CaesarNeRF: Calibrated Semantic Representation for Few-Shot Generalizable Neural Rendering

Supplementary Material

In this supplementary material, we provide more details in addition to the main manuscript, including specifics of datasets, along with additional analysis and discussions with visualizations, where we also discuss the GIF examples attached in the supplementary material.

6 Dataset Details

In this section, we discuss the further details of the datasets used in our experiments. Our generalizable experimental settings consist of two different configurations, the details of which are presented as follows:

(a) LLFF, Shiny, and mip-NeRF 360. In this configuration, we conduct experiments on the LLFF [42], Shiny [67], and mip-NeRF 360 [4] datasets with the GNT training settings as specified in [60]. We adopt the GPNR settings [58] to sample every eighth frame in each category for testing. Specifically, for LLFF [42], we evaluate on eight categories: trex, fern, flower, leaves, room, fortress, horns, and orchids. For Shiny [67], we test on eight categories: CD, crest, food, giants, lab, pasta, seasoning, and tools. For mip-NeRF 360 [4], we test on seven categories that are available without external restrictions: bicycle, bonsai, counter, garden, kitchen, room, and stump.

(b) MVImgNet. The MVImgNet dataset [77] comprises 6.5 million frames from 219,188 videos across 238 classes, making it infeasible to train on all sequences or categories. We adhere to the official split and focus on the *container* category, which includes category ID 0, 1, 14, 26, 28, 37, 39, 43, 48, 49, 83, 119, 145, and 160. We randomly subsample 2,500 training examples from the training set and select 108 sequences for the test set. Testing is conducted on the first example of each sequence, using the remaining samples as reference views. We attach the list of scenes used for training and inference to the supplementary material.

7 Additional Analysis

In this section, we present further analysis of CaesarNeRF, as well as its comparison with other methods along with more experimental details.

Per-scene Optimization on LLFF. We present detailed results for per-scene optimization in Table 8. We compare CaesarNeRF against the top-performing model from Table 5, GNT [60], as well as other methods, LLFF [42], NeX [67], and NeRF [44]. CaesarNeRF surpasses all other methods across all categories for LPIPS. It also shows the best performance in half of the LLFF dataset categories in terms of PSNR and SSIM. CaesarNeRF outperforms GNT [60], our baseline method, in all categories on three metrics with consistent improvements.

Table 8: Full table of evaluation for per-scene optimization of eight categories on the LLFF [42] dataset comparing CaesarNeRF with state-of-the-art methods.

Metric	Method	trex	fern	flower	leaves	room	fortress	horns	orchids
PSNR (↑)	LLFF [42]	27.48	28.72	20.72	21.13	24.54	21.79	23.22	18.52
	NeRF [44]	26.80	25.17	27.40	20.92	32.70	31.16	27.45	20.36
	NeX [67]	28.73	25.63	28.90	21.96	32.32	31.67	28.46	20.42
	GNT [60]	28.15	24.31	27.32	22.57	32.96	32.28	29.62	20.67
	Ours	28.30	25.63	28.29	23.11	32.21	32.47	29.56	21.52
LPIPS (↓)	LLFF [42]	0.222	0.247	0.174	0.216	0.155	0.173	0.193	0.313
	NeRF [44]	0.249	0.280	0.219	0.316	0.178	0.171	0.263	0.321
	NeX [67]	0.193	0.205	0.150	0.173	0.161	0.131	0.173	0.242
	GNT [60]	0.080	0.116	0.092	0.109	0.060	0.061	0.076	0.153
	Ours	0.076	0.111	0.068	0.095	0.057	0.055	0.071	0.139
SSIM (↑)	LLFF [42]	0.857	0.753	0.844	0.697	0.932	0.872	0.840	0.588
	NeRF [44]	0.880	0.792	0.827	0.690	0.948	0.881	0.828	0.641
	NeX [67]	0.953	0.887	0.933	0.832	0.975	0.952	0.937	0.765
	GNT [60]	0.936	0.846	0.893	0.852	0.963	0.934	0.935	0.752
	Ours	0.943	0.871	0.912	0.872	0.967	0.946	0.939	0.782

Comparison with input views. To investigate how CaesarNeRF handles a single reference view input, we present two visualizations in Figure 9, using the first two examples in Figure 1, "crest" and "lab" from the Shiny [67] dataset. We present the input view for these two examples along with the rendered results from GNT [60] and CaesarNeRF, comparing them with the ground-truth.

