taejoon Kim, Seung-Uk Yoon, Seong-Jae Lim, Bon-Woo Hwang, Kinam Kim, and Seung Wook Lee. 2026. Robust In-Engine Texture Optimization from Inconsistent Generative Targets. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers (SIGGRAPH Conference Papers '26), July 19–23, 2026, Los Angeles, CA, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3799902.3811170

Authors' Contact Information: Taejoon Kim, taejoonkim@etri.re.kr, Electronics and Telecommunications Research Institute (ETRI), Daejeon, South Korea; Seung-Uk Yoon, suyoon@etri.re.kr, Electronics and Telecommunications Research Institute (ETRI), Daejeon, South Korea; Seong-Jae Lim, sjlim@etri.re.kr, Electronics and Telecommunications Research Institute (ETRI), Daejeon, South Korea; Bon-Woo Hwang, bwhwang@etri.re.kr, Electronics and Telecommunications Research Institute (ETRI), Daejeon, South Korea; Kinam Kim, rlskal@etri.re.kr, Electronics and Telecommunications Research Institute (ETRI), Daejeon, South Korea; Seung Wook Lee, tajinet@etri.re.kr, Electronics and Telecommunications Research Institute (ETRI), Daejeon, South Korea.

SIGGRAPH Conference Papers '26, July 19–23, 2026, Los Angeles, CA, USA

ACM ISBN 979-8-4007-2554-8/2026/07
https://doi.org/10.1145/3799902.3811170

1 Introduction

In recent years, the demand for photorealistic visual experiences has driven the CG industry toward ultra-high-quality, intricately detailed textures. While traditionally resource-intensive, this texture creation paradigm is rapidly shifting with the advent of large-scale diffusion models [Rombach et al. 2022]. These models enable the

synthesis of high-fidelity textures from simple text prompts or reference images. However, realizing this potential within production pipelines remains challenging.

The primary challenge arises from *inconsistencies* in AI-generated multi-view images—geometric misalignment and semantic hallucinations such as misplaced facial features (e.g., faces incorrectly appearing in rear-facing views) or transient accessories (Figure 1). Under these conditions, standard pixel-wise optimization produces blurry textures and ghosting artifacts. In addition, practical deployment introduces a *rendering gap* between external optimization frameworks and deployment environments. While PBR parameters are theoretically portable, practical implementations of shading models (e.g., Fresnel approximations, split-sum IBL integration) vary across engines. When optimizing in an external differentiable renderer [Jakob et al. 2022; Li et al. 2018; Munkberg et al. 2022], the optimizer inevitably bakes renderer-specific biases into the texture maps to minimize the reconstruction loss. Consequently, these overfitted textures degrade when imported into a different engine, failing to reproduce the intended appearance.

In this paper, we propose a robust texture optimization framework for inconsistent generative targets. To handle target inconsistencies, we introduce two key components: (1) a per-view *uncertainty map* that automatically identifies and down-weights unreliable pixels, and (2) a per-view *deformation grid* that compensates for geometric misalignments between rendered and target images. This interplay between the two components is essential. If geometric alignment is not applied, valid texture details displaced by geometric discrepancies would be incorrectly rejected as outliers. Conversely, without uncertainty-based rejection, the deformation grids would warp hallucinated artifacts directly into the texture. A multi-phase optimization strategy further stabilizes convergence by sequentially refining textures, alignments, and appearance details. To enable practical deployment, we implement this formulation directly within the target engine's shader programs via finite-difference gradient estimation. This workflow ensures that the textures are optimized for the specific lighting and shading of the final application.

We validate our framework through extensive experiments on diverse AI-generated targets. Our results demonstrate that the proposed joint optimization effectively eliminates the blurring and ghosting artifacts, yielding production-ready assets with superior perceptual quality.

Our contributions can be summarized as follows:

- **Robust Optimization of Inconsistent Targets**: A joint formulation combining uncertainty estimation and deformation grids, with a multi-phase optimization strategy that decouples geometric misalignment from semantic hallucinations.
- **In-Engine Texture Optimization**: A fully shader-based, finite-difference approach for real-time engines (e.g., Unity and Unreal), enabling our robust formulation to operate without differentiable pipelines or third-party libraries.
- **Versatility and Efficiency**: A highly practical framework that supports arbitrary shading models and achieves rapid optimization (within a minute), enabling fast iteration loops in production workflows.

2 Related Work

2.1 Data-driven Inverse Rendering

Traditional inverse rendering approaches address intrinsic image decomposition [Li and Snavely 2018], light source estimation [Li et al. 2020; Srinivasan et al. 2020], BRDF estimation [Deschaintre et al. 2018; Zhu et al. 2022], and relighting [Philip et al. 2021]. Recently, data-driven techniques have been widely employed to address these problems based on deep priors learned from large-scale datasets. They typically rely on large, high-quality datasets, which reduce measurement requirements and relax constraints on input images such as HDR capture [Lin et al. 2025a] or controlled lighting conditions [Fei et al. 2024a].

Since ground-truth labeling for real-world scenes is challenging and time-consuming, some approaches tackle the problem by providing benchmarks or real-world datasets using carefully designed hardware capture systems [Choi et al. 2025; Kuang et al. 2023; Liu et al. 2023]. In contrast, several techniques address this limitation by utilizing synthetic large-scale datasets [Li et al. 2025, 2021; Liang et al. 2025; Roberts et al. 2021].

More recently, neural field representations have been extended for inverse rendering with improved efficiency and quality. TensoIR [Jin et al. 2023] leverages tensor factorization to jointly estimate geometry, materials, and illumination, while GS-IR [Liang et al. 2024] adapts 3D Gaussian Splatting for physically-based material decomposition. Concurrently, diffusion-based approaches have emerged for per-pixel intrinsic estimation: Kocsis et al. [Kocsis et al. 2024] fine-tune a diffusion model for single-image material prediction, and Zeng et al. [Zeng et al. 2024b] present a unified framework that bridges intrinsic decomposition and image synthesis through diffusion priors.

However, these methods assume clean, consistent training data and do not address the inconsistencies inherent in AI-generated multi-view images.

2.2 Optimization-based Inverse Rendering

Numerous approaches have explored inverse rendering by estimating scene parameters from input images using differentiable rendering, which propagates gradients from image pixels back to scene parameters. These gradients directly optimize geometry, lighting, and materials, or can be transferred to neural networks. However, this process relies heavily on primary rendering pipelines, requiring all renderer modules to be implemented as differentiable.

One of the early texture optimization methods [Zhou and Koltun 2014] addresses texture misalignment by refining the 3D-to-2D mapping through camera pose and non-rigid image deformation in a differentiable framework. Bi et al. [Bi et al. 2017] instead propose a patch-based texture synthesis approach that improves robustness to geometric and pose errors by reconstructing aligned target images via bidirectional patch similarity. However, both methods assume physically consistent multi-view observations, whereas our approach additionally handles semantic hallucinations and geometric inconsistencies within a non-differentiable rendering pipeline.

Physically-based methods [Bangaru et al. 2020; Jakob et al. 2022; Li et al. 2018] tackle global illumination with differentiable rendering frameworks. Differentiable path tracing [Yu et al. 2023] handles multi-bounce lighting at scene level but often requires significant time and struggles with high-order discontinuities. Volume-rendering techniques like NeRF [Mildenhall et al. 2021] represent scenes as implicit volume functions with pre-calculated irradiance, better suited for whole scenes than detailed objects. However, they face challenges in disentangling geometry, materials, and illumination. PBR-NeRF [Wu et al. 2025] addresses this limitation by using physics-based priors to jointly estimate these components for improved relighting and material editing. In parallel, another line of work focuses on accelerating NeRF through methods like SNeRG [Hedman et al. 2021], Mobile-NeRF [Chen et al. 2023b], and DIVeR [Wu et al. 2021], as well as reducing training time with IBRNet [Wang et al. 2021], MVSNeRF [Chen et al. 2021], Frugal-NeRF [Lin et al. 2025b] and others.

