Forward Flow for Novel View Synthesis of Dynamic Scenes
– Supplementary Material –

Xiang Guo[1], Jiadai Sun[1,3], Yuchao Dai[†1], Guanying Chen[2], Xiaoqing Ye[3], Xiao Tan[3], Errui Ding[3],
Yumeng Zhang[3], Jingdong Wang[3]

[1]Northwestern Polytechnical University, [2]FNii and SSE, CUHK-Shenzhen, [3]Baidu Inc.

Table 1. **Preliminaries in the paper.**

Variables	Description
$\mathbf{V}_{\text{Rf}}^{\text{Can}}$	voxel grid contains canonical radiance feature
$\mathbf{V}_{\text{p}}^{\text{Can}}$	voxel grid contains coordinate of each voxel
F_{θ_1}	light weight MLP network estimating canonical radiance field
$\mathbf{V}_{\text{R}}^{\text{Can}}$	canonical radiance field estimated by F_{θ_1}
$\mathbf{V}_{\sigma}^{\text{Can}}$	canonical density voxel grid in $\mathbf{V}_{\text{R}}^{\text{Can}}$
$\mathbf{V}_{\text{cf}}^{\text{Can}}$	canonical color feature voxel grid in $\mathbf{V}_{\text{R}}^{\text{Can}}$
$\mathbf{V}_{\text{Df}}^{\text{Can}}$	voxel grid contains canonical deformation feature
F_{θ_2}	light weight MLP network estimating trajectory of each voxel
$\mathbf{V}_{T}^{\text{Can}}$	trajectory voxel grid, containing DCT params of each voxel
$\mathbf{V}_{\text{flow}}^{t}$	deformation flow of each voxel from canonical to time t
F_{warp}	average splatting operation
$\mathbf{V}_{\text{R}}^{t}$	radiance field at time t warped from $\mathbf{V}_{\text{R}}^{\text{Can}}$
F_{θ_3}	Inpaint Network
$\mathbf{V}_{\text{R}_{\text{Inp}}}^{t}$	inpainted radiance field at time t by F_{θ_3}
$\mathbf{V}_{\text{R}_{\text{Up}}}^{t}$	upsampled radiance field at time t
F_{render}	volume rendering function
$\mathbf{C}_{\text{Inp}}(\mathbf{r})$	color of ray r rendered from field $\mathbf{V}_{\text{R}_{\text{Inp}}}^{t}$
$\mathbf{C}_{\text{Up}}(\mathbf{r})$	color of ray r rendered from field $\mathbf{V}_{\text{R}_{\text{Up}}}^{t}$

1. More Implementation Details

1.1. Preliminaries

We provide preliminaries in Table 1

1.2. Network Architecture

There are four networks in proposed network: canonical radiance network F_{θ_1}, canonical deformation network F_{θ_2}, view dependent color network F_{θ_4} and inpaint network F_{θ_3}.

Canonical radiance network F_{θ_1} is a 3 layer MLP, with width set to 128. The input contains the radiance feature sampled from radiance feature grid $\mathbf{V}_{\text{Rf}}^{\text{Can}}$ with dimension 12, and embedded position with dimension 33. The output contains density whose dimension is 1, and color feature whose dimension is 3 or 12, depending the training stage.

Canonical deformation network F_{θ_2} is a 4 layer MLP, with width set to 64. Similar with canonical radiance network, the input of F_{θ_2} contains the deformation feature and embedded position with dimension 33. We set the deformation feature dimension to be 0 and the number of dct bases to be 15 for D-NeRF dataset. For Hypernerf dataset, we set the deformation feature dimension to be 12 and the number of dct bases to be 25.

View dependent color network F_{θ_4} is a simpler MLP with only 2 layers and width is 64. For input, the dimension of embedded view direction is 27 and color feature is 12. The output is the rgb color with dimension 3.

Inpaint network F_{θ_3} consist of a UNet structure and a upsample layer. The UNet structure has 4 encode layers and 3 decode layers. Each encode layer consists of one max pooling layer and two convolution layers. For one convolution layer, there is one instance norm, followed by convolution and ReLU activation. Each decode layer consists of one up-sample interpolation layer and two convolution layers, which are the same with encode layer. The up-sample layer has the same structure with the decode layer.

1.3. Average Splatting vs. Softmax Splatting

We use average splatting in this paper, rather then more complex splatting methods in [3], like softmax splatting. We tried to predict weights for softmax splatting, and there is no obvious improvement compared with average splatting. It may introduce too much complexity and we find the UNet could refine the voxel grid to some extend. Average splatting is enough in our setting.

