DiffusionSfM: Predicting Structure and Motion via Ray Origin and Endpoint Diffusion

Supplementary Material

Overview

The supplementary material includes sections as follows:

- Section A: Implementation details.
- Section B: Additional analysis on integrating RayDiffusion [41] with MoGe [35].
- Section C: More qualitative comparisons of predicted geometry and camera poses against baseline methods.
- Section D: Details and evaluation of the sparse-to-dense training strategy employed in DiffusionSfM.
- Section E: More analysis of the homogeneous representation.
- Section F: Converting predicted ray origins and endpoints into camera poses.

A. Implementation Details

Inference. DiffusionSfM utilizes x_0-parameterization to predict clean ray origin and endpoint maps as the model output, employing 100 diffusion denoising timesteps. In Fig. 9, we evaluate the accuracy of x_0-prediction at each timestep with eight input images on CO3D [21] unseen categories. Interestingly, we find that DiffusionSfM achieves its most accurate clean sample predictions at an early timestep ($T = 90$), rather than at the final denoising step ($T = 0$). This observation remains consistent across different numbers of input images (Zhang *et al*. [41] also have a similar observation that early stopping helps improve performance). To capitalize on this property, we limit inference to 10 denoising steps and use the x_0-prediction at $T = 90$ as the final output, significantly reducing inference time. Moreover, we find that the optimal timestep varies across datasets: $T = 80$ yields the best results on Habitat [24], while $T = 70$ performs best on RealEstate10k [42].

Resolving Ambiguities in GT. We transform camera poses so the first camera has an identity rotation and is positioned at the world origin. For scale, we unproject the first image in the input views using GT depth and scale the world coordinates so the "median point" lies at a unit distance from the origin. Our model is trained to conform to this scene configuration.

B. RD+MoGe Baseline: More Details

To minimize the scale difference for the predicted camera poses from RayDiffusion [41] and depths from MoGe [35] to form a single consistent output, we follow these procedures: (1) We match the MoGe depth with the ground-truth (GT) depth using a 1D optimal alignment (thus giving this baseline some privileged information). (2) We align the predicted camera centers from RayDiffusion with GT cameras using an optimal similarity transform. (3) Finally, we unproject image pixels using the updated camera parameters and the aligned depths. We find that a naive combination of RayDiffusion and MoGe yields poor Chamfer Distance, even though RayDiffusion estimates relatively accurate focal length. This is because the MoGe depth estimates for different input views are inconsistent with each other. Therefore, to predict consistent 3D geometry from multiple images, the model must learn to reason over the entire set of views, rather than relying on mono-depth predictions from individual images. We also include more visualizations in Fig. 6, where duplicated structures are observed due to significant pose errors or a minor misalignment between views.

C. More Qualitative Comparisons

We include more qualitative comparisons with baselines on the predicted geometry (Fig. 6) and camera poses (Fig. 7).

Discussion. We show that DiffusionSfM can handle challenging input images where objects present highly symmetric patterns (*e.g*., the tennis ball example in Fig. 6 and the donut example in Fig. 7), while RayDiffusion [41] and DUSt3R [36] fail to predict correct camera poses. Compared to RayDiffusion, our approach leverages the prediction of *dense* scene geometry (*i.e*., pixel-aligned ray origins and endpoints) rather than relying on patch-wise "depth-agnostic" rays. We find that predicting dense pixel-aligned outputs improves performance (see Sec. D). When compared to DUSt3R, our model benefits from attending to all input images simultaneously and utilizing a diffusion framework to effectively manage the high uncertainties inherent to this task. Additionally, we observe that DUSt3R often predicts precise camera rotations but struggles with camera centers in many cases (*e.g*., the keyboard example in Fig. 6). This observation aligns with our quantitative results for camera center evaluation, presented in Tab. 1.

D. Sparse-to-Dense Training Details and Evaluation

As outlined in Sec. 3.3, we follow a sparse-to-dense strategy to train our model as we find that training the high-resolution model (*i.e*., the dense model) from scratch yields suboptimal performance. We visualize the output of the sparse model and dense model in Fig. 8. In the following, we introduce the details of training DiffusionSfM.

