NeRF Project Report

Tianyi Sun
Department of Electrical and Computer Engineering
Peking University
1900012920@pku.edu.cn

Peiheng Wang
Department of Electrical and Computer Engineering
Peking University
1900012916@pku.edu.cn

Hongbo Ning
Department of Electrical and Computer Engineering
Peking University
1800017711@pku.edu.cn

Abstract

NeRF is effective in synthesizing novel views of complex scenes by optimizing an underlying continuous volumetric scene function using a sparse set of input. Based on NeRF, two types of dynamic NeRF were proposed to incorporate the time dimension. One method is to add time dimension directly into the basic NeRF, while another one is to construct two networks dealing with deformation and canonical information respectively. However, the training of present models, whether for static or dynamic scenes, is time-consuming and presents low-resolution results when not well-trained. Here we show our methods to achieve better training effects with less time consumption based on our understanding of the current NeRF and its implementation in dynamic scenes. We implement two novel kinds of method to achieve data argumentation by improving the sampling process. One is guided by loss function and another is instructed by intuitive moving regions segmentation. Our method may contribute to the faster training of NeRF or enhance the performance of NeRF with limited training consumption.

1 Introduction

NeRF is effective in synthesizing novel views of complex scenes by optimizing an underlying continuous volumetric scene function using a sparse set of input [2]. Based on NeRF, two types of dynamic NeRF were proposed to incorporate the time dimension. One method is to add time dimension directly into the basic NeRF, making the network keep information in both time and space dimensions [1]. Another one is to construct two networks, one for canonical NeRF, memorizing the original space information, another Φ_t for deformation presenting the offset of each sample point along the sample ray between the first and the t moment [3]. We compare the two methods and analyze its effect.

In Sec. 2, we implement a 2D-NeRF to fit a single image with positional encoding. In Sec. 3, we implement a NeRF and fit on multi-view images. In Sec. 4, we implement two types of NeRF for dynamic scenes. In Sec. 5, we propose a new sampling method based on D-NeRF to generate results with better perspective quality with the same iterations. In Sec. 6, we utilize the two types of dynamic NeRF for data augmentation and segment dynamic parts from static parts. Based on our understanding

(a) **Subfigure 1**. original.

(b) **Subfigure 2**. ours.

Figure 1: A comparison between the original picture and the corresponding reconstruction of the best model with 5 layers, 1024 dimensions, 10 frequencies and with positional embedding, which are similar to each other.

of the current NeRF and its implementation in dynamic scenes, we propose our methods to achieve better training effects with less time consumption.

2 Fit a single image with positional encoding

We write a program according to the instruction and train separate models with different layers, dimensions, frequencies and with or without positional embedding. All models are trained on the picture in Fig. 1a for 40 epoches. We show the results in Tab. 1. With more layers, more dimensions, larger frequency and with positional embedding, we obtain better results. The removal of positional embedding has the most significant effect, with a drop of 7.35 in PSNR. The reconstruction of the best model with 5 layers, 1024 dimensions, 10 frequencies and with positional embedding is shown in Fig. 1b, which is similar to the original picture Fig. 1a.

Table 1: PSNR results of separate models with different layers, dimensions, frequencies and with or without positional embedding trained on a single image.

layers	dimensions	freq	positional_embedding	psnr
3	1024	10	TRUE	78.5557
4	1024	10	TRUE	79.7047
5	1024	10	TRUE	80.081
4	64	10	TRUE	73.7507
4	256	10	TRUE	76.7787
4	1024	2	TRUE	73.5915
4	1024	5	TRUE	76.5248
4	1024	10	FALSE	72.3555

3 Implement NeRF and fit on multi-view images

3.1 Method

NeRF is a view synthesis method that renders new views of complex static scenes from various perspectives by given a set of sparse views with known camera poses in a static scene. In this part, we implement 3D-NeRF and 5D-NeRF including computing the origin and direction of camera rays, sampling points along each camera ray, implementing the fully connected network architecture, and volume rendering.

The vanilla NeRF accepts the location coordinates in the 3D scene and the 2D viewing direction as input, maps the input to a high-dimensional space through positional encoding, and outputs the emitted RGB radiance in the specified viewing direction and the volume density of the specified

point in the 3D scene through a fully connected network. Finally, the RGB value of each pixel in the rendered view is calculated by volume rendering.Fig. 2 shows the network architecture of NeRF.