[†] Corresponding author `(daiyuchao@gmail.com)`. The first three authors also with the Shaanxi Key Laboratory of Information Acquisition and Processing.

1.4. Losses and Hyper-parameter Settings

Following DVGO [6], we use $\mathcal{L}^{\text{ptc}}$ to directly supervise the color of sampled points. The intuition is that sampled points with bigger weights contribute more to the rendered color.

$$\mathcal{L}^{\text{ptc}} = \frac{1}{|\mathcal{R}|} \sum_{r \in \mathcal{R}} \mathcal{L}(\mathbf{C}_{\text{Inp}}(\mathbf{r})) + \frac{1}{|\mathcal{R}|} \sum_{r \in \mathcal{R}} \mathcal{L}(\mathbf{C}_{\text{Up}}(\mathbf{r})), \quad (1)$$

$$\mathcal{L}(\mathbf{C}(\mathbf{r})) = \frac{1}{K} \sum\nolimits_{k=1}^{K} A_{\text{accum}}(k) \left\| \mathbf{c}(w_k) - \mathbf{C}_{\text{gt}}(\mathbf{r}) \right\|_2^2 , \quad (2)$$

$$A_{\text{accum}}(k) = T(w_k)\, \alpha\, (\sigma(w_k)\delta_k) \, . \quad (3)$$

Also, we use the background entropy loss $\mathcal{L}^{\text{bg}}$ to encourage the densities concentrating on either the foreground or the background.

$$\mathcal{L}^{\text{bg}} = \frac{1}{|\mathcal{R}|} \sum_{r \in \mathcal{R}} \mathcal{L}(\mathbf{A}_{\text{Inp}}(\mathbf{r})) + \frac{1}{|\mathcal{R}|} \sum_{r \in \mathcal{R}} \mathcal{L}(\mathbf{A}_{\text{Up}}(\mathbf{r})), \quad (4)$$

$$\begin{aligned} \mathcal{L}(\mathbf{A}(\mathbf{r})) = -\frac{1}{K-1} \sum\nolimits_{k=1}^{K-1} & A_{\text{accum}}(k) log(A_{\text{accum}}(k)) \\ & + (1 - A_{\text{accum}}(k)) log(1 - A_{\text{accum}}(k)), \end{aligned} \quad (5)$$

As show in Eq. (15) of the paper, the overall loss can be written as

$$\begin{aligned} \mathcal{L} = & \mathcal{L}^{\text{photo}} + w_1 \mathcal{L}^{\text{ptc}} + w_2 \mathcal{L}^{\text{bg}} + w_3 \mathcal{L}^{\text{flow}} + w_4 \mathcal{L}^{\text{vdiff}} \\ & + w_5 \mathcal{L}^{\text{tv}}(\mathbf{V}_{\sigma}^{\text{Can}}) + w_6 \mathcal{L}^{\text{tv}}(\mathbf{V}_{\text{flow}}^{t}) + w_7 \mathcal{L}^{\text{tv}}(\mathbf{D}), \end{aligned} \quad (6)$$

where w_1, w_2, w_3, w_4, w_5, w_6 and w_7 are weights to balance each component in the final coarse loss. In experiments, we set $w_1 = 1e-1$, $w_2 = 1e-2$, $w_3 = 1e-5$, $w_4 = 0.$, $w_5 = 1e-6$, $w_6 = 1e-3$ and $w_7 = 1e-1$ in coarse stage for all datasets, and set $w_1 = 1e-2$, $w_2 = 1e-3$, $w_3 = 1e-5$, $w_4 = 1e-5$, $w_5 = 1e-6$, $w_6 = 1e-3$ and $w_7 = 1e-1$ in fine stage for all datasets.

For progressive training in fine stage, we first train with 10 images with closest time steps with canonical step, and progressively add image with the closest time step every 60 iterations.

1.5. Training Strategy and Settings

We propose a coarse-to-fine training strategy. For the coarse stage, we set the expected voxel number to 67^3, the scale factor of the inpaint network is 1.5, and the iteration number is 20k. As the purpose of the coarse stage is learning a proxy geometry to calculate the bounding box for the fine stage, we do not use inpaint network during the coarse stage. For the fine stage, the expected voxel number is set to 80^3 for D-NeRF dataset and 70^3 for HyperNeRF dataset. The scale factor of the inpaint network is 2.0 and the iteration number is 100k. We use Adam optimizaer and set learning rate 1e-1 for voxel grids and 1e-3 for networks. The training process of a scene takes around 1 day on a GeForce RTX 3090 GPU. Our final model size on average is 260M for D-NeRF dataset and 440M for HyperNeRF dataset. The rendering peed is 7s per image for D-NeRF dataset and 15s per image for HyperNeRF dataset.

