OccludeNeRF: Geometry-aware 3D Scene Inpainting with Collaborative Score Distillation in NeRF

Supplementary Material

7. Analysis on Multi-view CDS

We have proposed the core distillation sampling strategy in our main manuscript, namely Equation (8). In this Supplementary section, we discuss how Equation (8) helps propagate the information from limited views to the collaborative update of the NeRF parameters.

7.1. Kernel-Weighted Noise Prediction

7.1.1. Noise Prediction

For each view i, the noise prediction $\hat{\epsilon}^{(i)}$ is computed as a kernel-weighted combination of noise predictions from all views:

$$\hat{\epsilon}^{(i)} = \frac{1}{N}\sum_{j=1}^{N} k(\mathbf{z}^{(i)}, \mathbf{z}^{(j)})\epsilon^{(j)} \tag{11}$$

where:

- $\epsilon^{(j)}$ is the noise prediction for view j.
- $k(\mathbf{z}^{(i)}, \mathbf{z}^{(j)})$ is the kernel function, which measures similarity between the latents $\mathbf{z}^{(i)}$ and $\mathbf{z}^{(j)}$.

  The kernel $k(\mathbf{z}^{(i)}, \mathbf{z}^{(j)})$ ensures that:
- Views j are from the rendered set as view i. Those from views j with relevant information (e.g., visible occluded areas) contribute more strongly to the noise prediction $\hat{\epsilon}^{(i)}$ of view i.
- Occluded view i to incorporate details from views j where the occluded area is visible.

7.2. Loss Function and Gradient

7.2.1. Loss Function

The loss for each view i is defined as:

$$L^{(i)} = \hat{\epsilon}^{(i)} - \epsilon_{\text{gt}}^{(i)} \tag{12}$$

where $\epsilon_{\text{gt}}^{(i)}$ is the ground-truth noise for view i. Substituting $\hat{\epsilon}^{(i)}$, the loss becomes:

$$L^{(i)} = \frac{1}{N}\sum_{j=1}^{N} k(\mathbf{z}^{(i)}, \mathbf{z}^{(j)})\epsilon^{(j)} - \epsilon_{\text{gt}}^{(i)} \tag{13}$$

7.2.2. Gradient of the Loss

The gradient of this loss with respect to the latent $\mathbf{z}^{(i)}$ is:

$$\nabla_{\mathbf{z}^{(i)}} L^{(i)} = \frac{1}{N}\sum_{j=1}^{N} \nabla_{\mathbf{z}^{(i)}} \left(k(\mathbf{z}^{(i)}, \mathbf{z}^{(j)})\right)\epsilon^{(j)} \tag{14}$$

For a Gaussian RBF kernel with scale h:

$$k(\mathbf{z}^{(i)}, \mathbf{z}^{(j)}) = \exp\left(-\frac{1}{h}\|\mathbf{z}^{(i)} - \mathbf{z}^{(j)}\|_2^2\right) \tag{15}$$

the gradient is:

$$\nabla_{\mathbf{z}^{(i)}} k(\mathbf{z}^{(i)}, \mathbf{z}^{(j)}) = -\frac{2}{h}(\mathbf{z}^{(i)} - \mathbf{z}^{(j)})k(\mathbf{z}^{(i)}, \mathbf{z}^{(j)}) \tag{16}$$

Substituting this into $\nabla_{\mathbf{z}^{(i)}} L^{(i)}$, we get:

$$\nabla_{\mathbf{z}^{(i)}} L^{(i)} = \frac{1}{N}\sum_{j=1}^{N} \left(-\frac{2}{h}(\mathbf{z}^{(i)} - \mathbf{z}^{(j)})k(\mathbf{z}^{(i)}, \mathbf{z}^{(j)})\right)\epsilon^{(j)} \tag{17}$$

This gradient shows how information propagates between views i and j, with the kernel $k(\mathbf{z}^{(i)}, \mathbf{z}^{(j)})$ modulating the strength of interaction.

7.3. NeRF Parameter Updates

7.3.1. Gradient Propagation

The latent updates are propagated to the NeRF parameters θ through backpropagation. The total gradient for θ is:

$$\nabla_\theta L = \sum_{i=1}^{N} \nabla_\theta L^{(i)} \tag{18}$$

Using the chain rule:

$$\nabla_\theta L^{(i)} = \nabla_{\mathbf{z}^{(i)}} L^{(i)} \cdot \nabla_\theta \mathbf{z}^{(i)} \tag{19}$$

Substituting $\nabla_{\mathbf{z}^{(i)}} L^{(i)}$ from above, we get:

$$\nabla_\theta L = \sum_{i=1}^{N} \nabla_\theta \mathbf{z}^{(i)} \cdot \left(\frac{1}{N}\sum_{j=1}^{N} \left(-\frac{2}{h}(\mathbf{z}^{(i)} - \mathbf{z}^{(j)})k(\mathbf{z}^{(i)}, \mathbf{z}^{(j)})\right)\epsilon^{(j)}\right) \tag{20}$$

