Perspective-aware 3D Gaussian Inpainting with Multi-view Consistency

Supplementary Material

A. Preliminaries

In this section, we first briefly review some preliminaries related to 3D Gaussian splatting and 2D diffusion inpainter used in PAInpainter's framework.

3D Gaussian Splatting. 3D Gaussian Splatting (3DGS) is proposed to represent 3D scenes with 3D Gaussian primitives. Given a training dataset $\mathbf{I}$ of multi-view 2D images with camera poses $\mathbf{P}$, 3DGS learns a set of colored 3D Gaussians $\mathcal{G} = \{\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_N\}$, where N denotes the number of 3D Gaussians in the scene, $\mathbf{g}_i = \{\mu, \Sigma, c, \alpha\}$ and $i \in \{1, \ldots, N\}$. Specifically, μ is the position where the Gaussian is centered, Σ denotes the 3D covariance matrix, c is the RGB color and α is the opacity attribute. Accordingly, 3DGS proposes a novel differentiable rasterization for efficient training and rendering. The rendering process can be formulated as

$$C = \sum_{i \in N} c_i \sigma_i \prod_{j=1}^{i-1} (1 - \alpha_j), \tag{4}$$

where $\sigma_i = \alpha_i e^{-\frac{1}{2}(x_i)^\intercal \Sigma^{-1}(x_i)}$ represents the influence of the Gaussian to the image pixel and x_i is the distance between the pixel and the center of the i-th Gaussian. Additionally, the 3DGS training process is based on successive iterations of rendering and comparing the resulting image to the training views in $\mathbf{I}$.

Notably, from the neural representation aspect, the 3DGS inpainting can be regarded as fine-tuning a pretrained 3DGS scene $\mathcal{G}_u$ with unknown region using a dataset of inpainted 2D images $\mathbf{I}_{inpainted}$.

2D diffusion inpainter. 2D diffusion inpainter is a variant of Latent Diffusion Models (LDMs) focusing on inpainting masked area of 2D image [34]. In LDMs, a powerful pretrained Vector Quantised-Variational AutoEncoder (VQ-VAE) model [38] is employed to encode and decode the images to and from latent representations and the UNet [35] works for denoising the encoded image latent. Additionally, by introducing cross-attention layers into the UNet architecture, the generation can be controlled by text or other conditions. As a variant, the 2D diffusion inpainter expanded the UNet in LDMs to digest the mask conditioned features with unmasked area as priors and text as control condition. Thereby, the input of 2D diffusion inpainter is formulated as:

$$x_t = [z_t; \hat{\mathbf{M}}; z_{\mathbf{M}}] \in \mathbb{R}^{H \times W \times 9}, \tag{5}$$

where t indicates the time step in the diffusion; z_t denotes the 4-channel noised latent of input image; $\hat{\mathbf{M}}$ denotes the 1-channel binary-value mask down-sampled aligned with the size of image latent; $z_{\mathbf{M}}$ denotes the 4-channel noise-free latent feature in unmasked region. Together with encoded text prompt y by the textual CLIP model [33], the $\hat{\mathbf{M}}$ and $z_{\mathbf{M}}$ are concatenated as the input condition for UNet to get noise $\epsilon_\theta(x_t, t, y)$. The scheduler in 2D diffusion inpainter denoises the image latent in an iterative manner, and the final denoised latent is decoded to produce the inpainted image.

B. Implementation Details

B.1. Experiment Setup

Method implementation. The implementation of our 3D Gaussian Splatting (3DGS) is built upon the Nerfstudio framework. For scene initialization, we encountered a significant challenge: the large masked regions with black color in multi-view images prevent COLMAP from extracting valid 3D point clouds for 3DGS initialization. To address this, we leverage the available camera poses from the datasets and adopt different initialization strategies based on scene characteristics. For most scenes, we normalize the camera poses and randomly initialize 50k points within a unit cube to form the point cloud. However, for the scenes from SPIn-NeRF dataset, which feature uniform facet camera poses that make reconstruction from random initialization particularly challenging, we utilize their pre-computed 3D point clouds for initialization. This choice is justified by the difficulty in achieving stable reconstruction from random initialization under such camera configurations. To ensure fair comparison, all baseline methods in our experiments share identical experimental conditions, including multi-view images, camera poses, initial masked 3D Gaussian scene representations, and the optimization process of 3D Gaussians during inpainting.