1.6. Lego Complete Dataset

We build a new dataset, named *Lego Complete Dataset*, that animates the object *LEGO* with three different motion patterns. For each scene, the test set is split into three categories to evaluate three abilities: *space interpolation*, *time interpolation*, and *canonical interpolation* abilities. For space interpolation ability, we test four random views for each training time step. Also, we interpolate three time steps between two near training time steps to test the model's generalization ability over unseen times. Finally, to evaluate the learned canonical radiance field, we test 50 random views in canonical space. This results in $200 + 197 + 50 = 447$ test images in the test set.

2. More Results

2.1. Results on NHR dataset

We also test our method on NHR dataset [7]. We test all four scenes with 100 frames selecting 90% views for train and 10% views for test. Quantitative results are shown in Table 2, which shows ours achieving best results consistently. We show qualitative comparison in Figure 1, and our method could render clean and detailed images. We also visualize the learnt trajectories in Figure 2. Use Acrobat to view animations.

2.2. Quantitative Results

Detailed results for D-NeRF Dataset We show more detailed results in Table 3 for D-NeRF dataset. As show in Table 3, the inpaint network plays a relative important role. Also, the up-sample layer could improves the performance, which proves this layer learns to recover the details of the 3D voxel grid. This point gives some insight for image super-resolution direction, which is working in 3D dimension may benefit the 2D image tasks. Finally, the total variation losses helps the training to be more stable and get cleaner images.

Resolutions We report performance with different resolutions on HyperNeRF dataset in Figure 3. According to Figure 3, the resolution of voxel grid plays an important role when it is relative small and the improvement of the performance decrease when resolution increasing.

Table 2. **Quantitative comparison on NHR dataset.** The **red** text indicates the best and **blue** text is the second best result.

		Sport 1			Sport 2			Sport 3			Bacsketball			Mean		
Methods	type	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
TiNeuVox-S [1]	NPC	26.06	**0.93**	**0.10**	25.98	0.93	0.11	25.90	**0.93**	**0.11**	23.75	0.91	0.14	25.42	0.92	0.12
TiNeuVox-B [1]	NPC	**26.44**	**0.93**	**0.10**	**26.68**	**0.94**	**0.10**	**26.09**	**0.93**	**0.11**	**25.06**	**0.92**	**0.12**	**26.07**	**0.93**	**0.11**
NDVG [2]	PC	23.66	0.89	0.15	24.43	0.91	0.13	22.54	0.88	0.16	22.55	0.89	0.17	23.29	0.89	0.15
Ours	PC	**27.71**	**0.95**	**0.08**	**27.89**	**0.95**	**0.08**	**27.57**	**0.94**	**0.08**	**24.85**	**0.93**	**0.11**	**27.00**	**0.94**	**0.09**

Table 3. **Quantitative comparison.** Comparison of our method with others on LPIPS (lower is better) and PSNR/SSIM (higher is better) on eight dynamic scenes of the D-NeRF dataset.

		Hell Warrior			Mutant			Hook			Bouncing Balls		
Methods	type	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
TNeRF [5]	N	23.19	0.93	0.08	30.56	0.96	0.04	27.21	0.94	0.06	32.01	0.97	0.04
TiNeuVox-S (100^3) [1]	NPC	27.00	0.95	0.09	31.09	0.96	0.05	29.30	0.95	0.07	39.05	0.99	0.06
TiNeuVox-B (160^3) [1]	NPC	28.17	0.97	0.07	33.61	0.98	0.03	31.45	0.97	0.05	40.73	0.99	0.04
DNeRF [5]	PC	25.03	0.95	0.07	31.29	0.98	0.03	29.26	0.97	0.12	38.93	**0.99**	0.10
NDVG [2]	PC	25.53	0.95	0.07	**35.53**	**0.99**	**0.01**	29.80	0.97	**0.04**	34.58	0.97	0.11
Ours	PC	**27.71**	**0.97**	**0.05**	34.97	0.98	0.03	**32.29**	**0.98**	**0.04**	**40.02**	**0.99**	**0.04**
Ours_notv [1]	PC	27.96	0.97	0.05	35.26	0.98	0.03	29.57	0.96	0.06	40.40	0.99	0.05
Ours_noup [2]	PC	27.60	0.96	0.06	34.15	0.98	0.04	31.51	0.97	0.04	38.89	0.99	0.05
Ours_noinp [3]	PC	27.15	0.96	0.06	33.98	0.98	0.03	31.77	0.97	0.04	38.22	0.99	0.04