7.4. Role of the Two Terms in the Kernel Update

The kernel update involves two key terms:

$$\nabla_{\mathbf{z}^{(i)}} k(\mathbf{z}^{(i)}, \mathbf{z}^{(j)}) = -\frac{2}{h}(\mathbf{z}^{(i)} - \mathbf{z}^{(j)})k(\mathbf{z}^{(i)}, \mathbf{z}^{(j)}) \tag{21}$$

and:

$$k(\mathbf{z}^{(i)}, \mathbf{z}^{(j)}). \tag{22}$$

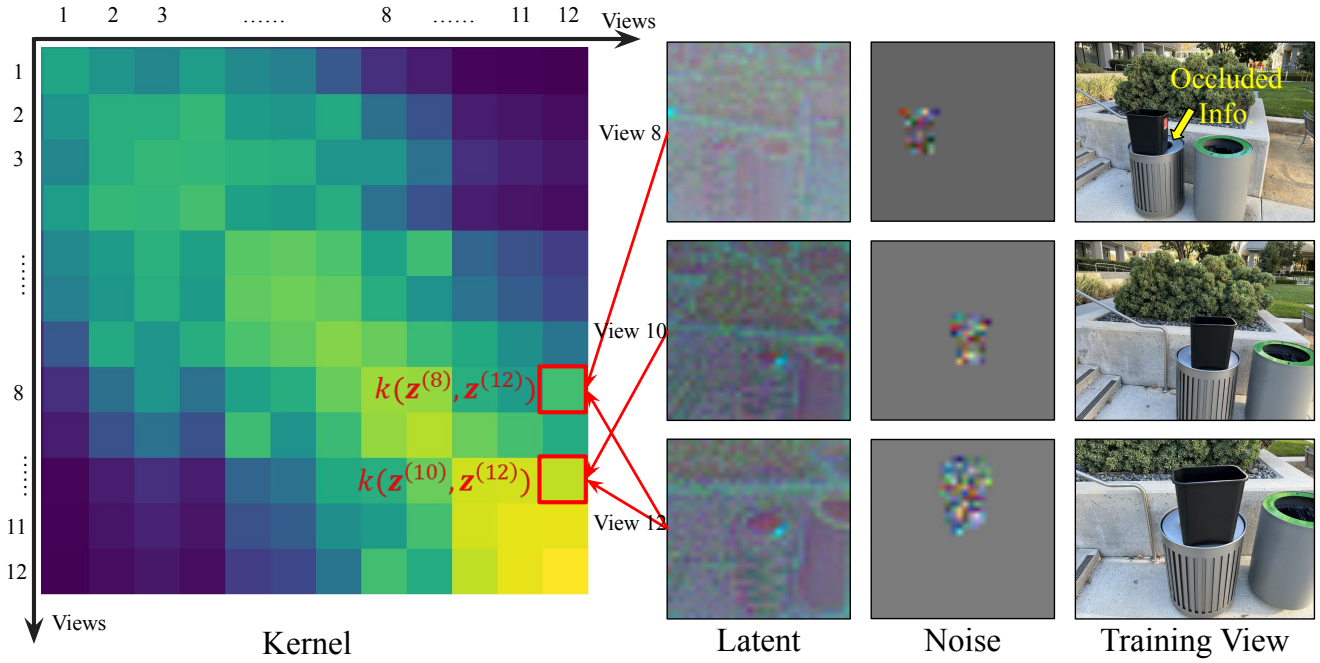


Figure 7. Illustration of the effect of our CDS kernel in one iteration. The heatmap of the kernel is on the left. The warmer the color, the higher the kernel value. We can see the corresponding views on the right-hand side. The closer (in the latent space) the views are, the higher the correspondence value in the kernel (View 10 & 12). Meanwhile, views with further distance but containing important information can also be related to the update (e.g. the kernel has a relatively high value between (View 8 & 10). In this way, the information about the occluded area (e.g. View 8, where the hole of the left trash bin is visible) can be propagated to the update of other views (e.g. View 12, where the hole is completely occluded).

First Term: Gradient of the Kernel ($\nabla_{\mathbf{z}^{(i)}} k(\mathbf{z}^{(i)}, \mathbf{z}^{(j)})$)

This term serves several critical purposes in the kernel-based updates:

1. **Repulsive Force to Maintain Diversity**:

$$\nabla_{\mathbf{z}^{(i)}} k(\mathbf{z}^{(i)}, \mathbf{z}^{(j)}) = -\frac{2}{h}(\mathbf{z}^{(i)} - \mathbf{z}^{(j)}) k(\mathbf{z}^{(i)}, \mathbf{z}^{(j)}) \quad (23)$$

The term $(\mathbf{z}^{(i)} - \mathbf{z}^{(j)})$ computes the directional vector pointing from $\mathbf{z}^{(j)}$ to $\mathbf{z}^{(i)}$. The negative sign ensures that the gradient drives $\mathbf{z}^{(i)}$ away from $\mathbf{z}^{(j)}$, creating a repulsive effect between similar latents. This repulsion prevents all latents from collapsing into a single representation, ensuring sufficient diversity among the latent representations for different views.