Pretrained models. Our framework leverages several state-of-the-art pretrained models from official repositories. For image inpainting, we adopt the "stable-diffusion-2-inpainting" model from stabilityai (Hugging Face), denoted as SD2, which serves as the primary inpainting engine for all baseline methods except MVInpainter (which employs its proprietary pretrained models). Meanwhile, we adopt the same setting in NeRFiller, i.e. all image inpaintings are performed under the default SD2's scheduler with twenty diffusion steps. This choice is motivated by SD2's superior performance and stability in general inpainting tasks.

The pipeline integrates multiple specialized models for different components:

- **Depth Estimation**: The pretrained ZoeDepth model ("ZoeD-NK") along with off-the-shelf weights from the

official torch hub, chosen for its robust depth prediction capability in diverse scenarios. Although our framework implements the inpaint content propagation through visual projection, it remains robust against flaws caused by depth estimation under extreme condition, thanks to our iterative inpainting strategy. Additionally, the inpaint content propagation module in our framework only provides SD2 with prior information during inpainting process.To verify the reliability and generalization of ZoeDepth within our framework for 3D inpainting across various scenarios, including object-centric, indoor and outdoor scenes, we conducted comprehensive experiments on the three datasets mentioned in the main paper.

- **Feature Extraction**: We utilize the pretrained ResNet18 model from torchvision (default IMAGENET1K-V1 version), where we remove the last layer classification head and extract intermediate features for dual-feature consistency verification. This lightweight architecture enables efficient inference while maintaining high-quality feature representation
- **Geometric Correspondence**: The official LoFTR model for perspective graph construction, utilized without modifications due to its proven effectiveness in establishing reliable cross-view correspondences

We maintain all models in inference mode without fine-tuning, leveraging their well-established performance as strong baselines in their respective domains. This design choice ensures reproducibility and demonstrates the generalization capability of our method. The consistent application of these models across all experimental comparisons guarantees fair evaluation.

Hardware Configuration and Runtime Environment. All experiments are conducted on a server equipped with two NVIDIA RTX 3090 GPUs. We optimize the computational pipeline by dedicating one GPU to 3D Gaussian scene optimization tasks, while the other GPU handles the inference of pretrained models for image inpainting, depth estimation, and feature extraction. This parallel processing strategy significantly enhances computational efficiency while maintaining stable performance.

B.2. Hyper-parameters Explanation

There are several hyper-parameters used in our PAInpainter implementation and we explain and discuss them here.

τ for perspective graph construction. In our graph construction process, we employ feature matching to establish correspondences between multi-view images and utilize the average confidence score of matches to define the perspective distance between views. A higher average confidence score indicates closer perspective distance. Despite the promising performance of state-of-the-art feature matching models like LoFTR, challenging cases (e.g., significant viewpoint changes, textureless regions) may still produce unreliable matches with low confidence scores. To enhance the robustness of our graph construction method, we introduce a confidence threshold τ to filter out potentially unreliable matches. This filtering strategy effectively mitigates the impact of outliers and improves the overall stability of perspective distance estimation. We empirically set $\tau = 0.4$ across all scenes in our experiments for two main reasons: 1) This value maintains a balance between match quality and quantity, ensuring sufficient valid matches for reliable perspective distance computation, and 2) It demonstrates consistent performance across diverse scenes with different viewpoint distributions and geometric complexities.While the specific choice of τ may affect individual match selection, our experiments indicate that moderate variations in the perspective graph do not significantly impact the overall inpainting performance. This robustness can be attributed to our method's inpaint content propagation strategy and consistency verification mechanism. However, for scenes with sparse viewpoint sampling or challenging viewing conditions, a lower τ value might be necessary to retain adequate matches for meaningful perspective distance estimation.

k for adaptive adjacent images sampling. When performing consistent multi-view inpainting, we sample k adjacent images from the perspective graph for each anchor image. These sampled images form a batch for joint inpainting and subsequent 3D Gaussian optimization. In our experiments, we did not search the optimal value of k and consistently set $k = 8$ across all scenes to ensure fair comparison . While this parameter demonstrates robust performance in our framework, its value can be task-dependent and warrants careful consideration based on the following factors:

1. **Lower Bound Constraint:** An insufficient k may lead to disconnected sub-graphs during the sampling process, potentially hampering inpaint content propagation. Consider a scenario where $k = 2$ and three images form a cyclic nearest neighbor relationship. This configuration necessitates additional heuristic-based anchor image selection to bridge disconnected components, introducing computational overhead and potentially compromising propagation efficiency.
2. **Upper Bound Consideration:** Conversely, an excessive k can also impact computational efficiency. As demonstrated in our findings (Sec. 3.1), the effectiveness of content propagation diminishes with increasing perspective distance between views. Including too many distant views in the sampling set may introduce redundant computations without contributing meaningful priors, potentially diluting the consistency of the inpainting results.