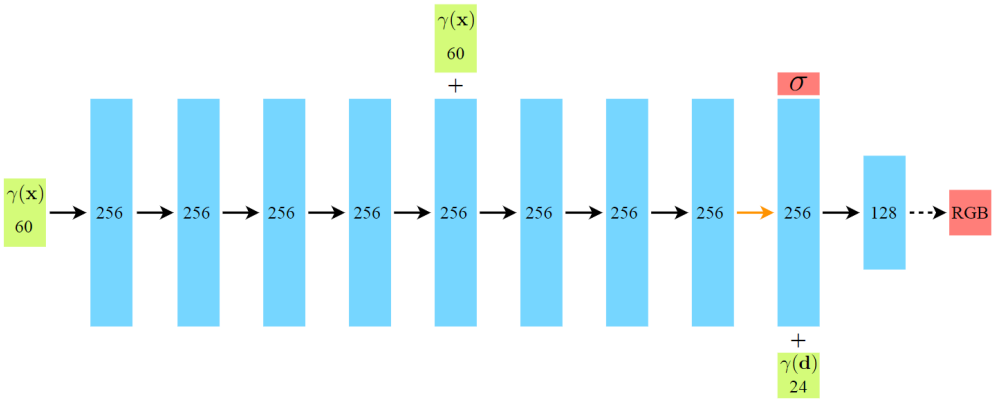
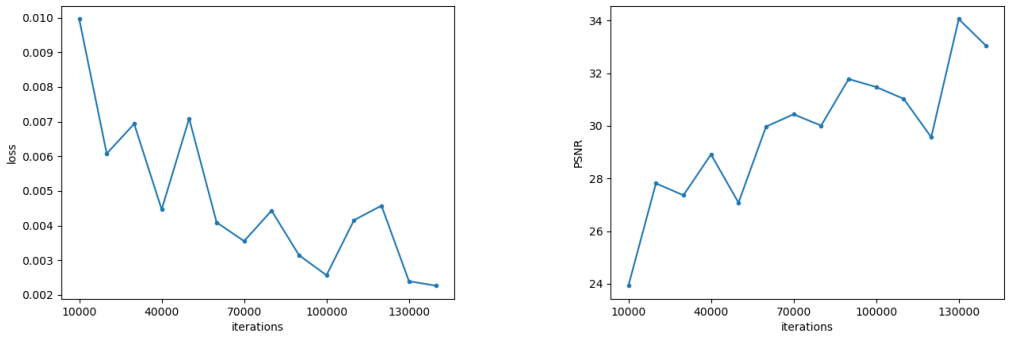


Figure 2: The fully-connected network architecture of 5D-NeRF. First, the input 3D coordinates are converted into 60-dimensional vector by position encoding function $\gamma(\cdot)$. Then the positional encoding of the input location is passed through 9 fully-connected layers, each with 256 channels, the last layer of the fully-connected layers outputs the volume density without the input viewing direction. Finally, the 24-dimensional positional encoding of the input viewing direction ($\gamma(d)$), the emitted RGB radiance is output through a small fully connected network.

3.2 Evaluation

We train the 5D-NeRF with positional encoding for 140,000 iterations on the lego dataset, which takes about 5 hours. Fig. 3 shows the changes in loss and PSNR on the training set during training. The training set loss drops off rapidly in the first 50,000 iterations and tends to slow down after about 100,000 iterations. PSNR increases almost linearly with the number of training iterations. After about 100,000 iterations, the model has achieved a very good restoration effect on the Lego scene. Fig. 4 shows the new view that rendered by 5D-NeRF w/ positional encoding after 140k iterations and ground truth camera view.



(a) **Subfigure 1**. Loss

(b) **Subfigure 2**. PSNR

Figure 3: Loss and PSNR on the training set during training

We experiment with different model architectures and the use of positional encoding or not, we trained 3D-NeRF with positional encoding, 5D-NeRF without positional encoding and 5D-NeRF with positional encoding for 10000 iterations. (Where 3D-NeRF only accepts the input of location coordinates without viewing direction; wo/ positional encoding means directly using 3D position coordinates or viewing direction as input without using positional encoding.) We found that if the 3D coordinates are directly used as the fully-connected network input without using high-frequency positional encoding, the rendered view will be too smooth and lose many detailed geometric structures and textures, indicating that position encoding can effectively help the network learn true distribution of the scene. And, using or not using the viewing direction as an additional input makes no significant difference when there is no specular reflection.