To address inefficiencies arising from repeatedly querying implicit volumetric representations, various 3D Gaussian-based approaches [Fei et al. 2024b; Kerbl et al. 2023] have been proposed recently. Since they reconstruct a scene as a collection of 3D Gaussians by computing the corresponding intensity of an intersecting ray like point-based rendering, they can be easily parallelized. Therefore, training and rendering time reduce significantly compared with NeRF-based approaches.

Most of those optimization-based inverse rendering methods require a differentiable renderer like nvDiffRast [Laine et al. 2020], Mitsuba 3 renderer [Jakob et al. 2022] or others implemented on PyTorch/Tensorflow. Although these frameworks provide useful gradients, they may not match the exact shader implementations, engine-specific optimizations, and asset workflows of production renderers such as Unity, Unreal, V-Ray, or Arnold. Since the performance of the primary renderer significantly impacts inverse rendering, generating photorealistic images of detailed objects from estimated scene parameters remains challenging due to the complexity of inverse rendering methods. Fischer et al. [Fischer and Ritschel 2023, 2024] approximate gradients via surrogate learning for non-differentiable pipelines, but focus on path tracing rather than production engine integration.

A few algorithms try to enable differentiation within standard rasterization pipelines to tackle this problem. Deliot et al. [Deliot et al. 2024] propose a stochastic gradient estimation method using random perturbations, while Bangaru et al. [Bangaru et al. 2023] introduce Slang.D to provide automatic differentiation via language extensions. While these methods successfully enable gradient-based optimization in real-time engines, they assume consistent target images.

2.3 Texture Synthesis

Unlike the inverse rendering approaches estimating scene parameters from images, there have recently been generative techniques that deal with geometry and textures of an object by adopting emerging latent diffusion models [Rombach et al. 2022]. Fantasia3D [Chen et al. 2023a] disentangles geometry and appearance using a hybrid DMTet representation with PBR materials. DreamMat [Zhang et al. 2024a] extends this by using geometry- and light-aware diffusion priors to generate high-quality PBR material maps. TexFusion [Cao et al. 2023] synthesizes textures for given 3D geometries by performing denoising diffusion iterations in multi-view and aggregating the trajectories on a latent texture map. TEXTure [Richardson et al. 2023] and Paint3D [Zeng et al. 2024a] use a pre-trained depth-to-image diffusion model to paint a 3D model from different viewpoints. Text2Tex [Chen et al. 2023c] also uses a pre-trained depth-aware image diffusion model to synthesize high-resolution partial textures from multiple viewpoints progressively. TexRO [Wu et al. 2024] utilizes a viewpoint selection strategy to adaptively optimize UV texture of a known 3D mesh. TexPainter [Zhang et al. 2024b] employs an optimization-based color fusion to keep multi-view consistency based on Denoising Diffusion Implicit Models. While these methods attempt to enforce multi-view consistency during generation, the resulting images still exhibit inconsistencies such as geometric misalignment and hallucinated details.

Recent works further improve texture generation by enforcing stronger multi-view consistency during diffusion [Yildirim et al. 2025] or by directly modeling textures in native 3D representations without relying on multi-view fusion [Zeng et al. 2025]. While these methods attempt to improve consistency across views or to represent textures as a function over 3D space during generation, fully resolving inconsistencies remains challenging due to geometric inaccuracies and the inherent limitations of generative priors.

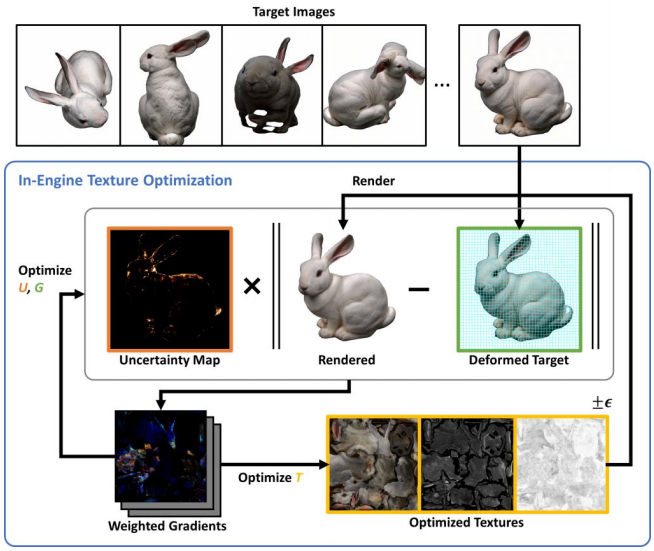


Fig. 2. Overview of our method. Our optimization estimates an uncertainty map U to down-weight unreliable pixels and a deformation grid G to correct geometric misalignments. The weighted gradients update textures T via finite-difference estimation. The entire pipeline runs within the game engine's shader programs.

3 Method

Unlike conventional inverse rendering, our formulation explicitly targets inconsistent generative supervision, where both geometric and semantic inconsistencies are present. Figure 2 and Algorithm 1

Algorithm 1 Texture Optimization

Require: Scene, initial textures $\{T\}$, target images $\{I^v_{\text{tgt}}\}$ for view v
Ensure: Optimized textures $\{T\}$

for each iteration **do**
  for each texture T_i **do**
    Generate random noise Z
    for each view v **do**
      $I^v, I^v_+ \leftarrow \text{Render}(T_i; v), \text{Render}(T_i + \epsilon Z; v)$
      Update U^v and G^v using I^v and I^v_{tgt}
      $I^v_{\text{warp}} \leftarrow W(I^v_{\text{tgt}}, G^v)$
      Compute ∇_p via $(I^v_+ - I^v)$, $(I^v - I^v_{\text{warp}})$, and U^v
      Render screen-to-UV mapping $M^v_{p\to t}$
      Accumulate ∇_t from ∇_p using $M^v_{p\to t}$
    end for
    Update texture T_i using ∇_t
  end for
end for

illustrate our overall pipeline. The objective of our method is to refine given textures to produce rendered images that closely match a set of target images from various views while handling inherent inconsistencies across the views. For each view v, we update the uncertainty map U^v and deformation grid G^v to align the inconsistent target I^v_{tgt} with the current rendering I^v. We then estimate the screen-space gradient ∇_p by perturbing the texture with random noise Z and measuring the response against the warped target I^v_{warp}, weighted by the uncertainty U^v. Finally, these gradients are back-projected to the UV texture space ∇_t via the screen-to-UV mapping $M^v_{p\to t}$ to update the texture.

3.1 Problem Formulation

Given a 3D mesh with initial textures $\{T\}$ and a set of target images $\{I^v_{\text{tgt}}\}$ captured from multiple viewpoints v, our goal is to optimize the textures $\{T\}$ such that the rendered images $\{I^v\}$ match the target images. However, directly minimizing the pixel-wise difference is insufficient when the target images lack consistency across views: the optimization would attempt to encode conflicting information into the textures, leading to blur and ghosting artifacts.

To address this, we jointly optimize three components: the texture maps T, per-view deformation grids G^v that align target images to the rendered images, and per-view uncertainty maps U^v that down-weight unreliable regions in the target images. The optimization problem can be formulated as:

$$\mathcal{L}_{total} = \sum_v \left(\mathcal{L}_{photo}(T, G^v, U^v) + \lambda_{reg}\mathcal{L}_{smooth}(G^v)\right), \tag{1}$$

where $\mathcal{L}_{photo}$ measures the photometric error and $\mathcal{L}_{smooth}$ regularizes the deformation grid to prevent grid distortions in textureless regions.