2. **Propagation of Occlusion Information**: When a view j contains visible information about an occluded area, its latent $\mathbf{z}^{(j)}$ contributes gradients to the update of $\mathbf{z}^{(i)}$ through this term:

$$\nabla_{\mathbf{z}^{(i)}} L^{(i)} = \frac{1}{N} \sum_{j=1}^{N} \nabla_{\mathbf{z}^{(i)}} k(\mathbf{z}^{(i)}, \mathbf{z}^{(j)}) \epsilon^{(j)}. \quad (24)$$

If $\mathbf{z}^{(j)}$ is close to $\mathbf{z}^{(i)}$, the kernel $k(\mathbf{z}^{(i)}, \mathbf{z}^{(j)})$ will be large, amplifying the influence of $\mathbf{z}^{(j)}$ on $\mathbf{z}^{(i)}$. This ensures that visible details in $\mathbf{z}^{(j)}$ are propagated into the occluded representation $\mathbf{z}^{(i)}$.

3. **Modulation by Kernel Weight**: The term $k(\mathbf{z}^{(i)}, \mathbf{z}^{(j)})$ modulates the strength of the gradient, ensuring that only nearby latents significantly influence $\mathbf{z}^{(i)}$. Mathematically, the magnitude of the gradient is proportional to the similarity between $\mathbf{z}^{(i)}$ and $\mathbf{z}^{(j)}$, as measured by $k(\mathbf{z}^{(i)}, \mathbf{z}^{(j)})$.

Second Term: Kernel Weight ($k(\mathbf{z}^{(i)}, \mathbf{z}^{(j)})$)

This term determines how much influence view j has on view i in the kernel-weighted noise prediction:

$$\hat{\epsilon}^{(i)} = \frac{1}{N} \sum_{j=1}^{N} k(\mathbf{z}^{(i)}, \mathbf{z}^{(j)}) \epsilon^{(j)}. \quad (25)$$

1. **Weighted Contribution of Nearby Views**: The kernel $k(\mathbf{z}^{(i)}, \mathbf{z}^{(j)})$ assigns higher weights to views j with similar latents $\mathbf{z}^{(j)}$ to $\mathbf{z}^{(i)}$, ensuring that these views have a stronger influence on the noise prediction for view i. This is

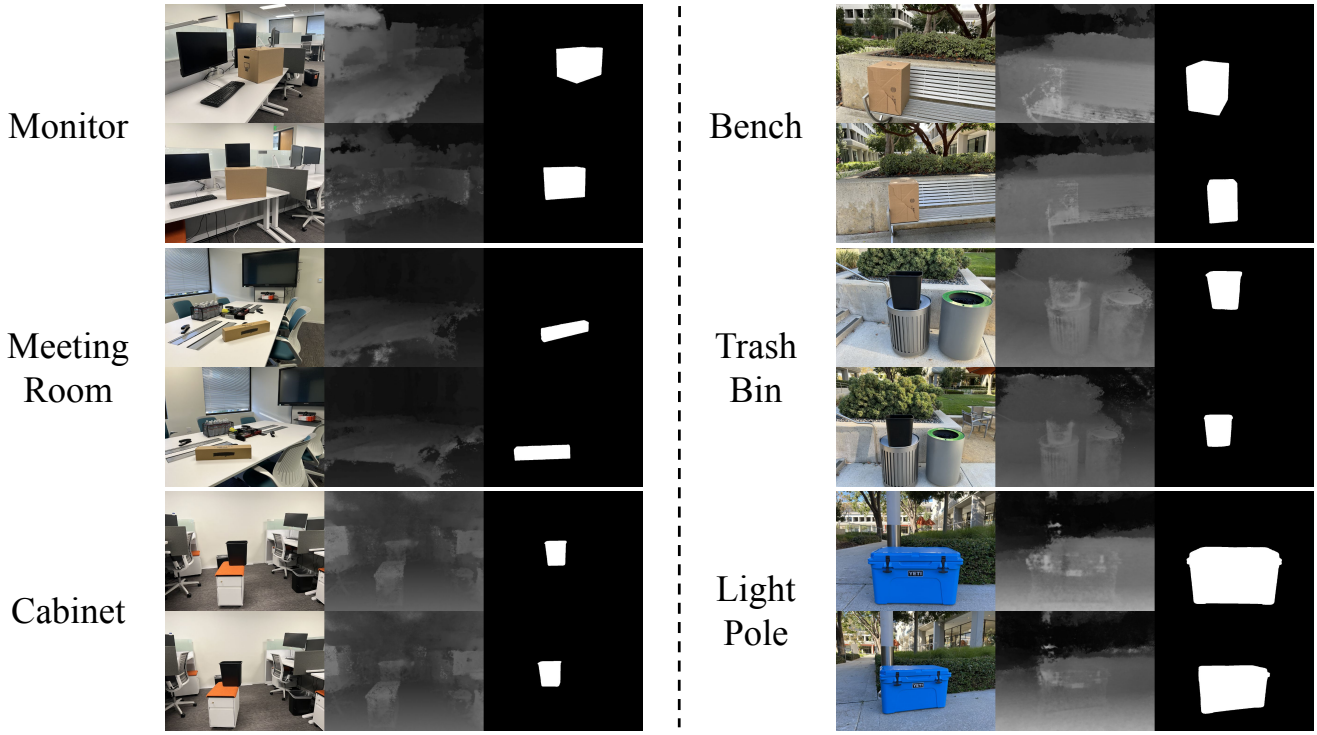


Figure 8. Visualization of our customized dataset. For each scene, we set up an obstacle blocking an area and mask the obstacle for inpainting tasks. We collect testing views and training views with RGB images and masks. We also generate the pseudo-depth maps (visualized as the disparity maps) corresponding to each view.