(a) **Subfigure 1**. ours

(b) **Subfigure 2**. GT

Figure 4: The view rendered NeRF after 140k iterations and the ground truth camera view.

(a) **Subfigure 1**. 5D wo/ positional encoding

(b) **Subfigure 2**. 3D w/ positional encoding

(c) **Subfigure 3**. 5D w/ positional encoding

(d) **Subfigure 4**. Ground Truth

(e) **Subfigure 5**. 5D wo/ positional encoding

(f) **Subfigure 6**. 3D w/ positional encoding

(g) **Subfigure 7**. 5D w/ positional encoding

(h) **Subfigure 8**. Ground Truth

Figure 5: The views rendered with different model architectures after training for 10k iterations

4 NeRF extension: NeRF for dynamic scenes

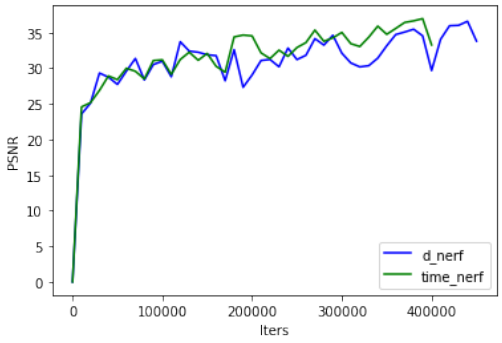
4.1 Description

To implement NeRF into dynamic scenes, there are two existing ideas. Firstly, time dimension could be directly added into the basic NeRF, which means the new network could remember information according to the input of both time and space indexs. Another one is to construct two network, one for canonical NeRF, memorizing the space information, another Φ_t for deformation presenting the offset of each sample point along the sample ray between the first and the t moment. The second idea use Φ_t to calculate Δx and use $x + \Delta x$ as input to train the canonical NeRF. In this part, we implement two ideas based on the NeRF code. Since the complete NeRF code wasn't available due to Jan.18th, we construct this part based on the open sourse code of D-NeRF and add into our implementation of the first idea as Time-NeRF.

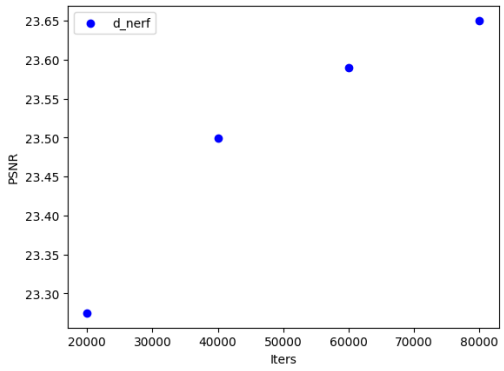
4.2 Evaluation

We train above two methods for 40,0000 iterations on NVIDIA GeForce RTX 3090, each taking about 19 hours. We compare the training process by their PSNR. (Fig. 6) After about 5,0000 iterations,

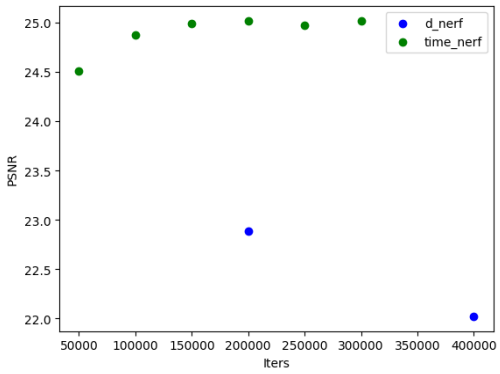
PSNR could reach a relatively high level(Fig. 7), and then slowly goes to convergence. According to rendering test, Time-NeRF performs better than D-NeRF in PSNR and SSIM. It seems reasonable for short-time and less-mobile data(dataset: lego). And according to Fig. 8, D-NeRF made great mistakes between trailer's shovel and body, while Time-NeRF performs relatively better with the same training rounds.



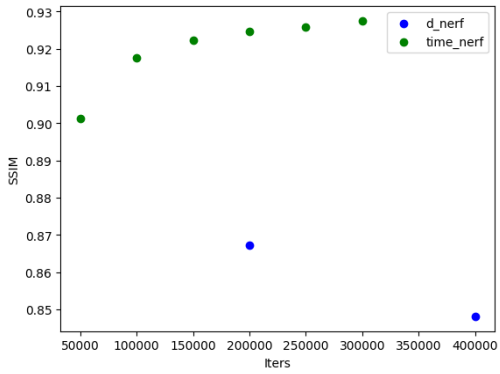
(a) **Subfigure 1**. Time-NeRF/D-NeRF training comparision.



