Pygmalion Effect in Vision: Image-to-Clay Translation for Reflective Geometry Reconstruction

Gayoung Lee[1,2] **Junho Kim**[1] **Jin-Hwa Kim**[1,3†] **Junmo Kim**[2†]
{gayoung.lee, jhkim.ai, j1nhwa.kim}@navercorp.com, junmo.kim@kaist.ac.kr

[1]NAVER AI Lab [2]KAIST [3]SNU AIIS

Abstract

Understanding reflection remains a long-standing challenge in 3D reconstruction due to the entanglement of appearance and geometry under view-dependent reflections. In this work, we present the Pygmalion Effect in Vision, a novel framework that metaphorically "sculpts" reflective objects into clay-like forms through image-to-clay translation. We then introduce a dual-branch design in which a BRDF-based reflective branch and a clay-guided branch share the same Gaussian geometry but operate through independent rendering paths. The clay-guided branch, supervised by the synthesized clay images, provides reflection-free geometric guidance that complements the photometric supervision of the BRDF branch. Experiments on both synthetic and real datasets show consistent improvements in both geometry and photometric metrics. Beyond technical gains, our framework reveals that seeing by unshining, translating radiance into neutrality, can serve as a powerful inductive bias for reflective object geometry learning.

1 Introduction

Reconstructing three-dimensional structures from images has long been a fundamental problem in computer vision. The emergence of Neural Radiance Fields (NeRF) [1, 2, 3] marked a major breakthrough in continuous scene representation and photorealistic novel-view synthesis. Subsequent studies [4, 5, 6] bridge neural rendering and mesh-based modeling by extracting explicit geometry. More recently, 3D Gaussian Splatting (3DGS) [7] introduced a point-based differentiable representation that enables real-time, high-fidelity rendering without neural queries at inference.

Despite these advances, reconstructing reflective objects remains a significant challenge. While NeRF- and 3DGS-based models can capture certain non-Lambertian cues, they still implicitly rely on view-invariant radiance consistency, which reflective surfaces violate since their appearance changes with view-dependent environmental reflections. To address this, several inverse-rendering approaches attempt to recover surface reflectance and environment illumination by explicitly modeling specular reflectance parameters within a BRDF formulation [8, 9, 10]. However, jointly optimizing geometry and material properties is challenging, as appearance and shape are entangled through reflection, making it difficult to attribute observed color variations to either geometry or reflectance. A parallel line of work [11, 12] introduces auxiliary normal supervision from off-the-shelf estimators to constrain geometry directly. Yet such signals are tightly coupled to the reflective rendering path: even small normal errors are amplified when computing reflection directions for environment-map lookup. Moreover, accurate per-pixel normal estimation on reflective surfaces remains an open problem.

In contrast to prior works that build increasingly complex models for reflective objects, we reformulate the problem as reconstruction by converting reflective appearances into non-reflective ones using image-to-image translation models. For this conversion, we adopt white clay as the target material.

†Corresponding authors.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).



(a) Ground Truths (b) Clay Images (c) Ours (d) MaterialRefGS

Figure 1: By translating reflective images (a) into clay-like renderings (b) for geometric guidance, our method produces sharper, more accurate reflective-surface geometry (c) than a strong baseline (d).

Clay renders, widely used in pipelines such as Blender [13], effectively convey geometric structure without interference from material appearance. This process can be viewed as an instance of the Pygmalion effect in vision (for our analogy, see Section A.1), where the model learns to reshape the visual world according to its own learned expectations, turning reflective complexity into a self-consistent, interpretable form. We find that clay-like appearances can be reliably synthesized by modern generative models, due to the abundance of similar matte-white objects in real-world images. To ensure cross-view consistency of the translated clay images, we adapt the pretrained video diffusion model LTX-2 [14] with lightweight LoRA [15] fine-tuning, reinterpreting its temporal coherence as a view-consistency prior.

To leverage these clay images for geometry optimization, we introduce a dual-branch reflective Gaussian splatting design that can be readily integrated into existing reflective splatting frameworks. Our clay-guided branch shares the Gaussian geometry with the BRDF-based branch but uses an auxiliary clay color parameter, decoupling reflection-free geometric guidance from specular evaluation. We adopt MaterialRefGS [12] as our primary backbone and show that adding clay supervision substantially improves geometry. Notably, our clay supervision is complementary to the monocular normal prior estimated by an off-the-shelf model, operating through an independent rendering path. Moreover, we will show that it can stand alone as a sufficient source of geometric guidance, even without any normal prior. Experimental results on multiple datasets demonstrate superior mesh quality compared to existing approaches, as illustrated in Fig. 1.

In summary, our contributions are summarized as follows:

- A novel framework that translates reflective images into clay-like renderings as a reflection-free supervision signal, providing geometric guidance.
- A multi-view-consistent clay translation model built on LTX-2 [14], a pretrained video diffusion backbone. We reinterpret its temporal axis as a view axis, enabling cross-view coherence with only lightweight LoRA adaptation.
- A dual-branch reflective Gaussian splatting design that augments existing BRDF-based pipelines with a clay-guided branch sharing the same geometry, providing reflection-free geometric supervision through an independent rendering path.
- Experimental validation demonstrates clear and consistent gains in mesh accuracy and reconstruction stability across multiple reflective object datasets.

2 Related work

Novel-view synthesis with implicit geometry. NeRF [1] introduced volumetric radiance fields, inspiring extensions for anti-aliasing [2], training efficiency [3], and high-fidelity surface reconstruction [16, 17]. In parallel, surface-based approaches [4, 18, 19] explicitly model signed-distance fields for improved geometry. Recently, point-based representations such as 3D Gaussian Splatting [7] and 2DGS [20] have enabled real-time rendering with improved geometric consistency.

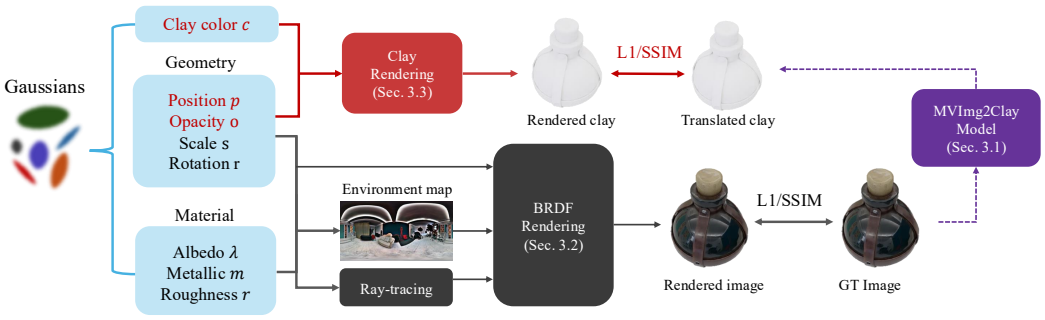


Figure 2: Overview of our dual-branch pipeline. A BRDF-based reflective branch and a clay-guided branch share the same Gaussian geometry. Clay-like images provide reflection-free supervision during early training, stabilizing geometry and improving surface normals.

Reflective reconstruction. Reconstructing reflective objects is challenging because appearance and geometry are entangled under view-dependent color variations. Early attempts such as TensoSDF [21] and NeRO [22] combine volumetric radiance fields with explicit reflectance modeling to jointly learn geometry and BRDF parameters. Gaussian-based approaches address the same issue through differentiable point representations. GShader [23] integrates shading functions into Gaussian splatting for reflective rendering, while GS-IR [24] formulates inverse rendering within the Gaussian domain to estimate reflectance and lighting. R3DG [25] introduces a ray-tracing formulation to better handle high-frequency specular highlights. Further, Ref-GS [26] and RGS [10] focus on disentangling view-dependent radiance for more accurate reflective reconstruction. GS-2DGS [11] and MaterialRefGS [12] enhance geometric consistency by supervising reflective Gaussians with external models. Recent works [12, 27, 28] also utilize ray-tracing to more accurately model indirect illumination.

S2Dnet [29] previously explored the idea of translating specular images into diffuse counterparts for geometry reconstruction, using a multi-view-aware GAN. However, S2Dnet relies on local correspondence-based consistency and uses diffuse translation as preprocessing for classical MVS, without BRDF learning and with limited mesh reconstruction accuracy. In contrast, our approach leverages a video diffusion prior with an anchor mechanism for cross-view consistency and integrates clay supervision into reflective Gaussian splatting optimization to support both geometry reconstruction and BRDF learning. An empirical comparison is provided in Section A.6.3.

Image and video editing. Image-to-image translation learns mappings between visual domains for controllable appearance transformation using paired adversarial learning [30], unpaired cycle-consistency [31], disentangled multimodal [32], contrastive [33] formulations, and unified spatial-semantic control [34]. Extending these ideas to the temporal domain, instruction-based video editors [35, 36] and stylization methods [37, 38, 39] enable coherent edits across frames. Unlike prior works that primarily focus on temporal coherence, our goal is multi-view consistent material translation: we reinterpret the temporal coherence prior of a pretrained video diffusion model as a view-consistency prior, generating clay-like views that provide reflection-free supervision for accurate and efficient geometry optimization.

3 Method

Our overall framework is illustrated in Fig. 2. We first develop a multi-view image-to-clay translation network in Section 3.1, trained on the dataset described in Section A.2. We then introduce our clay-guided reflective Gaussian splatting method in Section 3.2 and Section 3.3.

3.1 Multi-view image-to-clay translation

To translate reflective images into diffuse clay-like renderings that can serve as multi-view geometric supervision, we require a translation model that not only suppresses specular cues but also produces outputs consistent across viewpoints. Per-image translation models [40, 34] can produce high-quality clay translations on individual images, but offer no mechanism to enforce coherence across views.

