Supplementary Material for "Neural Deformable Voxel Grid for Fast Optimization of Dynamic View Synthesis"

Xiang Guo[1*], Guanying Chen[2*], Yuchao Dai[1†], Xiaoqing Ye[3], Jiadai Sun[1], Xiao Tan[3], and Errui Ding[3]

[1] Northwestern Polytechnical University
[2] FNii and SSE, CUHK-Shenzhen
[3] Baidu Inc.
{guoxiang,sunjiadai}@mail.nwpu.edu.cn, chenguanying@cuhk.edu.cn, daiyuchao@nwpu.edu.cn, {yexiaoqing,dingerrui}@baidu.com, tanxchong@gmail.com

To the best of our knowledge, we are one of the first to speed up the training of dynamic NeRF by integrating the voxel-grid optimization with a deformable radiance field. Note that making this idea work is non-trivial. One hand, it is difficult to optimize the voxel gird to achieve reasonable results due to its discontinuity nature, which requires us to design different constraints while keeping the model capacity. On the other hand, to further speed up the training, we design a coarse-to-fine strategy with filtering strategies specifically designed for dynamic scenes. We believe our method, which can speed up training by 70× while maintaining comparable rendering quality, is a good step forward in the fast optimization of dynamic view synthesis. Here, we provide more details and analysis of the design, followed with more experiment results, to further discuss the method we proposed.

1 More Details for the Proposed Method

1.1 Empty space filtering

The following two strategies are used to speed up the training. First, we locate the smallest box region that fully covers the whole scene, which allows us to make full use of the grid resolution. To achieve this, we query all grid vertices of the coarse deformation feature grid in all training time steps to get the alpha values. We assume a point belongs to an object point rather than the empty space if its alpha value is larger than a predefined threshold. Using alpha values of all grid vertices at all training time steps, we could find the smallest bounding box for the dynamic scene.

Second, we filter out the sampled points in a ray if they are recognized as empty points, reducing the query numbers to the light-weight MLP. For sampled points in the deformation module, as we assume empty points to be static, we

* Authors contributed equally to this work. † Yuchao Dai is the corresponding author.

filter out sampled points whose alpha is always below a threshold during all training times. For sampled points in the canonical module, we only filter out the empty points at the canonical time.

1.2 Fine Model Design

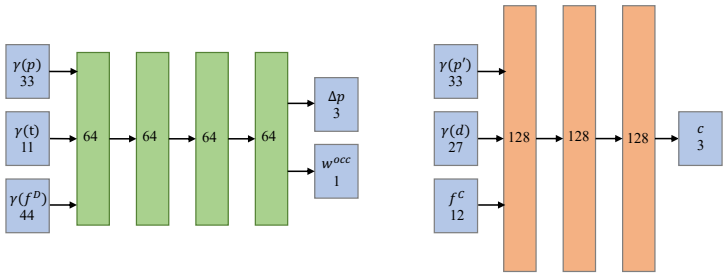
The MLP architecture of deformation module in the fine model is the same as in the coarse model, such that the trained network weights in the coarse deformation module can be used for initializing the fine model. We initialize the deformation feature grid with the optimized feature grid in the coarse model by interpolation. This initialization can result in higher performance and shorten the training time.

We model the view-dependent effect in the fine model by using a light-weight MLP to decode the interpolated color feature. We also use a progressive scale training, which doubles the voxel resolution after some training iterations [2].

1.3 Network Architecture

We show the architecture of our networks in Fig. 1. For deformation network $F^d_{\theta_1}$, we use four layers of fully connected layer with the width set to 64. For the input layer, the dimensions of position, time and feature are 33, 11 and 44.

For the color network $F^c_{\theta_2}$, we use three layers of fully connected layer with the width set to 128. For the input layer, the dimensions of position, view direction, and feature are 33, 27 and 12. Between the fully connected layer, we use ReLU as activation functions. For the occlusion output w^{occ} and color output c, we apply sigmoid activation to transfer outputs into range of $(0, 1)$.