		Lego			T-Rex			Stand Up			Jumping Jacks		
Methods	type	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
TNeRF [5]	N	23.82	0.90	0.15	30.19	0.96	0.13	31.24	0.97	0.02	32.01	0.97	0.03
TiNeuVox-S (100^3) [1]	NPC	24.35	0.88	0.13	29.95	0.96	0.06	32.89	0.98	0.03	32.33	0.97	0.04
TiNeuVox-B (160^3) [1]	NPC	25.02	0.92	0.07	32.70	0.98	0.03	35.43	0.99	0.02	34.23	0.98	0.03
DNeRF [5]	PC	21.64	0.84	0.17	**31.76**	**0.98**	**0.04**	32.80	0.98	**0.02**	32.80	**0.98**	0.04
NDVG [2]	PC	25.23	0.93	**0.05**	30.15	0.97	0.05	34.05	0.98	**0.02**	29.45	0.96	0.08
Ours	PC	**25.27**	**0.94**	**0.05**	30.71	0.96	**0.04**	**36.91**	**0.99**	**0.02**	**33.55**	**0.98**	**0.03**
Ours_notv [1]	PC	24.33	0.89	0.11	35.02	0.99	0.02	37.01	0.99	0.02	35.14	0.98	0.03
Ours_noup [2]	PC	25.20	0.93	0.06	30.24	0.97	0.05	35.47	0.98	0.02	33.14	0.98	0.04
Ours_noinp [3]	PC	25.42	0.93	0.06	30.00	0.97	0.04	36.46	0.99	0.02	31.24	0.98	0.04

[1] not use the three total variation losses [2] not up-sample the voxel grid [3] not use the inpaint network

Table 4. **More results.** Mean of Hell Warrior, Mutant, Hook and Bouncing Balls in DNeRF dataset.

Method	ours	w/o $\mathcal{L}^{\text{tv}}$	w/o $\mathcal{L}^{\text{flow}}$	w/o $\mathcal{L}^{\text{vdiff}}$	w/o coarse	can-time t at mid	w/o view dir	w/o $\mathbf{V}_{R_{\text{Up}}}$ photo
PSNR	33.75	33.30	33.74	33.57	31.63	34.09	22.81	31.56
SSIM	0.979	0.974	0.978	0.977	0.967	0.980	0.925	0.966
LPIPS	0.040	0.048	0.039	0.041	0.058	0.037	0.096	0.057

Regularization terms We study the effect of all regularization terms we proposed, including $\mathcal{L}^{\text{flow}}$, $\mathcal{L}^{\text{vdiff}}$ and $\mathcal{L}^{\text{tv}}$ in Table 4. We could observe that all these three regularization terms has positive effect on the performance, but the improvement is minor. This proves the improvement of our method compared with others come from the forward warping desgin we proposed in the paper. Also, backward flow based method NDVG [2] use similar regularization terms with ours, and our method has clear advantage compared with NDVG [2].

Training strategy We also set our method without coarse stage training, show in Table 4 (w/o coarse). The performance drops significantly that is reasonable, because the voxel grid covers bigger space without filtering with proxy geometry trained by coarse training.

Canonical setup In our method, we set canonical time

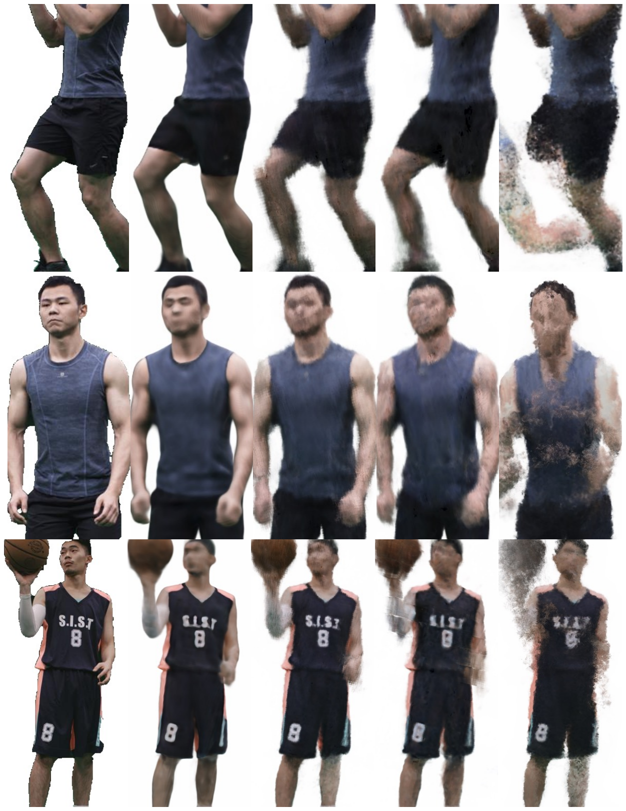


Figure 1. **NHR dataset qualitative comparison.** We show some synthesized images on NHR dataset.