In practical applications, the selection of k should prioritize addressing the lower bound constraint to ensure connected graph components and effective content propagation. The upper bound consideration is less critical due to our consistency verification mechanism, which filters out incon-

sistent inpainting candidates during the refinement stage. While a larger k might affect computational efficiency, it does not significantly compromise the final inpainting quality thanks to this verification safeguard.

m for inpainted candidates in consistency verification. To achieve consistency verification, we need to generate multiple (m) inpainted candidates for each adjacent image. Thanks to our inpaint content propagation before images inpainting, most inpainted candidates are highly consistent with the anchor image. However, due to the randomness attribute of diffusion model, the consistency verification is still really important to enhance the multi-view consistency of 3D inpainting, which can be seen from our experiment results in ablation study Sec. 4. To avoid the high time consumption overhead, we set $m = 4$ across all our experiments.

η for dual-feature consistency score. In our consistency verification mechanism, we propose a weighted dual-feature consistency score that combines texture and depth features, formulated as $S = \eta S_{rgb} + (1 - \eta)S_{depth}$, where S_{rgb} and S_{depth} represent the respective similarity scores. Through extensive experiments, we empirically set $\eta = 0.7$ to prioritize fine-grained texture consistency while maintaining the benefits of geometric constraints. This weighting strategy reflects our emphasis on texture features, which directly capture the visual quality of inpainted regions, while also leveraging depth information as a valuable complementary cue. The relatively lower weight assigned to depth similarity helps mitigate potential errors introduced by the pretrained depth estimator in challenging scenes, while still providing crucial geometric constraints. This is particularly important given our use of a lightweight ResNet18 for texture feature extraction, which, while computationally efficient, may occasionally struggle to discriminate subtle texture differences under low-light conditions or in regions with repetitive patterns. In such scenarios, the depth features computed from colored depth maps demonstrate superior discriminative power, contributing significantly to the robustness of our consistency verification. Our experiments show that this balanced weighting approach provides consistent and reliable performance across diverse scenes without requiring scene-specific parameter tuning, effectively combining the strengths of both texture and geometric features while maintaining computational efficiency.

Notably, the consistent performance achieved with these empirically determined hyper-parameters (τ, k, m and η), without scene-specific tuning, underscores the robustness and practical utility of our method, making it readily applicable to real-world scenarios while maintaining its effectiveness.

Algorithm 1 Adaptive sampling algorithm

Input: perspective graph $\mathbf{G}$, adjacent hyper-parameter k, Iterations $iters$, threshold of consistency score T_s
Initialize: Anchor set $\mathcal{A} \leftarrow \emptyset$, Inpainted set $\mathcal{P} \leftarrow \emptyset$, Masked image indices set $\mathcal{I} = I_i, i \in \{1, ..., N\}$
Select initial anchor I_0 randomly from $\mathbf{G}$
while step $< iters$ & $\mathcal{I} \neq \emptyset$ **do**
    $\mathcal{I}_{adj} \leftarrow k$ nearest neighbors of I_t from $\mathbf{G}$
    Update $\mathcal{A} \leftarrow \mathcal{A} \cup \{I_t\} \cup \mathcal{I}_{adj}[: \lfloor k/2 \rfloor]$
    $\mathcal{I}_{adj} \leftarrow \mathcal{I}_{adj} \cap \mathcal{I} \cup \{I_t\}$
    if $\mathcal{I}_{adj} \neq \emptyset$ **then**
        $\mathcal{I}'_{adj} \leftarrow$ inpainted $\mathcal{I}_{adj}$
        $\mathcal{S}_{adj} \leftarrow$ consistency score of inpainted $\mathcal{I}'_{adj}$
        Update $\mathcal{I} \leftarrow \mathcal{I} \setminus \mathcal{I}'_{adj}[\mathcal{S}_{adj} > T_s]$
        Update $\mathcal{P} \leftarrow \mathcal{P} \cup \mathcal{I}'_{adj}$
        Optimize 3D Gaussains with $\mathcal{P}$
    end if
    Select $I_t \leftarrow$ random sample from $(\mathcal{I} \setminus \mathcal{A}) \cap \mathcal{P}$
end while
while step $< iters$ **do**
    Optimize 3D Gaussains with $\mathcal{P}$
end while

