FlowCam: Training Generalizable 3D Radiance Fields without Camera Poses via Pixel-Aligned Scene Flow

— Supplementary Material —

Cameron Smith **Yilun Du** **Ayush Tewari** **Vincent Sitzmann**
{camsmith, yilundu, ayusht, sitzmann}@mit.edu
MIT CSAIL
cameronosmith.github.io/flowcam

Contents

1 Additional Results

Below we present additional results for both video reconstruction and odometry. Please see accompanying video for novel view synthesis results.

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

1.1 Video Reconstruction

In figures 2 to 5, we plot random additional video reconstruction renderings from our model and all baselines.

1.2 Pose Estimation

In Fig. 6, we plot additional comparisons with the Video Autoencoder in short-video odometry (20 frames) on all datasets, and in Fig. 7, we plot additional comparisons with ORB-SLAM3 and DROID-SLAM on longer videos (∼150 frames) from the CO3D 10-Category dataset. We also report quantitative comparison with DROID-SLAM and ORB-SLAM3 on CO3D 10-category in Tab. **??**. As ORB-SLAM3 and DROID-SLAM predict a set of sparse poses per video, we evaluate only on the temporal overlap of their predictions and the ground truth poses, as is standard in odometry evaluation. Also note that for the "frame-density" metric reported in Tab. **??**, we define a failed tracking for a video as yielding less than 5 poses for that video, rather than measuring the pose density within sequences.

In Tab. 1, we quantitatively measure our method after fine-tuning as well as BARF [1] and NoPe-NeRF [2] on the Caterpillar sequence from the Tanks and Temples [3] dataset. In Fig. 1 we also plot the poses for visual reference.

Table 1: Tanks and Temples' Caterpillar Sequence

	Ours RE10K	Ours Fine-Tuned	BARF	NoPe NeRF
ATE	0.0543	**0.0020**	0.400	0.521

Table 2: Here we present additional quantitative pose estimation evaluations for the Tanks and Temples's "Caterpillar" trajectory for our zero-shot evaluation with out RealEstate10K-trained model, our model fine-tuned on the "Caterpillar" sequence, BARF, and NoPe NeRF.

1.3 Pose Estimation Parameterization Ablations

in Tab. 3 we perform the following ablations to our model pipeline and report the corresponding PSNR for the Hydrants dataset: pose regression via an MLP (no solver), not using flow weights (non-weighted Procrustes), using bidirectional consistency flow weights (as opposed to flow weight prediction), including a scale factor in the Procrustes solver, regressing depth via and MLP instead of using the pixelNeRF's depth estimate, using a pose solver based on 2D correspondences instead of 3D correspondences, and finally the full model formulation. We show here that all design choiecs are critical, except for the depth estimation via rendering, which we show is comparable.

	MLP Pose Regression	No Flow Weights	Bidirectional Consistency Flow Weights	Scale Adjusting Procrustes	Depth Regression	2D-Only Pose Solver	Full Model
PSNR	18.50	17.87	18.68	11.93	23.56	16.63	**23.64**

Table 3: Ablations for parameterizing our model's pose-prediction.

1.4 Intrinsics Source Ablations

In Tab. 4 we ablate different sources for camera intrinsics on the Hydrants dataset: a constant value near the average of the dataset's intrinsics, predicted values via a CNN, and the dataset-provided ground-truth. Interestingly, our CNN-predicted intrinsics score higher for view-synthesis than using the dataset's intrinsics. We leave more extensive intrinsics-prediction evaluation and formulation to future work.

	Constant	Predicted	Ground Truth
PSNR	23.15	**25.6**	23.64

Table 4: Intrinsics-source ablations.

1.5 Scale Ambiguity Illustration

In Tab. 5 we randomly scale the model's estimated poses various amounts and report the corresponding rendering quality on the Hydrants dataset to illustrate the difficulty posed by a scale discrepancy between the coordinate system of the poses and that predicted implicitly by the scene representation. Though pixelNeRF's multiview rendering can partially compensate for scale perturbations, the rendering quality is negatively affected by the scale magnitude.