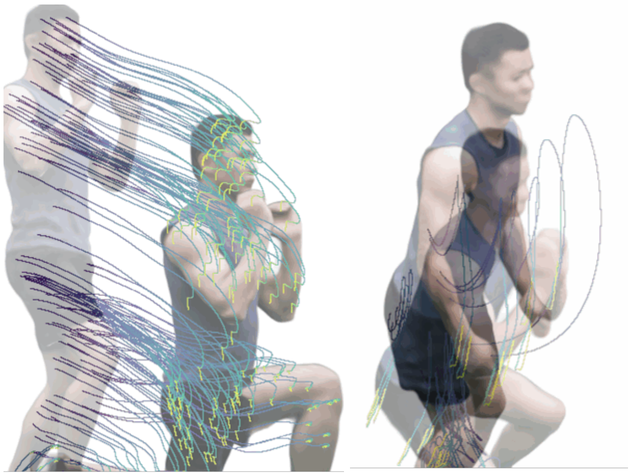


Figure 2. **Trajectory visualization** Use Acrobat to view animations.

to be the first frame for D-NeRF dataset to compare with *physical canonical based method* and the middle frame for HyperNeRF dataset for better performance. Setting canonical time to be the middle frame helps improving the performance as the state of the middle frame geometry is closer to other time steps compared with the first frame. To prove this, we set canoical time to be middel frame in Table 4 (can-time t at mid), and the PSNR improves slightly.

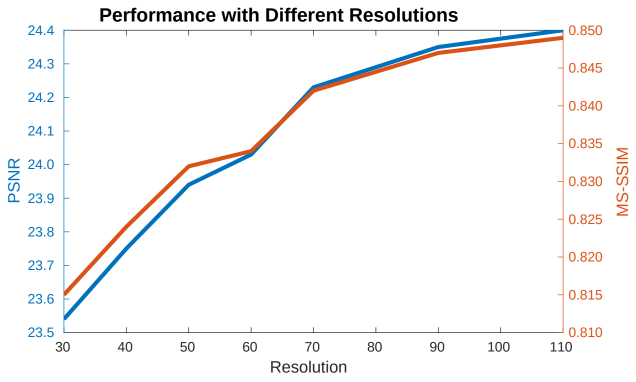


Figure 3. **Performance with different resolutions**

Ray direction modeling According to Nerfies[4], we need to transfer the current view orientation to the directions in the canonical workspace. But we think using them directly is an acceptable solution for most papers in this area. We leave this as an open question for further study. We test with setting all directions to (0,0,1), the PSNR drops obviously in Table 4 (w/o view dir). This proves the current solution works well to some extend.

Photmetric loss for $\mathbf{V}_{\mathbf{R}_{\mathbf{Up}}}$ We use photometric terms on $\mathbf{V}_{\mathrm{R}_{\mathrm{Up}}}$ to make sure the warped grid before inpainting could already render reasonable images. This make sure the UNet is actually doing 'inpainting'. Figure 5 (bottom right) shows an example of how inpainting works. Also, we do observe inpainting and upsampling could refine grids. For videos and Fig. 8 in the main paper, the trajectories are reasonable which means we do learn trajectories without inpainting overfitting. We also test w/o photometric terms of $\mathbf{V}_{\mathrm{R}_{\mathrm{Up}}}$ in Table 4 (w/o $\mathbf{V}_{\mathrm{R}_{\mathrm{Up}}}$ photo) and the performance drops as there is no direct supervise signal for trajectory training.

2.3. Qualitative Results

We show rendered images of our method with different resolutions on HyperNeRF dataset in Figure 4. With bigger resolutions, our method could recover more details, like the pattern of the 3D printer (first row), details of broom (second row) and details of the head in peel-banana (last row). Also, we provide more visual comparison with other methods in Figure 4.

We show the rendered images and depths of our methods and D-NeRF[5] in Figure 5, compared with ground truth. Since we aims to synthesis dynamic scenes from monocular camera, which is a nontrivial problem, the model is highly possible to over-fit the training images. In Figure 5, DNeRF is an example, which produce some clouds in the space which cause artifacts in other views. This is one of the reasons to build lego complete dataset, testing the abilities of the model to interpolate the time and space (including