B.3. Adaptive Sampling Algorithm

We formalize the adaptive sampling algorithm detailed in Sec. 3.3 into pseudo-code format (Algorithm 1) with the following key implementation details:

- **State 6:** We maintain an anchor image set $\mathcal{A}$ to prevent repetitive selection of previous anchors. Additionally, the $k/2$ adjacent images of any anchor are excluded from future anchor selection to avoid local saturation in the perspective graph, ensuring comprehensive coverage of the view space.
- **State 7:** The masked image set $\mathcal{I}$ adaptively tracks views requiring inpainting or refinement. Following our adaptive strategy described in Sec. 3.3, images with lower consistency scores remain in this set for subsequent refinement iterations.
- **State 11:** Images achieving consistency scores above the empirically determined threshold $T_s = 0.9$ are removed from the masked set $\mathcal{I}$, effectively identifying well-inpainted views that require no further processing.
- **State 12:** An inpainted image set $\mathcal{P}$ is maintained to track all processed views throughout the algorithm's execution.
- **State 15:** New anchor images are selected exclusively from the inpainted set $\mathcal{P}$, excluding both previous anchors ($\mathcal{A}$) and well-inpainted views. This ensures effective propagation of high-quality inpainting results while avoiding redundant processing.

C. Quantitative and Qualitative Results

We provide comprehensive scene-specific evaluation results to complement the average performance metrics presented in our main comparisons against state-of-the-art baseline methods. The detailed quantitative results for individual scenes are presented in Tab. 1, Tab. 2 and Tab. 3 for PSNR metrics, Tab. 4, Tab. 5 and Tab. 6 for SSIM metrics, and Tab. 7, Tab. 8 and Tab. 9 for LPIPS metrics across NeRF Blender dataset, SPIn-NeRF dataset and NeRFiller dataset. In addition, we discuss the performance variation with regards to mask types and area ratios in Tab. 10 (please find the examples of different mask types in main part Fig. 7), which demonstrate that performance variation is primarily influenced by mask type at reasonable ratios. PAInpainter performs better on real-world scenes and textured object scenes with more priors (SPIn-NeRF & NeRFiller) despite larger mask ratios, compared to synthetic Blender scenes. This also reveals the 2D diffusion inpainting model's input pattern sensitivity.

We provide more supplementary qualitative results to show the details results of PAInpainter and other state-of-the-art approaches in Fig. 1, Fig. 2, Fig. 3 and Fig. 4.

	ficus	ship	lego	drums	hotdog	microphone	materials	chair	**Avg.** ↑
Masked 3DGS	9.89	13.81	12.04	11.65	12.92	9.90	11.36	11.03	11.57
SD2	20.92	20.22	19.68	17.88	22.69	17.64	22.14	22.18	20.42
MVinpainter	19.56	23.03	17.05	16.14	25.41	12.92	20.15	21.10	19.42
Grid Prior + DU	23.34	22.97	21.33	20.31	24.95	22.22	22.17	24.91	22.77
NeRFiller	26.86	24.32	22.73	21.63	24.89	20.61	20.12	25.05	23.27
PAInpainter (ours)	25.39	24.29	21.70	21.33	26.05	23.28	24.84	26.64	24.19

Table 1. PSNR 3D inpainting results for NeRF Blender dataset.

	1(bench)	2(tree)	3(backpack)	4(stairs)	7(well)	9(wall)	10(yard)	12(garden)	book	trash	**Avg.** ↑
Masked 3DGS	12.07	12.77	11.88	9.86	12.74	13.98	16.89	12.00	14.74	17.72	13.46
SD2	22.68	24.57	21.69	25.96	26.82	21.35	22.08	21.38	23.42	24.89	23.48
MVinpainter	22.94	23.25	20.85	28.15	28.35	23.41	24.17	23.93	26.72	26.18	24.80
Grid Prior + DU	22.06	24.69	21.25	28.28	27.89	24.43	24.61	22.04	28.55	28.07	25.19
NeRFiller	23.08	24.74	21.76	28.03	26.42	24.66	24.11	23.72	28.10	27.41	25.20
PAInpainter (ours)	23.73	24.93	21.26	29.39	28.59	25.05	25.40	24.31	29.25	28.41	26.03

Table 2. PSNR 3D inpainting results for NeRF SPIn-NeRF dataset.