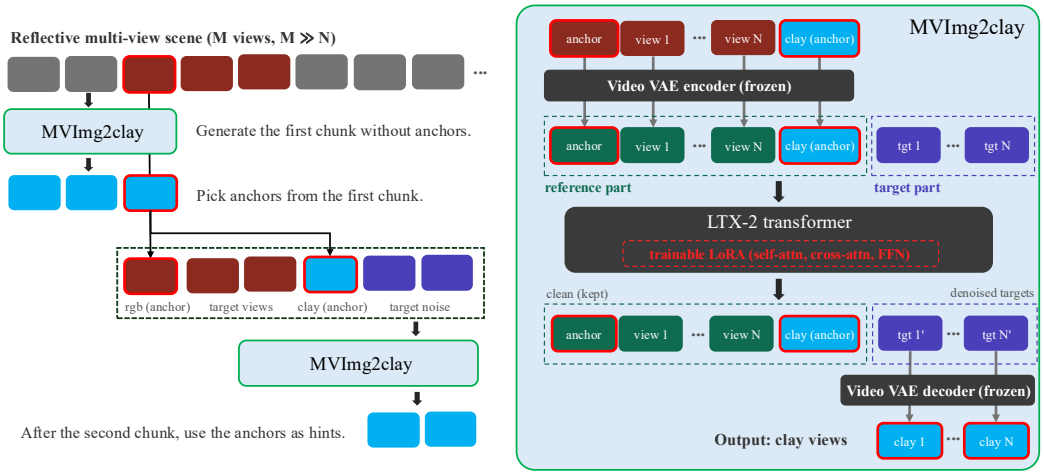


Figure 3: Pipeline of our LTX-2-based multi-view image-to-clay translation model. We arrange multi-view images along the temporal axis of the video diffusion model, repurposing temporal coherence as cross-view consistency. For scenes split into multiple chunks, we use anchor views and their clay translations from the first chunk as conditioning to maintain consistency across chunks.

We address this by repurposing a pretrained video diffusion model, LTX-2 [14], which natively enforces coherence across video frames through its temporal attention. Our key insight is that the temporal axis of a video model can be reinterpreted as a view axis. While multi-view images are not a continuous video sequence in the strict sense, we find that the inter-frame coherence prior learned by the video model transfers naturally to multi-view inputs with only lightweight fine-tuning, producing translations that remain consistent across views. We provide a brief overview of LTX-2 and In-Context LoRA in Section A.3. An overview of our multi-view image-to-clay translation pipeline is shown in Fig. 3.

Multi-view sequence construction. Given N multi-view images of a reflective object, we encode each view independently using LTX-2's video VAE encoder, producing N single-frame latent tokens. We then concatenate these latents along the temporal axis and use them as conditioning.

The full token sequence consists of two parts:

$$\underbrace{[\boldsymbol{Z}_1^{\text{cond}}, \ldots, \boldsymbol{Z}_N^{\text{cond}}]}_{\text{conditioning view (clean)}} \;\Big|\; \underbrace{[\boldsymbol{Z}_1^{\text{tgt}}, \ldots, \boldsymbol{Z}_N^{\text{tgt}}]}_{\text{target (noisy)}}, \tag{1}$$

where the conditioning part contains the input reflective views as clean condition tokens, and the target part contains the corresponding clay views as the diffusion target. Following IC-LoRA [41], we assign per-token timesteps to distinguish conditioning tokens within a unified sequence:

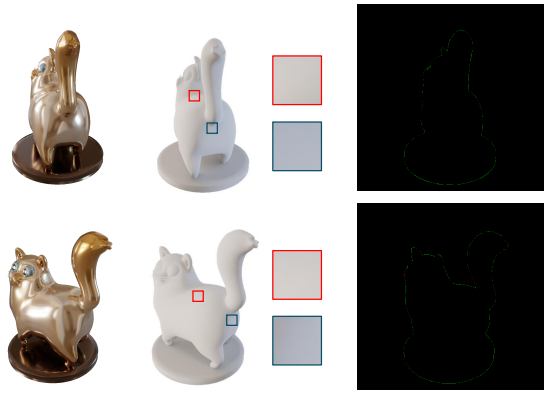
$$t_i = \begin{cases} 0 & \text{if } m_i = 1 \\ \sigma & \text{if } m_i = 0 \end{cases}, \tag{2}$$

where m_i denotes conditioning mask and σ is the noise level sampled from a shifted logit-normal distribution. This per-token timestep design lets the transformer treat reference tokens as fully observed while denoising target tokens, all within a single forward pass.

Within the pseudo-video sequence, we order the views randomly rather than by camera angle proximity. We empirically found that random ordering generalizes better across datasets with diverse camera trajectories.

Training data and objective. We train the translation model on a dataset of paired multi-view renderings. Each pair shares the same camera pose, HDRI lighting, and geometry, but differs in material. The full rendering details are provided in Section A.2.

We train the lightweight LoRA [15] using a standard flow-matching velocity prediction objective. Given a clean target latent $\boldsymbol{Z}^{\text{tgt}}$ and a noise sample $\boldsymbol{\epsilon}$, the noisy latent is $\boldsymbol{Z}_\sigma = (1-\sigma)\boldsymbol{Z}^{\text{tgt}} + \sigma\boldsymbol{\epsilon}$, and



(a) RGB images (b) Clay images (c) Silhouette difference

Table 1: **Quantitative analysis of clay image consistency.** Top: foreground/background MSE between RGB and clay images, and IoU between the clay and the GT mask. Bottom: cross-view color variance at the same 3D point compared with the overall variance.

MSE_{fg}	MSE_{bg}	Mask IoU
0.1407	0.0001	0.9915

Type	Cross var.	Global var.	Ratio $R\downarrow$
RGB	0.0575	0.1318	0.4109
Clay	0.0163	0.0436	0.3661

Figure 4: **Clay images preserve object silhouettes and view-consistent appearance.** Visualization of clay images alongside GT RGB on Cat (from Glossy Synthetic). The clay translation removes view-dependent specular cues while preserving object silhouettes.

the model predicts the velocity $\mathbf{v} = \boldsymbol{\epsilon} - \boldsymbol{Z}^{\text{tgt}}$. We apply the loss only on noisy target tokens, masking out reference tokens and any clean anchor tokens:

$$\mathcal{L}_{\text{flow}} = \mathbb{E}_{\sigma,\boldsymbol{\epsilon}} \left[\sum_{i:m_i=0} \|\mathbf{v}_\theta(\boldsymbol{Z}_\sigma; t)_i - (\boldsymbol{\epsilon}_i - \boldsymbol{Z}_i^{\text{tgt}})\|_2^2 \right], \tag{3}$$

where $\mathbf{v}_\theta$ is the LTX-2 transformer with our LoRA adaptation.

Anchor mechanism and inference for arbitrary view counts. Training uses fixed-length sequences ($N = 24$), but real scenes often contain hundreds of views. To bridge this gap, we introduce an anchor mechanism trained jointly with the denoising objective. During training, with probability 0.5 we replace $n_{\text{anchor}} = 4$ randomly selected target slots with clean ground-truth clay latents, while the remaining targets are denoised as usual; otherwise, all targets remain noisy. This yields two modes within one model: anchor-free generation, and inpainting-like generation conditioned on a few clean anchor translations. At inference, for a scene with $M \gg N$ views, we partition the views into chunks of size N. The first chunk is generated in the anchor-free mode; n_{anchor} of its input views (*anchors*) and their translated outputs (*anchor-clay* images) are then injected into every subsequent chunk, maintaining coherence across the full capture. A fixed text prompt describing the clay material is provided in both modes.

Silhouette preservation and view consistency. Fig. 4 and Table 1 show that our generated clay images effectively suppress view-dependent color variations while preserving object silhouettes. Fig. 4 (b) shows that the same surface point across different views is mapped to consistent colors. Fig. 4 (c) shows the silhouette mismatch between GT and clay-derived masks (red: GT-only, green: clay-only). Discrepancies appear only along thin boundaries, consistent with the near-zero background MSE and near-unity mask IoU in Table 1 (top). The bottom of Table 1 reports the color variance at corresponding 3D points obtained via GT depth. Variances and ratios are computed per scene and then averaged. Although clay images appear visually more uniform than the original images, the cross-view variance at a given surface point is substantially smaller than the global variance, yielding a lower variance ratio than the original images. This indicates that the clay translation produces view-independent appearances, providing an ideal supervision signal for geometry learning. We further explain the metric and analyze this in Section 4.5. Additional examples of the generated clay images are provided in Section A.6.1.

3.2 Reflective Gaussian splatting baseline

We build upon MaterialRefGS [12], a deferred shading-based reflective Gaussian Splatting framework that extends 2D Gaussian Splatting (2DGS) [20] with physically-based material modeling. Following recent BRDF-based reflective splatting approaches [10, 26, 11], each 2D Gaussian primitive is

augmented with material attributes such as albedo λ, metallic m, roughness r, and surface normal $\mathbf{n}$, which are alpha-blended into screen-space material maps:

$$\mathbf{B}(\boldsymbol{x}) = \sum_{i=1}^{N} \boldsymbol{b}_i \, \alpha_i \, \hat{\mathcal{G}}_i(\mathbf{u}(\boldsymbol{x})) \prod_{j=1}^{i-1} \Big(1 - \alpha_j \, \hat{\mathcal{G}}_j(\mathbf{u}(\boldsymbol{x}))\Big), \tag{4}$$

where $\boldsymbol{b}_i = [\lambda_i, m_i, r_i, \mathbf{n}_i]$ are the per-Gaussian material parameters, and $\hat{\mathcal{G}}_i$, α_i denote the 2D Gaussian and opacity. The rasterized material maps are then evaluated under a physically-based BRDF, where the specular term is approximated via the split-sum formulation [42], separating the integral into a view-dependent BRDF lookup and a prefiltered environment-map evaluation. A complete formulation of 2DGS rasterization, BRDF, and split-sum integration is provided in Section A.4.

Beyond the standard BRDF framework, MaterialRefGS introduces a multi-view material consistency loss $\mathcal{L}_{mv}$ that aligns rasterized material maps across nearby views via homography warping. Also, a monocular normal prior $\mathcal{L}_n$ from a pretrained estimator [43] provides auxiliary geometric supervision. Indirect illumination is additionally handled through differentiable Gaussian ray tracing [44]. The total baseline loss is

$$\mathcal{L}_{\text{base}} = \mathcal{L}_{rgb} + \lambda_{n\text{-}d}\mathcal{L}_{n\text{-}d} + \lambda_n \mathcal{L}_n + \lambda_{\text{mv}}\mathcal{L}_{\text{mv}} + \lambda_{\text{ref}}\mathcal{L}_{\text{ref}}, \tag{5}$$

combining photometric, depth-normal consistency, normal prior, multi-view material, and reflection-strength regularization terms.

3.3 Clay-guided reflective Gaussian splatting

In addition to the BRDF-based rendering branch (*BRDF branch*) of MaterialRefGS (Section 3.2), we introduce an auxiliary *clay branch* that renders view-independent, clay-like images supervised by our multi-view image-to-clay translation outputs (Section 3.1). The two branches share the same underlying Gaussian geometry but operate on independent appearance parameters, providing geometric supervision via a rendering path that is fully decoupled from the specular color evaluation.