For a single-view input, the difference between the input and the target view can be decomposed into three parts: affine transformation, changes in occlusions, and information outside the image. CaesarNeRF can generate reasonable results for the affine transformation that is visible in the input view. For different occlusions between the input and target views, such as the center bottom of the first example, where the base occludes more of the pillar in the background in the rendered image compared with the input view, CaesarNeRF can predict the occlusion correctly, indicating it can handle the object relationships in the image instead of treating the scene as a flat image. For areas not captured in the input images, CaesarNeRF cannot provide rendering results if there is a large patch missing. Additionally, compared with large view shifts in the first example, small shifts between the poses of the input and target views, as in the second example, yield more accurate rendering.

8 Visualizations.

In this section, we provide more visualizations and analysis on the fours datasets we used in our experiment. We present two different variations, framewise results as attached to this document, and the video results in the form of GIF files, which are included in the supplementary material.

Framewise results. We provide more examples comparing CaesarNeRF to GNT [60], our baseline method, as it has outperformed other baseline methods in Table 1. We show results on the LLFF [42] and Shiny [67] datasets in Figure 10 and on the mip-NeRF 360 [4] and MVImgNet [77] datasets in Figure 11. For each dataset, we present two examples using 1, 2, and 3 views as input reference views. We observe that with a limited

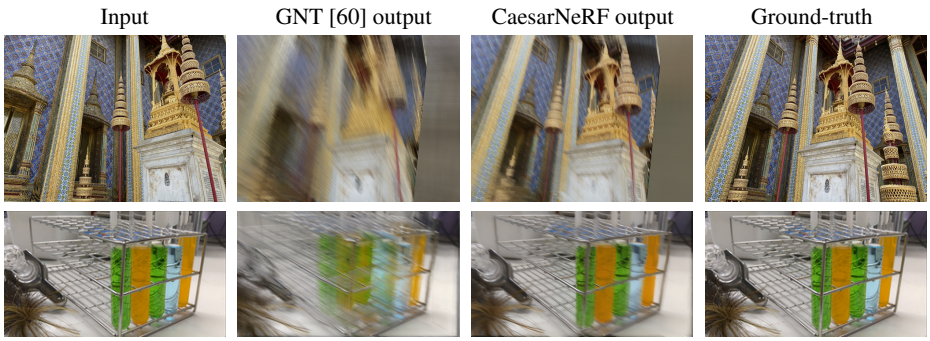


Fig. 9: Visualization comparison of CaesarNeRF and GNT with the input frame using one reference view as input.

number of input views, the overall reconstruction quality is constrained, especially for mip-NeRF 360 [4], where the differences between reference and target camera views are substantial. Nevertheless, when compared to GNT [60] across varying numbers of views, CaesarNeRF consistently generates images with better rendered quality, featuring sharper boundaries and fewer miscolored areas.

In addition to scenes with multiple objects, we also present additional visualizations from the recently introduced MVImgNet [77] dataset in Figure 12. These scenes in the MVImgNet [77] dataset mostly focus on object-centric scenarios, and we use just one reference view as input as it is a simpler case. Different from scenes featuring multiple object combinations and intricate geometrical relationships, the object-centric scenes in MVImgNet [77] provide enhanced quality with even a single input view for both GNT [60] and our CaesarNeRF. CaesarNeRF markedly surpasses GNT [60].

