RoGUENeRF: A Robust Geometry-Consistent Universal Enhancer for NeRF Supplementary Material

Sibi Catley-Chandar[1,2], Richard Shaw[1],
Gregory Slabaugh[2], and Eduardo Pérez-Pellitero[1]

[1] Huawei Noah's Ark Lab
[2] Queen Mary University of London

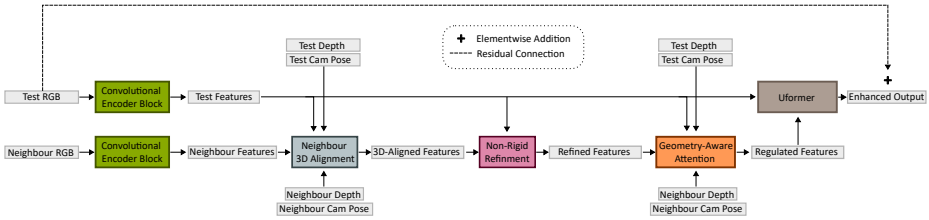

Fig. 1: An overview of our entire pipeline. We present implementation details for each component of our pipeline in Section 4.

We present further detailed qualitative and quantitative evaluation of our method in Sections 1 and 5. We evaluate the amount of training data required to train our model compared to the state-of-the-art in Section 2 and compare inference speeds in Section 3. We present full implementation details of each component of our method in Section 4.

1 Video Results

We present video results of our proposed enhancer compared to NeRFLiX [13], MipNeRF360 [1], Nerfacto [9] and NeRF [6] on our project page: `https://sib1.github.io/projects/roguenerf/`. We demonstrate noticeable improvements over the baseline NeRF models and NeRFLiX, whilst retaining geometric consistency and restoring high-frequency textures. We encourage viewing the video results to fully appreciate the improvement in quality provided by our method.

2 Real World Data Efficiency

Real world NeRF datasets are typically small, sometimes only containing tens of images per scene. Exisiting NeRF enhancers like NeRFLiX [13] struggle to

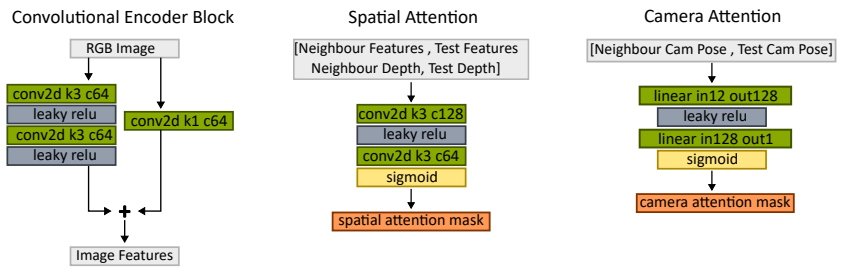

Fig. 2: We present detailed architectures of our convolutional encoder and geometry-aware attention modules. k indicates kernel size, c indicates output channel size, in/out indicate input/output dimension size.

leverage such small datasets, requiring several orders of magnitude more training data. In contrast, our novel geometry-consistent alignment and fusion mechanisms allow our enhancer to be pre-trained and fine-tuned on relatively small datasets. We compare the results of training NeRFLiX on different dataset sizes with our method in Table 1. We show that NeRFLiX requires several orders of magnitude more simulated training data (Vimeo90K [11]) than our method to achieve any improvement at all over the baseline NeRF model, TensoRF [2]. When trained on only the real world LLFF dataset, NeRFLiX actually degrades the performance of TensoRF by 0.6dB. In contrast, our method can be pre-trained and fine-tuned on small datasets to achieve large improvements in performance.

Table 1: We show that our method is substantially more data efficient than NeRFLiX as it can be pre-trained and fine-tuned on very small datasets. NeRFLiX is unable to learn an improvement over the baseline using only a small real world dataset. †Results as reported by authors.

Model	Training Data	# Training Frames	PSNR
TensoRF	LLFF	262	26.88
TensoRF + NeRFLiX†	LLFF	262	26.28 (↓ 0.60)
TensoRF + NeRFLiX†	LLFF + Vimeo90K-10%	45,490	26.71 (↓ 0.17)
TensoRF + NeRFLiX†	LLFF + Vimeo90K-50%	226,404	27.08 (↑ 0.30)
TensoRF + NeRFLiX	LLFF + Vimeo90K-100%	452,546	27.38 (↑ 0.50)
TensoRF + Ours	LLFF	262	27.58 (↑ 0.70)

3 Inference Speed

We compare the inference speed of NeRFLiX and NeRFLiX++ [12] with our method for different image resolutions in Table 2. We measure inference speed

on a single NVidia V100 GPU at full float32 precision using the Pytorch event time measurement function [7]. For NeRFLiX++, we report inference times directly from the original work. We note that for all quantitative and qualitative comparisons in this work, NeRFLiX performs test-time augmentation (TTA) by running 4× forward passes during inference with horizontally and vertically flipped inputs and averaging the result. We perform a single inference pass and do not use TTA. We show that RoGUENeRF is approximately 13× faster than NeRFLiX with TTA, 3× faster than NeRFLiX without TTA, and 3× slower than NeRFLiX++.