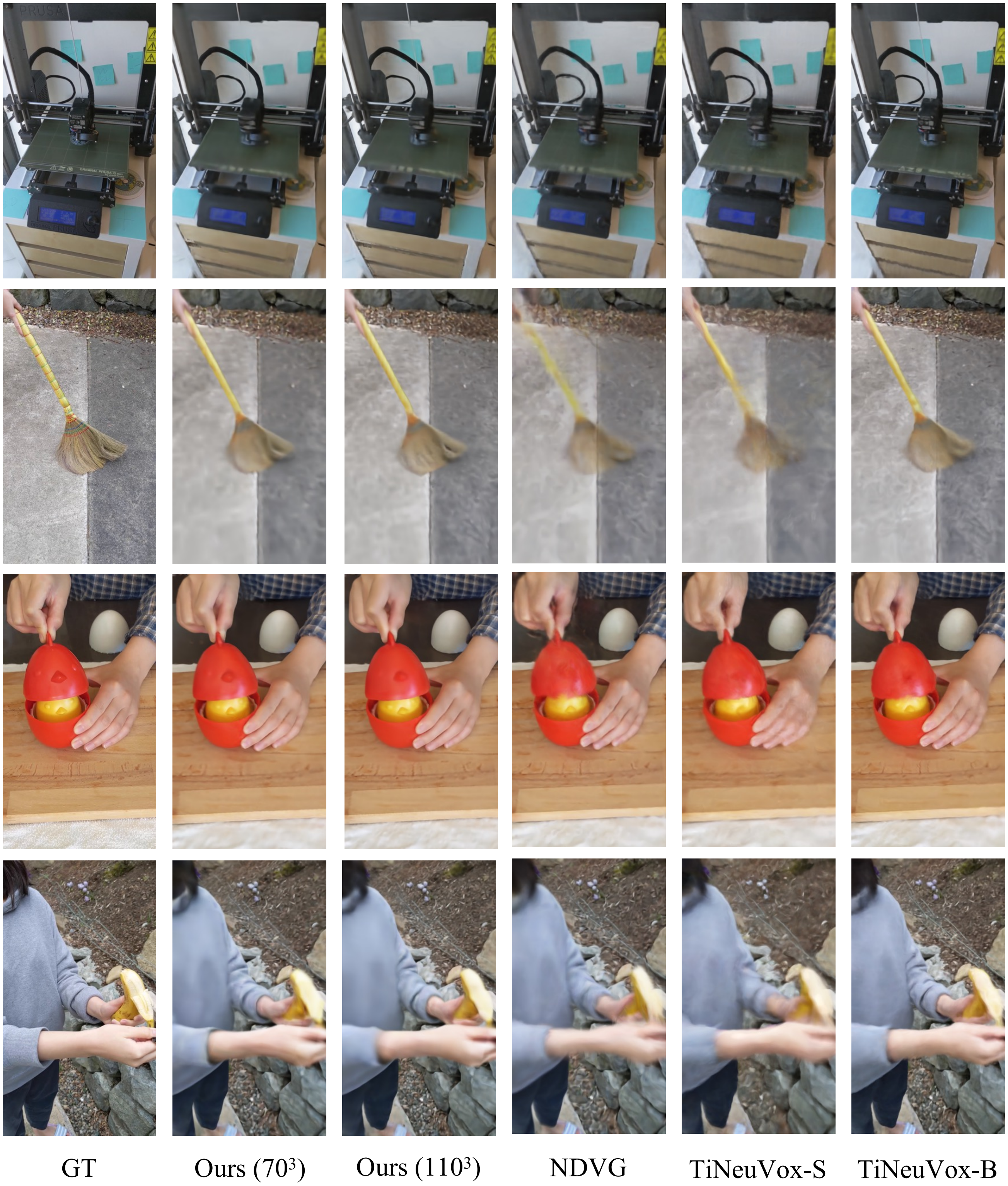