Scale Range	.10-10	.25-4	.5-2	.75-1.33
PSNR	18.71	19.23	19.47	20.56

Table 5: Scale ambiguity demonstration.

1.6 Additional Flow Weights and Tanks and Temples Pose Plots

In Fig. 1 we plot an example where our model's flow mask prediction ignores a dynamic object. Note that since our model only needs theoretically needs a few points to model camera motion, the weights can be relatively sparse and do not explicitly correspond to "dynamic object" masks.

2 Reproducibility

2.1 Hardware

We train our models on a single Tesla-V100 GPU (32GB memory).

2.2 Architecture Details

Our CNN image encoder follows [4] with a pre-trained ResNet34 feature pyramid encoder, though we modify the first encoder's first convolutional layer to accept optical flow channels as well as RGB. Our renderer, which maps 3D query points and image features to density and color, is implemented as 6 layer MLP with 64 hidden units, with FILM [5] conditioning in the first four layers instead of concatenation. Also following pixelNeRF, we only use the feature-wise addition, as opposed to scaling features as well. We train a separate linear conditioning layer mapping per network level. Our renderer does not use any view-dependent conditioning. Our flow confidence predictor, which maps concatenated image features to a confidence score for optical flow correspondence, is implemented as a two-layer 128-unit MLP which accepts the concatenated CNN image features. Recall that the purpose of this network is to assign weights to each scene flow vector for more robust pose solving.

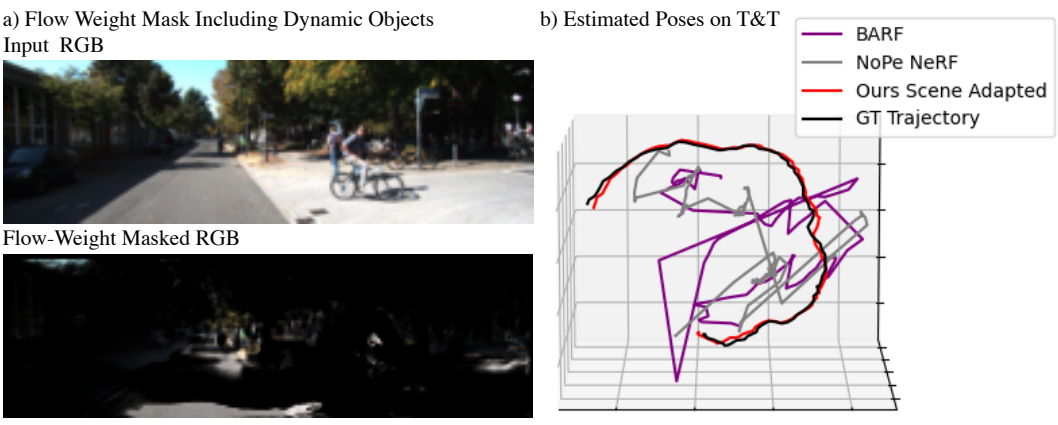

Figure 1: In (**a**) we present an example where our model's flow weights mask includes a bicyclist in motion. In (**b**) we plot the poses estimated by our model and additional baselines on the Tanks and Temples Caterpillar sequence.

2.3 Training Details

We train each dataset for 1 to 2 days, using a constant learning rate of 0.0001 and the Adam optimizer. The CO3D-10Category model is initialized with the Hydrants model. We first train with a batch size of 2 and video length 6, and then train with a single batch of video length 12.

2.4 Dataset Details

For the two CO3D datasets (Hydrant and 10-Category), we use the second dataset release version, and randomly skip 1 or 2 frames when training for larger baseline. On RealEstate10K, we skip 9 frames, and on KITTI, we do not skip any frames. We note that manually defining an "appropriate" frameskip amount per dataset is not particularly scalable, and a more principled way to handle this problem is to dynamically determine the frame skip, per sequence, via the average amount of optical flow in the image. We employ this dynamic video definition on the YouTube and Ego4D videos.

3 Baseline Details

Below we describe model details for all model comparisons.