	billiards	norway	drawing	office	turtle	kitchen	bear	boot	cat	dumptruck	**Avg.** ↑
Masked 3DGS	10.26	14.58	14.32	12.06	18.91	11.94	13.13	9.76	15.70	8.82	12.95
SD2	25.41	20.81	22.99	24.99	19.16	25.00	19.65	14.72	16.50	14.37	20.36
MVinpainter	28.43	24.93	22.73	22.64	20.35	20.77	22.24	14.66	17.73	16.84	21.13
Grid Prior + DU	29.76	27.76	27.95	31.87	22.61	27.91	26.40	26.63	23.85	24.99	26.97
NeRFiller	27.32	25.00	27.35	25.75	20.77	25.31	20.90	13.88	20.33	16.86	22.35
PAInpainter (ours)	29.43	30.72	29.26	33.14	30.29	30.39	28.33	29.55	26.06	27.90	29.51

Table 3. PSNR 3D inpainting results for NeRFiller dataset.

	ficus	ship	lego	drums	hotdog	microphone	materials	chair	**Avg.** ↑
Masked 3DGS	0.85	0.75	0.85	0.84	0.85	0.84	0.84	0.85	0.83
SD2	0.91	0.86	0.89	0.88	0.92	0.93	0.93	0.91	0.90
MVinpainter	0.80	0.82	0.76	0.74	0.90	0.79	0.87	0.84	0.81
Grid Prior + DU	0.93	0.87	0.90	0.91	0.93	0.96	0.94	0.93	0.92
NeRFiller	0.94	0.88	0.92	0.91	0.94	0.93	0.92	0.93	0.92
PAInpainter (ours)	0.94	0.87	0.91	0.91	0.94	0.95	0.95	0.93	0.92

Table 4. SSIM 3D inpainting results for NeRF Blender dataset.

	1(bench)	2(tree)	3(backpack)	4(stairs)	7(well)	9(wall)	10(yard)	12(garden)	book	trash	**Avg.** ↑
Masked 3DGS	0.28	0.13	0.31	0.64	0.50	0.18	0.50	0.12	0.71	0.77	0.41
SD2	0.61	0.72	0.73	0.83	0.81	0.54	0.78	0.65	0.81	0.80	0.73
MVinpainter	0.58	0.67	0.70	0.83	0.84	0.64	0.80	0.81	0.76	0.81	0.74
Grid Prior + DU	0.57	0.71	0.74	0.87	0.86	0.68	0.86	0.78	0.91	0.89	0.79
NeRFiller	0.60	0.72	0.75	0.86	0.85	0.71	0.84	0.80	0.89	0.87	0.79
PAInpainter (ours)	0.63	0.75	0.74	0.88	0.85	0.70	0.89	0.83	0.91	0.91	0.81

Table 5. SSIM 3D inpainting results for SPIn-NeRF dataset.

	billiards	norway	drawing	office	turtle	kitchen	bear	boot	cat	dumptruck	**Avg.** ↑
Masked 3DGS	0.68	0.66	0.66	0.72	0.87	0.73	0.87	0.77	0.87	0.74	0.76
SD2	0.86	0.83	0.77	0.87	0.86	0.79	0.91	0.85	0.87	0.82	0.84
MVinpainter	0.85	0.75	0.65	0.83	0.85	0.66	0.89	0.82	0.85	0.81	0.80
Grid Prior + DU	0.92	0.91	0.86	0.95	0.91	0.86	0.96	0.95	0.94	0.93	0.92
NeRFiller	0.89	0.88	0.86	0.90	0.89	0.80	0.92	0.85	0.90	0.87	0.88
PAInpainter (ours)	0.92	0.93	0.88	0.95	0.96	0.90	0.96	0.96	0.94	0.95	0.94

Table 6. SSIM 3D inpainting results for NeRFiller dataset.

	ficus	ship	lego	drums	hotdog	microphone	materials	chair	**Avg.** ↓
Masked 3DGS	0.21	0.26	0.17	0.18	0.19	0.19	0.17	0.18	0.19
SD2	0.07	0.13	0.09	0.11	0.09	0.09	0.05	0.08	0.09
MVinpainter	0.20	0.12	0.19	0.21	0.08	0.35	0.08	0.15	0.17
Grid Prior + DU	0.07	0.12	0.09	0.10	0.08	0.05	0.06	0.07	0.08
NeRFiller	0.06	0.13	0.08	0.09	0.08	0.08	0.08	0.08	0.09
PAInpainter (ours)	0.07	0.13	0.09	0.09	0.07	0.06	0.04	0.06	0.08

Table 7. LPIPS 3D inpainting results for NeRF Blender dataset.