To be specific, we introduce an additional clay color parameter $\hat{\boldsymbol{c}}_i$ for each Gaussian i, alongside the existing BRDF parameters $[\lambda_i, m_i, r_i, \mathbf{n}_i]$. Both BRDF and clay branches share the same geometric attributes: position $\boldsymbol{p}_i$, tangential direction $\boldsymbol{t}_i$, scaling $\boldsymbol{s}_i$, opacity α_i, and surface normal $\mathbf{n}_i$. So, the geometric supervision from the clay branch directly improves the shared geometry. The clay color is rendered through the same alpha blending used for material maps:

$$\hat{\boldsymbol{c}}(\boldsymbol{x}) = \sum_{i=1}^{N} \hat{\boldsymbol{c}}_i \, \alpha_i \, \hat{\mathcal{G}}_i(\boldsymbol{u}(\boldsymbol{x})) \prod_{j=1}^{i-1} \Big(1 - \alpha_j \, \hat{\mathcal{G}}_j(\boldsymbol{u}(\boldsymbol{x}))\Big). \tag{6}$$

Unlike the specular branch, $\hat{\boldsymbol{c}}(\boldsymbol{x})$ is view-independent by design as the clay surfaces exhibit only diffuse shading variations. The clay branch is supervised by the multi-view-consistent clay images $\mathbf{I}_{\text{clay}}$ generated by our translation model, using a weighted combination of $\mathcal{L}_1$ and SSIM losses:

$$\mathcal{L}_{\text{clay}} = (1 - \lambda_{\text{dssim}}) \, \|\hat{\mathbf{I}}_{\text{clay}} - \mathbf{I}_{\text{clay}}\|_1 + \lambda_{\text{dssim}} \big(1 - \text{SSIM}(\hat{\mathbf{I}}_{\text{clay}}, \mathbf{I}_{\text{clay}})\big), \tag{7}$$

with $\lambda_{\text{dssim}} = 0.2$, following the standard photometric weighting in 2DGS-based methods.

Selective gradient routing. A naive joint optimization can produce conflicting gradients on shared parameters: the RGB loss ties geometry to view-dependent specular evaluation, while the clay loss prefers geometry that explains diffuse shading. We resolve this conflict by selectively routing gradients. Rotation $\boldsymbol{t}_i$ and scale $\boldsymbol{s}_i$ receive both gradients at full strength, since surface normals (derived from rotation) determine reflection directions and must remain refinable by the BRDF branch. Position $\boldsymbol{p}_i$ and opacity α_i, in contrast, are optimized primarily by $\mathcal{L}_{\text{clay}}$ with only a weak residual gradient from $\mathcal{L}_{\text{rgb}}$:

$$\tilde{\nabla}_\theta = \nabla_\theta \mathcal{L}_{\text{clay}} + \beta \, \nabla_\theta \mathcal{L}_{\text{rgb}}, \quad \theta \in \{\boldsymbol{p}_i, \alpha_i\}, \quad \beta = 0.1, \tag{8}$$

implemented via a stop-gradient with a residual scale on the RGB path.

Table 2: Chamfer-$\mathcal{L}_1$ ($\downarrow$) distances of 3D reconstruction results on the Glossy Synthetic dataset. The intensity of the red color signifies a better result.

Method	Angel	Bell	Cat	Horse	Luyu	Potion	Tbell	Teapot	Average
GShader [23]	0.0060	0.0078	0.0175	0.0072	0.0101	0.0382	0.0308	0.0178	0.0169
GS-IR [24]	0.0110	0.1097	0.0566	0.0149	0.0224	0.0593	0.0989	0.0693	0.0553
R3DG [25]	0.0090	0.0403	0.0326	0.0117	0.0151	0.0380	0.0472	0.0488	0.0303
Ref-GS [26]	0.0049	0.0084	0.0170	0.0060	0.0098	0.0090	0.0064	0.0084	0.0087
GS-2DGS [11]	0.0061	0.0037	0.0074	0.0050	0.0076	0.0088	0.0078	0.0083	0.0068
RGS [10]	0.0050	0.0071	0.0161	0.0069	0.0093	0.0081	0.0065	0.0090	0.0085
MaterialRefGS [12]	0.0038	0.0087	0.0130	0.0047	0.0061	0.0067	0.0049	0.0059	0.0067
Ours	0.0035	0.0069	0.0057	0.0047	0.0057	0.0064	0.0046	0.0057	0.0054

The total training loss combines the MaterialRefGS baseline loss $\mathcal{L}_{\text{base}}$ (Section 3.2) with our clay supervision as follows:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{base}} + \lambda_{\text{clay}} \mathcal{L}_{\text{clay}}. \tag{9}$$

We use $\lambda_{\text{clay}} = 1.0$ in all experiments. After the initial phase in which geometry is learned through clay loss with gradient routing, we disable the clay branch entirely and continue training with the RGB loss only for synthetic data. For real data, we instead keep both losses active, jointly optimizing with RGB and clay losses without further gradient routing. This is because reflections in real scenes are more complex and harder to disentangle, making continued reflection-free supervision beneficial.

4 Experiments

In this section, we compare our method with existing approaches both quantitatively and qualitatively to demonstrate its effectiveness. Our method shows notably superior performance in mesh quality, confirming the effectiveness of utilizing clay images for geometry learning.

4.1 Implementation details

For our multi-view image-to-clay translation model, we adapt LTX-2 [14] with LoRA [15] modules of rank 32 and train for $50{,}000$ iterations on $8\times$ NVIDIA H100 GPUs with a batch size of 1 per GPU. Each sample consists of 24 multi-view images of a single object. To enable the anchor-based inference, with probability 0.5 we expose four randomly selected target views as clean anchors during training. Our reconstruction model is built on the official codebase of MaterialRefGS [12]. Following the baseline, we estimate monocular normal maps using Metric3Dv2 [43] for the normal prior loss. Additional implementation details are provided in Section A.5.

4.2 Datasets and metrics

We use multiple datasets widely adopted for reflective object reconstruction. For synthetic evaluation, we use Shiny Blender [45] and Glossy Synthetic [22], both of which render non-Lambertian objects under environment maps. For real-world evaluation, we use Ref-Real [45], Glossy-Real [22], and NeRF-casting [27], which capture diverse reflective objects in real scenes. Ground-truth meshes are available for the Glossy Synthetic dataset, and ground-truth surface normals are additionally provided for Shiny Blender. We assess mesh quality using the Chamfer distance computed with the official evaluation code, and surface normal accuracy using the mean angular error (MAE). The photometric quality of novel-view synthesis is also evaluated by comparing rendered results with ground-truth images using PSNR, SSIM, and LPIPS.

4.3 Baselines

We compare our method with various state-of-the-art Gaussian-based approaches for reflective object reconstruction, including 2DGS [20], GShader [23], R3DG [25], 3DGS-DR [46], Ref-GS [26], RGS [10], and MaterialRefGS [12]. We use the official implementation of the baselines. Note that the reflection prior in the MaterialRefGS code is not yet available, so this component is excluded from our experiments. This may cause a discrepancy between the results reported in their paper and our reproduced results. Among these, MaterialRefGS serves as the underlying reflective Gaussian

Table 3: Generalization across backbones and the effect of normal prior. We report Chamfer-$\mathcal{L}_1$ (↓) distances on the Glossy Synthetic dataset. *"+ Clay"* integrates our clay-guided branch on top of each backbone. *"- Normal"* further removes the monocular normal prior to test whether clay supervision alone suffices. The intensity of the red color signifies a better result.

Method	**Angel**	**Bell**	**Cat**	**Horse**	**Luyu**	**Potion**	**Tbell**	**Teapot**	**Average**
RGS [10]	0.0050	0.0071	0.0161	0.0069	0.0093	0.0081	0.0065	0.0090	0.0085
RGS + *Clay*	0.0032	0.0067	0.0142	0.0042	0.0066	0.0077	0.0061	0.0110	0.0074
MaterialRefGS [12]	0.0038	0.0087	0.0130	0.0047	0.0061	0.0067	0.0049	0.0059	0.0067
MaterialRefGS + *Clay*	0.0035	0.0069	0.0057	0.0047	0.0057	0.0064	0.0046	0.0057	0.0054
MaterialRefGS + *Clay* - *Normal*	0.0035	0.0047	0.0087	0.0044	0.0063	0.0064	0.0034	0.0060	0.0054

Table 4: **Average image quality on five benchmark datasets.** For each dataset, we report the average PSNR / SSIM / LPIPS over all scenes. Per-scene scores are reported in Table A2 and Table A3. The intensity of the red color signifies a better result.

Datasets	**Shiny Blender** [45]			**Glossy Synthetic** [22]			**Ref-Real** [45]			**Glossy-Real** [22]			**NeRF-casting** [27]		
Metric	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
2DGS	29.58	0.946	0.084	26.07	0.918	0.088	24.15	0.661	0.292	21.20	0.804	0.243	24.66	0.822	0.225
Ref-GS	34.80	0.973	0.056	30.16	0.956	0.055	24.44	0.682	0.225	22.33	0.814	0.228	28.09	0.855	0.185
RGS	35.04	0.973	0.056	30.37	0.958	0.049	24.61	0.685	0.252	22.00	0.802	0.254	28.35	0.849	0.211
MaterialRefGS	34.68	0.975	0.054	30.60	0.960	0.048	24.73	0.698	0.183	22.65	0.818	0.186	28.07	0.845	0.148
Ours	35.12	0.976	0.053	30.68	0.961	0.047	24.74	0.699	0.183	23.04	0.824	0.184	28.69	0.853	0.137

splatting backbone of our method (Section 3.2), upon which we add the proposed clay-guided supervision.

4.4 Comparisons

We first evaluate the geometric quality of reconstructed meshes, the primary focus of our method. Table 2 reports Chamfer-$\mathcal{L}_1$ distances on Glossy Synthetic, where our method achieves the lowest average distance, outperforming MaterialRefGS. On Shiny Blender (Table A4), our method also achieves the lowest mean angular error.

In Table 3, we further investigate whether our clay-guided branch enhances geometry independent of the backbone, and the role of the external normal prior. Adding our clay branch to RGS [10] reduces the Chamfer distance from 0.0085 to 0.0074, with consistent improvements across most scenes. Removing the normal prior from MaterialRefGS while keeping the clay branch (+ *Clay* − *Normal*) degrades some scenes (*e.g.*, Cat) but improves others (*e.g.*, Bell), where highly curved surfaces lead to noisy normal predictions. Unlike normal supervision, our clay branch operates through an independent rendering path that does not affect the environment map or BRDF parameters, making it more robust to such noise, and supporting its role as both a complementary signal and a standalone source of geometric guidance. Moreover, clay-like objects stay within the pretrained RGB domain of the video backbone, unlike normal or depth maps that require learning a new output modality.