(a) Deformation Network $F^d_{\theta_1}$ (b) Color Network $F^c_{\theta_2}$

Fig. 1. **Network Architecture.** We show architectures of our light-weight deformation network and color network.

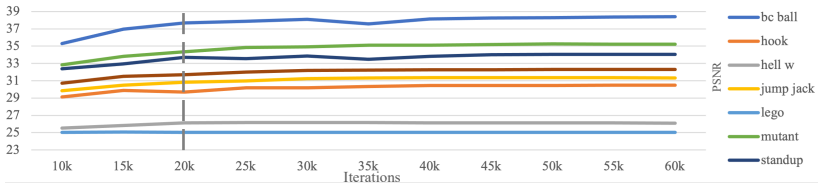


Fig. 2. PSNR with more training iterations

1.4 Hyper-parameter Settings in Experiments

As described in Eq. (16) of the paper, the overall loss function for the coarse stage and the fine stage can be written as

$$\mathcal{L} = \mathcal{L}^{\text{photo}} + w^{\text{ptc}} \cdot \mathcal{L}^{\text{ptc}} + w^{\text{bg}} \cdot \mathcal{L}^{\text{bg}} + w^{\text{d\_norm}} \cdot \mathcal{L}^{\text{d\_norm}} + w^{\text{d\_tv}} \cdot \mathcal{L}^{\text{d\_tv}}, \quad (1)$$

where w^{ptc}, w^{bg}, $w^{\text{df\_norm}}$, and $w^{\text{df\_tv}}$ are weights to balance each component.

In the coarse stage, we empirically set w^{ptc}, w^{bg}, $w^{\text{df\_norm}}$, and $w^{\text{df\_tv}}$ to 0.1, 0.01, 0.1, and 1, respectively. In the fine stage, these four weights were set to be smaller values as 0.01, 0.001, 0.01, and 1, respectively. As the motions of '`Hell Warrior`', '`Jumping Jacks`' and '`T-Rex`' are larger, we used a smaller weight for the deformation norm regularization, with $w^{\text{df\_norm}}$ equals to 0.01 and 0.001 for the coarse and fine stage.

As shown in Fig. 2, the PSNR generally increase slightly after 20k iteratins, but will takes significantly more iterations to reach the max. To balance the PSNR and time consumption, we set the training iteration to 20k.

1.5 Discussion of Tips to Speed Up Training

The purpose to use coarse-to-fine training is that we could get a relative good geometry and motion model quickly and other strategy like empty-space filtering is built on this coarse model to save training time in fine stage. In Table 1, $\text{NDGV}_{\text{(w/o filter)}}$ does not use any speed-up strategies and directly optimizes fine model without filtering. Under this setting, training speed, render speed and PSNR are all clearly worse than our full model $\text{NDGV}_{\text{(full)}}$. On the other hand, the train speed of $\text{NDGV}_{\text{(w/o filter)}}$ is on the same level with $\text{NDGV}_{\text{(full)}}$, which means these tips are not the main factors to speed up training.

1.6 Illustration of Empty Space Filtering

As described in Section 4.2 of the paper, we speed up the training with the empty space filtering strategy. Here, we visualize our empty space filtering strategy in Fig. 3. We first locate the smallest box region that fully covers the whole scene, which is denoted as coarse geometry prior (see Fig. 3 (a)-(c)). We obtain the coarse geometry prior by warping object points in a canonical space into all

Table 1. More results of ablations (**half:400x400, full:800x800**)

Metrics	DNeRF (half)	NDGV (half)	NDGV (full, w/o filter)	NDGV (full, w/o grid)	NDGV (full)
train speed (s/scene)	99034	1380	2487	1450	2087
render speed (s/image)	8.7	0.4	3.5	1.3	1.7
model size (M)	13.2	995.8	988.3	668.1	994.2
PSNR	30.02	30.32	27.85	30.62	31.08

w/o filter: without any filter methods w/o grid: without deformation grid

other times, so that we can identify all possible positions that could be object points in the world coordinates. With this coarse geometry prior, we could reduce the volume of the grid in fine stage to avoid as many empty point as possible.