3.2 Gradient Estimation via Finite-Differences

A core design principle is performing texture optimization entirely within the target production pipeline. This eliminates the rendering gap that occurs when textures optimized in external differentiable renderers are imported into production engines, which often leads to suboptimal results due to differences in rendering algorithms, lighting models, and shader implementations. To enable gradient-based optimization within non-differentiable rasterizers (e.g., Unity, Unreal), we implement a pixel-parallel finite-difference method directly in shaders. Rather than perturbing each texel independently, which would require an impractical number of renderings, we perturb all texels simultaneously with a random $\pm\epsilon$ per texel [Deliot et al. 2024]. The gradients are then accumulated in UV space across all views:

$$\nabla_t = \sum_v M^v_{p\to t}(\nabla^v_p), \tag{2}$$

where $M^v_{p\to t}$ maps screen space pixels to texture UV space for view v, and ∇^v_p is the per-pixel gradient estimated in screen space.

(a) Rendered (b) Target (c) Uncertainty Map

Fig. 3. Estimated uncertainty visualization. (a) The rendered image with optimized textures. (b) The target image from a diffusion model. (c) The estimated uncertainty map visualized as a heatmap. The optimizer identifies inconsistent regions (e.g., hallucinated details) and assigns high uncertainty to prevent these artifacts from contaminating the texture.

3.3 Robust Photometric Loss

Standard photometric losses (e.g., MSE) are highly sensitive to outliers and inconsistencies in the target images. To mitigate this, we need a mechanism to automatically identify and down-weight unreliable pixels during optimization.

Let $\mathbf{e}^v_p$ denote the pixel-wise error at pixel p for view v:

$$e^v_p = I^v_p - W(I^v_{\text{tgt}}, G^v)_p, \tag{3}$$

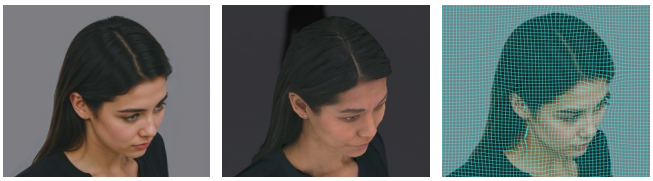
where $W(I^v_{\text{tgt}}, G^v)$ is the warped target image using the deformation grid G^v.

To handle outliers (e.g., hallucinations), we adopt a robust loss formulation based on predictive uncertainty [Kendall and Gal 2017; Martin-Brualla et al. 2021]. Let Φ be a color space transformation and $s^v_p = \log((\sigma^v_p)^2)$ be the predicted log-variance where σ^v_p represents the estimated uncertainty of the observation at pixel p. The photometric loss for a pixel p is defined as:

$$\mathcal{L}_{photo}(p) = \frac{||\Phi(e^v_p)||^2_{\mathbf{w}}}{2(\sigma^v_p)^2 + \epsilon_{\text{stab}}} + \frac{1}{2}s^v_p, \tag{4}$$

where ϵ_{stab} is a fixed tiny constant to avoid division by zero.

The first term penalizes the pixel-wise error weighted by the inverse variance, while the second term prevents the trivial solution of predicting infinite uncertainty. We employ a weighted norm in the numerator ($||\mathbf{x}||^2_{\mathbf{w}} = \sum w_c x_c^2$) to selectively emphasize specific error components depending on the texture type. For diffuse textures, we define Φ as the transformation to the YCoCg color space to explicitly



(a) Original Target (b) Rendered (c) Warped Target

Fig. 4. Deformation grid visualization. (a) The original target image from a diffusion model. (b) The current rendered image. (c) The target image warped by the optimized deformation grid to align with the rendered image.

separate luminance and chromaticity components:

$$||\Phi(\mathbf{e})||_{\mathbf{w}}^2 = w_Y(e_Y)^2 + w_{Chroma}(e_{Co})^2 + w_{Chroma}(e_{Cg})^2. \quad (5)$$

We set $w_Y = 0.1$ and $w_{Chroma} = 1.0$ to prioritize chromaticity differences over luminance, which helps the optimization focus on color fidelity rather than brightness variations. For non-color textures (e.g., metallic or roughness), we use standard RGB Euclidean distance ($\Phi(\mathbf{x}) = \mathbf{x}, \mathbf{w} = \mathbf{1}$).

This formulation enables adaptive outlier handling through the estimated uncertainty. When the optimization encounters inconsistent regions (e.g., hallucinated details in AI-generated images), the uncertainty estimation increases σ_p^v, which reduces the influence of these pixels on the texture update. Conversely, for reliable regions with consistent appearance across views, σ_p^v remains low, ensuring that these pixels strongly guide the optimization. Figure 3 visualizes an example where the estimated uncertainty map correctly identifies inconsistent regions in the target image.

3.4 Geometric Alignment via Deformation Grid

To handle misalignments between the rendered images and target images, we introduce a per-view 2D deformation grid G^v that warps the target image I_{tgt}^v. Each grid is a low-resolution (e.g., 64×64) offset field that warps the target image to better align with the rendered image I^v.

For each view v, the grid G^v stores 2D displacement vectors. Target image pixels are sampled using bilinear interpolation based on these displacements:

$$W(I_{\text{tgt}}^v, G^v)_p = I_{\text{tgt}}^v(p + \mathbf{interp}(G^v, p)). \quad (6)$$

Gradients for the deformation grid are computed using finite-differences at screen resolution. These high-resolution gradients are then efficiently downsampled to the grid resolution (e.g., 64×64) via GPU-accelerated mipmap generation. To prevent the grid from overfitting to background noise, we apply a depth mask during gradient computation. Figure 4 illustrates how the optimized deformation grid warps the target image to align with the rendered geometry.

Since the deformation grid optimization is an ill-posed problem, particularly in textureless regions, we enforce a smoothness constraint using a standard Laplacian regularizer $\mathcal{L}_{smooth}(G^v) = \sum ||\Delta G^v||^2$ to penalize high-frequency variations, preventing folds and ensuring physically plausible deformations.

3.5 Optimization Strategy

While our framework allows for the simultaneous optimization of all variables (T, G, U), a naive joint optimization can lead to suboptimal convergence, particularly when the initial texture significantly deviates from the target. In such cases, the photometric error is dominated by texture appearance rather than geometric misalignment, causing the grid optimization to converge to incorrect solutions.

To address this, we employ a multi-phase strategy that progressively refines the results:

(1) **Texture Warm-up**: Optimize textures T and uncertainties U with fixed identity grids G to generate *anchor textures* that roughly capture the target appearance.
(2) **Geometric Alignment**: Fix textures T and uncertainties U, optimize grids G to align target images to the rendered images based on the anchor textures. Since the anchor textures already approximate the target appearance, this phase effectively corrects geometric misalignments.
(3) **Appearance Refinement**: Finally, we refine the textures T and uncertainties U using the aligned target images to recover high-frequency details. The fixed grids G ensure that the texture optimization focuses solely on appearance without re-introducing alignment errors.

This phased approach is particularly beneficial when working with AI-generated target images that exhibit significant inconsistencies. When the target images are well aligned, single-phase joint optimization can achieve comparable performance to the proposed multi-phase strategy.

4 Experiments

4.1 Implementation Details

Our framework is implemented entirely in Unity, demonstrating seamless integration into standard production pipelines. Since all optimization steps are implemented in the fragment shader, this method can be easily transferred to other engines that support shader-based rendering (e.g., Unreal Engine).

Dataset construction. All 3D object examples used in our experiments are exactly those shown in the figures; no additional undisclosed data is used. Geometry is obtained from publicly available 3D models. Initial textures are generated using DreamMat [Zhang et al. 2024a], except for human characters, where artist-provided textures are used since DreamMat did not produce satisfactory results for human faces. All assets are rendered in Unity, and the rendered images are processed through a Stable Diffusion [Rombach et al. 2022] 1.5 img2img pipeline with the RealDream 12 checkpoint to produce the target images.