	1(bench)	2(tree)	3(backpack)	4(stairs)	7(well)	9(wall)	10(yard)	12(garden)	book	trash	**Avg.** ↓
Masked 3DGS	0.51	0.55	0.42	0.30	0.34	0.65	0.24	0.61	0.27	0.16	0.40
SD2	0.37	0.24	0.13	0.14	0.11	0.47	0.13	0.39	0.16	0.13	0.23
MVinpainter	0.42	0.34	0.19	0.11	0.10	0.31	0.13	0.17	0.17	0.18	0.21
Grid Prior + DU	0.44	0.35	0.18	0.12	0.14	0.26	0.11	0.21	0.08	0.08	0.20
NeRFiller	0.37	0.22	0.13	0.11	0.13	0.26	0.10	0.18	0.10	0.09	0.17
PAInpainter (ours)	0.35	0.19	0.17	0.08	0.13	0.19	0.09	0.15	0.08	0.08	0.15

Table 8. LPIPS 3D inpainting results for SPIn-NeRF dataset.

	billiards	norway	drawing	office	turtle	kitchen	bear	boot	cat	dumptruck	**Avg.** ↓
Masked 3DGS	0.33	0.32	0.33	0.28	0.21	0.29	0.19	0.30	0.21	0.33	0.28
SD2	0.11	0.17	0.19	0.16	0.17	0.16	0.11	0.21	0.20	0.26	0.17
MVinpainter	0.09	0.17	0.21	0.18	0.15	0.26	0.09	0.24	0.19	0.24	0.18
Grid Prior + DU	0.10	0.11	0.16	0.09	0.20	0.15	0.06	0.10	0.14	0.15	0.13
NeRFiller	0.10	0.13	0.14	0.14	0.16	0.15	0.08	0.24	0.15	0.22	0.15
PAInpainter (ours)	0.07	0.07	0.12	0.08	0.07	0.08	0.05	0.06	0.11	0.10	0.08

Table 9. LPIPS 3D inpainting results for NeRFiller dataset.

	Mask types				Avg. mask area ratios		
	object-centric removal	large indoor missing region	object-centric large missing region	multiple disjoint missing regions	$\leq 10\%$	$10\% \sim 20\%$	$20\% \sim 30\%$
PSNR	26.43	30.61	25.10	28.23	26.18	25.59	29.56
SSIM	0.817	0.920	0.931	0.953	0.823	0.907	0.924
LPIPS	0.145	0.085	0.077	0.077	0.147	0.086	0.090
FID	124.9	96.0	100.3	76.9	117.6	101.4	104.7

Table 10. Performance variation upon different mask types/ratios (All 28 scenes)