Second, we filter out the sampled points in a ray if they are recognized as empty points, reducing the query numbers to the light-weight MLP (see Fig. 3 (d)-(f)). Specifically, we filter our sampled points whose alpha values are always below a threshold during all training times. All these operations significantly reduce the time consumption of training and rendering.

2 More Analysis for the Proposed Method

2.1 Training, Rendering Speed and Model Size

We show the training and render speed of different settings in Table 1. To speed up training, we need to use a very small size MLP, which could harm the image quality. This is why we use a deformation feature grid to embed more information without adding too much complexity to the system. We can see that adding a deformation feature grid increases the PSNR at the cost of slightly more training time and bigger model size, which we believe is reasonable to balance these metrics.

We report model size in Table 1. The model size of DNeRF is the smallest thanks to the compact representation of MLP, but it doesn't mean it is faster and cheaper to train. In fact, we could train faster on the same device compared with DNeRF and set a much bigger batch size with a resolution of around 150^3 in our experiments.

In addition, We further show the training speed of each submodule in our system in Table 2. According to Table 2, we can notice a dramatic increase of time consumption of query deformation feature, deformation network forward and gradient computation process without our empty space filtering mechanism, since more empty points are involved in the optimization process without filtering.

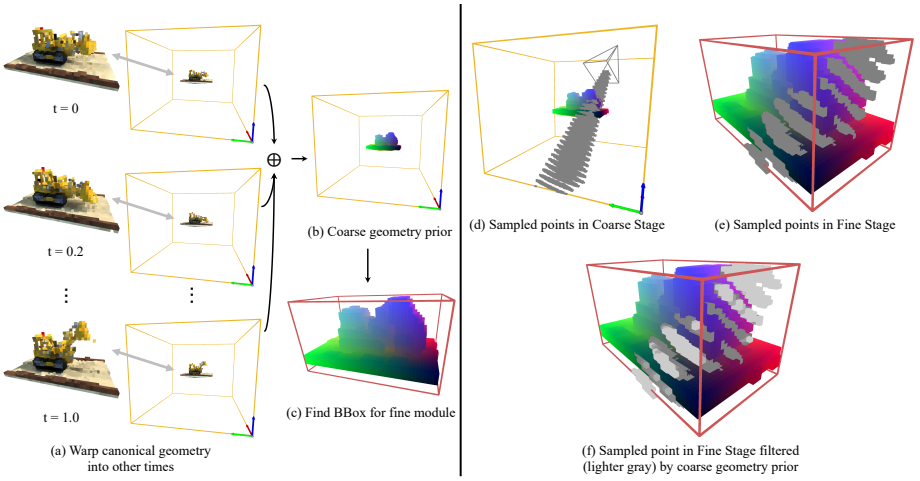


Fig. 3. **Illustration of Empty Space Filtering.** (a) warps the learnt canonical geometry during coarse training stage into other times. Be aware of the moving part in red circle. (b) combines object points in all times to form a coarse geometry prior, indicating all possible positions that could be an object point along all times. (c) with coarse geometry prior, find a smallest bounding box which could cover all possible object points. The volume of the grid is significantly reduced. (d) shows a example of sampled point in Coarse Stage masked by coarse bounding box. (e) shows a example of sampled point in Fine Stage masked by fine bounding box. (f) shows sampled points in (e) but filtered by coarse geometry prior. Be aware that the sampled points in empty space are marked with lighter gray color.

2.2 Ablation Study of Losses

We establish the ablation study of the losses we use during training. We show the results of models trained without the total variation loss $\mathcal{L}^{\mathrm{d}\_\mathrm{tv}}$ and without the deformation normalization loss $\mathcal{L}^{\mathrm{d}\_\mathrm{norm}}$ in Table 3. According to Table 3, there is a significant drop without $\mathcal{L}^{\mathrm{d}\_\mathrm{tv}}$. It is worth to mention that, even it achieve slightly better results without deformation normalization loss $\mathcal{L}^{\mathrm{d}\_\mathrm{norm}}$, the geometry of canonical module degenerated due to lack of normalization of the canonical space, which is show in Fig. 4. As shown in Fig. 4, without $\mathcal{L}^{\mathrm{d}\_\mathrm{tv}}$, the canonical images are shattered without this spatial smooth term. Also, without $\mathcal{L}^{\mathrm{d}\_\mathrm{norm}}$, the canonical images are distorted. Since there are no constraints on deformation estimation, there is more freedom of the deformation between spaces at other times and the canonical space, which could cause distortions in the canonical space.