particularly critical when $\mathbf{z}^{(j)}$ contains visible information about an occluded area in $\mathbf{z}^{(i)}$, as the kernel amplifies the contribution of $\mathbf{z}^{(j)}$.

2. **Occlusion-Aware Updates**: For occluded areas, $\mathbf{z}^{(j)}$ from views where the occlusion is visible dominates the noise prediction for $\mathbf{z}^{(i)}$, effectively propagating information about the occluded area across views:

$$\hat{\epsilon}^{(i)} \approx \frac{\sum_j^N k(\mathbf{z}^{(i)}, \mathbf{z}^{(j)})\epsilon^{(j)}}{\sum_j^N k(\mathbf{z}^{(i)}, \mathbf{z}^{(j)})}. \tag{26}$$

This weighted update ensures that the occluded representation $\mathbf{z}^{(i)}$ aligns with the visible views.

3. **Locality of Influence**: The kernel decays rapidly with distance in the latent space:

$$k(\mathbf{z}^{(i)}, \mathbf{z}^{(j)}) = \exp\left(-\frac{1}{h}\|\mathbf{z}^{(i)} - \mathbf{z}^{(j)}\|_2^2\right). \tag{27}$$

As $\|\mathbf{z}^{(i)} - \mathbf{z}^{(j)}\|$ increases, $k(\mathbf{z}^{(i)}, \mathbf{z}^{(j)}) \to 0$, ensuring that only nearby latents significantly influence the updates. In this way, we avoid distillation of 2D prior from views that are too far away, which may result in inconsistent 2D inpainting results, as also pointed out by prior work [56].

Combined Functionality of the Two Terms

The combined effect of the two terms is as follows:

- The **first term** ($\nabla_{\mathbf{z}^{(i)}} k$) ensures that information propagates between views and prevents collapse by introducing repulsive forces.
- The **second term** (k) amplifies contributions from relevant views, particularly those with visible occluded areas, ensuring effective information sharing.

Together, these terms propagate occlusion information across views, align latent representations, and maintain diversity in the latent space, enabling robust NeRF training.

Why Occlusion Information Propagates to θ

1. **Kernel-Based Weighting:** The kernel $k(\mathbf{z}^{(i)}, \mathbf{z}^{(j)})$ ensures that visible views j contribute more strongly to occluded views i, propagating occlusion details across the latent space, as shown in Figure 7

2. **Collaborative Updates to Latents:** The gradient $\nabla_{\mathbf{z}^{(i)}} k(\mathbf{z}^{(i)}, \mathbf{z}^{(j)})$ drives latent $\mathbf{z}^{(i)}$ of occluded views to align with $\mathbf{z}^{(j)}$ of visible views.

3. **Backpropagation to NeRF:** The updated latents $\mathbf{z}^{(i)}$ are used to refine the NeRF parameters θ, enabling the model to represent occluded areas consistently across all views.

8. Dataset Building

In this section, we introduce our procedure and details for constructing the Occlude-NeRF dataset.

8.1. Scene Setup

We collected data from six scenes in total, with three indoors and three outdoors, as shown in Figure 8. We name the three indoor scenes: *Cabinet*, *Monitor*, and *Meeting Room*, and the three outdoor scenes: *Bench*, *Trash Bin*, and *Light Pole*. The specific scene description is listed below:

- *Cabinet*: an office workspace with a symmetrical arrangement of two cubicles on either side. Each cubicle includes a white desk, a chair with a gray backrest, and an orange seat cushion. A black trash bin is placed on top of a mobile pedestal with an orange cushion. The black trash bin is masked.
- *Monitor*: an office workspace with a white desk and a cardboard box placed in the center. On the desk are two Dell monitors (one visible and turned off), a black keyboard, and a desk lamp on the left. The cardboard box is masked.
- *Meeting Room*: an indoor office meeting room with a white conference table surrounded by teal office chairs. On the table are various items, including a long cardboard box, staplers, and a black organizer containing stationery such as pens, highlighters, and sticky notes. The cardboard box is masked.
- *Bench*: a cardboard box placed on a silver metal bench in an outdoor area. The bench is positioned next to a concrete planter filled with green shrubs and small rocks. The cardboard box is masked.
- *Trash Bin*: two outdoor trash bins in front of a concrete planter with green shrubs. The left bin is metallic with vertical slits, and a black rectangular container is placed on its circular opening. The right bin is smooth, gray, and labeled "Compost" with a green rim. The black container is masked.
- *Light Pole*: a bright blue cooler placed on a concrete sidewalk, next to a metal pole and a neatly trimmed hedge. The cooler is masked.