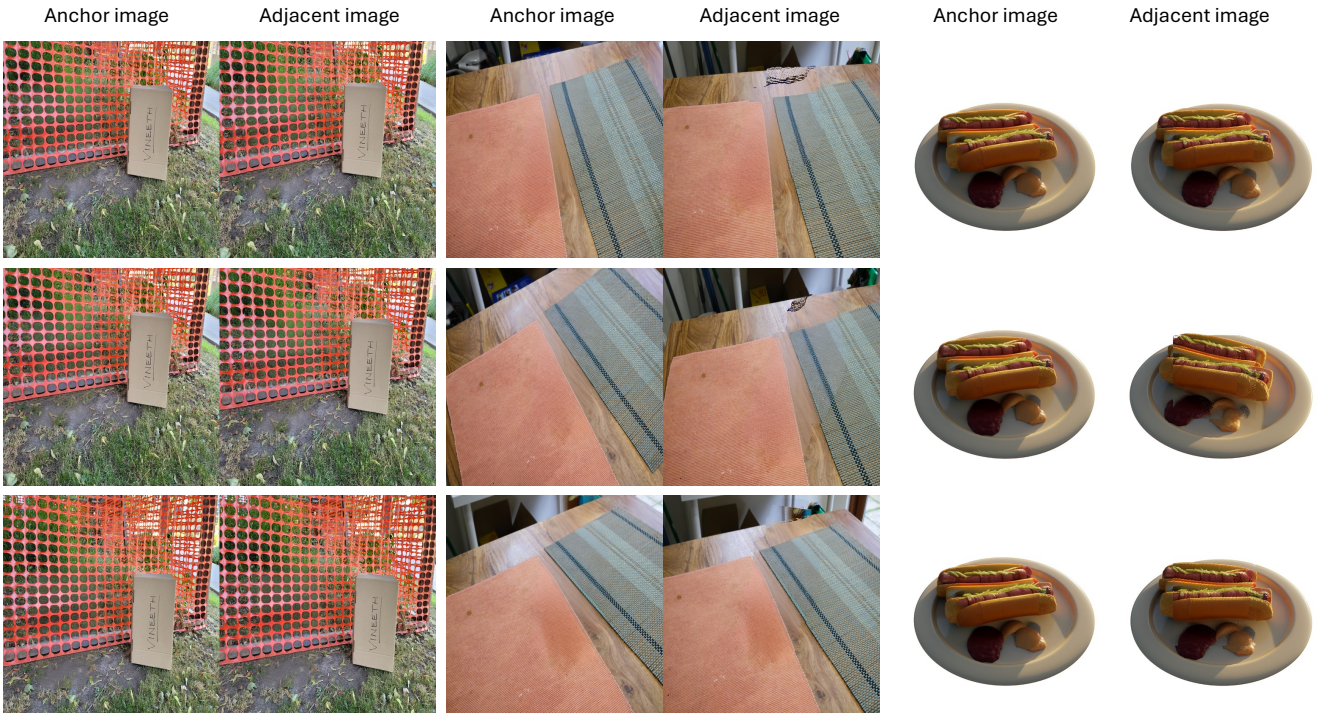

Figure 1. The inpaint content propagation between anchor images and corresponding adjacent images. With our perspective graph sampling strategy, the anchor image provides sufficient and accurate prior to adjacent images to guide consistent multi-view inpainting.

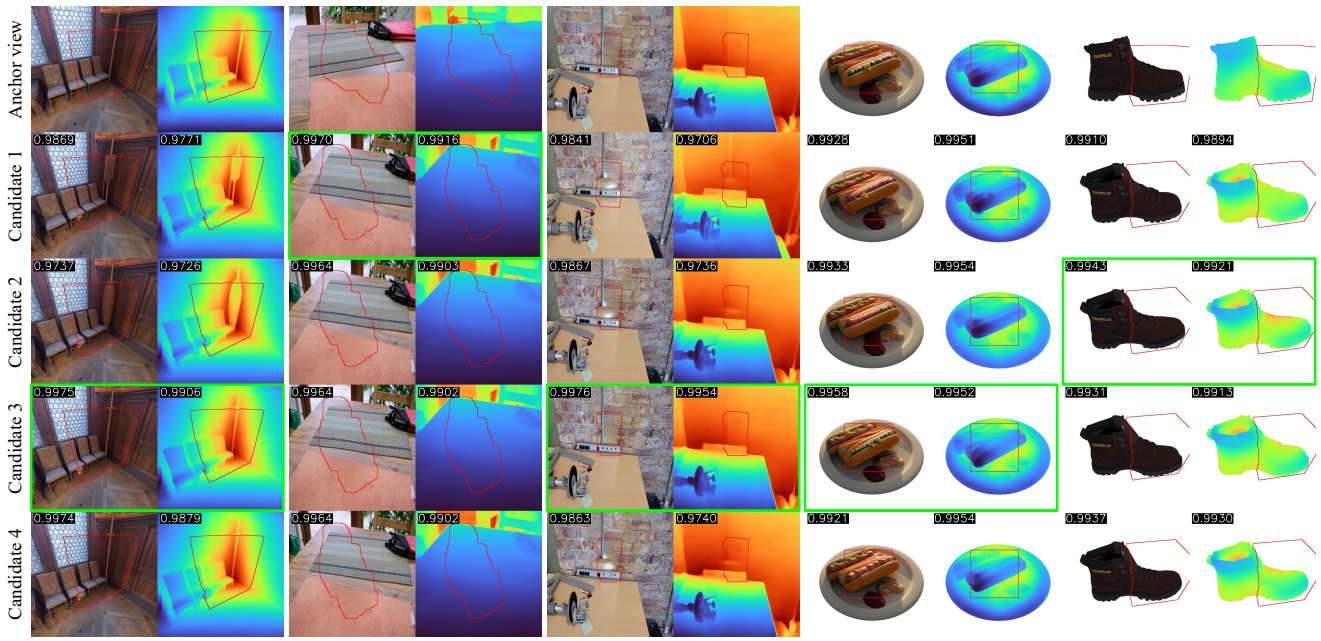

Figure 2. Visualization for consistency verification. Red contours delineate mask boundaries and green boxes highlight top-scoring candidates selected for 3DGS optimization. The upper-left number of each candidate represents the consistency score. This module reliably identifies inpainted regions exhibiting both textural and geometric consistency (zoom for details), enhancing performance and robustness.