Table 2: We present quantitative results from using different datasets for training. TTA = Test-Time Augmentation. NeRFLiX results reported in this work use TTA.

Model	TTA	Input Resolution	Inference Time (ms)
NeRFLiX	Yes	512x512	4208
NeRFLiX	No	512x512	1050
Ours	No	512x512	337
NeRFLiX++	No	512x512	109
NeRFLiX	Yes	1024x1024	18385
NeRFLiX	No	1024x1024	4579
Ours	No	1024x1024	1346
NeRFLiX++	No	1024x1024	433

4 Implementation Details

We present our entire pipeline in Figure 1. For each of the learnable components in our pipeline, we present detailed module architectures in Figures 2 and 3. We refer to a standard 2d convolutional layer as $conv2d$, and indicate the kernel size with k, channel output size with c, stride with s and input/output dimensions for linear layers with in/out. We use zero padding and He initialization [3] for learnable weights. We use leaky relu with a slope of 0.01.

4.1 3D Alignment

We first extract features using the convolutional encoder block shown in Figure 2. We use the Kornia [8] geometry remap function with bilinear interpolation and zero padding to perform the 3D alignment of neighbouring training images to a novel test view. The inputs to the function are formulated as described in the main manuscript.

4.2 Iterative Refinement Network

We present the implementation details of our iterative refinement network in Figure 3. We perform our optical flow estimation at $8\times$ downsampled resolution and apply bilinear upsampling to the estimated flow field to get back to full resolution. Our flow warping uses the PyTorch grid sample function with bilinear interpolation.

4.3 Geometry-Aware Attention

We present the architecture of our spatial attention and camera attention modules in Figure 2. We learn a bias for each convolutional and linear layer. Camera rotations are first converted to Euler angles before being processed by the attention module.

4.4 Uformer

We use the official implementation of the Uformer [10]. We modify the original architecture to accept 64 channel image features as input instead of 3 channel RGB images. Otherwise we keep the architecture identical to the original formulation.

5 Per-Scene Results

We present extended quantitative results in Tables 3 and 4 where we report results for each scene in the 360v2 [1] and LLFF [5] datasets. We show that RoGUENeRF consistently outperforms the baseline NeRF models, NeRFLiX and AligNeRF [4] on the majority of scenes across all metrics.

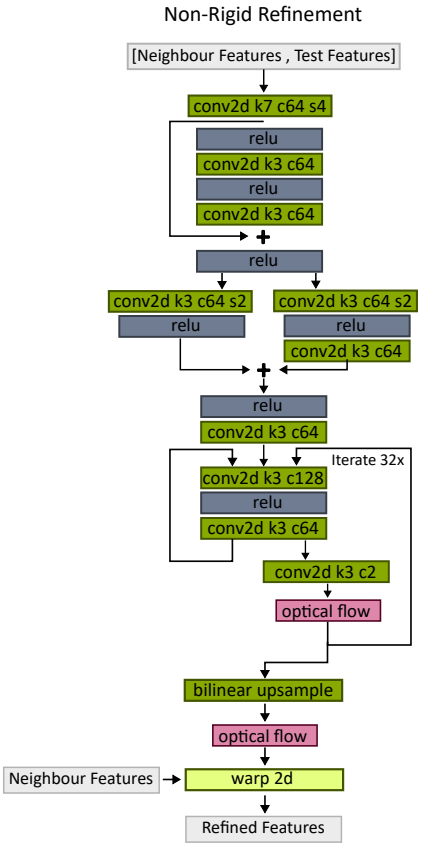

Fig. 3: We present the implementation details of our iterative flow network for non-rigid refinement. The optical flow estimation is performed at $8\times$ downsampled resolution and we apply bilinear upsampling to recover a full resolution flow field.

Table 3: Per-Scene results on the 360v2 dataset. All scenes are evaluated at 4× downsampled resolution. Red / orange highlights indicate best / second method respectively. †Results as reported by authors.