Beyond geometric metrics, we report photometric metrics on synthetic and real-world datasets in Table 4. Per-scene scores are provided in Tables A2 and A3 in the appendix. For older baselines, we report the original numbers from their papers, while results on Glossy-Real and NeRF-casting are obtained by re-running the most recent reflective splatting methods. On average, our method outperforms the baseline on both synthetic and real datasets.

Fig. 5 shows qualitative comparisons. In the upper row, the photographer's reflection is a view-dependent artifact that should not be baked into either the environment map or the diffuse color. However, RGS and Ref-GS visibly absorb this *transient* object into the diffuse component, indicating that these methods overfit diffuse color to explain view-dependent appearance. On the car window (bottom), our method clearly reproduces the reflection of the overhead building seen in the GT, whereas baselines blur or fail to capture it. Note that the transmission-like appearance of some baselines arises when the window fails to form a proper surface and is left as a hole, so the diffuse interior behind it becomes directly visible rather than being rendered through a reflective surface. Our method recovers the window surface, so the BRDF model operates on proper geometry and the reflection is correctly reproduced. Additional visualizations are provided in Section A.6.

Even with monocular normal supervision, the original BRDF path requires substantial training to disentangle geometry from BRDF parameters under a view-inconsistent RGB loss. Our clay branch



Figure 5: Qualitative comparison of our method with baselines. Our clay-guided supervision yields cleaner geometry, which in turn produces sharper and more faithful reflections.



Figure 6: **Multi-view image-to-clay translation comparison.** Qwen-Image-Edit [40] and LTX-2 [14] without anchors often produce color-inconsistent translations across different views.

sidesteps this by learning geometry from view-consistent clay images without specular rendering, similar to 2DGS on non-reflective objects. Fig. A5 shows that this facilitates the separation of specular and diffuse colors and accelerates the learning of the environment map.

4.5 Ablation studies

Multi-view consistency of clay translation. We measure how well our multi-view image-to-clay model maps the same surface point to consistent colors across views using a color *variance ratio*:

$$R = \frac{\mathbb{E}_{\mathbf{p}}\left[\mathrm{Var}_v(C_v(\mathbf{p}))\right]}{\mathrm{Var}_{\mathbf{p}\in\mathcal{I}}(C(\mathbf{p}))}, \tag{10}$$

where $C_v(\mathbf{p})$ is the color at view v corresponding to surface point $\mathbf{p}$, and $\mathcal{I}$ is the set of all foreground pixels. For each pixel in an anchor view, we use the GT depth to reproject it to its $K = 10$ nearest views (selected by camera-center distance) and collect the values observed at the same 3D point. The numerator measures the *cross-view variance* at corresponding surface points, while the denominator measures the *global variance* across all foreground pixels. A low R thus indicates that a method produces expressive outputs overall while keeping the same surface point consistent across views.

Table 5 compares per-image translation with Qwen-Image-Edit [40] and our LTX-2-based translator under three view-arrangement strategies. Qwen-Image-Edit yields strong shading contrast (high global variance) but lacks any mechanism for cross-view coherence, producing visibly inconsistent colors across views (Fig. 6). LTX-2 without anchors (Section 3.1) stays consistent within each generation group but drifts across groups, implicitly suppressing overall variance to compensate. LTX-2 with sorted ordering groups views by camera proximity, so each group only sees a partial portion of the object, raising cross-view variance at points observed from distant groups. Our random-input strategy with anchor conditioning achieves the lowest R, indicating the most view-consistent translation. Fig. 7 visualizes our translation on synthetic scenes, where regions reflecting the same environment lighting are mapped to consistent tones across views; real-world results are shown in Fig. A2.

Table 5: **Multi-view consistency of clay translation and its downstream effect.** Left: consistency of translated clay images on Shiny Blender; lower ratio R indicates more view-consistent supervision relative to its expressiveness. Right: reconstruction obtained by running the full pipeline with each translator's clay images, keeping everything else fixed. Best is bolded.

	Translation consistency			Downstream reconstruction			
Method	Cross-view Var ↓	Global Var ↑	Ratio R ↓	MAE ↓	PSNR ↑	SSIM ↑	LPIPS ↓
Qwen-Edit [40]	0.0214	**0.0555**	0.3875	1.53	34.60	0.9732	0.057
LTX-2 (sorted)	0.0181	0.0455	0.3966	1.39	34.78	0.9749	0.054
LTX-2 (no anchor)	**0.0157**	0.0398	0.4299	1.33	34.94	0.9750	0.054
LTX-2 (**ours**)	0.0163	0.0436	**0.3661**	**1.27**	**35.12**	**0.9760**	**0.053**



Figure 7: **Multi-view image-to-clay translation results on synthetic scenes.** Top: input RGB views from Glossy Synthetic; bottom: translated clay images. Regions reflecting the same environment lighting are translated into consistent tones across views. Real-world results are provided in Fig. A2.

The rightmost columns of Table 5 further link this consistency to final reconstruction quality. Our translator achieves the lowest variance ratio and the best downstream reconstruction metrics among the evaluated variants.

Selective gradient routing. We rely on selective gradient routing to make the early training stage learn geometry primarily from the clay loss. Table A6 shows our ablation. Applying the clay loss without routing already helps, but routing it to both position and opacity yields the best PSNR. Maintaining a weak residual gradient from the RGB loss is also beneficial, particularly on textureless or flat regions where clay images provide limited cues. We additionally ablate the lock duration: locking for around 40% of total iterations is most helpful, while longer durations leave the BRDF path with insufficient time to refine the geometry, leading to performance degradation.

5 Conclusion

We presented the Pygmalion Effect in Vision, a framework that reconstructs reflective objects by reducing specular ambiguity through clay-guided supervision. Combining a BRDF-based reflective branch with a clay-guided branch using synthesized clay images enables reliable disentanglement of reflection and geometry. Beyond practical gains, our results suggest that neutralizing specular cues offers a strong inductive bias for learning geometry, pointing toward new directions for challenging materials and appearance modeling.

Acknowledgments and Disclosure of Funding

We sincerely thank our colleagues at NAVER AI Lab, especially Sangdoo Yun, for their valuable support and insightful feedback. This work was supported by NAVER Corporation and NAVER Cloud Corporation through research funding and computing resources. Gayoung Lee, Junho Kim, and Jin-Hwa Kim are employees of NAVER Corporation.

References

[1] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020.

[2] Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. Mip-NeRF: A multiscale representation for anti-aliasing neural radiance fields. In *ICCV*, 2021.

[3] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM TOG*, 2022.

[4] Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume rendering of neural implicit surfaces. In *NeurIPS*, 2021.

[5] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. NeuS: learning neural implicit surfaces by volume rendering for multi-view reconstruction. In *NeurIPS*, 2021.

[6] Dejan Azinović, Ricardo Martin-Brualla, Dan B Goldman, Matthias Nießner, and Justus Thies. Neural RGB-D surface reconstruction. In *CVPR*, 2022.

[7] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D gaussian splatting for real-time radiance field rendering. *ACM TOG*, 2023.

[8] Addy Ngan, Frédo Durand, and Wojciech Matusik. Experimental analysis of BRDF models. In *Eurographics conference on Rendering Techniques*, 2005.

[9] Joakim Löw, Joel Kronander, Anders Ynnerman, and Jonas Unger. BRDF models for accurate and efficient rendering of glossy surfaces. *ACM TOG*, 2012.

[10] Yuxuan Yao, Zixuan Zeng, Chun Gu, Xiatian Zhu, and Li Zhang. Reflective gaussian splatting. In *ICLR*, 2025.

[11] Jinguang Tong, Xuesong Li, Fahira Afzal Maken, Sundaram Muthu, Lars Petersson, Chuong Nguyen, and Hongdong Li. GS-2DGS: Geometrically supervised 2dgs for reflective object reconstruction. In *CVPR*, 2025.

[12] Wenyuan Zhang, Jimin Tang, Weiqi Zhang, Yi Fang, Yu-Shen Liu, and Zhizhong Han. MaterialRefGS: Reflective gaussian splatting with multi-view consistent material inference. In *NeurIPS*, 2025.

[13] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, 1994.

[14] Yoav HaCohen, Benny Brazowski, Nisan Chiprut, Yaki Bitterman, Andrew Kvochko, Avishai Berkowitz, Daniel Shalem, Daphna Lifschitz, Dudu Moshe, Eitan Porat, et al. LTX-2: Efficient joint audio-visual foundation model. *arXiv preprint arXiv:2601.03233*, 2026.

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

[16] Zhaoshuo Li, Thomas Müller, Alex Evans, Russell H Taylor, Mathias Unberath, Ming-Yu Liu, and Chen-Hsuan Lin. Neuralangelo: High-fidelity neural surface reconstruction. In *CVPR*, 2023.

[17] Yiming Wang, Qin Han, Marc Habermann, Kostas Daniilidis, Christian Theobalt, and Lingjie Liu. NeuS2: Fast learning of neural implicit surfaces for multi-view reconstruction. In *ICCV*, 2023.

[18] Michael Oechsle, Songyou Peng, and Andreas Geiger. UNISURF: Unifying neural implicit surfaces and radiance fields for multi-view reconstruction. In *ICCV*, 2021.

[19] Zehao Yu, Songyou Peng, Michael Niemeyer, Torsten Sattler, and Andreas Geiger. MonoSDF: Exploring monocular geometric cues for neural implicit surface reconstruction. In *NeurIPS*, 2022.

[20] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2d gaussian splatting for geometrically accurate radiance fields. In *ACM SIGGRAPH*, 2024.

[21] Jia Li, Lu Wang, Lei Zhang, and Beibei Wang. TensoSDF: Roughness-aware tensorial representation for robust geometry and material reconstruction. *ACM TOG*, 2024.

[22] Yuan Liu, Peng Wang, Cheng Lin, Xiaoxiao Long, Jiepeng Wang, Lingjie Liu, Taku Komura, and Wenping Wang. NeRO: Neural geometry and brdf reconstruction of reflective objects from multiview images. *ACM TOG*, 2023.

[23] Yingwenqi Jiang, Jiadong Tu, Yuan Liu, Xifeng Gao, Xiaoxiao Long, Wenping Wang, and Yuexin Ma. GaussianShader: 3d gaussian splatting with shading functions for reflective surfaces. In *CVPR*, 2024.

[24] Zhihao Liang, Qi Zhang, Ying Feng, Ying Shan, and Kui Jia. GS-IR: 3d gaussian splatting for inverse rendering. In *CVPR*, 2024.

[25] Jian Gao, Chun Gu, Youtian Lin, Zhihao Li, Hao Zhu, Xun Cao, Li Zhang, and Yao Yao. Relightable 3d gaussians: Realistic point cloud relighting with brdf decomposition and ray tracing. In *ECCV*, 2024.