2.3 Deformation Grid Resolution

We test the different deformation grid resolution (total voxel number) and analysis the corresponding images quality and training time consumption, which is

Table 2. **Detailed Training Time of Each Module**. We report the training time in details of each module in **second/iteration**. **Query Feat** is query the deformation feature grid. **Deform Net** means deformation network forward. **Canonical Module** means the whole process in canonical module. **Render** means the render process after get the densities and colors of all sampled points. **Loss** means the computation of loss. **Backward** means the computation of the gradients. **Optimization** is the process of optimizing all parameters.

Methods	Deformation Module		Canonical Module	Render	Loss	Backward	Optimization
	Query Feat	Deform Net					
NDVG (full)	9.23e-4	5.06e-3	5.28e-3	1.02e-3	5.61e-3	3.50e-2	7.23e-3
NDVG (w/o filter)	3.28e-3	2.82e-2	4.93e-3	1.08e-3	5.36e-3	6.68e-2	7.36e-3

Table 3. **Ablation Study of Losses**.

Methods	PSNR↑	SSIM↑	LPIPS↓
NDVG (w/o $\mathcal{L}^{\mathrm{d}}\_^{\mathrm{tv}}$)	30.45	0.963	0.061
NDVG (w/o $\mathcal{L}^{\mathrm{d}}\_^{\mathrm{norm}}$)	31.54	0.971	0.039
NDVG (full)	31.08	0.970	0.039

show in Fig. 5. With the total number of voxels is increasing, the overall trends of PSNR and training time consumption are also increase. We choose the final number of voxels as 190^3 to balance the image quality and training time consumption.

2.4 Visualization of the Feature Grid

The feature grids, including deformation feature grid in Deformation Module and color feature grid in Canonical Module, play important roles in our pipeline. We show the ablation study in paper with quantitative results. Here we visualize the feature learnt from training in Fig. 6. As shown of the first row in Fig. 6, we slice the feature grid along the axis x. We then treat the sliced feature slice as stacked rgb images and visualize them after normalization. In terms of deformation of feature grid slices, some of the columns (like the 4th and 5th) show clean clues of objects. Other columns (like the 6th and 7th), show clean clues of deformations of the objects (bouncing trajectory). In terms of color feature grid slices, the visualizations show clean clues of objects and no clues of movements, because it is defined in canonical space.

3 More Results

First, we show some of the test images of real scenes dataset synthesised by our method in Figure 7. The visual results demonstrate that our model could learn reasonable representation for the real dynamic scenes and enable novel view synthesis with fast optimization.

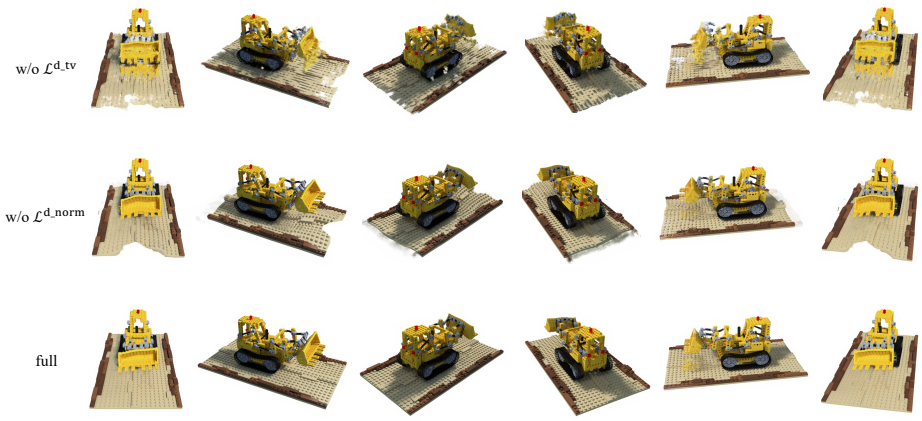


Fig. 4. **The Reconstructed Canonical space with different Losses**.