8.2. Data Collection

For each scene, we collect 60 training views and 40 testing views. For each training view, we obtain a mask by prompting a point at the object to mask, using the Segment Anything Model (SAM) [23]. Each mask is dilated with a 3×3 kernel for 3 iterations. To obtain relatively accurate camera pose estimations, we mark the objects' location in the scene with a marker, place the object to take one image, and remove the object for another while the camera remains static. In this way, we obtain a testing view with and without the object in the scene. We then put the object back according to the mark and move the camera for other views. We conducted this procedure because we found prior work's [36] method for estimating camera poses resulted in unstable accuracy since they use COLMAP [43, 44] to perform structure from motion with images with and without objects. Therefore, in our case, we obtain extra images with objects for the testing views, so that we can estimate the poses for both training views and testing views together. We then obtain the pseudo depth maps for each view following SPIn-NeRF [36]. The collected dataset samples can be found in Figure 8.

9. Hyperparameter Details

In this section, we elucidate the hyperparameters we have used in our experiments, as well as some findings exploring the hyperparameters.

9.1. Implementation

We implement our Occlude-NeRF method on 2 NVIDIA H100 GPUs, trained for 10,000 iterations for each scene with the Adam optimizer with a learning rate of $1e-4$ scheduled with a cosine annealing scheduler (max number of iterations: 50 and min learning rate:0). For the distillation sampling, we follow prior work [6, 56] and choose timesteps uniformly increasing with the training from $t_{min} = 0.02$ to $t_{max} = 0.98$. For the classifier-free guidance [15], we choose a uniform value for all scenes to see the generalizability of our methods, in contrast to MVIP-NeRF [6]. Specifically, we set:

$$\hat{\epsilon} = \epsilon_{uncond} + \gamma \times (\epsilon_{text} - \epsilon_{uncond}) \tag{28}$$

where $\hat{\epsilon}$ is the final noise prediction, ϵ_{uncond} and ϵ_{text} are the noise prediction with no condition and conditioned by text, respectively, and γ is the guidance scale, which we set uniformly $\gamma = 7.5$. During training, the size of latent z is set to 256×256 for collaborative distillation and 512×512 for geometry distillation due to GPU RAM limits. For each iteration, we set the batch (of rays) size to 1024 and the number of samples along the ray to 32. Additionally, we set the number of samples for fine networks to 32. We test with three different numbers of the Grid-based Denoising $M = 1, 4, 8$ and choose $M = 4$ to balance performance and training time. The textual prompts to diffusion models are listed in Table 3:

9.2. Exploratory Study on Hyperparameters

In addition to the hyperparameter choices we have reported above, we explore the hyperparameter space and report interesting findings in this subsection.

9.2.1. Noise Scheduling

We test two noise scheduling methods. Namely, we first implemented a random sampling schedule, where a random

(a) Random Timestep Scheduling (b) Progressive Timestep Scheduling

Figure 9. Comparison between (a) random timestep scheduling and (b) progressive timestep scheduling. We can easily observe the better convergence of shape and appearance of the latter.

noise timestep between t_{min} and t_{max} is chosen. We then implemented a progressive sampling schedule similar to [6, 65]:

$$t = t_{max} - (t_{max} - t_{min}) * iter/max\_iter) \quad (29)$$

where $iter$ is the current iteration number and max_iter is the total number of iterations.

Qualitatively, we found that the progressive sampling schedule fosters convergence toward clearer and sharper inpainting, as shown in Figure 9. This can be attributed to the larger changes in the earlier stages to form the 3D representations and smaller changes in the later stages, instead of randomly changing the update scale mid-training, which aligns with the similar findings in [56].

9.2.2. Randomization during Grid-based Denoising

We experimented with different numbers of times shuffling during Grid-based Denoising, namely $M = 1, 4, 8$. Our experiments qualitatively showed that 8 times of shuffling yields fewer artifacts in the inpainted area, followed by $M = 4$, and then $M = 1$, as shown in Figure 10. This can be attributed to the averaging effect of the shuffling step in our pipeline, where the influence of multiple views is merged into one update of distillation. However, increasing M by one means calling the U-Net for one iteration of denoising, which severely increases the training time. Therefore, we made a trade-off between training efficiency and performance by setting $M = 4$.