(a) PSNR

Model	Bicycle	Bonsai	Counter	Garden	Kitchen	Room	Stump	Flowers	Treehill
ZipNeRF	25.88	35.65	29.5	28.24	33.28	34.03	27.32	22.32	23.91
ZipNeRF + NeRFLiX	25.9	36.22	30.06	27.33	33.42	34.16	27.44	22.42	24.05
ZipNeRF + Ours	26.2	36.4	29.84	28.57	333.67	34.32	27.55	22.55	24.02
MipNeRF360	24.31	33.99	29.83	27.00	32.91	32.53	26.36	21.68	25.72
MipNeRF360 + NeRFLiX	24.22	34.98	30.49	26.40	33.31	32.91	26.49	21.64	25.56
MipNeRF360 + AligNeRF†	24.75	-	-	27.07	-	-	26.69	21.61	-
MipNeRF360 + Ours	24.81	35.17	30.53	27.49	34.0	33.06	26.82	22.05	26.07
Nerfacto	23.16	29.88	27.14	25.28	30.09	30.77	25.31	20.71	22.69
Nerfacto + NeRFLiX	23.26	31.92	28.71	25.62	31.86	31.78	25.59	20.79	22.75
Nerfacto + Ours	23.79	32.79	28.71	26.46	32.57	32.56	25.86	21.26	23.08

(b) SSIM

Model	Bicycle	Bonsai	Counter	Garden	Kitchen	Room	Stump	Flowers	Treehill
ZipNeRF	0.7729	0.968	0.9215	0.8631	0.9522	0.953	0.7881	0.6366	0.6749
ZipNeRF + NeRFLiX	0.7611	0.9714	0.9337	0.8245	0.9521	0.9536	0.7898	0.6257	0.6731
ZipNeRF + Ours	0.7877	0.972	0.9313	0.8732	0.956	0.9569	0.8002	0.6567	0.6842
MipNeRF360	0.6798	0.9554	0.9082	0.8130	0.9425	0.9374	0.7428	0.5786	0.6876
MipNeRF360 + NeRFLiX	0.6696	0.9656	0.9279	0.7843	0.9499	0.9442	0.7467	0.5666	0.6776
MipNeRF360 + AligNeRF†	0.7052	-	-	0.8250	-	-	0.7650	0.5880	-
MipNeRF360 + Ours	0.7225	0.9669	0.9301	0.8428	0.9565	0.9508	0.7716	0.6174	0.7136
Nerfacto	0.5286	0.9082	0.8299	0.7151	0.9005	0.9001	0.6628	0.474	0.5216
Nerfacto + NeRFLiX	0.5334	0.9448	0.8954	0.7386	0.9378	0.926	0.679	0.4768	0.5369
Nerfacto + Ours	0.5848	0.9496	0.8985	0.7976	0.9413	0.9408	0.7063	0.5398	0.571

(c) LPIPS

Model	Bicycle	Bonsai	Counter	Garden	Kitchen	Room	Stump	Flowers	Treehill
ZipNeRF	0.2299	0.0886	0.1227	0.1283	0.0726	0.1266	0.2410	0.3107	0.2811
ZipNeRF + NeRFLiX	0.2747	0.101	0.1230	0.1858	0.0831	0.1485	0.2605	0.3348	0.3292
ZipNeRF + Ours	0.2096	0.0924	0.1149	0.1189	0.0756	0.1290	0.2209	0.263	0.2715
MipNeRF360	0.3406	0.1065	0.1504	0.1911	0.0879	0.1505	0.3072	0.3763	0.3568
MipNeRF360 + NeRFLiX	0.3696	0.1098	0.1332	0.2248	0.0853	0.1615	0.3172	0.4032	0.3919
MipNeRF360 + Ours	0.2892	0.1055	0.1227	0.1516	0.0761	0.1393	0.2653	0.3206	0.3177
Nerfacto	0.4988	0.1749	0.2524	0.2956	0.1401	0.2138	0.3852	0.4646	0.5135
Nerfacto + NeRFLiX	0.5067	0.1362	0.1703	0.2696	0.0989	0.1871	0.3707	0.4742	0.5258
Nerfacto + Ours	0.4436	0.1298	0.1598	0.2077	0.0944	0.1518	0.3394	0.4044	0.4723

Table 4: Per-Scene results on the LLFF dataset. All scenes are evaluated at 4× downsampled resolution. Red / orange highlights indicate best / second method respectively.