[26] Youjia Zhang, Anpei Chen, Yumin Wan, Zikai Song, Junqing Yu, Yawei Luo, and Wei Yang. Ref-GS: Directional factorization for 2d gaussian splatting. In *CVPR*, 2025.

[27] Dor Verbin, Pratul P Srinivasan, Peter Hedman, Ben Mildenhall, Benjamin Attal, Richard Szeliski, and Jonathan T Barron. Nerf-casting: Improved view-dependent appearance with consistent reflections. In *SIGGRAPH Asia 2024 Conference Papers*, 2024.

[28] Nicolas Moenne-Loccoz, Ashkan Mirzaei, Or Perel, Riccardo de Lutio, Janick Martinez Esturo, Gavriel State, Sanja Fidler, Nicholas Sharp, and Zan Gojcic. 3d gaussian ray tracing: Fast tracing of particle scenes. *ACM TOG*, 2024.

[29] Shihao Wu, Hui Huang, Tiziano Portenier, Matan Sela, Daniel Cohen-Or, Ron Kimmel, and Matthias Zwicker. Specular-to-diffuse translation for multi-view reconstruction. In *ECCV*, 2018.

[30] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, 2017.

[31] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV*, 2017.

[32] Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz. Multimodal unsupervised image-to-image translation. In *ECCV*, 2018.

[33] Taesung Park, Alexei A Efros, Richard Zhang, and Jun-Yan Zhu. Contrastive learning for unpaired image-to-image translation. In *ECCV*, 2020.

[34] Zhenxiong Tan, Songhua Liu, Xingyi Yang, Qiaochu Xue, and Xinchao Wang. OminiControl: Minimal and universal control for diffusion transformer. In *ICCV*, 2025.

[35] Chong Mou, Qichao Sun, Yanze Wu, Pengze Zhang, Xinghui Li, Fulong Ye, Songtao Zhao, and Qian He. InstructX: Towards unified visual editing with MLLM guidance. *arXiv preprint arXiv:2510.08485*, 2025.

[36] Ruonan Yu, Zhenxiong Tan, Zigeng Chen, Songhua Liu, and Xinchao Wang. ViFeEdit: A video-free tuner of your video diffusion transformer. *arXiv preprint arXiv:2603.15478*, 2026.

[37] Mengtian Li, Jinshu Chen, Songtao Zhao, Wanquan Feng, Pengqi Tu, and Qian He. DreamStyle: A unified framework for video stylization. *arXiv preprint arXiv:2601.02785*, 2026.

[38] Shiwen Zhang, Xiaoyan Yang, Bojia Zi, Haibin Huang, Chi Zhang, and Xuelong Li. TeleStyle: Content-preserving style transfer in images and videos. *arXiv preprint arXiv:2601.20175*, 2026.

[39] Pengze Zhang, Yanze Wu, Mengtian Li, Xu Bai, Songtao Zhao, Fulong Ye, Chong Mou, Xinghui Li, Zhuowei Chen, Qian He, and Mingyuan Gao. OmniTransfer: All-in-one framework for spatio-temporal video transfer. *arXiv preprint arXiv:2601.14250*, 2026.

[40] Chenfei Wu, Jiahao Li, Jingren Zhou, Junyang Lin, Kaiyuan Gao, Kun Yan, Sheng-ming Yin, Shuai Bai, Xiao Xu, Yilei Chen, Yuxiang Chen, Zecheng Tang, Zekai Zhang, Zhengyi Wang, An Yang, Bowen Yu, Chen Cheng, Dayiheng Liu, Deqing Li, Hang Zhang, Hao Meng, Hu Wei, Jingyuan Ni, Kai Chen, Kuan Cao, Liang Peng, Lin Qu, Minggang Wu, Peng Wang, Shuting Yu, Tingkun Wen, Wensen Feng, Xiaoxiao Xu, Yi Wang, Yichang Zhang, Yongqiang Zhu, Yujia Wu, Yuxuan Cai, and Zenan Liu. Qwen-Image technical report. *arXiv preprint arXiv:2508.02324*, 2025.

[41] Lianghua Huang, Wei Wang, Zhi-Fan Wu, Yupeng Shi, Huanzhang Dou, Chen Liang, Yutong Feng, Yu Liu, and Jingren Zhou. In-context LoRA for diffusion transformers. *arXiv preprint arXiv:2410.23775*, 2024.

[42] Jacob Munkberg, Jon Hasselgren, Tianchang Shen, Jun Gao, Wenzheng Chen, Alex Evans, Thomas Müller, and Sanja Fidler. Extracting Triangular 3D Models, Materials, and Lighting From Images. In *CVPR*, 2022.

[43] Mu Hu, Wei Yin, Chi Zhang, Zhipeng Cai, Xiaoxiao Long, Hao Chen, Kaixuan Wang, Gang Yu, Chunhua Shen, and Shaojie Shen. Metric3D v2: A versatile monocular geometric foundation model for zero-shot metric depth and surface normal estimation. *IEEE TPAMI*, 2024.

[44] Tao Xie, Xi Chen, Zhen Xu, Yiman Xie, Yudong Jin, Yujun Shen, Sida Peng, Hujun Bao, and Xiaowei Zhou. EnvGS: Modeling view-dependent appearance with environment gaussian. In *CVPR*, 2025.

[45] Dor Verbin, Peter Hedman, Ben Mildenhall, Todd Zickler, Jonathan T Barron, and Pratul P Srinivasan. Ref-NeRF: Structured view-dependent appearance for neural radiance fields. *IEEE TPAMI*, 2024.

[46] Keyang Ye, Qiming Hou, and Kun Zhou. 3d gaussian splatting with deferred reflection. In *ACM SIGGRAPH*, 2024.

[47] Team Hunyuan3D, Bowen Zhang, Chunchao Guo, Dongyuan Guo, Haolin Liu, Hongyu Yan, Huiwen Shi, Jiaao Yu, Jiachen Xu, Jingwei Huang, Kunhong Li, Lifu Wang, Linus, Penghao Wang, Qingxiang Lin, Ruining Tang, Xianghui Yang, Yang Li, Yirui Guan, Yunfei Zhao, Yunhan Yang, Zeqiang Lai, Zhihao Liang, and Zibo Zhao. HY3D-Bench: Generation of 3d assets. *arXiv preprint arXiv:2602.03907*, 2026.

[48] Chendi Lin, Heshan Liu, Qunshu Lin, Zachary Bright, Shitao Tang, Yihui He, Minghao Liu, Ling Zhu, and Cindy Le. Objaverse++: Curated 3d object dataset with quality annotations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, 2025.

[49] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. In *CVPR*, 2023.

[50] Matt Deitke, Ruoshi Liu, Matthew Wallingford, Huong Ngo, Oscar Michel, Aditya Kusupati, Alan Fan, Christian Laforte, Vikram Voleti, Samir Yitzhak Gadre, Eli VanderBilt, Aniruddha Kembhavi, Carl Vondrick, Georgia Gkioxari, Kiana Ehsani, Ludwig Schmidt, and Ali Farhadi. Objaverse-XL: A universe of 10m+ 3d objects. In *NeurIPS*, 2023.

[51] Poly Haven. Poly Haven: The public 3d asset library. `https://polyhaven.com/`, 2025.

[52] Zhengcong Fei, Di Qiu, Debang Li, Changqian Yu, and Mingyuan Fan. Video diffusion transformers are in-context learners. *arXiv preprint arXiv:2412.10783*, 2024.

[53] Bruce Walter, Stephen R Marschner, Hongsong Li, and Kenneth E Torrance. Microfacet models for refraction through rough surfaces. In *Eurographics conference on Rendering Techniques*, 2007.

[54] Claus Müller. *Spherical Harmonics*. Lecture Notes in Mathematics. Springer, 1966.

Technical Appendices and Supplementary Material

A.1 Analogy to the Pygmalion effect

The Pygmalion effect, originally a psychological phenomenon describing how expectations can shape outcomes, finds an intriguing parallel in our study. In the myth, the sculptor *Pygmalion* creates a statue so lifelike that his belief in its beauty brings it to life. Similarly, our model is trained on datasets curated through the expectations of "what a correct reflective object should look like." These expectations, embedded in the proposed image-to-clay translation, recursively reinforce the model's perception of the object.

In our work, this metaphor captures the essence of geometry-aware generative reconstruction: our learning framework not only reconstructs the observed world but also projects the internal geometric belief of the image-to-clay translation model back into the image domain, closing a loop between seeing and shaping. By acknowledging this Pygmalion effect in vision, we emphasize the importance of critical evaluation, ensuring that the visual worlds our models bring to life remain faithful to reality rather than to their own internal expectations through evaluation benchmarks.

A.2 Multi-view dataset for image-to-clay translation

To train our multi-view image-to-clay translation model, we construct a dataset of paired multi-view renderings — each sample contains the same object rendered with a reflective material and with a diffuse clay material, sharing identical camera poses. Our dataset comprises $7{,}996$ objects sourced from HY3D-Bench [47] and an additional 154 objects from Objaverse++ [48], all rendered using our pipeline described below.

Mesh sources. We primarily use HY3D-Bench [47], a curated collection of 252K+ watertight, manifold meshes that have been cleaned and normalized through a professional preprocessing pipeline. Compared to raw Objaverse [49, 50] meshes, which often contain non-manifold geometry, holes, and inconsistent normal orientations, HY3D-Bench provides geometry that is reliably renderable across diverse camera poses without artifacts such as backface leakage or inconsistent shading. This is particularly important for our paired-rendering setup, where any geometric inconsistency between the reflective and clay passes would inject spurious supervision signal. We additionally include 154 objects from Objaverse++ [48], a quality-annotated subset of Objaverse, to retain a small portion of original surface textures.

Synthetic rendering pipeline. We render each mesh using Blender's Cycles path tracer at 768×768 resolution. Each mesh is first normalized so that its bounding-box maximum dimension fits within a unit cube centered at the origin, with consistent up-axis orientation. For each object, we sample one HDRI environment map from PolyHaven [51] as the background, applied through the world shader without making the film transparent — preserving realistic illumination across all views.

Multi-view camera sampling. We generate 24 camera poses per object using Fibonacci-sphere sampling with stochastic perturbation. Camera elevations are distributed in $[15°, 100°]$ from the top, covering the upper, side, and slightly lower regions of the object. Azimuthal angles follow the golden-ratio sequence and are jittered by ± 0.15 rad in azimuth and ± 0.1 rad in elevation, with the camera distance and look-at point also randomly perturbed. This yields well-distributed yet diverse views per object, avoiding the regularity of a fixed sampling pattern that the model could overfit to.