We set both the rendering and target image to 1024×1024 resolution. For each iteration, we use randomly sampled 36 viewpoints distributed uniformly around the object. All textures, including intermediate results, are stored and processed in 32-bit floating-point format to offer sufficient precision for the texture perturbation, gradient estimation and optimization. We employ different learning rates for each component: 0.001 for texture optimization, 0.0001 for grid deformation, and 0.003 for uncertainty estimation. The smoothness regularization weight λ_{reg} is set to 5. The perturbation

magnitude ϵ is set to 0.001 and we employ **Adam** for optimization. The optimization typically completes 400 steps in approximately 1 minute for a single 2048×2048 texture using 36 views on an NVIDIA RTX 4090 GPU; computation time scales linearly with the number of textures and views. This optimization time is suitable for iterative offline asset refinement workflows where artists repeatedly update assets.

4.2 Qualitative Comparisons

We evaluate our method on challenging scenarios using AI-generated multi-view images as targets, constructed as described in Section 4.1. Importantly, our dataset includes cases with intentional inconsistencies such as local shape deviations, appearing/disappearing accessories, and inconsistent surface details across views.

Our method consistently produces high-fidelity textures across diverse geometries and materials, maintaining sharp details even under significant target inconsistencies (Figure 10). We compare our method against two categories of methods:

Text-driven generation. Fantasia3D [Chen et al. 2023a] and DreamMat [Zhang et al. 2024a], which generate textures from scratch based on text prompts. While not a direct inverse rendering comparison, they serve as a reference for generative quality.

Inverse rendering. nvdiffrec [Munkberg et al. 2022] and Deliot et al. [Deliot et al. 2024], which optimize textures using the same target images as ours.

Figure 11 presents a visual comparison across various subjects. All baselines except **nvdiffrec** were imported into Unity for fair comparison. The **nvdiffrec** results were rendered using their own differentiable renderer. **Fantasia3D** and **DreamMat** successfully generate coherent textures but often struggle with human subjects and generally lack photorealism compared to inverse rendering approaches. While **nvdiffrec** produces high-quality results within its native framework, the quality degrades when assets are imported into Unity due to the rendering gap–discrepancies in lighting models and shader implementations. In contrast, **Deliot et al.** maintains quality as it optimizes directly in the target environment. **nvdiffrec** shows some noise and **Deliot et al.** suffers from blurring and ghosting artifacts due to inconsistencies in the target images. Our method, however, produces clean and sharp textures, effectively handling the inconsistencies through the uncertainty map and deformation grid.

We provide additional results in the supplementary material, including optimization on real photographs and application to Unity's High Definition Render Pipeline (HDRP). These results further validate our method's robustness and practical applicability across different scenarios and rendering pipelines.

4.3 Quantitative Comparison

Quantitative evaluation of texture optimization from AI-generated targets is inherently challenging due to the absence of ground-truth images. To enable quantitative analysis, we design controlled experiments using *Synthetic Corruption*. For all target images, two types of corruption are applied to validate our key components: (1) *Random Elastic Deformation*, applying random 2D geometric warping to simulate geometric misalignments, and (2) *Random Occlusion*, drawing random colored boxes at arbitrary locations to simulate semantic hallucinations. This setup allows us to measure reconstruction accuracy against known ground truth.

Table 1. Ablation study on synthetic corruption. Uncertainty estimation alone outperforms all evaluated baselines, demonstrating its central role in handling inconsistent targets. Multi-phase optimization further improves performance by resolving the interference between geometric alignment and appearance refinement.

Method	PSNR ↑	SSIM ↑	LPIPS ↓
nvdiffrec	30.7	0.917	0.086
Deliot et al.	32.5	0.932	0.074
Ours (Uncertainty only)	34.0	0.944	0.050
Ours (Deformation only)	30.7	0.927	0.064
Ours (Both, Single Phase)	31.7	0.937	0.050
Ours (Both, Multi-phase)	**36.2**	**0.965**	**0.025**

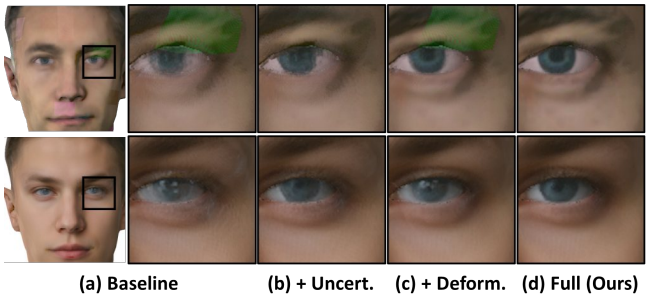


Fig. 5. Visualization of the impact of each component on both synthetic corruption (top) and real AI-generated targets (bottom). The baseline (a) suffers from both blurring and baked-in artifacts. Using only uncertainty estimation (b) removes artifacts (occlusions in top, hallucinated highlights on the eye in bottom) but the texture remains blurry due to geometric misalignment. Using only grid deformation (c) sharpens features but accumulates inconsistent artifacts from multi views. Our full method (d) combines both components to achieve clean, sharp, and artifact-free optimization.

Table 1 summarizes the quantitative results using PSNR, SSIM, and LPIPS metrics. **nvdiffrec** is evaluated in its native differentiable renderer, as importing its results into Unity would conflate optimization quality with the rendering gap. **nvdiffrec** and **Deliot et al.** suffer significantly from the corruptions, blurring the texture to average out the errors. The key finding is that uncertainty estimation is the primary driver of improvement: using uncertainty alone achieves 34.0 dB, outperforming Deliot et al. (32.5 dB) by +1.5 dB. This demonstrates that identifying and down-weighting unreliable pixels is crucial for handling inconsistent targets. In contrast, adding grid deformation alone does not improve quality (30.7 dB), and even combining both components in single-phase optimization (jointly optimizing all variables) degrades performance (31.7 dB) compared to uncertainty alone (34.0 dB). This is because the initial texture differs substantially from the target appearance, causing the deformation grid to establish incorrect correspondences. We observe that this failure is not only reflected in quantitative metrics, but

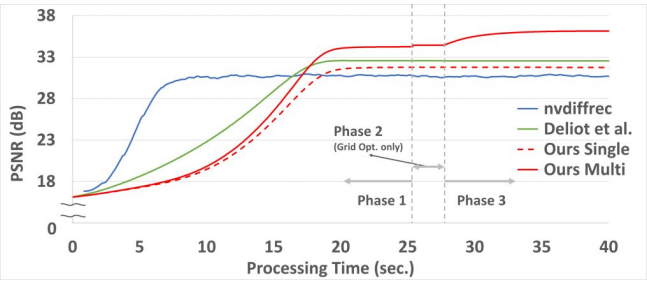


Fig. 6. Convergence comparison on synthetic corruption. Our multi-phase approach initially matches the single-phase trajectory during **Phase 1** (texture warm-up). After geometric alignment in **Phase 2**, **Phase 3** achieves a significant quality jump, surpassing all baselines. Note that nvdiffrec converges faster initially but saturates at a lower PSNR and oscillates due to its inability to handle target inconsistencies.

Table 2. Sensitivity analysis on hyperparameters. (Left) Robustness to perturbation magnitude ϵ. The method remains stable across two orders of magnitude. (Right) Impact of deformation grid resolution. The 64×64 grid achieves the most balanced performance.