Figure 3. Details comparison in renderings of inpainted 3D scene, among PAInpainter, GridPrior+DU, SD2

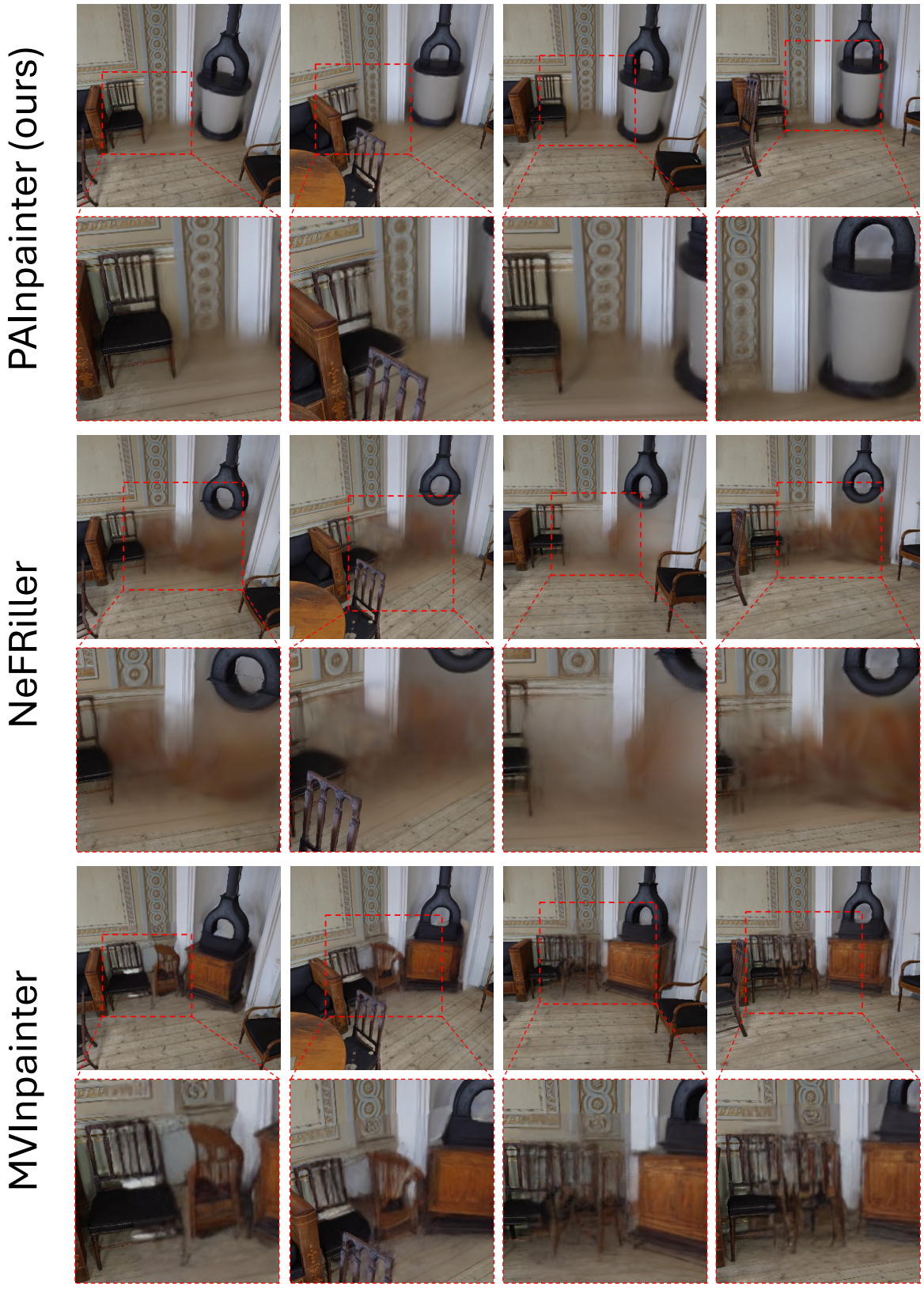

Figure 4. Details comparison in renderings of inpainted 3D scene, among PAInpainter, NeRFiller, MVInpainter