Paired material rendering. For each object and camera pose, we render two passes that share identical geometry, illumination, and viewpoint:

Reflective pass. With probability 0.5 we assign a metallic appearance, choosing one of ten preset metal colors (e.g., gold, silver, copper, gunmetal) with metallic factor 1.0 and roughness sampled uniformly from $[0.0, 0.15]$. Otherwise, we assign a dielectric appearance with random HSV color, metallic factor 0, roughness in $[0.0, 0.1]$, specular IOR in $[0.8, 1.0]$, and a random clearcoat layer in $[0, 1]$. This split exposes the model to both pure-metal and glossy-painted surface types observed in real-world reflective objects.



Figure A1: Visualization of our multi-view dataset. Top: rendered reflective objects, where each pair of columns shows two views of the same object. Bottom: corresponding clay renderings under the same camera and lighting.

Clay pass. The same scene is re-rendered with all materials overridden to a neutral diffuse appearance: base color $(0.3, 0.3, 0.3)$, metallic 0, roughness 1.0, and specular 0. The result approximates a Lambertian gray-clay rendering that conveys geometric structure without specular cues.

Because both passes share the same camera trajectory and HDRI, the resulting 24-view paired sequence forms a clean multi-view supervision signal for our pseudo-video training (Section 3.1). Object masks are additionally rendered via Blender's compositor to support background-free training setups. Camera intrinsics and extrinsics are stored in a NeRF-style `transforms.json` per object. Example renderings are shown in Fig. A1.

A.3 Preliminary: LTX-2 and In-Context LoRA

LTX-2 [14] is a latent video diffusion transformer trained with rectified flow matching. Video frames are compressed into compact latent tokens by a 3D causal VAE, then processed by a DiT-style backbone. Spatiotemporal positions are injected via 3D rotary position embeddings (RoPE), and the model accepts per-token timesteps, allowing clean and noisy tokens to coexist within a single sequence.

In-Context LoRA (IC-LoRA) [41] is a parameter-efficient adaptation scheme that recasts conditional generation as in-context inpainting on a frozen diffusion transformer. Conditioning and target tokens are concatenated into a single sequence, with the conditioning portion kept clean and the loss applied only to the noisy target tokens. While originally proposed for image diffusion transformers, IC-LoRA has been extended to video diffusion models for controlled editing and generation [52]. Since only a small set of LoRA adapters is trained, the pretrained joint-attention prior is preserved while the adapters learn the conditioning-to-target mapping.

A.4 Detailed formulation of reflective 2D Gaussian Splatting

2D Gaussian Splatting (2DGS) [20] improves upon Gaussian-based representations by defining oriented 2D disks instead of volumetric Gaussians. Each Gaussian is represented on a local tangent plane, parameterized by its center $\mathbf{p}$, two tangential directions $\mathbf{t}_u, \mathbf{t}_v$, and scaling factors s_u, s_v. A surface point on the disk is expressed as:

$$\mathbf{P}(u, v) = \mathbf{p} + s_u \mathbf{t}_u u + s_v \mathbf{t}_v v. \tag{11}$$

The Gaussian weight over this plane is:

$$\mathcal{G}(u, v) = \exp\left(-\frac{u^2 + v^2}{2}\right). \tag{12}$$

During rendering, a pixel $\mathbf{x}$ is projected onto the local plane of each Gaussian, yielding the corresponding local coordinates $\mathbf{u}(\mathbf{x}) = [u(\mathbf{x}), v(\mathbf{x})]^\top$ through the inverse projection of $\mathbf{P}(u, v)$. The

color at $\mathbf{x}$ is then obtained by the standard alpha-compositing rule:

$$\mathbf{c}(\mathbf{x}) = \sum_{i=1}^{N} \mathbf{c}_i \, \alpha_i \, \hat{\mathcal{G}}_i(\mathbf{u}(\mathbf{x})) \prod_{j=1}^{i-1} \left(1 - \alpha_j \, \hat{\mathcal{G}}_j(\mathbf{u}(\mathbf{x}))\right), \tag{13}$$

where $\mathbf{c}_i$ is the Gaussian color, α is the Gaussian opacity, and $\hat{\mathcal{G}}$ is the visibility-corrected Gaussian weight of $\mathcal{G}$.

We build upon MaterialRefGS [12], which is based on Reflective Gaussian Splatting (RGS) [10], where each Gaussian is associated with both geometric and BRDF parameters. Each Gaussian primitive is associated with a feature vector $\boldsymbol{b}_i = [\lambda_i, m_i, r_i, \mathbf{n}_i]$, where λ_i, m_i, r_i, and $\mathbf{n}_i$ denote albedo, metallic, roughness, and surface normal, respectively, for the i-th of N Gaussians. These attributes are composited using the same alpha blending as in 2DGS:

$$\mathbf{B}(\boldsymbol{x}) = \sum_{i=1}^{N} \boldsymbol{b}_i \, \alpha_i \, \hat{\mathcal{G}}_i(\mathbf{u}(\boldsymbol{x})) \prod_{j=1}^{i-1} \left(1 - \alpha_j \, \hat{\mathcal{G}}_j(\mathbf{u}(\boldsymbol{x}))\right), \tag{14}$$

where $\mathbf{B} = [\Lambda, M, R, \boldsymbol{N}]$ represents the accumulated material features, while α and $\hat{\mathcal{G}}$ denote opacity and 2D Gaussian evaluated at the local tangent coordinate $\mathbf{u}(\boldsymbol{x})$, respectively.

The final outgoing radiance $L(\omega_o)$ at the viewing direction ω_o is computed using the physically-based rendering equation:

$$L(\omega_o) = \int_{\Omega} L_i(\omega_i) \, f(\omega_i, \omega_o) \, (\omega_i \cdot \boldsymbol{N}) \, d\omega_i, \tag{15}$$

where $L_i(\omega_i)$ denotes the incoming radiance from direction ω_i, and $f(\omega_i, \omega_o)$ is the bidirectional reflectance distribution function (BRDF).

The BRDF is decomposed into diffuse and specular terms $f = f_d + f_s$. While the diffuse term f_d is straightforward to integrate, the specular term follows the GGX microfacet model [53]:

$$f_s(\omega_i, \omega_o) = \frac{D \, G \, F}{4(\omega_o \cdot \boldsymbol{N})(\omega_i \cdot \boldsymbol{N})}, \tag{16}$$

where D, G, and F denote the normal distribution, geometry, and Fresnel terms, respectively. To efficiently evaluate the specular reflection, we compute the specular radiance $L_s(\omega_o)$ using the precomputed split-sum formulation:

$$\begin{aligned} L_s(\omega_o) \approx & \left(\int_{\Omega} f_s(\omega_i, \omega_o)(\omega_i \cdot \boldsymbol{N}) \, d\omega_i \right) \\ & \cdot \left(\int_{\Omega} L_i(\omega_i) D(\omega_i, \omega_o)(\omega_i \cdot \boldsymbol{N}) \, d\omega_i \right), \end{aligned} \tag{17}$$

where the first term depends only on the view direction and roughness and can therefore be precomputed and stored in a 2D lookup texture, while the second term is efficiently evaluated from prefiltered environment maps.

Following the previous work [10], we further separate the lighting into direct L_{dir}, the second term in Eq. (17), and indirect L_{ind} components to account for inter-reflection:

$$\begin{aligned} L'_s(\omega_o) \approx & \left(\int_{\Omega} f_s(\omega_i, \omega_o)(\omega_i \cdot \boldsymbol{N}) \, d\omega_i \right) \\ & \cdot \big(L_{\text{dir}} \cdot \chi_v + L_{\text{ind}} \cdot (1 - \chi_v)\big), \end{aligned} \tag{18}$$

where $\chi_v \in \{0, 1\}$ is a binary visibility indicator computed by ray tracing on the extracted mesh surface. The indirect light L_{ind} is defined as follows:

$$L_{\text{ind}} = \sum_{i=1}^{N} \boldsymbol{l}_{\text{ind},i} \, \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j), \tag{19}$$

where $\boldsymbol{l}_{\text{ind},i}$ denotes an additional view-dependent color evaluated along the reflected direction and modeled using spherical harmonics [54].

A.5 Additional implementation details

We fine-tune LTX-2 [14] for $50{,}000$ iterations on $8\times$ NVIDIA H100 GPUs with a batch size of 1 per GPU, where each sample contains all 24 multi-view images. For each sample, the 24 views are randomly resampled with replacement and augmented with random cropping. We adapt the model with LoRA [15] modules of rank 32 and use AdamW with a learning rate of 2×10^{-4}. All training samples share a unified text prompt: `"A 3D rendered object with diffuse white clay-like material and surface properties"`. To enable the anchor-based inference scheme described in Section 3.1, with probability 0.5 we expose four randomly selected target views as clean anchors.

For the reflective object reconstruction stage, we use a single H100 GPU. Following the baselines, we train for $50{,}000$ iterations on synthetic data and $30{,}000$ iterations on real-world data, except for the NeRF-Casting scenes, which we train longer for $50{,}000$ iterations. Selective gradient routing is applied during the first $20{,}000$ iterations for synthetic data and the first $6{,}000$ iterations for real data. For the mask loss, the normal prior schedule, and other regularization terms, we follow the hyperparameters provided in the official MaterialRefGS [12] codebase. Following the official recommendation, we estimate monocular normal maps using the official implementation of Metric3D v2 [43] and use them as normal supervision.

Since the MaterialRefGS codebase provides no configuration for the Glossy-Real and NeRF-Casting datasets, we adopt its Ref-Real setting as the default and adjust a few options so that the unbounded backgrounds remain intact: we set the distortion loss weight to zero on both datasets, and for Glossy-Real we additionally disable the normal propagation phase, whose periodic perturbation, designed under the assumption of object-centric synthetic data, introduces background floaters in unbounded real scenes. These adjusted settings are applied identically to the MaterialRefGS baseline and to our clay-guided variant for a fair comparison.

Timing breakdown. Table A1 reports the wall-clock timing of our full pipeline, measured on the *angel* scene of Glossy Synthetic (112 views) using a single NVIDIA H100 GPU. Clay translation accounts for only 3.5% of the total pipeline time (4.4% including the one-time model loading), and is a one-time pre-processing cost per scene, independent of the number of optimization iterations. The clay translation stage is therefore far from a runtime bottleneck.

Table A1: **Wall-clock timing breakdown.** Measured on the *angel* scene of Glossy Synthetic (112 views) with a single NVIDIA H100 GPU.

Stage	Time
LTX-2 model loading (one-time)	1m 16s
Clay translation of all 112 training views	4m 59s
MaterialRefGS optimization ($50{,}000$ iterations)	2h 15m 7s
Total	2h 21m 22s

A.6 Additional visualization and results

In this section, we provide additional visualizations of the image-to-clay transformation results and our reconstructed outputs.