ϵ	PSNR ↑	SSIM ↑	LPIPS ↓	Grid Size	PSNR ↑	SSIM ↑	LPIPS ↓
0.0005	36.0	0.962	0.025	16×16	36.3	0.959	0.034
0.001	**36.2**	0.965	**0.025**	32×32	**36.7**	0.964	0.029
0.002	36.1	**0.966**	0.026	**64×64**	36.2	**0.965**	0.025
0.01	36.1	0.966	0.027	128×128	35.8	0.963	**0.023**
0.05	36.1	0.966	0.027	256×256	35.3	0.960	0.029

Table 3. Sensitivity analysis on number of views. Quality improves consistently with more views but saturates beyond 16 views.

Views	PSNR ↑	SSIM ↑	LPIPS ↓	Views	PSNR ↑	SSIM ↑	LPIPS ↓
1	20.4	0.675	0.298	8	27.2	0.890	0.100
2	24.0	0.797	0.191	16	32.7	0.939	0.035
4	25.6	0.856	0.134	32	35.7	0.963	0.025

also manifests as visible geometric drift and texture baking artifacts, which we demonstrate in the accompanying video.

Our multi-phase strategy resolves this interference, achieving 36.2 dB (an additional +2.2 dB over uncertainty alone). Figure 6 validates this strategy: a distinct quality jump occurs immediately after enabling geometric alignment (**Phase 2**), confirming that the anchor textures from **Phase 1** successfully prevent the incorrect correspondences observed in single-phase optimization. Notably, when applying our *texture initialization strategy* (see the supplementary material) to provide a better starting point, single-phase optimization achieves comparable results (36.9 dB/0.965/0.027) to multi-phase (37.1 dB/0.964/0.024), confirming that single-phase optimization suffices when reliable initial textures are available. Note that both grid deformation and uncertainty estimation are implemented as simple shader passes (a few blits per iteration), adding negligible computational overhead to the baseline optimization.

Fig. 7. Comparison of results across three Unity shaders (left to right): Standard, Standard (Specular Setup), and Mobile/Bumped Diffuse.

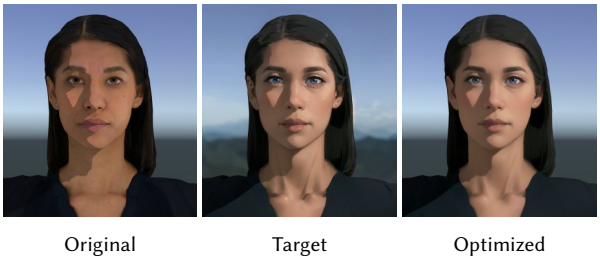


Fig. 8. Our optimization method can also be applied to custom shaders. This is an example of applying a cartoon-like shader.

4.4 Sensitivity Analysis

Table 2 analyzes sensitivity to two hyperparameters: the perturbation magnitude ϵ and the resolution of the deformation grid. Our method shows remarkable stability across two orders of magnitude: PSNR varies by only 0.2 dB between $\epsilon = 0.0005$ and $\epsilon = 0.05$. Such insensitivity to hyperparameter selection simplifies practical deployment. Regarding the deformation grid resolution, coarse grids (e.g., 16×16) lack the flexibility to correct complex local misalignments, resulting in higher perceptual error (LPIPS 0.034). Conversely, overly fine grids (256×256) tend to overfit to high-frequency noise or artifacts, leading to unstable convergence and a drop in PSNR. We found that a resolution of 64×64 provides the optimal balance, achieving the highest structural similarity (SSIM 0.965) and low perceptual error.

We also analyze the impact of the number of views (See Table 3). Quality improves consistently as views increase but saturates beyond 16 views (32.7 dB vs. 35.7 dB at 32 views), suggesting that 32 views provide a practical trade-off between quality and computation.

4.5 Shader Variation

To verify that our optimization is not limited to a single shader, we tested four different shaders available in Unity: *Standard*, *Standard (Specular Setup)*, *Mobile/Bumped Diffuse*, and a custom shader. As a custom shader, we chose a cartoon-like shader to show that it can be applied to completely different types of shaders. Figure 7 and 8 show that our method successfully optimizes in all four cases, though the final image fidelity depends on each shader's ability to represent the surface.

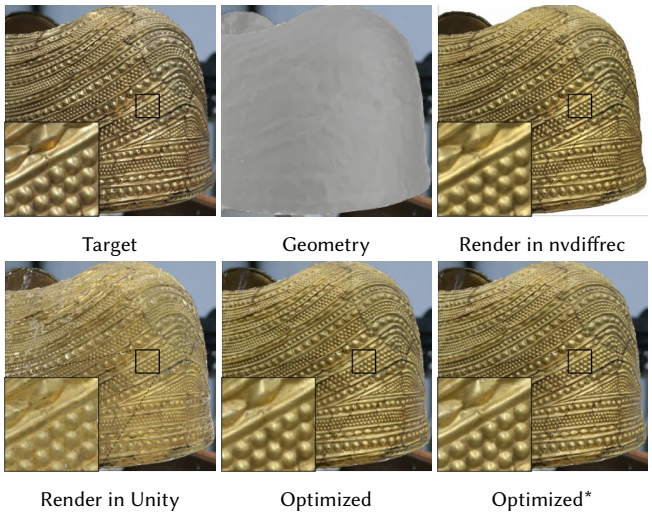


Fig. 9. Comparison with nvdiffrec [Munkberg et al. 2022] that uses a differentiable renderer. **Top**: the ground-truth target from the *Mold Gold Cape* dataset (left), nvdiffrec-generated geometry (center), and the renderer's native result (right). **Bottom**: the rendering result in Unity (left), our in-engine texture optimization (center), and an outcome of using higher-resolution target images for the texture optimization (right).

4.6 Comparison with Analytical Gradients

We conduct a comparative analysis between our numerical finite-difference approach and the analytical differentiation used in standard differentiable renderers like nvdiffrec [Munkberg et al. 2022], evaluating them in terms of gradient accuracy and computational efficiency.

A primary concern with finite-differences is approximation error. To quantify this, we implemented our finite-difference approach within nvdiffrec's framework. With geometry and lighting fixed, nvdiffrec's analytical gradients achieved 24.3 dB PSNR while our finite-difference method achieved 24.1 dB. The marginal difference (< 0.2 dB) confirms that numerical gradients are a viable alternative to analytical gradients.

Beyond comparable accuracy, our per-texel formulation also offers practical flexibility: since our approach computes gradients per-texel without batching constraints, it naturally supports higher-resolution target images, yielding sharper details (Figure 9, Optimized*).

5 Limitations and Future Work

Although our method provides high-quality results in diverse scenarios, several challenges remain. First, assets with complex transparency or fine geometric details, such as hair strands and alpha-heavy assets, remain challenging. Second, while our uncertainty map effectively handles semantic hallucinations, extreme cases where the majority of target pixels are unreliable can still degrade optimization quality. Third, our method provides only partial illumination decoupling: generative targets may contain view-consistent but physically incorrect lighting patterns (e.g., specular highlights fixed across views), which can be partially baked into the recovered textures. This is due to ambiguity in the input, where reflectance and illumination cannot be fully separated, although the effect is reduced when the targets have sufficient lighting variation.

There are several interesting directions for future work. While we currently target rasterization-based engines (e.g., Unity, Unreal), extending our method to ray-tracing pipelines could improve fidelity for complex light interactions such as caustics and global illumination. Additionally, while we prioritize preserving artist-created geometry in this work, extending to joint optimization of shape and environment maps would broaden the applicability of our approach. Finally, another exciting avenue is tighter integration with generative models. Instead of treating image generation and texture optimization as separate stages, a unified pipeline that iteratively refines both prompts and the texture parameters could yield even higher fidelity and consistency.