3.1 Video Autoencoder

For the Video Autoencoder, we initialize training with their RealEstate10K model and train with a batch size of 7 videos of clip length 6 for 1 to 2 days. We tried offering optical flow as additional input channels to their CNN encoders, but found training was unstable and just fine-tuned their pretrained model.

3.2 RUST

Since no code release exists for RUST, We implement RUST by following their writeup and modifying the endorsed code base for the Scene Representation Transformer [6]. Training is performed using a batch size of 12 videos of length 9 frames for 1 to 2 days.

3.3 BARF

We use the official code base for BARF and optimize each scene for about 12 hours.

3.4 ORB-SLAM3

We use the official ORB-SLAM3 release and use the monocular RGB mode of operation. We found that the default settings for running ORB-SLAM3 suffered in the low textured scenes in CO3D. We therefore increased the number of features in ORB-SLAM3 to 4000 and initial and minimum fast thresholds to 5 and 1 respectively to improve performance on low textured scenes.

3.5 DROID-SLAM

We use the official code base for DROID-SLAM and their pretrained model, i.e., we do not fine-tune it on the CO3D datasets. We use two bundle-adjustment iterations (the default setting) per scene.

References

[1] Chen-Hsuan Lin, Wei-Chiu Ma, Antonio Torralba, and Simon Lucey. Barf: Bundle-adjusting neural radiance fields. In *Proc. CVPR*, 2021.

[2] Wenjing Bian, Zirui Wang, Kejie Li, Jiawang Bian, and Victor Adrian Prisacariu. Nope-nerf: Optimising neural radiance field with no pose prior. 2023.

[3] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *Proc. TOG*, 36(4), 2017.

[4] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelNeRF: Neural radiance fields from one or few images. In *Proc. CVPR*, 2021.

[5] Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[6] Mehdi SM Sajjadi, Henning Meyer, Etienne Pot, Urs Bergmann, Klaus Greff, Noha Radwan, Suhani Vora, Mario Lucic, Daniel Duckworth, Alexey Dosovitskiy, et al. Scene representation transformer: Geometry-free novel view synthesis through set-latent scene representations. *arXiv preprint arXiv:2111.13152*, 2021.