(a) PSNR

Model	Fern	Flower	Fortress	Horns	Leaves	Orchids	Room	T-Rex
NeRF	24.90	27.76	31.43	27.55	21.06	20.24	32.62	27.03
NeRF + NeRFLiX	25.48	28.42	31.89	28.57	21.24	20.41	33.44	27.90
NeRF + Ours	25.44	28.43	32.55	29.86	22.02	20.37	33.95	28.71
TensoRF	24.37	28.87	31.41	28.98	21.46	19.45	32.77	27.70
TensoRF + NeRFLiX	25.90	29.21	31.72	29.20	21.50	20.16	33.58	27.76
TensoRF + Ours	24.76	29.27	32.05	30.51	21.85	19.62	33.87	28.71

(b) SSIM

Model	Fern	Flower	Fortress	Horns	Leaves	Orchids	Room	T-Rex
NeRF	0.7917	0.8393	0.8863	0.8342	0.6995	0.6466	0.9511	0.8873
NeRF + NeRFLiX	0.8382	0.8752	0.9045	0.8954	0.7543	0.6912	0.9638	0.9188
NeRF + Ours	0.8470	0.8791	0.9183	0.9239	0.7945	0.7039	0.9676	0.9356
TensoRF	0.7875	0.8791	0.9001	0.9006	0.7709	0.6363	0.9578	0.9136
TensoRF + NeRFLiX	0.8533	0.8916	0.9072	0.9161	0.7795	0.6842	0.9658	0.9240
TensoRF + Ours	0.8184	0.8935	0.9123	0.9364	0.8095	0.6607	0.9674	0.9376

(c) LPIPS

Model	Fern	Flower	Fortress	Horns	Leaves	Orchids	Room	T-Rex
NeRF	0.2747	0.2036	0.1546	0.2590	0.3038	0.3083	0.1639	0.2430
NeRF + NeRFLiX	0.1902	0.1490	0.1486	0.1615	0.1779	0.2224	0.1248	0.1818
NeRF + Ours	0.1799	0.1377	0.1128	0.1295	0.1469	0.2043	0.1198	0.1651
TensoRF	0.2373	0.1417	0.1266	0.1522	0.2067	0.2657	0.1402	0.1927
TensoRF + NeRFLiX	0.1693	0.1304	0.1352	0.1288	0.1510	0.2086	0.1159	0.1720
TensoRF + Ours	0.1887	0.1259	0.1206	0.1131	0.1406	0.2220	0.1179	0.1660

References

1. Barron, J.T., Mildenhall, B., Verbin, D., Srinivasan, P.P., Hedman, P.: Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In: CVPR (2022)
2. Chen, A., Xu, Z., Geiger, A., Yu, J., Su, H.: Tensorf: Tensorial radiance fields. In: ECCV (2022)
3. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: ICCV. pp. 1026–1034 (2015)
4. Jiang, Y., Hedman, P., Mildenhall, B., Xu, D., Barron, J.T., Wang, Z., Xue, T.: Alignerf: High-fidelity neural radiance fields via alignment-aware training. In: CVPR. pp. 46–55 (2023)
5. Mildenhall, B., Srinivasan, P.P., Ortiz-Cayon, R., Kalantari, N.K., Ramamoorthi, R., Ng, R., Kar, A.: Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. ACM TOG (2019)
6. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. In: ECCV (2020)
7. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: PyTorch: An Imperative Style, High-Performance Deep Learning Library. In: Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché Buc, F., Fox, E., Garnett, R. (eds.) NeurIPS. pp. 8024–8035. Curran Associates, Inc. (2019)
8. Riba, E., Mishkin, D., Ponsa, D., Rublee, E., Bradski, G.: Kornia: an open source differentiable computer vision library for pytorch. In: Winter Conference on Applications of Computer Vision (2020)
9. Tancik, M., Weber, E., Ng, E., Li, R., Yi, B., Kerr, J., Wang, T., Kristoffersen, A., Austin, J., Salahi, K., Ahuja, A., McAllister, D., Kanazawa, A.: Nerfstudio: A modular framework for neural radiance field development. In: ACM SIGGRAPH 2023 Conference Proceedings. SIGGRAPH '23 (2023)
10. Wang, Z., Cun, X., Bao, J., Zhou, W., Liu, J., Li, H.: Uformer: A general u-shaped transformer for image restoration. In: CVPR. pp. 17683–17693 (June 2022)
11. Xue, T., Chen, B., Wu, J., Wei, D., Freeman, W.: Video enhancement with task-oriented flow. IJCV **127** (08 2019)
12. Zhou, K., Li, W., Jiang, N., Han, X., Lu, J.: From nerflix to nerflix++: A general nerf-agnostic restorer paradigm. IEEE TPAMI pp. 1–17 (2023)
13. Zhou, K., Li, W., Wang, Y., Hu, T., Jiang, N., Han, X., Lu, J.: Nerflix: High-quality neural view synthesis by learning a degradation-driven inter-viewpoint mixer. In: CVPR. pp. 12363–12374 (2023)