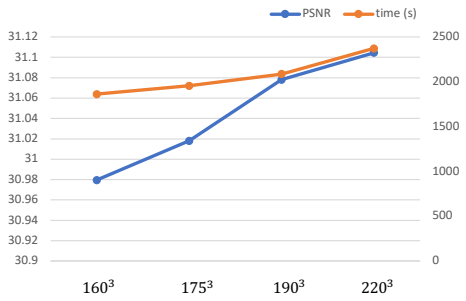


Fig. 5. **Effects of Using Different Resolutions for the Deformation Grid**.

Then, we present more results for the learned geometries in Fig. 8. Our method can reconstruct accurate canonical geometry and render high-quality images for different time steps. We notice that the reconstructed canonical geometry of 'Bouncing Balls' has some missing parts in the white plate. One of the main reasons is that the background for this scene is white, and the plate has a color similar to the background. The image reconstruction error for this region is low even if the learned density of the plate is close to zero. Then, these low-density regions might be filtered out by our empty space filtering strategy. This problem can be easily alleviated by using a background with different color or disabling the filtering strategy.

Finally, we compare the results rendered by the coarse stage and fine stage in Fig. 9, and we show more results to compare our method with D-NeRF [1] in Fig. 10.

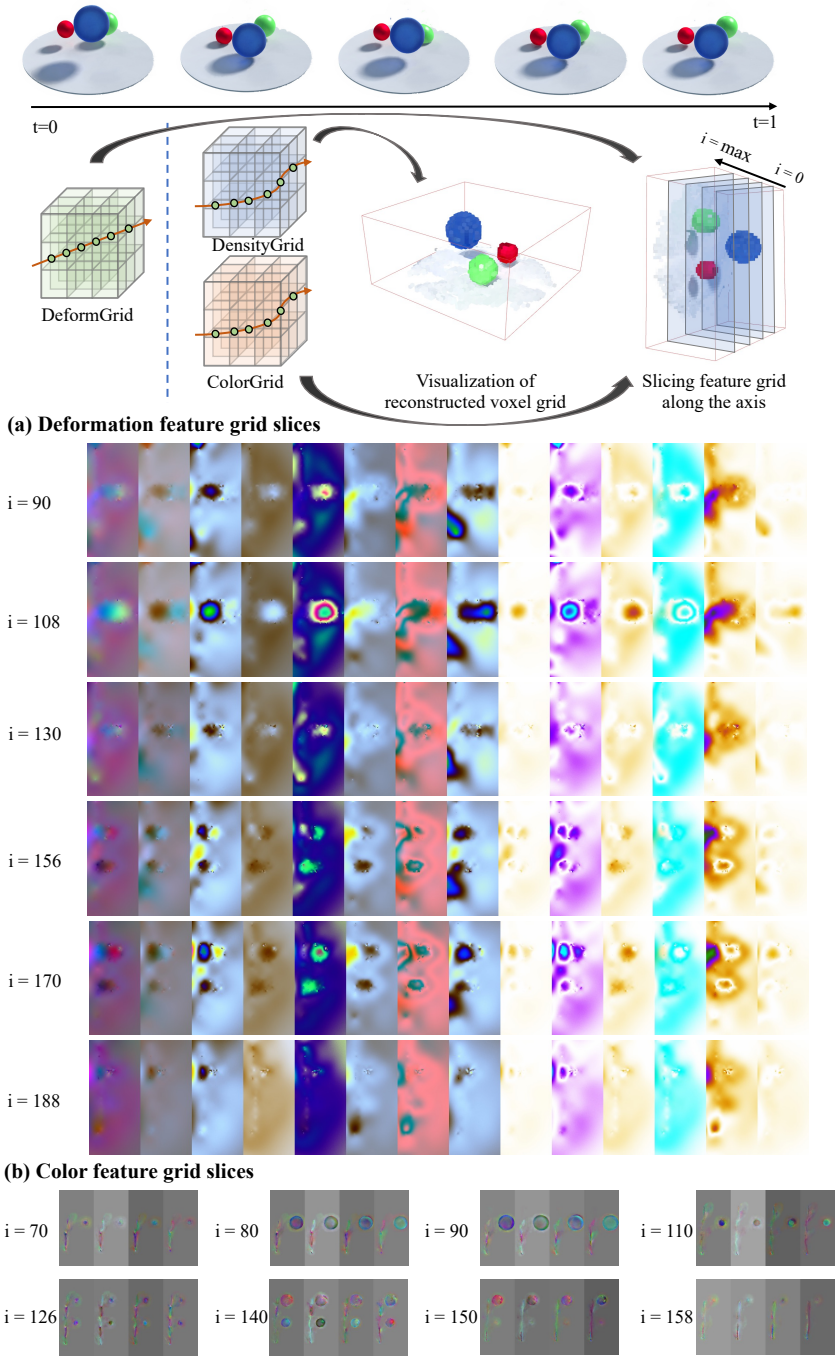


Fig. 6. **Visualization of Feature Grid.** We slice the feature grid along the axis to get feature slices. Then we visualize the feature slices every three dimension as rgb images after normalization. Best viewed in color and zoom in for details.