A.6.1 Multi-view image-to-clay model outputs

Figure A2 shows our image-to-clay translation results on real-world reflective objects. Results on synthetic scenes are visualized in Fig. 7 of the main paper. Despite being trained only on synthetic data, our model generalizes well to real-world scenes thanks to the strong prior of LTX-2. Notably, for object-centric scenes, the model also translates auxiliary structures into clay, but since the underlying geometry remains aligned with the RGB image, this does not affect downstream training. For car windows—a region where monocular normal estimators frequently fail—our model produces clean matte surfaces. Although our synthetic training pairs contain only a single material per object, objects with spatially varying materials (*e.g.*, the car in Fig. A2, where glass, metal, and rubber coexist)



Figure A2: Visualization of our multi-view image-to-clay outputs on real-world scenes. Synthetic results are shown in Fig. 7.

Table A2: Per-scene image quality comparison on synthetic datasets. The intensity of the red color signifies a better result. The Avg. column is bolded for readability.

Datasets	**Shiny Blender** [45]							**Glossy Synthetic** [22]							
Scenes	**Ball**	**Car**	**Coffee**	**Helmet**	**Teapot**	**Toaster**	**Avg.**	**Angel**	**Bell**	**Cat**	**Horse**	**Luyu**	**Potion**	**Tbell**	**Teapot**	**Avg.**
							PSNR ↑									
2DGS	25.97	26.38	32.31	27.42	44.97	20.42	29.58	26.95	24.79	30.65	25.18	26.89	29.50	23.28	21.29	26.07
GShader	30.99	27.96	32.39	28.32	45.86	26.28	31.97	25.08	28.07	31.81	26.56	27.18	30.09	24.48	23.58	27.11
R3DG	23.64	25.92	30.10	25.01	43.15	18.80	27.77	24.90	23.51	27.59	23.37	24.68	27.29	21.25	20.47	24.13
3DGS-DR	33.43	30.48	34.53	31.44	47.04	26.76	33.95	29.07	30.60	32.59	26.17	28.96	32.65	29.03	25.77	29.36
Ref-GS	36.10	30.94	34.38	33.40	46.69	27.28	34.80	30.34	31.70	33.15	27.45	29.46	32.64	30.08	26.47	30.16
RGS	37.01	31.04	34.63	32.32	47.16	28.05	35.04	30.38	32.86	33.01	27.05	30.04	33.07	29.84	26.68	30.37
MaterialRefGS	35.38	31.27	34.34	32.87	44.97	29.25	34.68	31.42	31.40	33.57	27.52	30.25	33.47	30.21	26.99	30.60
Ours	35.53	31.31	34.45	33.08	47.00	29.35	35.12	31.53	31.93	33.46	27.62	30.34	33.45	30.05	27.04	30.68
							SSIM ↑									
2DGS	0.934	0.930	0.972	0.953	0.997	0.892	0.946	0.918	0.911	0.958	0.909	0.918	0.939	0.902	0.886	0.918
GShader	0.966	0.932	0.971	0.951	0.996	0.929	0.958	0.914	0.919	0.961	0.933	0.914	0.936	0.898	0.901	0.922
R3DG	0.888	0.922	0.963	0.931	0.995	0.858	0.926	0.894	0.888	0.934	0.878	0.889	0.911	0.875	0.869	0.892
3DGS-DR	0.979	0.963	0.976	0.971	0.997	0.942	0.971	0.942	0.959	0.973	0.933	0.943	0.959	0.958	0.942	0.951
Ref-GS	0.981	0.961	0.973	0.975	0.997	0.950	0.973	0.955	0.965	0.973	0.948	0.946	0.957	0.956	0.944	0.956
RGS	0.981	0.964	0.976	0.971	0.998	0.948	0.973	0.954	0.969	0.973	0.944	0.952	0.963	0.962	0.947	0.958
MaterialRefGS	0.984	0.967	0.976	0.973	0.997	0.954	0.975	0.960	0.959	0.977	0.949	0.955	0.965	0.965	0.953	0.960
Ours	0.984	0.967	0.976	0.974	0.997	0.955	0.976	0.960	0.963	0.977	0.950	0.955	0.966	0.964	0.953	0.961
							LPIPS ↓									
2DGS	0.156	0.052	0.079	0.079	0.008	0.127	0.084	0.072	0.109	0.060	0.071	0.066	0.097	0.125	0.101	0.088
GShader	0.121	0.044	0.078	0.074	0.007	0.079	0.067	0.082	0.098	0.056	0.056	0.064	0.088	0.091	0.122	0.083
R3DG	0.214	0.058	0.090	0.125	0.013	0.170	0.112	0.085	0.125	0.089	0.081	0.080	0.117	0.156	0.115	0.106
3DGS-DR	0.105	0.033	0.076	0.050	0.006	0.082	0.059	0.052	0.050	0.042	0.057	0.048	0.068	0.059	0.060	0.054
Ref-GS	0.098	0.034	0.082	0.045	0.006	0.070	0.056	0.042	0.049	0.041	0.046	0.046	0.076	0.073	0.064	0.055
RGS	0.098	0.033	0.076	0.049	0.006	0.074	0.056	0.042	0.040	0.040	0.048	0.043	0.064	0.058	0.058	0.049
MaterialRefGS	0.090	0.032	0.076	0.048	0.009	0.067	0.054	0.038	0.052	0.036	0.045	0.042	0.061	0.057	0.053	0.048
Ours	0.089	0.031	0.076	0.046	0.006	0.066	0.053	0.038	0.047	0.036	0.044	0.041	0.061	0.056	0.052	0.047

are translated reliably without noticeable artifacts at material boundaries, which we attribute to the material diversity of our training set and the strong object prior of the video backbone.

A.6.2 Clay-guided reflective object reconstruction results

We provide full per-scene photometric results corresponding to the abbreviated Table 4 in Tables A2 and A3. We also report the Mean Angular Error (MAE) of surface normals in Table A4. Additional qualitative comparisons are shown in Fig. A3.

A.6.3 More analysis

To illustrate why the clay loss benefits geometry learning, we visualize the diffuse color, specular color, and rendered normals at intermediate training iterations in Fig. A5. Both methods share the same normal supervision. However, the clay pass bypasses specular evaluation and thus strongly regularizes geometry, preventing Gaussians from distorting their normals and opacities to entangle view-dependent reflections into the diffuse color. Consequently, the environment map also converges

Table A3: Per-scene image quality comparison on real-world datasets. "Ours" uses MaterialRefGS as the reflective Gaussian splatting backbone. The intensity of the red color signifies a better result.

Datasets	**Ref-Real** [45]				**Glossy-Real** [22]						**NeRF-casting** [27]				
Scenes	**Garden**	**Sedan**	**Toycar**	**Avg.**	**Bear**	**Bunny**	**Coral**	**Maneki**	**Vase**	**Avg.**	**Compact**	**Grinder**	**Hatchback**	**Toaster**	**Avg.**
							PSNR ↑								
2DGS	22.53	26.23	23.70	24.15	19.47	23.59	18.94	22.37	21.61	21.20	23.38	25.33	23.96	25.99	24.66
Ref-GS	22.48	26.63	24.20	24.44	17.17	26.02	19.09	24.65	24.70	22.33	28.19	30.46	25.27	28.45	28.09
RGS	22.97	26.60	24.27	24.61	19.89	24.56	18.83	23.70	23.01	22.00	28.32	30.67	26.12	28.28	28.35
MaterialRefGS	23.29	26.04	24.86	24.73	20.38	25.59	19.32	24.26	23.70	22.65	27.47	30.89	25.77	28.16	28.07
Ours	23.21	26.13	24.89	24.74	20.79	25.72	19.66	24.66	24.39	23.04	27.66	31.56	26.20	29.34	28.69
							SSIM ↑								
2DGS	0.609	0.778	0.597	0.661	0.750	0.861	0.762	0.842	0.806	0.804	0.828	0.787	0.799	0.871	0.822
Ref-GS	0.607	0.783	0.656	0.682	0.704	0.892	0.750	0.875	0.851	0.814	0.863	0.822	0.833	0.902	0.855
RGS	0.617	0.777	0.660	0.685	0.752	0.861	0.743	0.846	0.808	0.802	0.857	0.819	0.819	0.903	0.849
MaterialRefGS	0.645	0.765	0.685	0.698	0.768	0.880	0.754	0.858	0.830	0.818	0.843	0.821	0.812	0.903	0.845
Ours	0.642	0.770	0.684	0.699	0.769	0.885	0.757	0.865	0.843	0.824	0.847	0.829	0.823	0.915	0.853
							LPIPS ↓								
2DGS	0.254	0.225	0.396	0.292	0.275	0.203	0.252	0.236	0.248	0.243	0.235	0.254	0.244	0.166	0.225
Ref-GS	0.242	0.196	0.236	0.225	0.332	0.164	0.263	0.190	0.189	0.228	0.178	0.221	0.187	0.154	0.185
RGS	0.256	0.245	0.256	0.252	0.273	0.218	0.276	0.243	0.261	0.254	0.215	0.244	0.230	0.157	0.211
MaterialRefGS	0.180	0.179	0.191	0.183	0.233	0.133	0.212	0.173	0.178	0.186	0.159	0.156	0.167	0.110	0.148
Ours	0.184	0.174	0.191	0.183	0.232	0.130	0.233	0.164	0.161	0.184	0.152	0.143	0.154	0.099	0.137

Table A4: Mean Angular Error of normals on Shiny Blender dataset. Lower is better. The intensity of the red color signifies a better result.

	Car	**Ball**	**Helmet**	**Teapot**	**Toaster**	**Coffee**	**Average**
GS-IR	28.31	25.79	25.58	15.35	33.51	15.38	23.99
GShader	14.05	7.03	9.33	7.17	13.08	14.93	10.93
3DGS-DR	2.32	0.85	1.67	0.53	6.99	2.21	2.43
Ref-GS	2.02	1.05	1.99	0.69	3.92	3.61	2.21
RGS	1.77	1.17	2.07	0.70	3.90	3.07	2.11
MaterialRefGS	1.55	0.65	1.18	0.43	2.51	1.72	1.34
Ours	1.48	0.63	1.19	0.38	2.25	1.70	1.27

faster with early clay supervision. This rapid geometric stabilization improves geometry metrics (Table 2 and Table A4) and allows the BRDF branch to focus on material refinement, yielding better qualitative results. In Fig. A4, we visualize material decomposition. RGS and MaterialRefGS bake reflections into the diffuse color (red boxes), indicating poor disentanglement between geometry and view-dependent appearance. Our method instead produces clean, view-independent diffuse colors and sharper normal maps, confirming that clay-guided supervision improves both geometry and material decomposition.