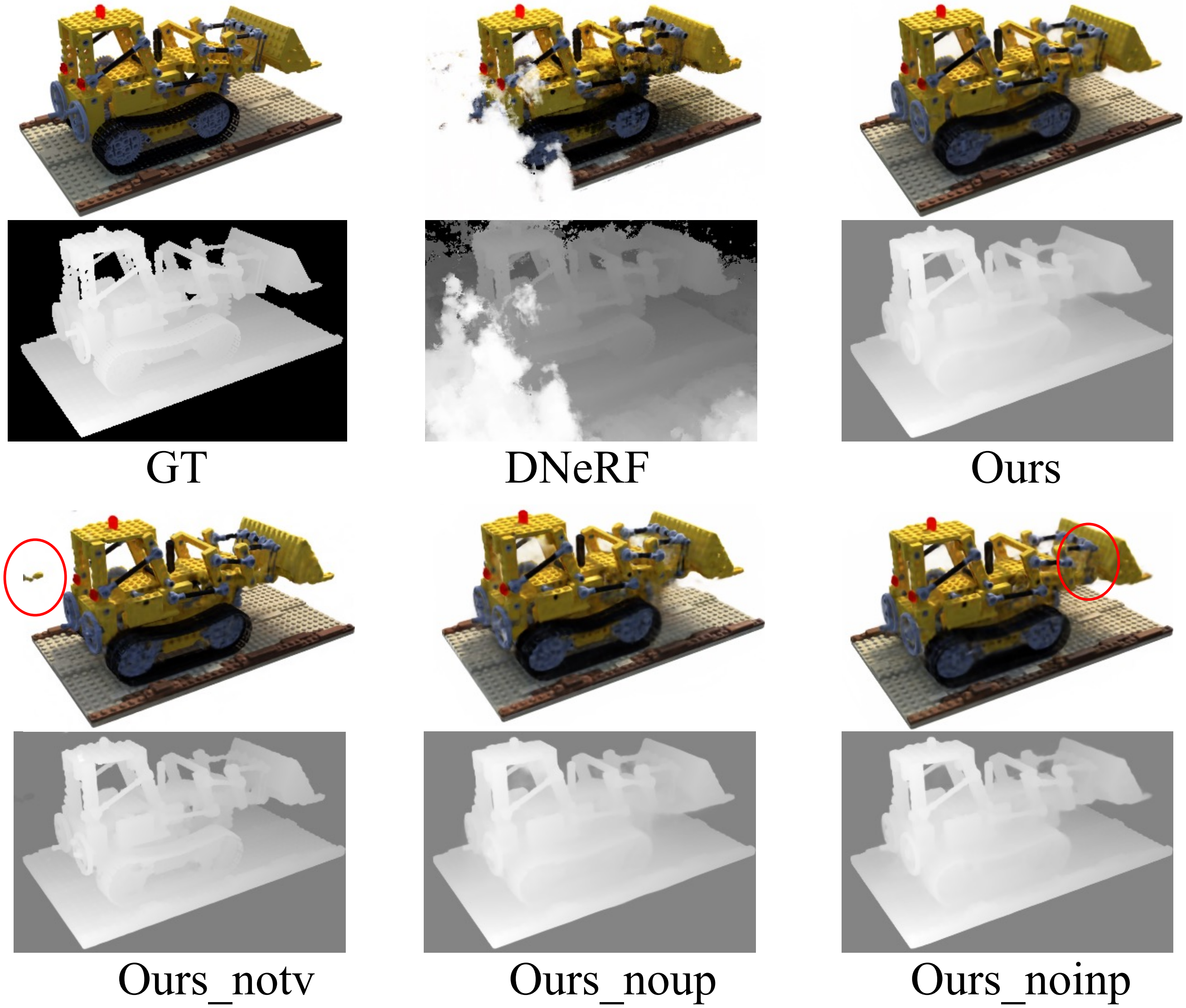