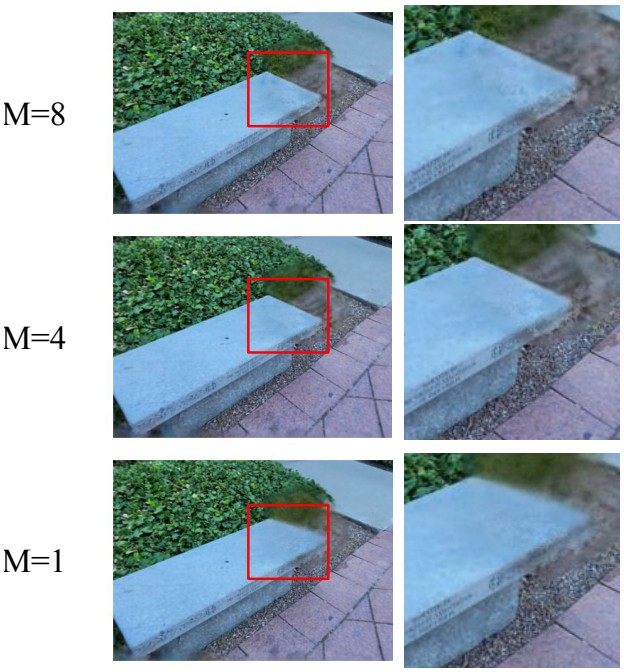


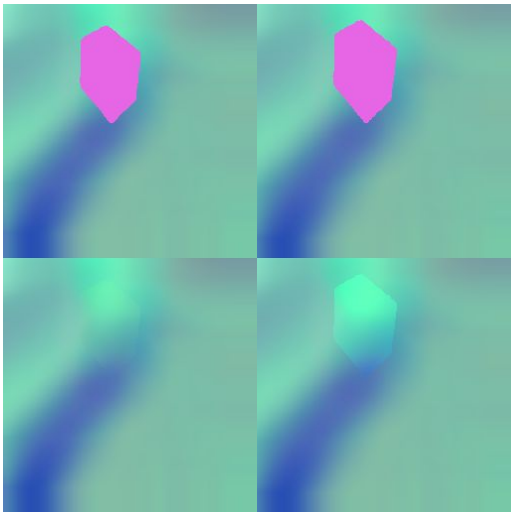
Figure 10. Comparison among different values of M. While $M = 8$ yields the best visual results regarding the sharpness of the inpainting. We made a trade-off between the computation cost in the training phases and the performance and eventually chose $M = 4$ for our experiment.

Dataset Name	Scenes	Prompts
SPIn-NeRF	1	”a stone park bench”
	2	”a wooden tree trunk on dirt”
	3	”a red fence”
	4	”stone stairs”
	7	”a grass ground”
	9	”a corner of a brick wall and a carpeted floor”
	10	”a wooden bench in front of a white fence”
	12	”grass ground”
	Book	”a brick wall with an iron pipe”
	Trash	”a brick wall”
Occlude-NeRF	Monitor	”a computer monitor on a white office desk”
	Meeting Room	”a black stapler on a white office table”
	Bench	”a silver metallic bench with slats and armrests”
	Trash Bin	”a metallic garbage bin with a round opening”
	Light Pole	”a metallic light pole next to a green ground plant”
	Cabinet	”a white filing cabinet with an orange cushion on top, next to a white wall”
LLFF	Fern	”plant and planter on dirt”
	Fortress	”a wooden tabletop”
	Horns	”glass windows and a white support pillar on carpet”
	Orchids	”a conference room with black office chairs and brown carpet”
	Room	”green leaves of a plant”

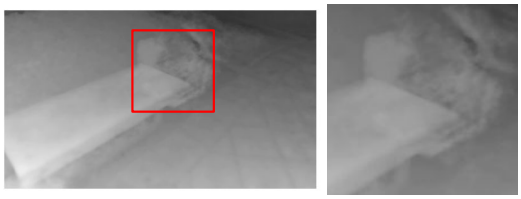
Table 3. Textual prompts used for each scene in our experiments.

9.2.3. Qualitative Comparison with 3D-attention-based SDS

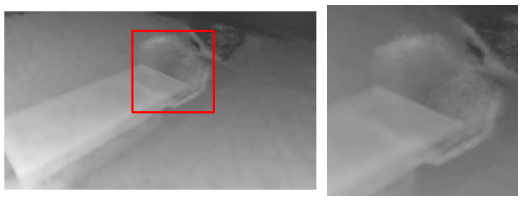
Inspired by prior works on 3D-attention in Diffusion Models [4, 10, 48], we attempted 3D-attention in SDS in our early trials. Specifically, we replaced the diffusion model we used with Stable Video Diffusion (SVD) [4] and MVDream (MVD) [48]. For SVD, we trained the NeRF with SD2 for 3000 iterations to get the initial reconstruction of the scene and then switched to SVD. During each iteration, we rendered 14 views with the first one used as the reference and passed then 14 views as a batch to SVD and back-propagated the distillation loss. For MVD, we disabled the mask condition and passed four rendered views as well as their camera poses as the conditions for the MVD model. For both methods, the gradients are masked and only enabled in the masked area. Derived from text-to-image Stable Diffusion [41], SVD takes a sequence of images as input and allows an additional channel in the input to the U-Net by adding 3D attention layers across the time dimension to compute the self-attention within the batch of input images. Similarly, MVD utilizes 3D spatial attention to assess the view consistency of the 3D generation from 2D. The purpose of our trial is to investigate the possibility of directly using such models off-the-shelf in our 3D inpainting tasks to address cross-view consistency. The visual results are shown in Figure 12. We found that, although efficient in 3D generation tasks, these models are not satisfactory without being further modified and fine-tuned for 3d inpainting tasks, because they do not enable inpainting conditions