6 Conclusion

In this paper, we present a robust texture optimization framework that integrates seamlessly into existing non-differentiable rendering pipelines while handling the inherent inconsistencies of AI-generated target images. By combining per-view deformation grids and uncertainty-aware optimization, our approach effectively decouples geometric misalignment from semantic hallucinations, recovering sharp textures where previous methods produce blur and ghosting artifacts. The entire optimization runs directly within shader programs, avoiding the constraints of specialized differentiable renderers and maintaining compatibility with widely used engines such as Unity and Unreal.

Experimental results across diverse scenarios demonstrate that our method outperforms existing approaches in both visual quality and robustness. Overall, our framework addresses the growing need for practical tools that bridge generative AI and production rendering pipelines. By enabling artists to leverage AI-generated imagery without sacrificing fidelity or workflow efficiency, our approach provides a robust and efficient solution for studios, game developers, and researchers seeking to integrate emerging generative techniques into their existing infrastructure.

Acknowledgments

We thank the creators and maintainers of the datasets and asset repositories used in our experiments: Objaverse [Deitke et al. 2022], licensed under CC BY 4.0, for the cat, turtle, and apple models; Stanford-ORB [Kuang et al. 2023] for the teapot and car models; the NeRD dataset, licensed under Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International, for the mold gold cape model; Reallusion for the human character assets; the McGuire Computer Graphics Archive for the Bunny and teapot assets in Figure 10; Keenan Crane's repository for the Nefertiti model in Figure 10; and Real-World Textured Things [Maggiordomo et al. 2020] for the remaining Figure 10 assets. This work was supported by Electronics and Telecommunications Research Institute (ETRI) grant funded by the Korean government [26ZC1120, Development of Spatial Media Technology and Interaction Technology for Convergence of the Real and Virtual World].

References

Sai Praveen Bangaru, Tzu-Mao Li, and Frédo Durand. 2020. Unbiased warped-area sampling for differentiable rendering. *ACM Trans. Graph.* 39, 6, Article 245 (Nov. 2020), 18 pages.

Sai Praveen Bangaru, Lifan Wu, Tzu-Mao Li, Jacob Munkberg, Gilbert Bernstein, Jonathan Ragan-Kelley, Frédo Durand, Aaron Lefohn, and Yong He. 2023. SLANG.D: Fast, Modular and Differentiable Shader Programming. *ACM Trans. Graph.* 42, 6, Article 264 (Dec. 2023), 28 pages. https://doi.org/10.1145/3618353

Sai Bi, Nima Khademi Kalantari, and Ravi Ramamoorthi. 2017. Patch-based optimization for image-based texture mapping. *ACM Trans. Graph.* 36, 4, Article 106 (July 2017), 11 pages. https://doi.org/10.1145/3072959.3073610

Tianshi Cao, Karsten Kreis, Sanja Fidler, Nicholas Sharp, and KangXue Yin. 2023. TexFusion: Synthesizing 3D Textures with Text-Guided Image Diffusion Models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

Anpei Chen, Zexiang Xu, Fuqiang Zhao, Xiaoshuai Zhang, Fanbo Xiang, Jingyi Yu, and Hao Su. 2021. Mvsnerf: Fast generalizable radiance field reconstruction from multi-view stereo. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 14124–14133.

Dave Zhenyu Chen, Yawar Siddiqui, Hsin-Ying Lee, Sergey Tulyakov, and Matthias Nießner. 2023c. Text2Tex: Text-driven Texture Synthesis via Diffusion Models. *arXiv preprint arXiv:2303.11396* (2023).

Rui Chen, Yongwei Chen, Ningxin Jiao, and Kui Jia. 2023a. Fantasia3D: Disentangling Geometry and Appearance for High-quality Text-to-3D Content Creation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 22246–22256.

Zhiqin Chen, Thomas Funkhouser, Peter Hedman, and Andrea Tagliasacchi. 2023b. MobileNeRF: Exploiting the Polygon Rasterization Pipeline for Efficient Neural Field Rendering on Mobile Architectures. In *The Conference on Computer Vision and Pattern Recognition (CVPR)*.

Seokjun Choi, Hoon-Gyu Chung, Yujin Jeon, Giljoo Nam, and Seung-Hawn Baek. 2025. A Real-world Display Inverse Rendering Dataset. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) 2025*.

Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. 2022. Objaverse: A Universe of Annotated 3D Objects. *arXiv preprint arXiv:2212.08051* (2022).

Thomas Deliot, Eric Heitz, and Laurent Belcour. 2024. Transforming a Non-Differentiable Rasterizer into a Differentiable One with Stochastic Gradient Estimation. *Proc. ACM Comput. Graph. Interact. Tech.* 7, 1, Article 3 (May 2024), 16 pages.

Valentin Deschaintre, Miika Aittala, Fredo Durand, George Drettakis, and Adrien Bousseau. 2018. Single-image SVBRDF capture with a rendering-aware deep network. *ACM Trans. Graph.* 37, 4, Article 128 (July 2018), 15 pages.

Ben Fei, Jingyi Xu, Rui Zhang, Qingyuan Zhou, Weidong Yang, and Ying He. 2024b. 3D Gaussian Splatting as New Era: A Survey. *IEEE Transactions on Visualization and Computer Graphics* (2024), 1–20.

Fan Fei, Jiajun Tang, Ping Tan, and Boxin Shi. 2024a. VMINer: Versatile Multi-view Inverse Rendering with Near-and Far-field Light Sources. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 11800–11809. https://doi.org/10.1109/CVPR52733.2024.01121

Michael Fischer and Tobias Ritschel. 2023. Plateau-Reduced Differentiable Path Tracing. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 4285–4294. https://doi.org/10.1109/CVPR52729.2023.00417

Michael Fischer and Tobias Ritschel. 2024. ZeroGrads: Learning Local Surrogates for Non-Differentiable Graphics. *ACM Trans. Graph.* 43, 4, Article 49 (July 2024), 15 pages. https://doi.org/10.1145/3658173

Peter Hedman, Pratul P. Srinivasan, Ben Mildenhall, Jonathan T. Barron, and Paul Debevec. 2021. Baking Neural Radiance Fields for Real-Time View Synthesis. *ICCV* (2021).

Wenzel Jakob, Sébastien Speierer, Nicolas Roussel, and Delio Vicini. 2022. Dr.Jit: A Just-In-Time Compiler for Differentiable Rendering. *Transactions on Graphics (Proceedings of SIGGRAPH)* 41, 4 (July 2022). https://doi.org/10.1145/3528223.3530099

Haian Jin, Isabella Liu, Peijia Xu, Xiaoshuai Zhang, Songfang Han, Sai Bi, Xiaowei Zhou, Zexiang Xu, and Hao Su. 2023. TensoIR: Tensorial Inverse Rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Alex Kendall and Yarin Gal. 2017. What uncertainties do we need in Bayesian deep learning for computer vision?. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (Long Beach, California, USA) *(NIPS'17)*. Curran Associates Inc., Red Hook, NY, USA, 5580–5590.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 2023. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Transactions on Graphics* 42, 4 (July 2023). https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/

Peter Kocsis, Vincent Sitzmann, and Matthias Niessner. 2024. Intrinsic Image Diffusion for Indoor Single-view Material Estimation. *Conference on Computer Vision and Pattern Recognition (CVPR)* (2024).

Zhengfei Kuang, Yunzhi Zhang, Hong-Xing Yu, Samir Agarwala, Shangzhe Wu, and Jiajun Wu. 2023. Stanford-ORB: a real-world 3d object inverse rendering benchmark. In *Proceedings of the 37th International Conference on Neural Information Processing Systems* (New Orleans, LA, USA) *(NIPS '23)*. Curran Associates Inc., Red Hook, NY, USA, Article 2033, 20 pages.