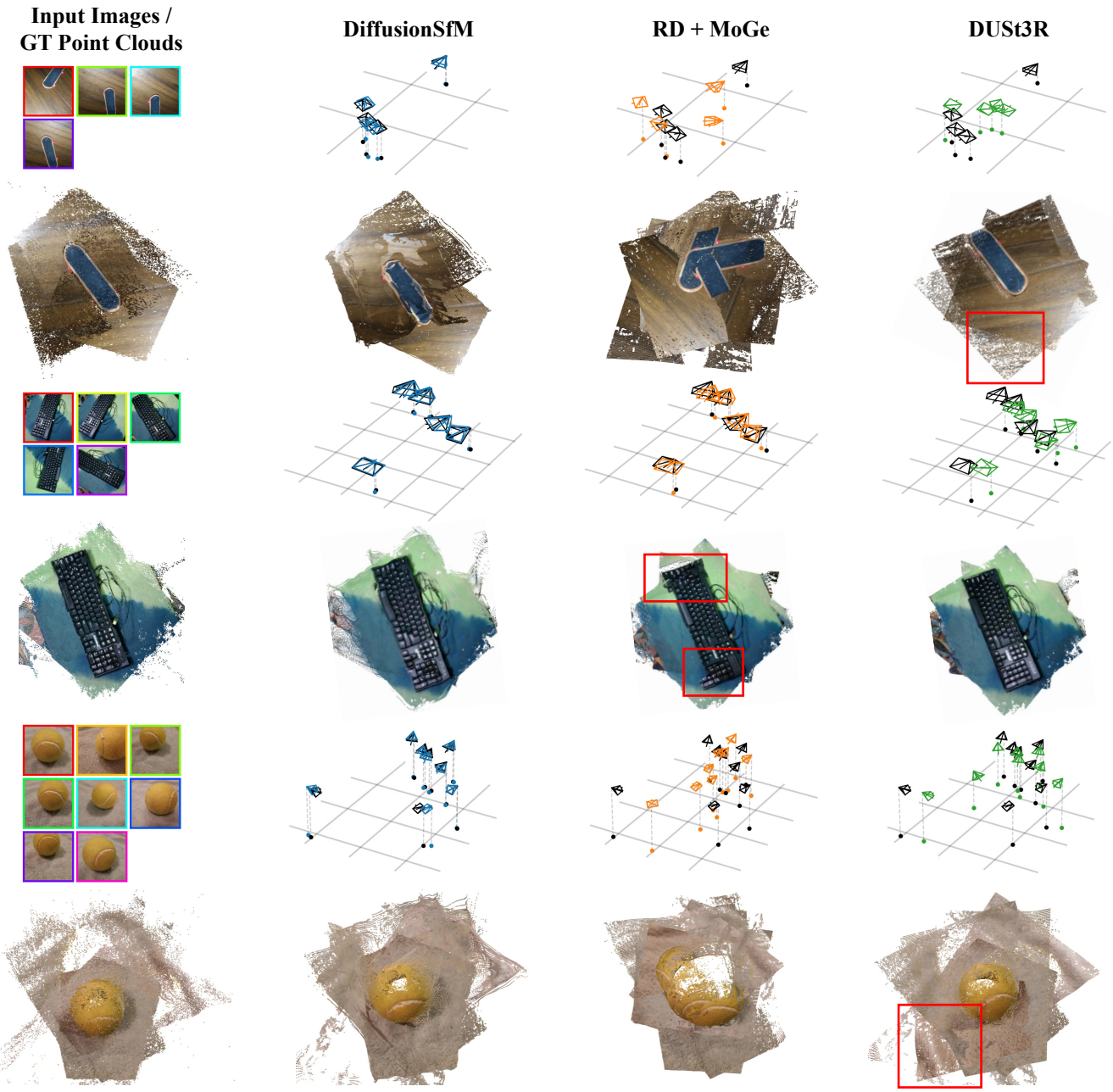

Figure 6. **More Qualitative Comparisons on Predicted Geometry and Camera Poses.** DiffusionSfM shows superior capabilities in handling challenging samples, *e.g.*, the skateboard and tennis ball. Additionally, while we observe that DUSt3R can predict highly precise camera rotations, it often struggles with camera centers (see the keyboard example).

Details. Our model leverages DINOv2-ViTs14 [18] as the feature backbone and takes 224×224 images as input. This results in 16×16 image patches, each with patch size 14. We first train a sparse model that outputs patch-wise (*i.e.*, 16×16) ray origins and endpoints. Since the spatial resolution of the ground-truth ray origins and endpoints for the sparse model aligns with the DINOv2 feature map, we use a single linear layer to embed the noisy ray origins and endpoints (without spatial downsampling), rather than a convolutional layer as shown in Eq. 5. We also remove the DPT [20] decoder in our sparse model. Subsequently, we initialize our dense model from the pre-trained sparse model to predict dense (*i.e.*, 256×256) ray origins and endpoints. We copy-paste the DiT [19] weights from the sparse model. Whereas for the convolutional layer used to embed ray origins and endpoints, we duplicate the linear-layer weights by 16×16 (as the patch size of the conv-layer is 16) and then divide them by 256 to account for the patch-wise addition.