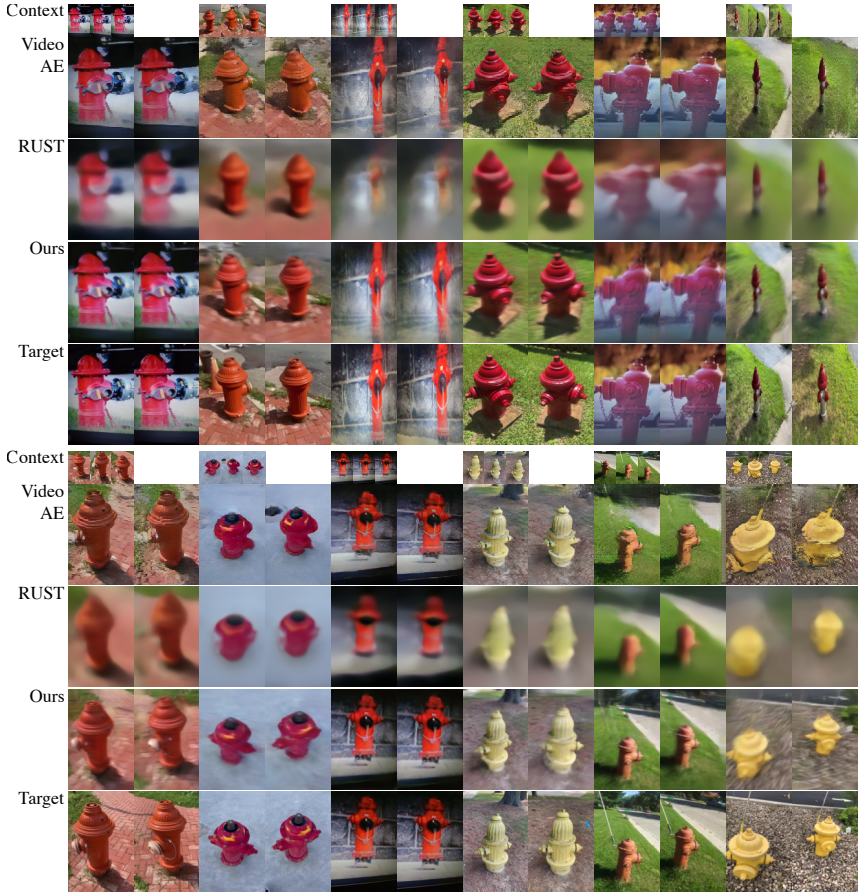

Figure 2: **Additional Results on CO3D Hydrants.**

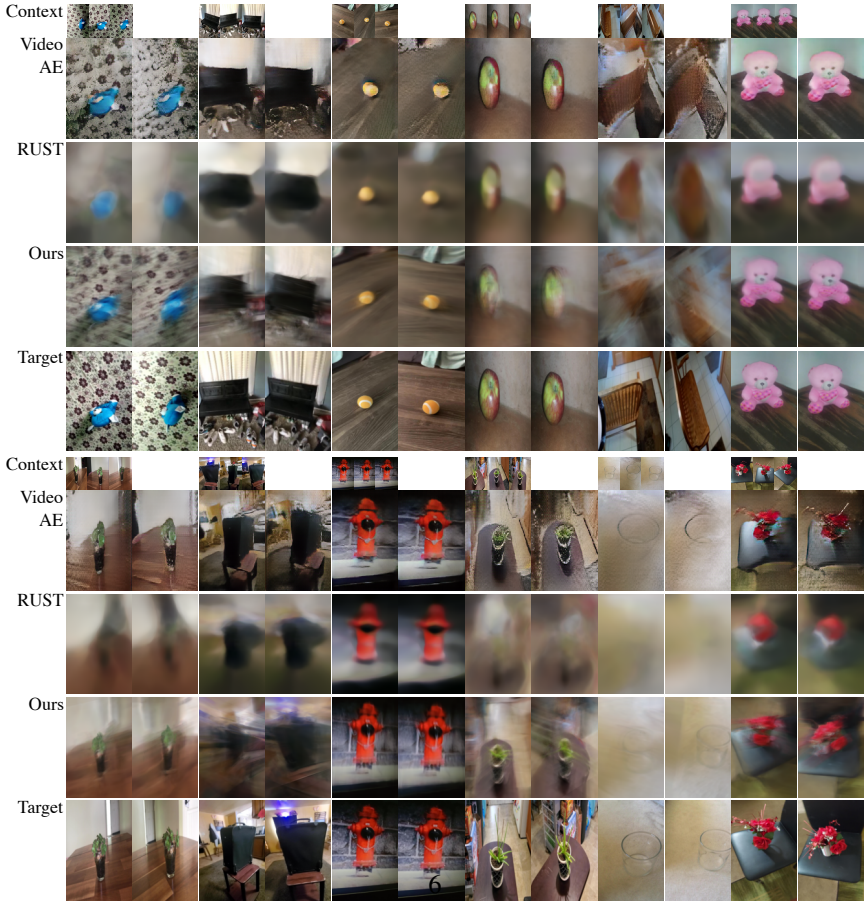

Figure 3: **Additional Results on CO3D 10Category.**

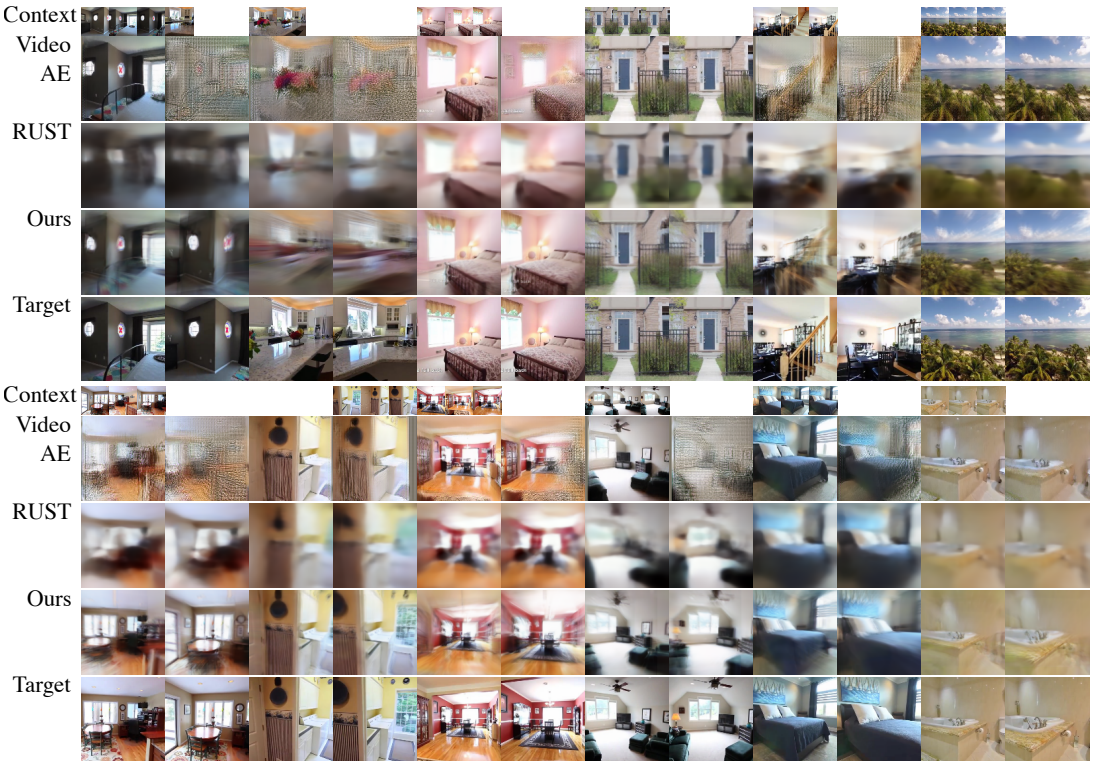