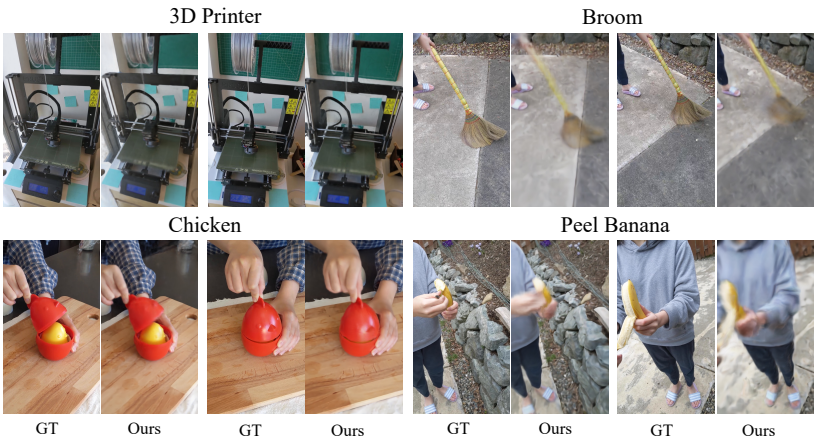


Fig. 7. Results of real scenes

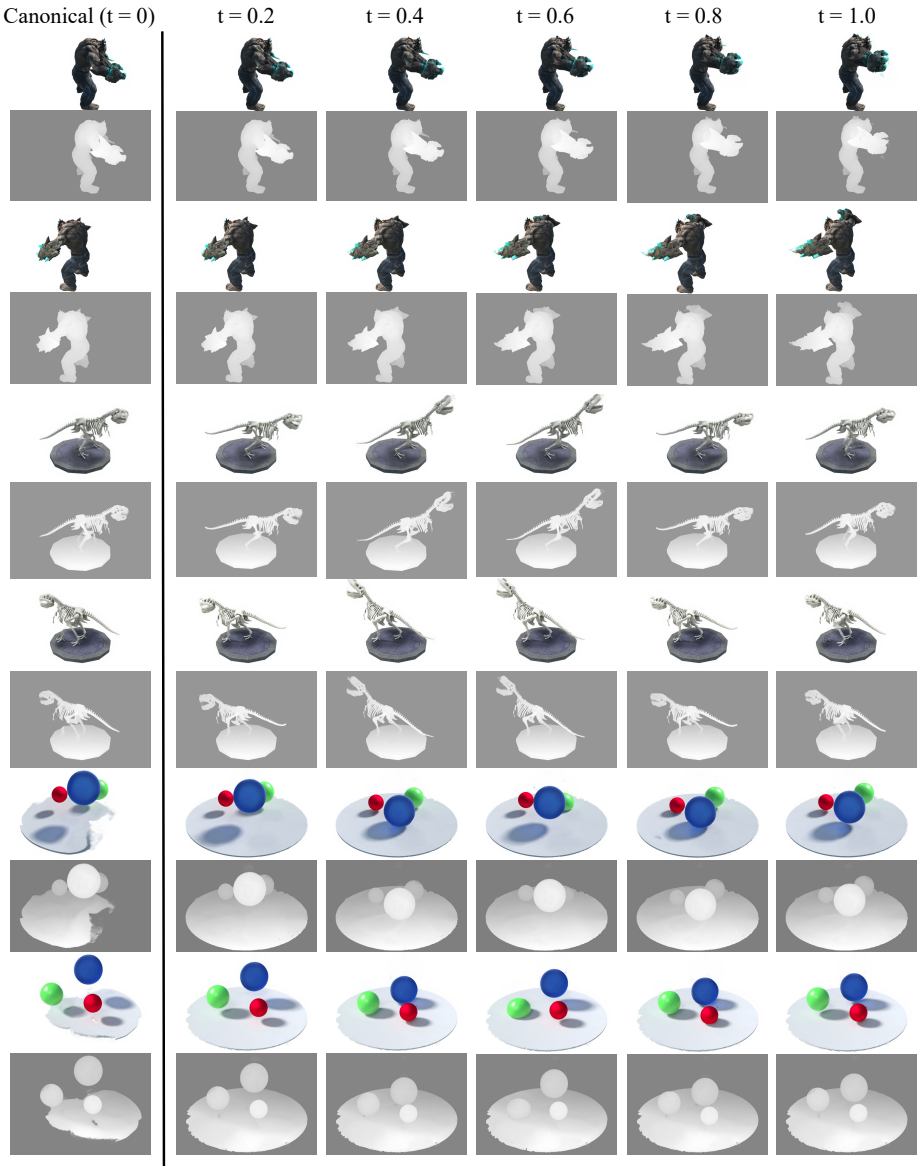


Fig. 8. **More Results for the Learned Geometry.** We show examples of geometries learned by our model. For each, we show rendered images and corresponding disparity under two novel views and six time steps.