(a) Collaborative Inpainting of Normal Maps



(b) Final Disparity Map w/ CDS Geometry



(c) Final Disparity Map w/ Vanilla SDS Geometry

Figure 11. (a) Collaborative Inpainting with Grid-based denoising does not yield consistent inpainting. The top row is the masked normal maps and the bottom row is the inpainted normal maps (normal maps of the bench scene in SPIn-NeRF). We observed inconsistent inpainting results. Comparing the final results with Vanilla SDS geometry guidance (c) and CDS geometry guidance (b), we observed increased artifacts in the latter.

where the masks and masked images are passed to the U-Net to specify the area and the context to inpaint. As a result, the 3D generation diffusion models will be prompted to generate 2D prior, which will change the entire image/view rather than just the masked region. The resulting distillation results do not constitute to a promising 3D inpainting even if we constrain the gradient flow to only the masked region.

9.3. Vanilla Geometry SDS v.s. CDS Geometry SDS

As a major contribution of our method, we apply collaborative SDS in the color space to tackle the occlusion problem in 3D inpainting. Yet, for the geometry SDS of a NeRF scene, we only apply vanilla geometry SDS, where we denoise a single normal map per iteration without collaboratively computing the cross-view loss. This is because, during the experiments, we found that CDS Geometry SDS does not yield consistent distillation in the geometry space. The inconsistency among the different views of normal maps easily leads to convergence into inpainting with artifacts in the geometry space as shown in Figure 11.

As a result, we do not apply collaborative SDS to the geometry space of our inpainting method. This can be attributed to the priors in diffusion models being trained mostly on large-scale datasets of natural RGB images paired with textual descriptions (e.g., "a mountain at sunset"), while normal maps are specialized data representing surface orientations using encoded RGB values (usually indicating x, y, z surface normals), which differ fundamentally from natural RGB images in structure and meaning.

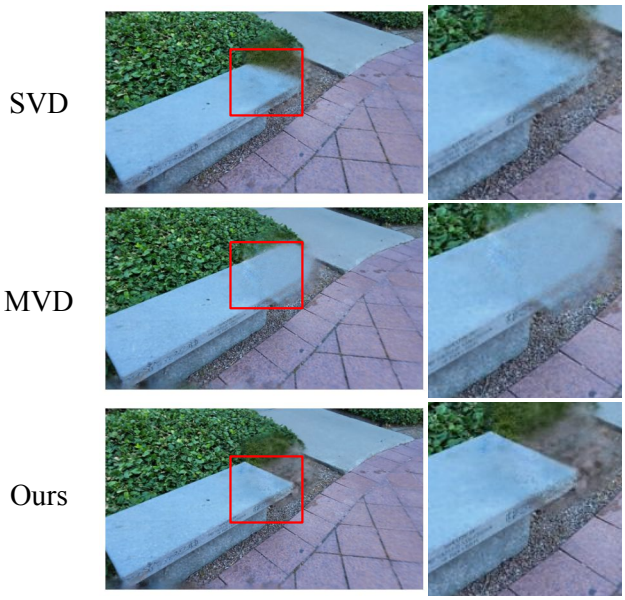


Figure 12. Qualitative comparison between 3D-attention-based diffusion models (SVD & MVD) and our choice of SD. We observe that SVD does not converge to a sharp reconstruction. Meanwhile, MVD does not yield a correct or faithful reconstruction of the scene.