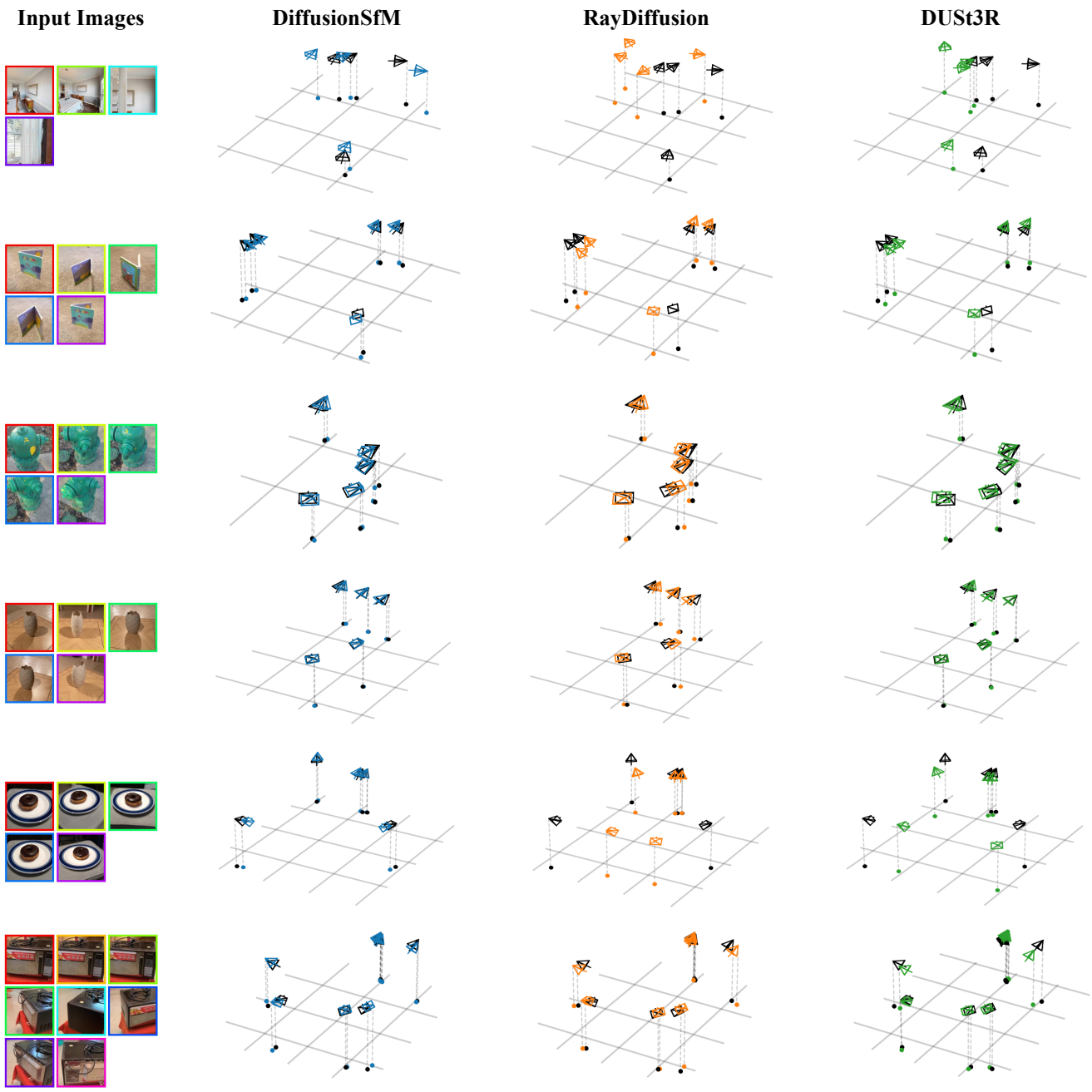

Figure 7. **More Qualitative Comparisons on Predicted Camera Poses.**

While the DiT in the dense model has learned meaningful representations, the DPT decoder is initialized from scratch. To avoid breaking the learned DiT weights in early training iterations, we freeze its weights while only training the convolutional embedding layer and the DPT decoder for a few iterations. This warm-up model is referred to as Dense Model (1). After that, we train the whole model together, including the DINOv2 encoder as well (which was frozen in the previous stage). During this phase, we apply a lower learning rate ($0.1\times$) to both the DINOv2 encoder and DiT compared to the DPT decoder. The fully trained model is referred to as Dense Model (2). We compare the performance of DiffusionSfM-CO3D at each stage in Tab. 5.