(b) **Subfigure 2**. D-NeRF PSNR with 20,000-100,000 iterations.



(c) **Subfigure 3**. Time-NeRF/D-NeRF PSNR.



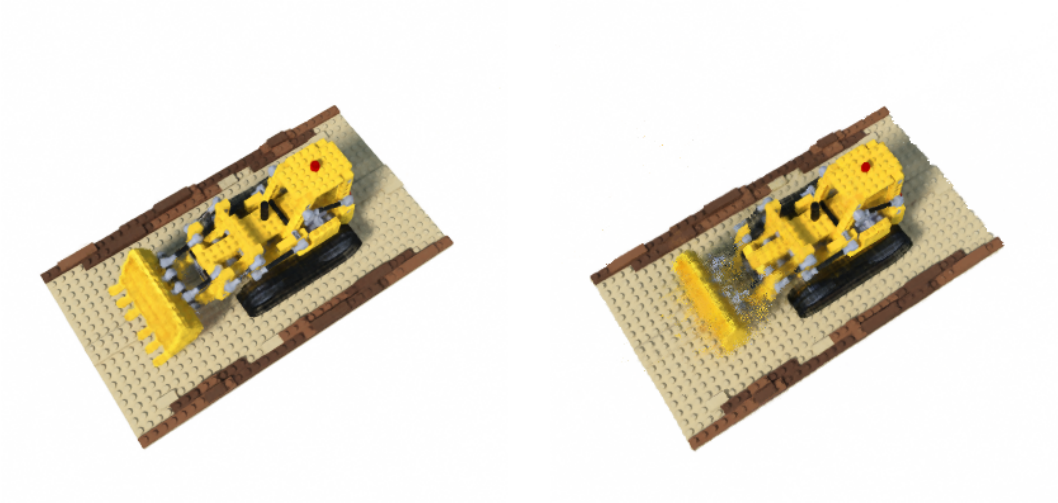
(d) **Subfigure 4**. Time-NeRF/D-NeRF SSIM.

Figure 6: Training and rendering test results of Time-NeRF/D-NeRF

(a) **Subfigure 1**. Time-NeRF rendering result.

(b) **Subfigure 2**. D-NeRF rendering result.

Figure 7: The 13th frame of the video generated by models after training for 50,000 iterations.



(a) **Subfigure 1**. Time-NeRF. (b) **Subfigure 2**. D-NeRF.

Figure 8: The 1st frame of the video generated by models after training for 200,000 iterations.

5 Propose new ideas for dynamic nerf

Considering that some of the regions in the picture are smooth and easy to predict(*e.g*. the white margin of the lego example), while others have complex texture, we sample at different rate according to the PSNR. After training for 50000 iterations, we run the trained model on all the training set. We calculate MSE for each 10x10 patch in all pictures. Pixels in a patch with MSE N times larger than another patch has a $N^{0.4}$ higher probability of being sampled. This method is similar to the idea of Hierarchical volume sampling. We implement the method based on the DNeRF code written in Sec. 4. Taken Fig. 10 as an example, Fig. 10a is a visualization of $MSE^{0.4}$ and Fig. 10b is an example of our sampling result. The black dots denote random samples according to the generated weights, so samples concentrate around details.

Fig. 9 shows the PSNR for anchor method (*i.e*. DNeRF implemented in Sec. 4) and our method. For both methods, we start from loading the pretrained model with 50000 iterations and continue training. We validate both methods in 100000, 140000 and 190000 iterations on the test set. Unexpectedly, PSNR goes down with larger iterations, though perception quality increases with iterations, which may demonstrate that PSNR isn't an ideal metrics. Our method performs worse than anchor method in terms of PSNR. Then we evaluate the perspective quality of both methods in Fig. 11 and Fig. 12. Our method shows better perspective quality, especially for details. So with our method, we can obtain results with high resolution more quickly than anchor method, though the dropping PSNR has to be solved in the future.

6 Delta-Improved Time-NeRF

6.1 Method

From the previous experiments, we come to following conclusions in dynamic scenes. First, both models can reach relatively good perceptual effect within 50,000 iterations. Second, moving parts have poor details which mainly impact the perceptual and quantitative performance a lot. Third, the deformation model of D-NeRF tells apart the moving sections clearly as shown in Fig. 13. With these observations, it comes to us another idea–using a pretrained deformation model to give moving parts more weight in the process of ray sampling, using it as an induction to argument data.