Samuli Laine, Janne Hellsten, Tero Karras, Yeongho Seol, Jaakko Lehtinen, and Timo Aila. 2020. Modular Primitives for High-Performance Differentiable Rendering. *ACM Transactions on Graphics* 39, 6 (2020).

Tzu-Mao Li, Miika Aittala, Frédo Durand, and Jaakko Lehtinen. 2018. Differentiable Monte Carlo ray tracing through edge sampling. *ACM Trans. Graph.* 37, 6, Article 222 (Dec. 2018), 11 pages.

Zhengqin Li, Mohammad Shafiei, Ravi Ramamoorthi, Kalyan Sunkavalli, and Manmohan Chandraker. 2020. Inverse rendering for complex indoor scenes: Shape, spatially-varying lighting and svbrdf from a single image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2475–2484.

Zhengqi Li and Noah Snavely. 2018. Learning Intrinsic Image Decomposition from Watching the World. In *Computer Vision and Pattern Recognition (CVPR)*.

Zhengqin Li, Dilin Wang, Ka Chen, Zhaoyang Lv, Thu Nguyen-Phuoc, Milim Lee, Jia-Bin Huang, Lei Xiao, Yufeng Zhu, Carl S. Marshall, Yuheng Ren, Richard Newcombe, and Zhao Dong. 2025. LIRM: Large Inverse Rendering Model for Progressive Reconstruction of Shape, Materials and View-dependent Radiance Fields. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 505–517. https://doi.org/10.1109/CVPR52734.2025.00056

Zhengqin Li, Ting-Wei Yu, Shen Sang, Sarah Wang, Meng Song, Yuhan Liu, Yu-Ying Yeh, Rui Zhu, Nitesh Gundavarapu, Jia Shi, Sai Bi, Hong-Xing Yu, Zexiang Xu, Kalyan Sunkavalli, Milos Hasan, Ravi Ramamoorthi, and Manmohan Chandraker. 2021. OpenRooms: An Open Framework for Photorealistic Indoor Scene Datasets . In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, Los Alamitos, CA, USA, 7186–7195.

Ruofan Liang, Zan Gojcic, Huan Ling, Jacob Munkberg, Jon Hasselgren, Chih-Hao Lin, Jun Gao, Alexander Keller, Nandita Vijaykumar, Sanja Fidler, and Zian Wang. 2025. DiffusionRenderer: Neural Inverse and Forward Rendering with Video Diffusion Models. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 26069–26080. https://doi.org/10.1109/CVPR52734.2025.02428

Zhihao Liang, Qi Zhang, Ying Feng, Ying Shan, and Kui Jia. 2024. GS-IR: 3D Gaussian Splatting for Inverse Rendering. (2024).

Chih-Hao Lin, Jia-Bin Huang, Zhengqin Li, Zhao Dong, Christian Richardt, Tuotuo Li, Michael Zollhöfer, Johannes Kopf, Shenlong Wang, and Changil Kim. 2025a. IRIS: Inverse Rendering of Indoor Scenes from Low Dynamic Range Images. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 465–474. https://doi.org/10.1109/CVPR52734.2025.00052

Chin-Yang Lin, Chung-Ho Wu, Chang-Han Yeh, Shih-Han Yen, Cheng Sun, and Yu-Lun Liu. 2025b. FrugalNeRF: Fast Convergence for Extreme Few-shot Novel View Synthesis without Learned Priors. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 11227–11238. https://doi.org/10.1109/CVPR52734.2025.01049

Isabella Liu, Linghao Chen, Ziyang Fu, Liwen Wu, Haian Jin, Zhong Li, Chin Ming Ryan Wong, Yi Xu, Ravi Ramamoorthi, Zexiang Xu, and Hao Su. 2023. OpenIllumination: a multi-illumination dataset for inverse rendering evaluation on real objects. In *Proceedings of the 37th International Conference on Neural Information Processing Systems* (New Orleans, LA, USA) *(NIPS '23)*. Curran Associates Inc., Red Hook, NY, USA, Article 1607, 12 pages.

Andrea Maggiordomo, Federico Ponchio, Paolo Cignoni, and Marco Tarini. 2020. Real-World Textured Things: A repository of textured models generated with modern photo-reconstruction tools. *Computer Aided Geometric Design* 83 (2020), 101943. https://doi.org/10.1016/j.cagd.2020.101943

Ricardo Martin-Brualla, Noha Radwan, Mehdi S. M. Sajjadi, Jonathan T. Barron, Alexey Dosovitskiy, and Daniel Duckworth. 2021. NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections. In *CVPR*.

Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. 2021. NeRF: representing scenes as neural radiance fields for view synthesis. *Commun. ACM* 65, 1 (Dec. 2021), 99–106. https://doi.org/10.1145/3503250

Jacob Munkberg, Jon Hasselgren, Tianchang Shen, Jun Gao, Wenzheng Chen, Alex Evans, Thomas Müller, and Sanja Fidler. 2022. Extracting Triangular 3D Models, Materials, and Lighting From Images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 8280–8290.

Julien Philip, Sébastien Morgenthaler, Michaël Gharbi, and George Drettakis. 2021. Free-viewpoint Indoor Neural Relighting from Multi-view Stereo. *ACM Transactions on Graphics* (2021). http://www-sop.inria.fr/reves/Basilic/2021/PMGD21

Elad Richardson, Gal Metzer, Yuval Alaluf, Raja Giryes, and Daniel Cohen-Or. 2023. TEXTure: Text-Guided Texturing of 3D Shapes. In *ACM SIGGRAPH 2023 Conference Proceedings* (Los Angeles, CA, USA) *(SIGGRAPH '23)*. Association for Computing Machinery, New York, NY, USA, Article 54, 11 pages.

Mike Roberts, Jason Ramapuram, Anurag Ranjan, Atulit Kumar, Miguel Angel Bautista, Nathan Paczan, Russ Webb, and Joshua M. Susskind. 2021. Hypersim: A Photorealistic Synthetic Dataset for Holistic Indoor Scene Understanding. In *International Conference on Computer Vision (ICCV) 2021*.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Bjorn Ommer. 2022. High-Resolution Image Synthesis with Latent Diffusion Models . In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, Los Alamitos, CA, USA, 10674–10685.

Pratul P. Srinivasan, Ben Mildenhall, Matthew Tancik, Jonathan T. Barron, Richard Tucker, and Noah Snavely. 2020. Lighthouse: Predicting Lighting Volumes for Spatially-Coherent Illumination. *CVPR* (2020).

Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul Srinivasan, Howard Zhou, Jonathan T. Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas Funkhouser. 2021. IBRNet: Learning Multi-View Image-Based Rendering. In *CVPR*.

Jinbo Wu, Xing Liu, Chenming Wu, Xiaobo Gao, Jialun Liu, Xinqi Liu, Chen Zhao, Haocheng Feng, Errui Ding, and Jingdong Wang. 2024. TexRO: Generating Delicate Textures of 3D Models by Recursive Optimization. arXiv:2403.15009 [cs.CV] https://arxiv.org/abs/2403.15009

Liwen Wu, Jae Yong Lee, Anand Bhattad, Yuxiong Wang, and David Forsyth. 2021. DIVeR: Real-time and Accurate Neural Radiance Fields with Deterministic Integration for Volume Rendering. arXiv:2111.10427 [cs.CV]

Sean Wu, Shamik Basu, Tim Brödermann, Luc Van Gool, and Christos Sakaridis. 2025. PBR-NeRF: Inverse Rendering with Physics-Based Neural Fields. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 10974–10984. https://doi.org/10.1109/CVPR52734.2025.01025

Ahmet Burak Yildirim, Mustafa Utku Aydogdu, Duygu Ceylan, and Aysegul Dundar. 2025. MD-ProjTex: Texturing 3D Shapes with Multi-Diffusion Projection. arXiv:2504.02762 [cs.CV] https://arxiv.org/abs/2504.02762

Bohan Yu, Siqi Yang, Xuanning Cui, Siyan Dong, Baoquan Chen, and Boxin Shi. 2023. MILO: Multi-Bounce Inverse Rendering for Indoor Scene With Light-Emitting Objects. *IEEE Trans. Pattern Anal. Mach. Intell.* 45, 8 (Aug. 2023), 10129–10142.