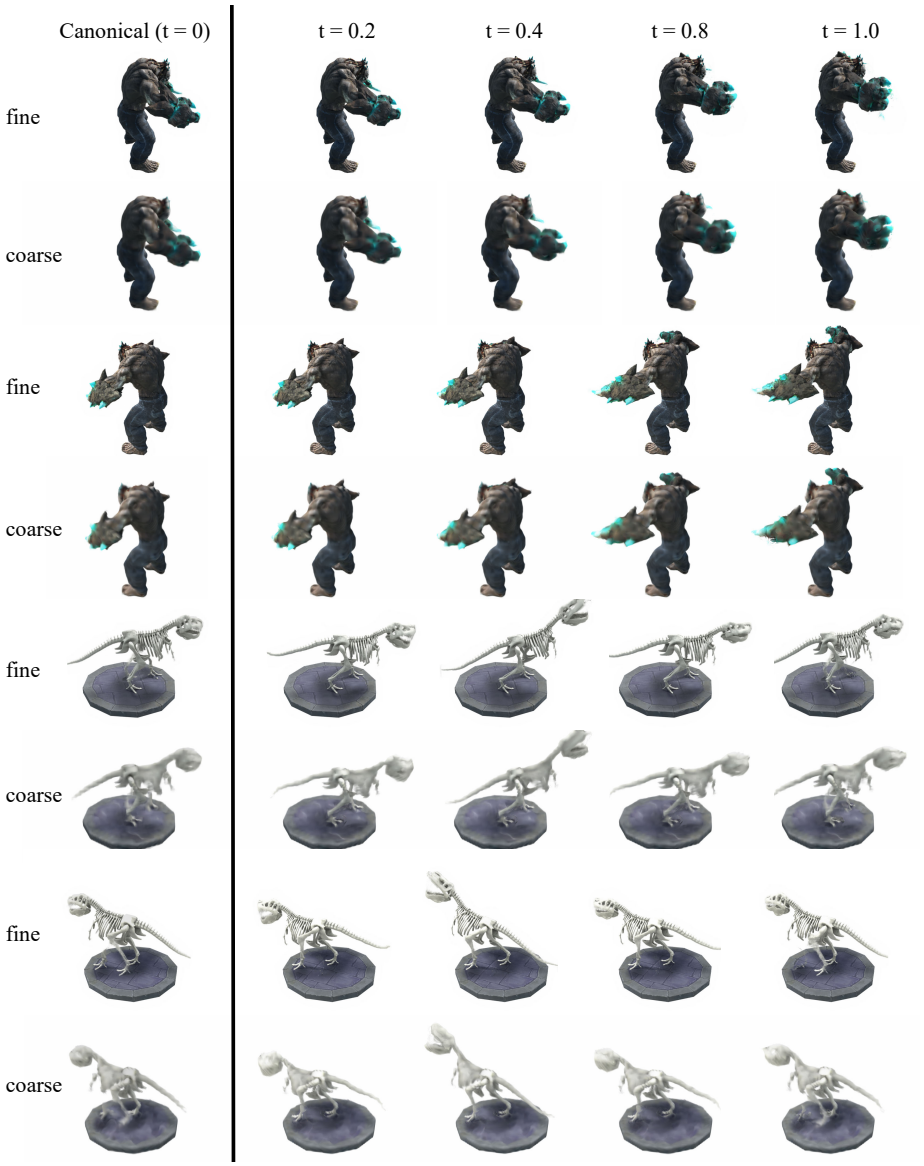


Fig. 9. **Comparison of Results of Coarse and Fine Stage.** Synthesized images rendered by modules trained from coarse stage and fine stage. The images of coarse stage module is relative over smoothed compared with images of fine stage module. Best viewed in color and zoom in for details.

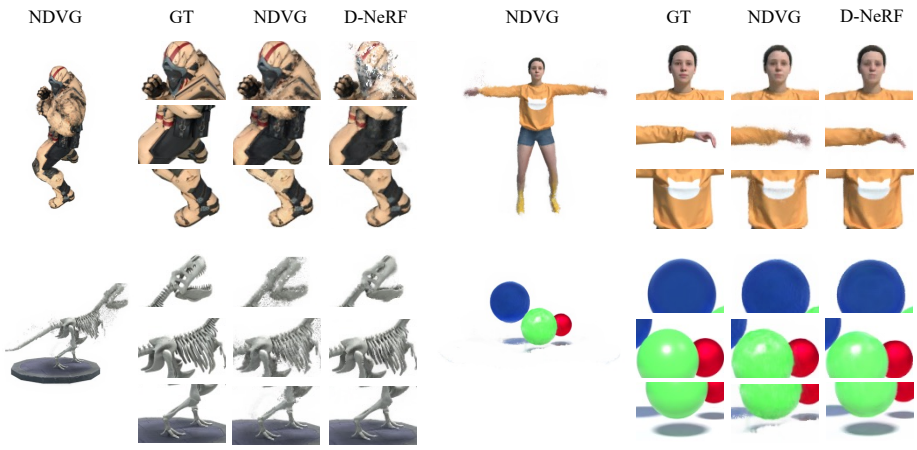


Fig. 10. **More Qualitative Comparison.** Synthesized images on test set of the dataset. For each scene, we show an image rendered at novel view, and followed by zoom in of ground truth, our NDVG, and D-NeRF [1]. Best viewed in color and zoom in for details.

References

1. Pumarola, A., Corona, E., Pons-Moll, G., Moreno-Noguer, F.: D-nerf: Neural radiance fields for dynamic scenes. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 10318–10327 (2021) 7, 12
2. Sun, C., Sun, M., Chen, H.T.: Direct Voxel Grid Optimization: Super-fast convergence for radiance fields reconstruction. arXiv preprint arXiv:2111.11215 (2021) 2