We use a deformation model from D-NeRF pretrained for 50,000 iterations to get delta value of images in the training set. Each pixel indicates a sample ray, each ray has equally spaced sample points. We add its offset in three dimensions to get delta of each point and then sum all points' delta value along one ray as final delta value of a pixel. With a fine-tuned threshold, we could tell apart the moving regions as Fig. 14 demonstrates. When sampling rays, in original method we choose pixels randomly, while in our approach, we give large-delta area 50% possibility to be choosen and other

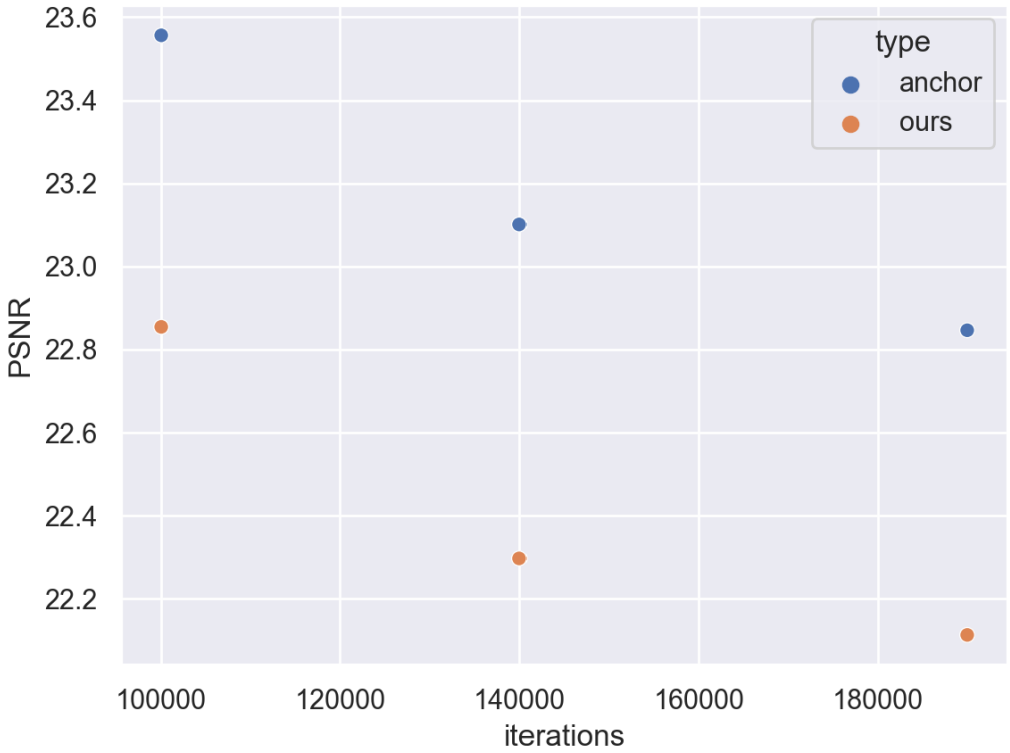
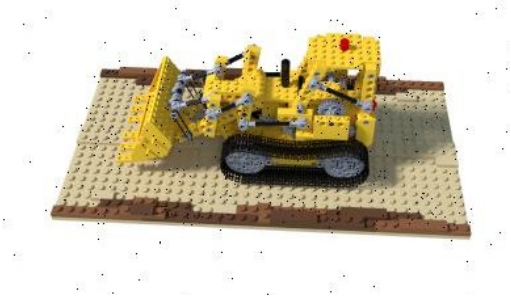


Figure 9: A comparison between the anchor method and our method in Sec. 5 after 100000, 140000 and 190000 iterations. Our method performs worse in terms of PSNR.

(a) **Subfigure 1**. A visualization of weight generated by a pretrained model(*i.e.* $MSE^{0.4}$). Mention that all pictures showed in the paper are removed of much of their margin for a more compact layout.