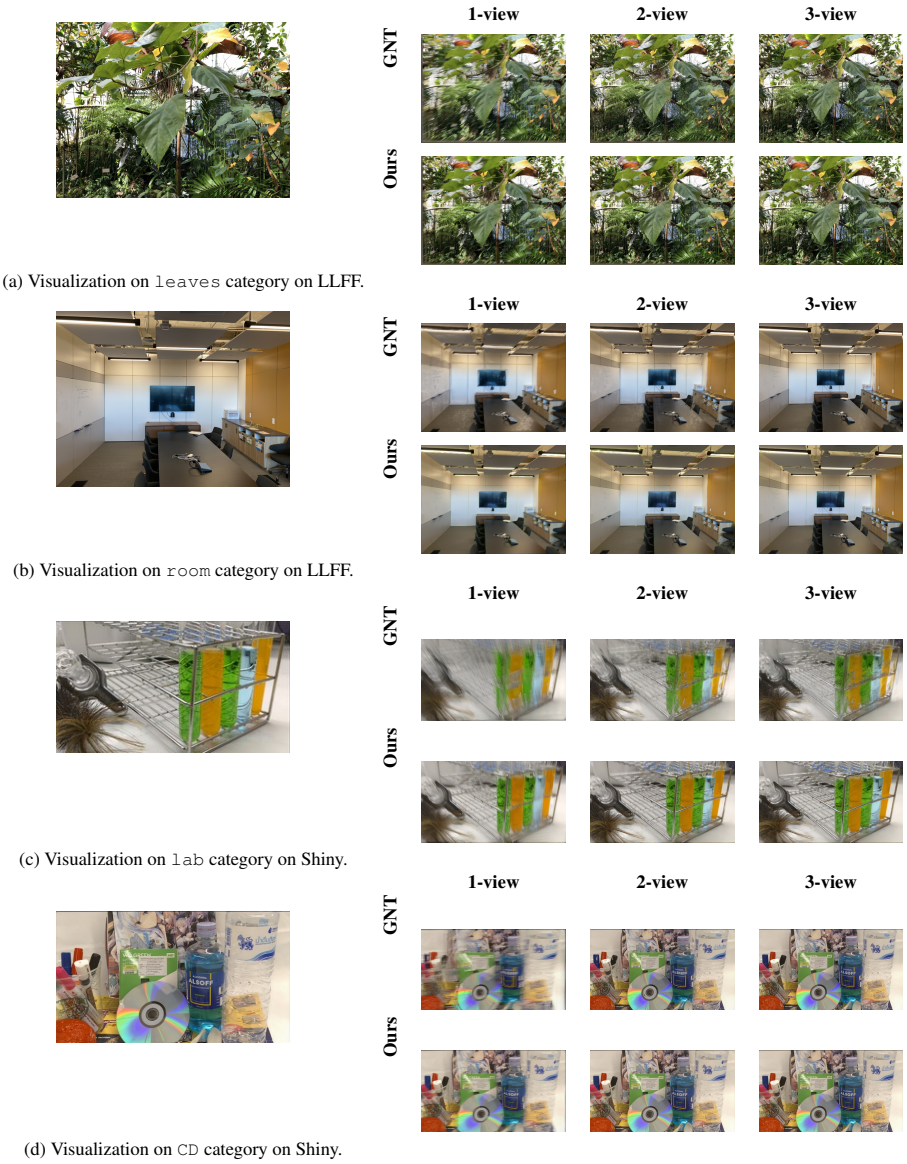
We further present the results for different numbers of views using CaesarNeRF in Figure 13. When the number of views exceeds 2, the overall quality of the reconstructed images remains consistent for CaesarNeRF, resulting in high-quality outcomes as evident in Table 4. In scenarios where the scene is object-centric with a straightforward background, CaesarNeRF excels with relatively fewer input views, maintaining its high quality across different numbers of images used as reference views.

Video results. Along with the framewise rendering, we also include rendered videos in the form of GIF files along with this document in the supplementary material. As we have focused on generalizable rendering with one or two reference views in framewise reconstructions, for video rendering, we provide examples for two other cases, including the rendering results with three reference views for generalizable rendering and per-scene optimization.

For the generalizable setting with three reference views, we have selected four scenes from LLFF with high-frequency pattern changes, including “flower”, “horns”, “leaves”, and “orchids”. We compare CaesarNeRF with its baseline, GNT [60]. When the input views are limited but sufficient, GNT exhibits more inconsistent fragments, such as the boundaries of leaves and flowers. In contrast, our proposed CaesarNeRF

demonstrates more consistent boundaries with the introduction of calibrated semantic representation across views, enabling a holistic understanding.

For per-scene optimization, we present an example involving “orchids”, comparing CaesarNeRF with GNT [60]. GNT primarily focuses on pixel-level rendering and lacks a holistic scene-level understanding. Consequently, the overall illumination across different viewpoints changes consistently in GNT, while CaesarNeRF produces a more consistent rendering. Additionally, we observe that in two patches we cropped out, GNT generates inconsistent structures across different views, whereas our rendering results remain more stable.



(a) Visualization on `leaves` category on LLFF.

(b) Visualization on `room` category on LLFF.

(c) Visualization on `lab` category on Shiny.

(d) Visualization on `CD` category on Shiny.

Fig. 10: Additional Visualizations on LLFF [42] and Shiny [67] datasets when using 1, 2 and 3 reference views as input comparing ours with GNT [60].

(a) Visualization on `kitchen` category on mip-NeRF 360.

(b) Visualization on `room` category on mip-NeRF 360.

(c) Visualization on `container` category on MVImgNet.

(d) Visualization on `container` category on MVImgNet.

Fig. 11: Additional Visualizations on mip-NeRF 360 [4] and MVImgNet [77] datasets when using 1, 2 and 3 reference views as input comparing our proposed method, CaesarNeRF, with our baseline, GNT [60].

(a) GNT (b) Ours (c) Groundtruth (a) GNT (b) Ours (c) Groundtruth

Fig. 12: Visualization for one reference view input comparing GNT [60] with CaesarNeRF on MVImgNet [77].

(a) 1-view (b) 2-view (c) 3-view (d) 4-view (e) 5-view (f) Groundtruth

Fig. 13: Visualization for different numbers of reference views with CaesarNeRF on MVImgNet [77].