Xianfang Zeng, Xin Chen, Zhongqi Qi, Wen Liu, Zibo Zhao, Zhibin Wang, Bin Fu, Yong Liu, and Gang Yu. 2024a. Paint3d: Paint anything 3d with lighting-less texture diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 4252–4262.

Yifei Zeng, Yajie Bao, Jiachen Qian, Shuang Wu, Youtian Lin, Hao Zhu, Buyu Li, Feihu Zhang, Xun Cao, and Yao Yao. 2025. TEXTRIX: Latent Attribute Grid for Native Texture Generation and Beyond. arXiv:2512.02993 [cs.CV] https://arxiv.org/abs/2512.02993

Zheng Zeng, Valentin Deschaintre, Iliyan Georgiev, Yannick Hold-Geoffroy, Yiwei Hu, Fujun Luan, Ling-Qi Yan, and Miloš Hašan. 2024b. RGB↔X: 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. https://doi.org/10.1145/3641519.3657445

Hongkun Zhang, Zherong Pan, Congyi Zhang, Lifeng Zhu, and Xifeng Gao. 2024b. TexPainter: Generative Mesh Texturing with Multi-view Consistency. In *ACM SIGGRAPH 2024 Conference Papers* (Denver, CO, USA) *(SIGGRAPH '24)*. Association for Computing Machinery, New York, NY, USA, Article 6, 11 pages. https://doi.org/10.1145/3641519.3657494

Yuqing Zhang, Yuan Liu, Zhiyu Xie, Lei Yang, Zhongyuan Liu, Mengzhou Yang, Runze Zhang, Qilong Kou, Cheng Lin, Wenping Wang, and Xiaogang Jin. 2024a. DreamMat: High-quality PBR Material Generation with Geometry- and Light-aware Diffusion Models. *ACM Trans. Graph.* 43, 4, Article 39 (jul 2024), 18 pages. https://doi.org/10.1145/3658170

Qian-Yi Zhou and Vladlen Koltun. 2014. Color map optimization for 3D reconstruction with consumer depth cameras. *ACM Trans. Graph.* 33, 4, Article 155 (July 2014), 10 pages. https://doi.org/10.1145/2601097.2601134

Rui Zhu, Zhengqin Li, Janarbek Matai, Fatih Porikli, and Manmohan Chandraker. 2022. IRISformer: Dense Vision Transformers for Single-Image Inverse Rendering in Indoor Scenes . In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, Los Alamitos, CA, USA, 2812–2821.

Fig. 10. Texture optimization results on diverse objects. For each example, we show two rendered views and the optimized PBR texture maps (albedo, roughness, metallic). Our method successfully handles a wide range of materials demonstrating strong generalization across object categories.

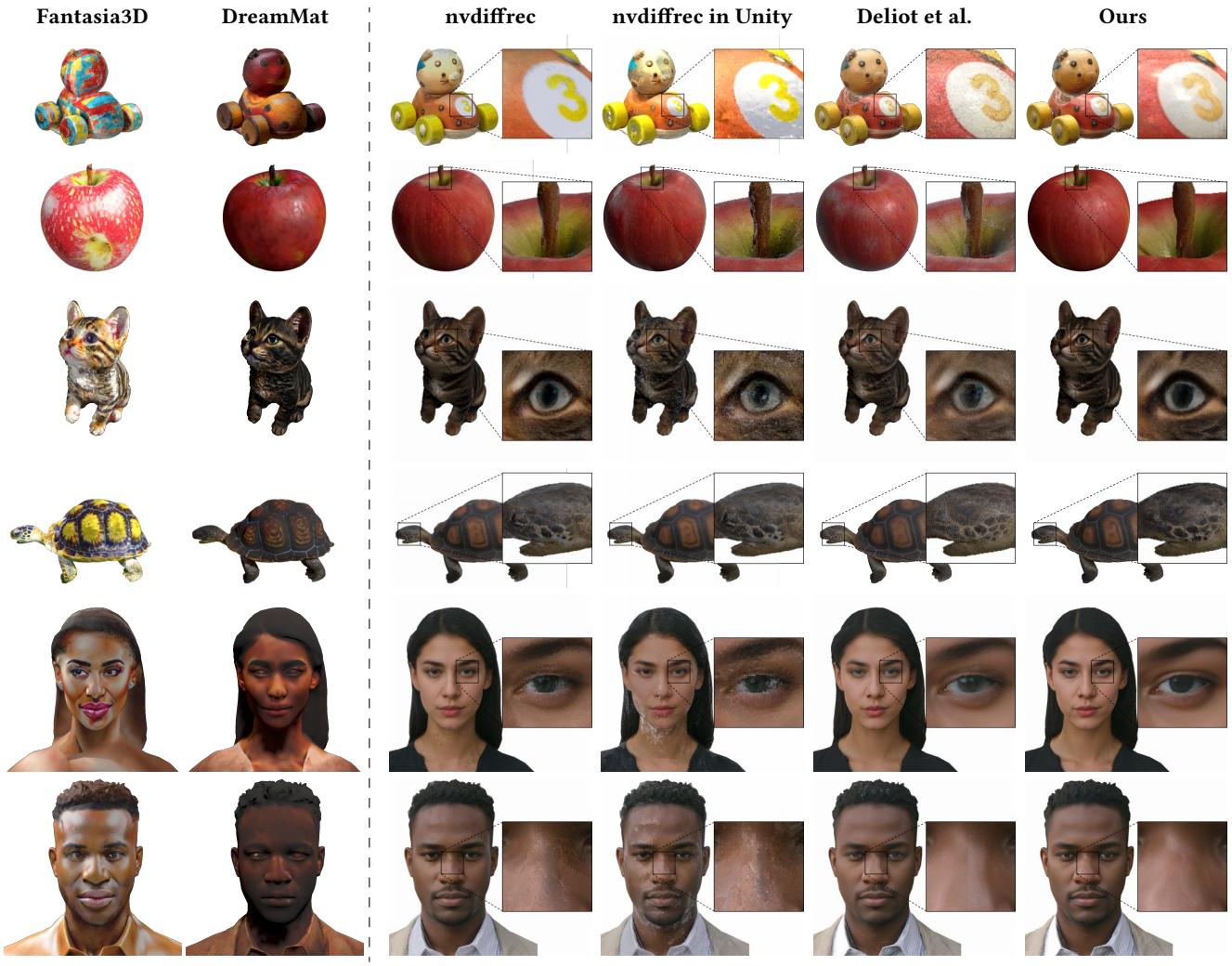


Fig. 11. Comparison against text-driven generation approaches (Fantasia3D [Chen et al. 2023a], DreamMat [Zhang et al. 2024a]) and inverse rendering methods (nvdiffrec [Munkberg et al. 2022], Deliot et al. [Deliot et al. 2024]). **Fantasia3D** and **DreamMat** produce coherent textures but lack photorealism, especially for human faces. **nvdiffrec** shows results rendered in its native framework, while **nvdiffrec in Unity** denotes the same assets imported into Unity, illustrating the rendering gap. **nvdiffrec** achieves high quality in its native renderer but exhibits noise and the quality degrades when imported into Unity. **Deliot et al.** optimizes directly in Unity but suffers from blurring and ghosting artifacts due to target inconsistencies. Our method produces clean, sharp textures by handling inconsistencies through uncertainty estimation and deformation grids.