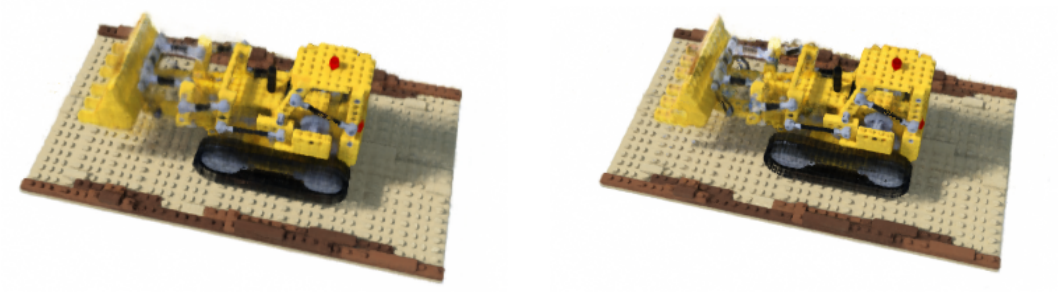
(b) **Subfigure 2**. An example of our sampling result. The black dots denote random samples according to the generated weights, so samples concentrate around details.

Figure 10: An example for Sec. 5.

regions share another 50%. In this way, we can cover all regions in the image. If we only choose rays in large-delta area, it will cause overfitting and bad perceptual effect on test set.

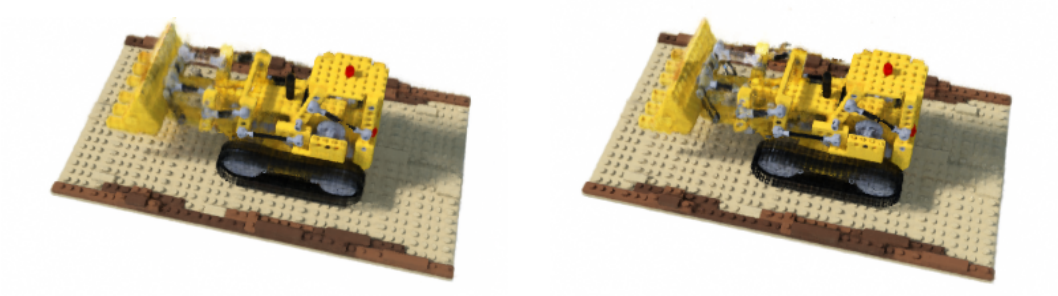
6.2 Impelmentation

The deformation model in D-NeRF displays the offset of sample points against their position at time t_0 as shown in Fig. 14, which performs great segmentation effect to tell apart static and dynamic part. In our experiment, we use the deformation model of pre-trained D-NeRF (after 50000 iterations) to segment parts with larger delta, shown as Fig. 14. Using the calculated delta values to guide the sampling of training rays, we then train time-nerf for 200000 iterations with later 150000 iterations implying the delta-guided sampling.



(a) **Subfigure 1**. anchor. (b) **Subfigure 2**. ours.

Figure 11: A comparison between the anchor method and our method in Sec. 5 after 100000 iterations. Our method shows relatively better perceptual quality.



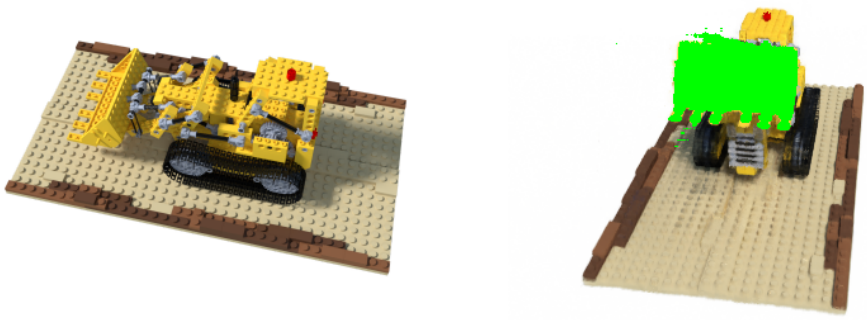
(a) **Subfigure 1**. anchor. (b) **Subfigure 2**. ours.

Figure 12: A comparison between the anchor method and our method in Sec. 5 after 140000 iterations. Our method still shows relatively better perceptual quality and both methods show better perceptual quality than their 100000-iteration version.

6.3 Evaluation

For qualitative observation, we could see our method obtain better details in shovel part which is also the moving part(Fig. 15), solving the question we raised before. The figure show the rendering result of the same frame in the test set after the same rounds of iterations,the left one is the normal Time-NeRF, the right one is our delta-improved method. The lego figure in the shovel part are more clear in our method though still with some noise.