Figure 4: **Additional Results on RealEstate10K.**

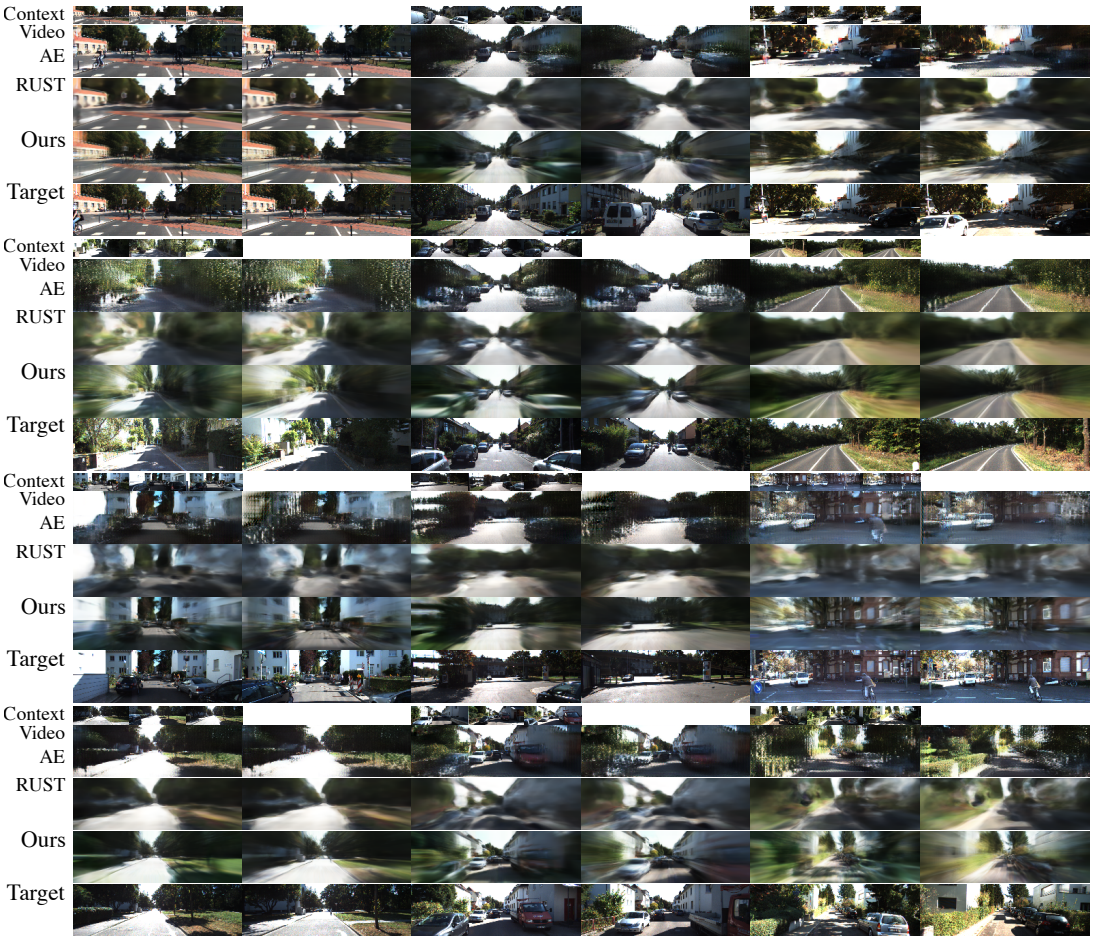

Figure 5: **Additional Results on KITTI.**

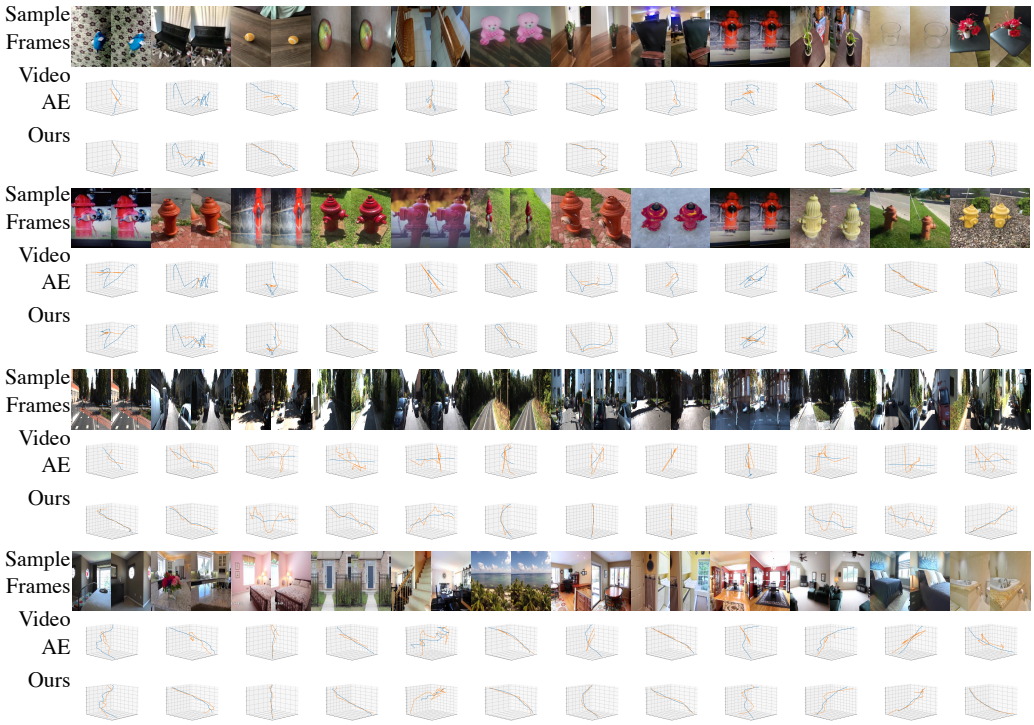

Figure 6: **Additional Odometry Comparisons with Video Autoencoder on 20 Frame Sequences** Please zoom in to individual sequences for easier viewing.

Figure 7: **Additional Odometry Comparisons with ORB-SLAM3 and DROID-SLAM on $\sim$ 150 Frame Sequences** Please zoom in to individual sequences for easier viewing.