Table A5: Reconstruction on Glossy Synthetic with different diffuse guidance.

	CD ↓	PSNR ↑	SSIM ↑	LPIPS ↓
Baseline (MaterialRefGS)	0.0067	30.60	0.960	0.048
Ours w/ S2Dnet diffuse images	0.0229	28.62	0.943	0.065
Ours w/ our clay images	**0.0054**	**30.68**	**0.961**	**0.047**

Here we provide details on the selective gradient routing ablation summarized in Table A6. All ablations are conducted on the Shiny Blender dataset varying only the routing configuration. As shown in the table, routing both position and opacity gives the best PSNR, and the best overall configuration adds a weak residual RGB gradient with a 20K-iteration lock period (PSNR 35.12, MAE 1.27). Locking for 30K iterations causes a sharp drop (PSNR 33.77). When the routing period is too long, the BRDF branch is left with too few iterations to refine the surface geometry using photometric supervision from the RGB images.

View ordering within chunks. To separate the effect of view diversity from that of the specific ordering, we additionally evaluate a stratified chunking strategy that draws the views of each chunk from strata over the ordered camera trajectory, guaranteeing angular diversity within every chunk. Evaluated with the same model trained under random ordering, it performs on par with random ordering, and both clearly outperform sorted ordering. This indicates that the key factor behind the benefit of random ordering is view diversity within each chunk rather than the ordering itself. We keep random ordering as the default since exposure to diverse view orders during training additionally acts as data augmentation, while stratified chunking remains a supported inference-time option.

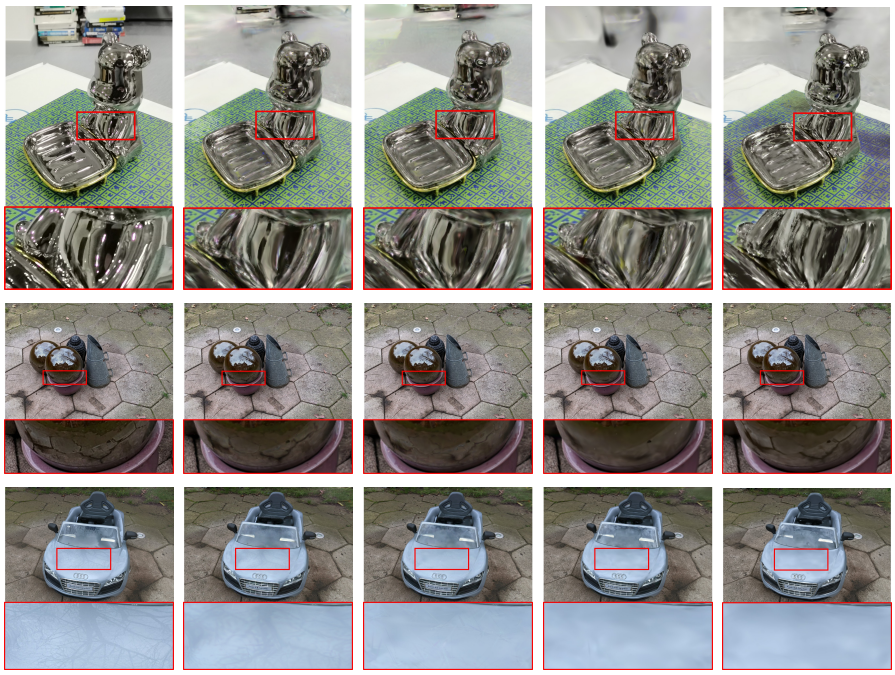


Figure A3: More qualitative comparison of our method with baselines.

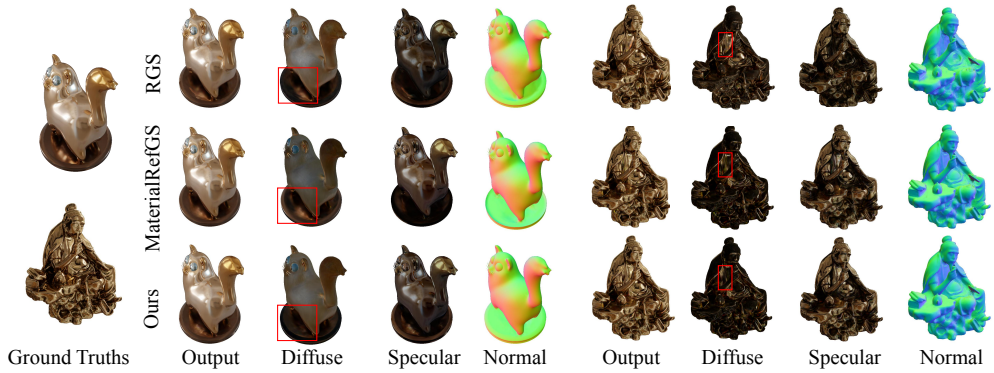


Figure A4: Material visualization. We visualize the rendered output, diffuse color, specular color, and surface normal separately. The red boxes show reflections that are incorrectly baked into the diffuse color instead of the specular color.

Comparison with specular-to-diffuse translation. We empirically compare with S2Dnet [29] by running its released pretrained model on Glossy Synthetic. Its diffuse conversion leaves reflective regions largely intact in most scenes, presumably because its training data lacks cases where the environment is mirrored on the object surface. Training our framework with these images degrades reconstruction below the baseline (Table A5): inaccurate diffuse guidance is worse than none, underscoring that faithful translation on challenging reflective data, enabled by the video prior, is central to our approach.

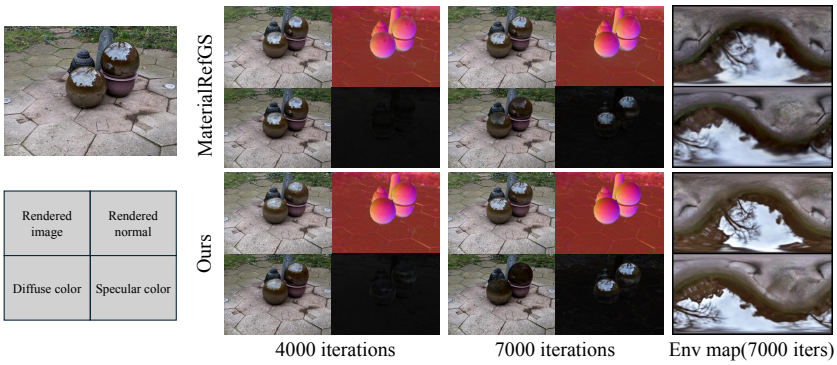


Figure A5: Convergence comparison during training. At both 4,000 and 7,000 iterations, our method produces fewer reflections baked into the diffuse color, and a more faithful environment map than MaterialRefGS, despite using the same normal supervision.

Table A6: Ablation on selective gradient routing on the Shiny Blender dataset. We jointly ablate (i) which Gaussian parameters are routed (position, opacity, rotation), (ii) whether a weak residual RGB gradient is applied to the routed parameters, and (iii) the lock duration. Our default configuration is highlighted.

Routed parameters				Lock duration	MAE ↓	PSNR ↑
Position	Opacity	Rotation	Weak RGB			
-	-	-	-	10K	1.32	35.01
✓	-	-	-	10K	1.29	35.04
✓	✓	-	-	10K	1.30	35.05
✓	✓	✓	-	10K	1.31	34.94
✓	✓	-	✓	5K	1.33	34.89
✓	✓	-	✓	10K	1.29	35.06
✓	✓	-	✓	20K	**1.27**	**35.12**
✓	✓	-	✓	30K	1.33	33.77

A.7 Broader impacts and limitations

A.7.1 Broader social impacts

Our work focuses on 3D reconstruction of reflective objects, which can be used for various applications such as digital content creation, AR/VR, and robotic perception. While 3D reconstruction in general could be misused for negative purposes such as military applications, as foundational research, we do not foresee direct negative societal impacts from our specific contribution.

A.7.2 Limitations

The quality of geometry learning is bounded by the accuracy of the image-to-clay translation model: incorrect translations can propagate into geometry optimization, particularly for materials or lighting conditions underrepresented in the training data. Training the translator on a broader range of materials and lighting conditions would likely further improve our method. In addition, the clay translation step introduces extra pre-processing cost, although it is negligible compared to the optimization phase. Translations of non-salient objects or multiple objects may fail because the translation model was trained on paired images of single foreground objects and their clay counterparts. Finally, while the method remains effective with moderately sparse inputs (8 views), extremely sparse captures (4 or 2 views) break the structure preservation of the translation.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly describe our main contributions: the multi-view image-to-clay translation model and the geometric enhancement provided by clay-guided supervision.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We provide the limitations in the appendix Section A.7.2.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: Our paper does not present theoretical results.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We describe all training and inference details (architecture, hyperparameters, dataset construction, evaluation protocols) in the main paper and appendix. Our implementation builds upon publicly available codebases (LTX-2 [14], MaterialRefGS [12]), and we will release our code upon publication.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: We do not release code at submission time. However, all implementation details are described in the main paper and appendix to enable reproducibility, and our method is built on publicly available codebases (LTX-2 [14] and MaterialRefGS [12]). We will release our full implementation upon publication.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We follow the experimental setup of the baseline (MaterialRefGS [12]), and provide detailed descriptions of our additional dataset construction in the appendix. Our newly proposed evaluation metric for the ablation study is also described in the main paper.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: Following the standard practice in 3D reconstruction [7, 20, 12], we report single-run results due to the high computational cost of training each scene-specific model. Our improvements are consistent across multiple scenes and benchmarks, supporting the robustness of our claims.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.

- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We specify the GPU type and number of training iterations for both the image-to-clay translation model ($8\times$ NVIDIA H100, 50K iterations) and the reconstruction model (single H100, 50K iterations on synthetic data and 30K on real data) in Section 4.1 and Section A.5.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Our work conforms with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: We discuss the broader impacts of our work in Section A.7.1.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: Our work does not release models or datasets with a high risk of misuse.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: All existing assets (datasets, pre-trained models, codebases) used in this work are properly credited in the main paper and appendix. We use each asset for research purposes in accordance with its original license and terms of use. PolyHaven [51] HDRIs are CC0, while other assets are used under their respective open-source licenses (e.g., MIT, Apache 2.0, BSD, CC-BY) or research-only terms.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: Implementation and training details are provided in the paper and supplementary material. We plan to release the code with accompanying documentation and may also release pretrained models.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: Our paper does not include any crowdsourcing experiments.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: Our paper does not include any crowdsourcing experiments.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: LLMs are not used as components of our core method.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.