Training Resources. (1) DiffusionSfM-CO3D: We train the sparse model using 4 H100 GPUs with a total batch size of 64 for 400,000 iterations, which takes approximately 2 days. To warm up the dense model, we freeze the DiT weights and train for 50,000 iterations. We then unfreeze the full model and continue training for another 250,000 iterations on 4 H100 GPUs with a batch size of 48, requir-

	# of Images	Rotation Accuracy (↑, @ 15°) 2	3	4	5	6	7	8	Center Accuracy (↑, @ 0.1) 2	3	4	5	6	7	8
Seen	Sparse Model	92.5	93.1	93.4	93.6	93.6	93.8	93.9	100	95.4	92.6	90.9	89.6	88.8	88.2
	Dense Model (1)	90.3	90.7	90.9	90.8	90.9	91.0	90.9	100.0	94.9	91.1	89.0	87.1	85.7	84.2
	Dense Model (2)	93.4	94.0	94.5	94.8	95.0	95.2	95.1	100.0	95.9	93.6	92.2	91.2	90.7	90.2
Unseen	Sparse Model	87.0	89.2	90.2	90.7	91.2	91.7	92.1	100.0	90.9	86.3	83.1	81.0	79.7	79.2
	Dense Model (1)	85.9	86.8	87.4	87.8	88.5	88.6	88.8	100.0	89.1	83.7	79.7	77.7	75.5	74.5
	Dense Model (2)	90.4	91.2	92.7	93.0	93.1	93.3	93.5	100.0	91.1	87.7	85.3	83.7	82.7	82.0

Table 5. **Camera Rotation and Center Accuracy on CO3D at Different Training stages.** On the left, we report the proportion of relative camera rotations within 15° of the ground truth. On the right, we report the proportion of camera centers within 10% of the scene scale, relative to the ground truth. To align the predicted camera centers to ground truth, we apply an optimal similarity transform (s, $\mathbf{R}$, $\mathbf{t}$). Hence the alignment is perfect at $N = 2$ but worsens with more images.

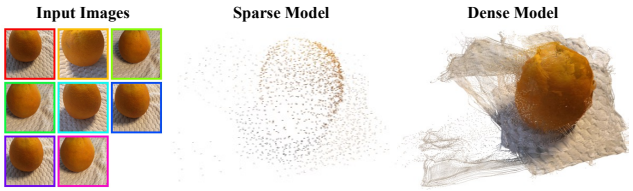

Figure 8. **Qualitative Comparison of Sparse and Dense Model Outputs.** The sparse model predicts the ray origin and endpoint for each image patch, limiting its ability to capture the fine-grained details of the scene.

ing an additional 2 days. (2) DiffusionSfM: This variant is trained with 8 H100 GPUs and a larger batch size. The sparse model is trained with a total batch size of 288 for 490,000 iterations over roughly 6.5 days. The dense model is trained with a batch size of 96 for 800,000 iterations (including 50,000 warm-up iterations as well), taking approximately 10.5 days. However, the training beyond 500,000 iterations yields only marginal improvements.

E. The Effect of Homogeneous Representation

To underscore the importance of the proposed homogeneous representation for ray origins and endpoints, we train a variant of DiffusionSfM using these components directly in $\mathbb{R}^3$ (*i.e.*, without using homogeneous coordinates). For this model, we employ a scale-invariant loss function, as used in DUSt3R [36]. The training loss curve for this model is shown in Fig. 10. Notably, the model fails to converge, with the training loss remaining persistently high. This failure occurs because our diffusion-based approach assumes input data within a reasonable range, as the Gaussian noise added during training has a fixed standard deviation of 1. Consequently, training scenes with substantial scale differences across components disrupt the model's learning process. In contrast, employing homogeneous coordinates enables the normalization of the input data to a unit norm, which not only stabilizes training and facilitates convergence but also provides an elegant representation of unbounded scene geometry.

F. Converting Ray Origins and Endpoints to Camera Poses

The camera centers for each input image are recovered by averaging the corresponding predicted ray origins. To determine camera rotations and intrinsics, we follow the method proposed by Zhang *et al*. [41], which involves solving for the optimal homography that aligns the predicted ray directions with those of an identity camera. For additional details, we refer readers to Zhang *et al*. [41].