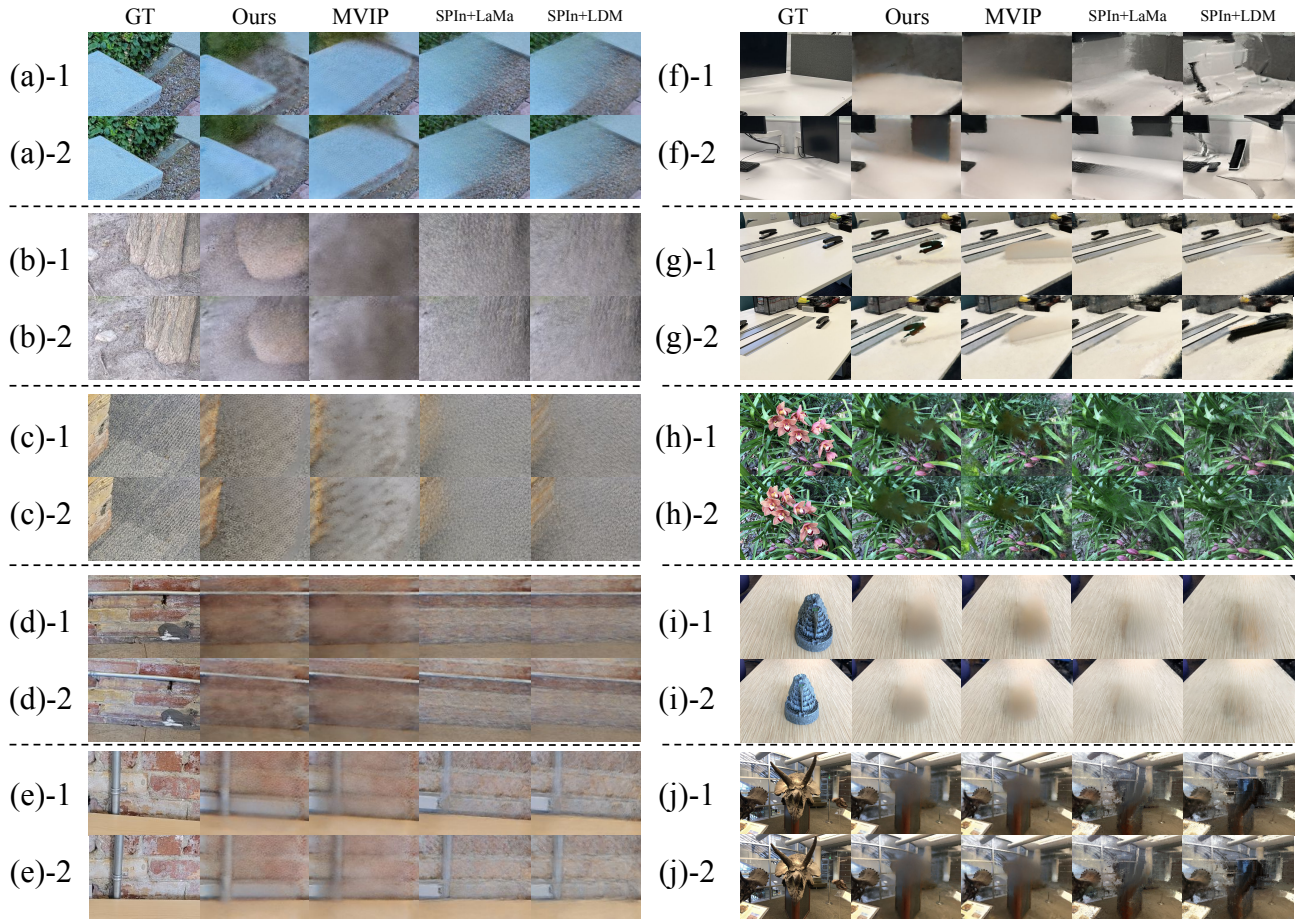


Figure 13. More qualitative results from our experiments. We visualize only the masked region for comparisons. Note that for the LLFF dataset ((h), (i), and (j)), there is no ground truth with the object removed. Thus we show the ground truths with objects instead. We observed more faithful reconstructions of the true scenes with our method in severely occluded cases ((a), (b), (c), (f), (g)). We also witness more cross-view consistency of our method compared to SPIn-NeRF-based methods ((f) and (j)). However, our method is limited in reconstructing high-frequency texture regions in the scene ((d) and (e) the brick wall, (h) the orchid leaves, and (i) the table), which is a common challenge in SDS-based methods.

10. Qualitative Results

In this section, we present more qualitative results and discuss the shortcomings of our Occlude-NeRF method.

As shown in Figure 13, our methods can faithfully reconstruct the occluded areas with limited information compared to the baseline methods ((a) the true edge of the stone bench, (b) the root location of the trunk, (c) the shape of the wall corner, (f) the edge of the monitor, and (g) the location of the stapler). While being able to reconstruct the occluded area, our method maintains satisfactory visual reconstruction in the cases where occlusion is not severe ((d) and (e) the location of the pipe).

10.1. Limitation: High-frequency Region Reconstruction

As prior works [25, 56] pointed out, recovering high-frequency regions remains a common challenge for 3D generative methods like SDS. In this subsection, we showcase Occlude-NeRF's limitation in generating high-frequency regions in Figure 13. Specifically, in scenes (d) and (e), we observe that the brick wall texture in ours and MVIP's is blurred compared to that in SPIn-NeRF-based methods. Similarly, in (c) the gray carpet is rendered with artifacts with dot texture instead of the real texture of the carpet. Moreover, the patterns of the orchid leaves in (h) and the texture of the table in (i) are both blurred in ours and MVIP's, while being sharper and clearer in those in SPIn-NeRF-based methods. This is attributed to the mechanism of SDS-based methods. Repetitive updates in SDS will av-

erage out the shape of high-frequency objects in the scene. To tackle this problem, we anticipate future work introducing a more advanced noise scheduling mechanism, where the shape of the reconstruction can be affirmed in the early stages and sharpened in the later stages to avoid blurriness.

11. Ethical Concerns

The ethical concerns of our method primarily revolve around its potential misuse and implications for privacy, authenticity, and societal impact [45, 46]. Similar algorithms have been applied to editing humanoid avatars [31, 47] and objects [12, 18, 19] in virtual environments. The capability of this type of algorithm might be exploited to fabricate or manipulate digital evidence, misrepresent physical spaces, or breach privacy by reconstructing obscured or private areas without consent. Additionally, biases inherent in training datasets could lead to unfair or inaccurate reconstructions, potentially reinforcing stereotypes or producing misleading results, this is an especially common downfall of text-prompted generative AIs.