Figure 4. **HyperNeRF dataset qualitative comparison.** We show some synthesized images on HyperNeRF dataset of our method with different resolutions and other methods.

canonical space). Without total variation losses, we could get sharper depth but there may some noise points on the images. Without up-sample layer, the image is blurer (better zoom in for details). Without inpaint network, the rub-



GT DNeRF Ours

Ours_notv Ours_noup Ours_noinp

Figure 5. **Lego Complete dataset qualitative comparison.** We show some synthesized images and depth rendered at the same test view selected from one scene of lego complete dataset with different settings.

ber band of the lego arms disappears. The rubber band at this time step is stretched and this motion is non-rigid. This non-rigid motion would cause one-to-many issue, compared with rigid motions of mechanical structures of this lego. This proves our inpaint network could handle the one-to-many issue of forward warping.

We show the comparison of canonical image between D-NeRF[5] and proposed method in Figure 6. Figure 6 shows our method could recover correct canonical geometry in the canonical compared with DNeRF[5], which shows the power and potential of the forward warping.

We show more results in our video, including canonical comparison, trajectory visualization and other images render at novel views with different setting. Please refer to the supplement video for more information.

References

[1] Jiemin Fang, Taoran Yi, Xinggang Wang, Lingxi Xie, Xiaopeng Zhang, Wenyu Liu, Matthias Nießner, and Qi Tian. Fast dynamic radiance fields with time-aware neural voxels. *ACM SIGGRAPH Asia*, 2022. 3

[2] Xiang Guo, Guanying Chen, Yuchao Dai, Xiaoqing Ye, Jiadai Sun, Xiao Tan, and Errui Ding. Neural deformable voxel grid for fast optimization of dynamic view synthesis. In *Proceedings of the Asian Conference on Computer Vision (ACCV)*, 2022. 3

[3] Simon Niklaus and Feng Liu. Softmax splatting for video frame interpolation. In *Proceedings of the IEEE Conference*

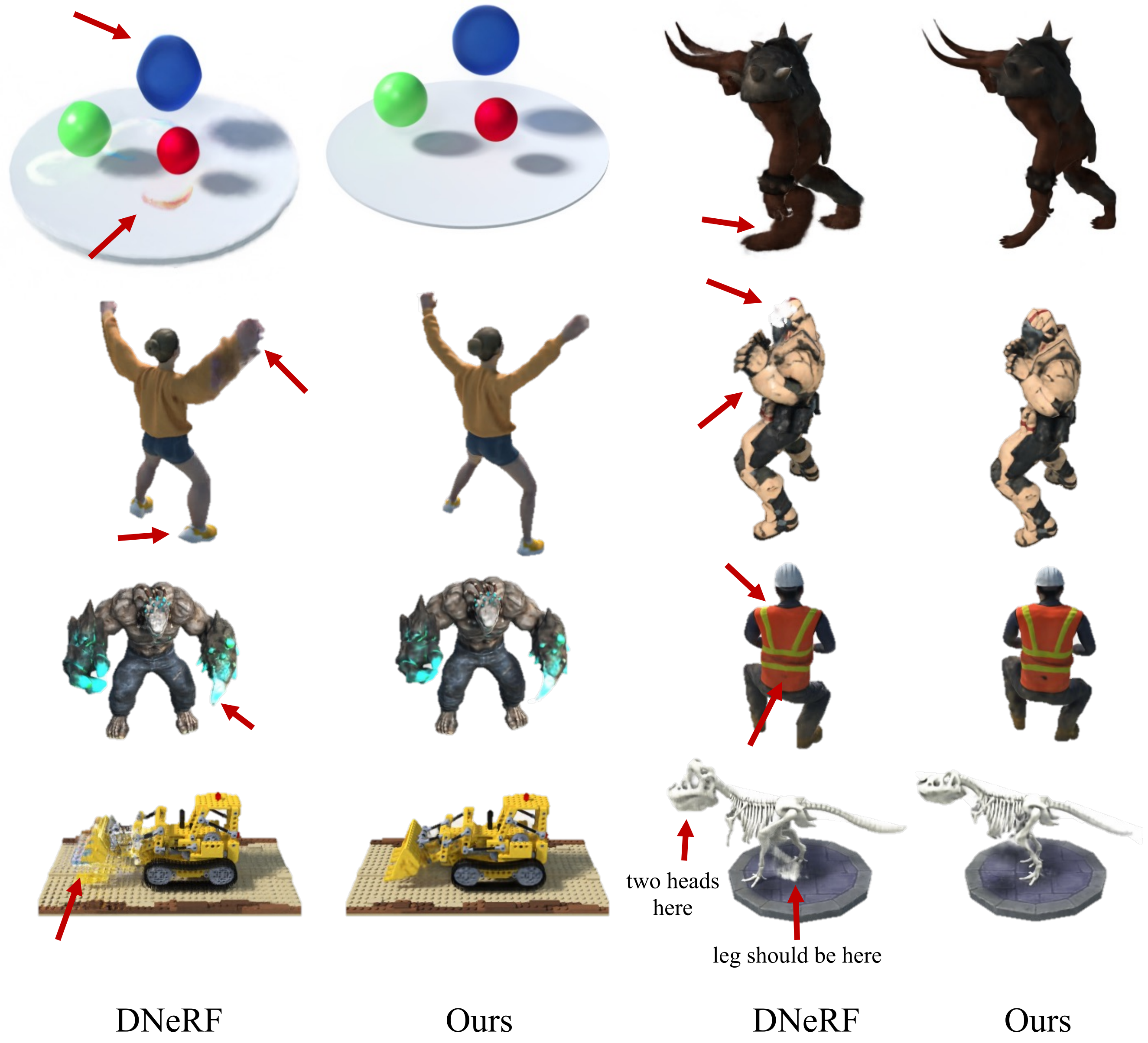


Figure 6. **Canonical qualitative comparison.** We show canonical comparison of the DNeRF[5] dataset. Note the differences highlighted by the arrows.

on Computer Vision and Pattern Recognition (CVPR), 2020. 1

[4] Keunhong Park, Utkarsh Sinha, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Steven M Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable neural radiance fields. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2021. 4

[5] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 3, 4, 6, 7

[6] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct Voxel Grid Optimization: Super-fast convergence for radiance fields reconstruction. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2

[7] Minye Wu, Yuehao Wang, Qiang Hu, and Jingyi Yu. Multi-view neural human rendering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2