For quantitative results, our method shows 0.2 improvement in PSNR and steady increasement in SSIM, both indicates effectiveness of our method(Fig. 17). As the ascending trend shown in the



(a) **Subfigure 1**. Frame N0.001 (b) **Subfigure 2**. Frame No.039.

Figure 13: The deformation model performs as segmentation method.

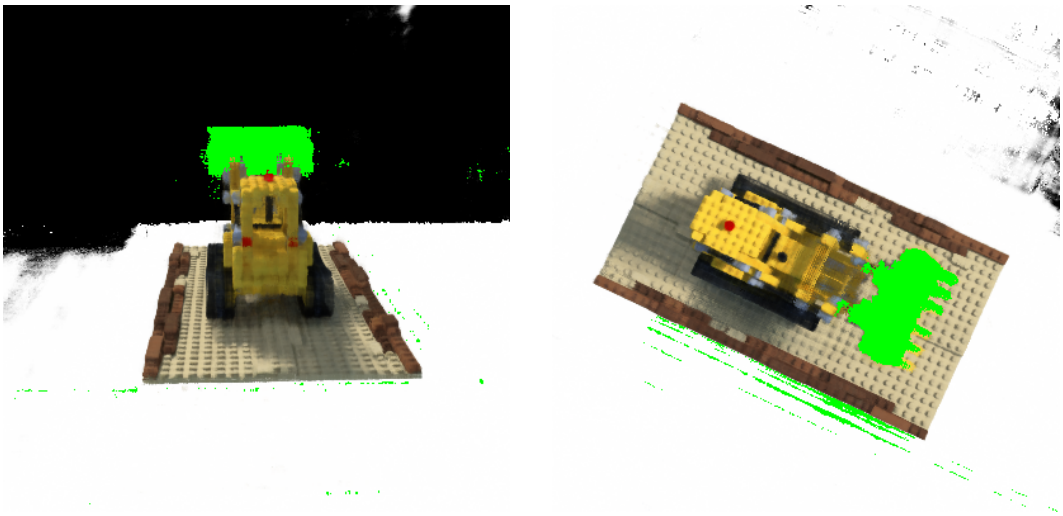
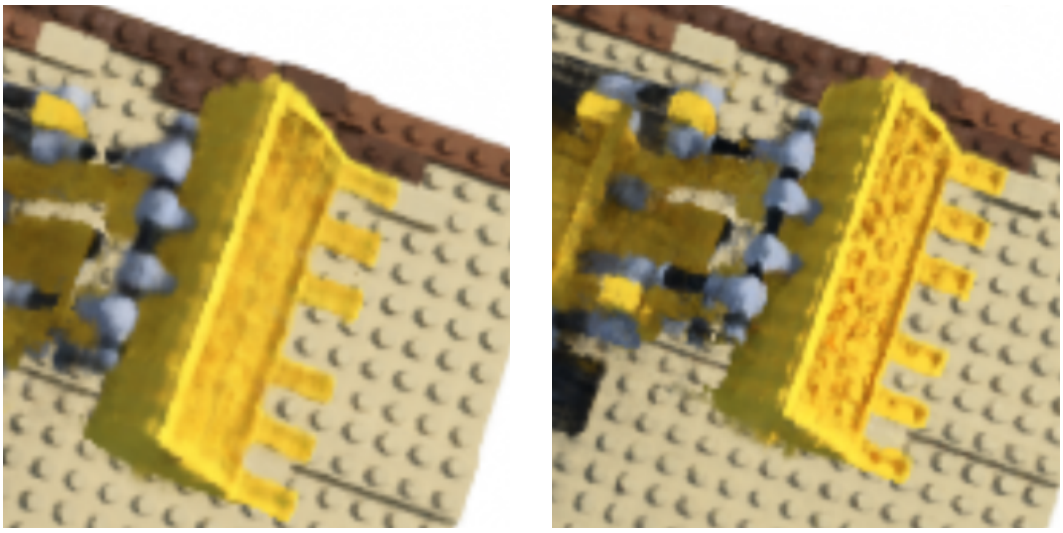


Figure 14: The green part indicates the moving regions of the tested frame against the 1st frame.



(a) **Subfigure 1** Time-NeRF (b) **Subfigure 2** Delta-Improved Time-NeRF

Figure 15: Details in shovel part–the moving regions of the trailer in dynamic scenes.