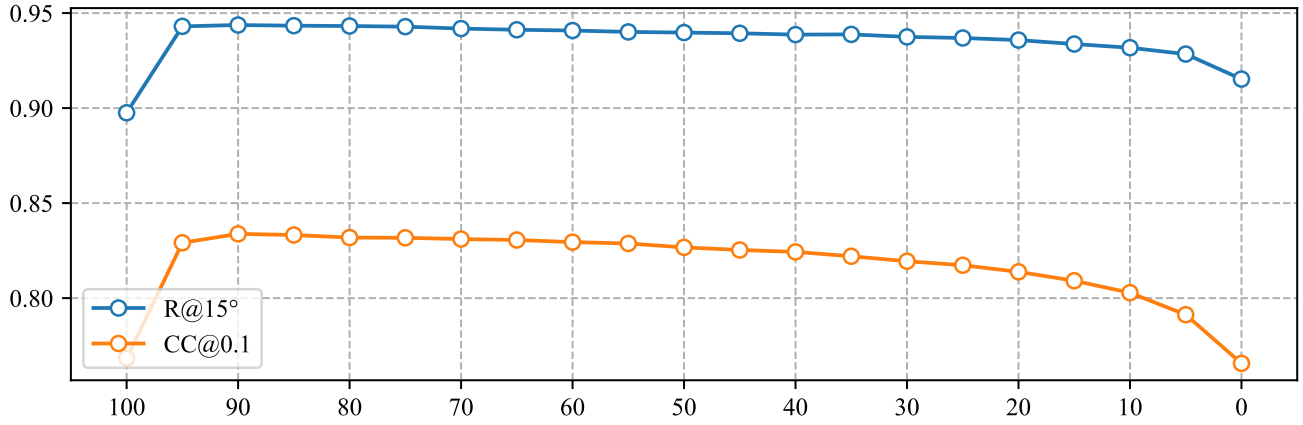

Figure 9. **Performance of x_0-Prediction on CO3D Unseen Categories across Diffusion Denoising Timesteps (N = 8).** The X-axis represents the diffusion denoising timesteps, with $T = 100$ indicating predictions starting from Gaussian noise and $T = 0$ corresponding to the clean sample. The Y-axis shows the accuracy for camera rotation (blue) and camera center (orange). Notably, DiffusionSfM achieves peak performance at $T = 90$. As a result, in inference, we perform only 10 diffusion steps, significantly improving inference speed.

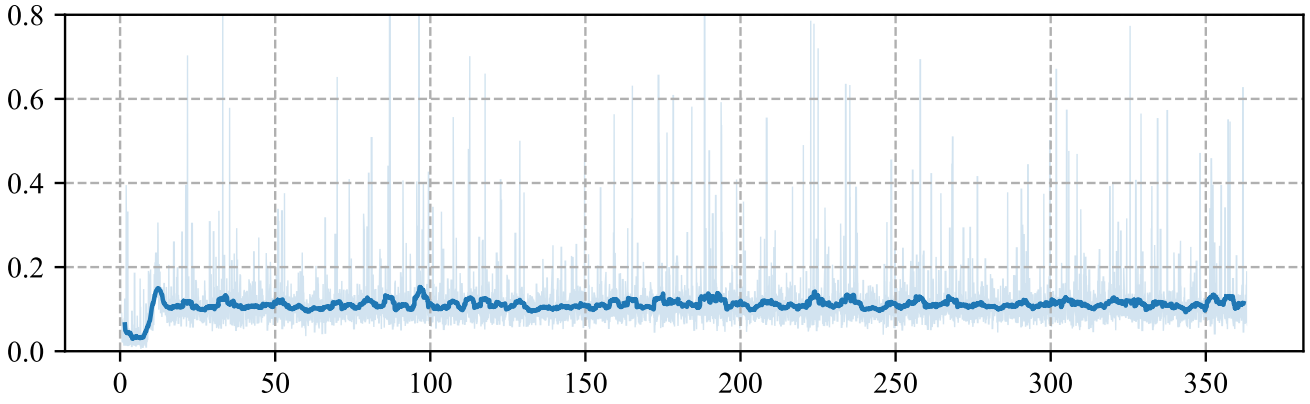

Figure 10. **Training Loss Curve for DiffusionSfM without Homogeneous Representation.** The X-axis represents training iterations (in thousands, k), and the Y-axis denotes the loss value. Without incorporating a homogeneous representation for ray origins and endpoints, the model struggles to train effectively due to significant scale differences across various scene components.