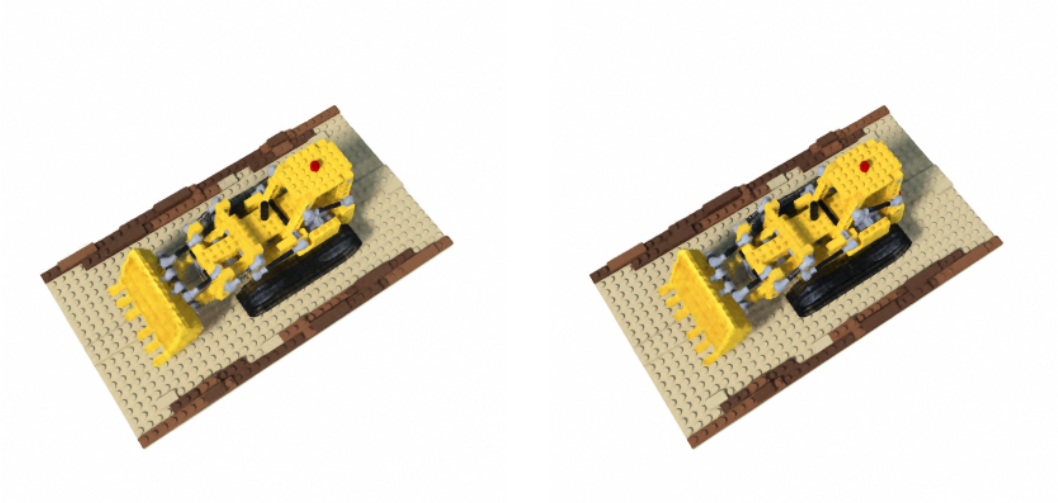
scatter plot, more training iterations may lead to better performance. In conclusion, adding more weights to the moving region and at the same time giving enough weight to normal area appears as a better sampling approach to synthesize novel views for dynamic scenes.

7 Contributions

This section states each author's contributions. Peiheng Wang contributes to Sec. 4 and Sec. 6. Hongbo Ning contributes to Sec. 3. Tianyi Sun contributes to Sec. 2 and Sec. 5.

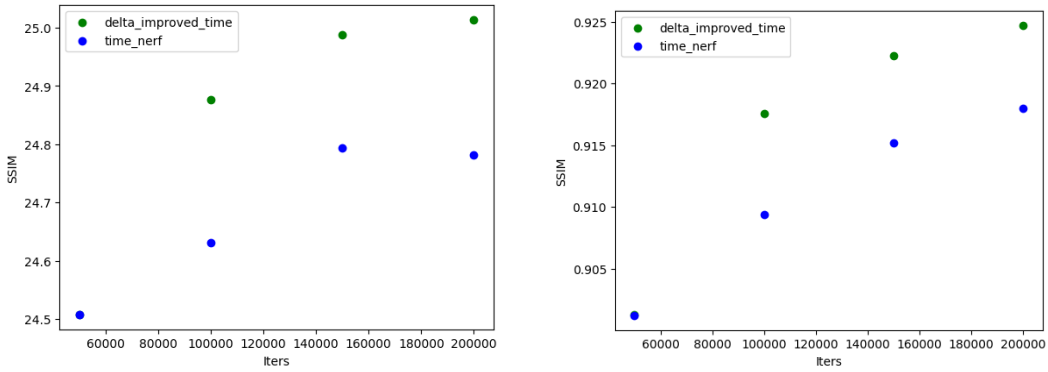
8 Conclusion

Considering that the training of present models is time-consuming and presents low-resolution results with poor details when not well-trained, we try methods to achieve better training effects with less time consumption based on our understanding of the current NeRF and the extension method proposed in dynamic scenes. We implement two novel kinds of data augmentation and gain better perceptual quality with both methods and higher PSNR and SSIM with Delta-Improved Time-NeRF.



(a) **Subfigure 1** time-NeRF (b) **Subfigure 2** delta_improved_time_NeRF

Figure 16: Comparision of time-NeRF and delta-improved time-NeRF.



(a) **Subfigure 1** Delta-Improved Time-NeRF PSNR (b) **Subfigure 2** Delta-Improved Time-NeRF SSIM

Figure 17: Delta-Improved Time-NeRF PSNR/SSIM after 200000 iterations.

Our method may contribute to the real-time application of NeRF or enhance the results of NeRF in the future.

References

[1] Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6498–6508, 2021. 1

[2] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021. 1

[3] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10318–10327, 2021. 1