Designing Difficult Datasets for Dynamic NeRFs

Daniel Acks
Stanford University
djacks1@stanford.edu

Elisse Chow
Stanford University
echow24@stanford.edu

Warren Xia
Stanford University
waxia@stanford.edu

Abstract

Dynamic NeRF models extend traditional NeRF models by allowing for the recreation of dynamic scenes. This project designs challenging synthetic datasets that expose the limitations and difficult cases for existing dynamic NeRF models. We focus on 4 potentially challenging aspects of dynamic scenes: many moving objects, objects leaving and entering scenes, objects obstructing views, and objects being subject to deformations such as squishing or fracturing. We evaluate three standard dynamic NeRF models and find that existing models struggle to reconstruct our more complex datasets to varying degrees.

1. Introduction

Traditional Neural Radiance Field models (NeRFs) [2] have been highly successful at reconstructing complex static scenes. In recent years, researchers have worked toward modifying existing NeRF approaches to support dynamic scenes, making them more reflective of the complexity seen in real world videos.

Despite these advancements, extending NeRF and similar techniques to handle dynamic scenes reconstructed from real world and generated video data presents several challenges. Dynamic scenes introduce complexities such as non-rigid deformations, occlusions, and the tracking of multiple moving objects, which challenge models designed for static environments. As a result, current methodologies often struggle with maintaining coherence and accuracy in the presence of motion.

Recent models such as D-NeRF [4], HyperNeRF [3], and TiNeuVox [1] have attempted to tackle the challenge of reconstructing dynamic scenes and have shown success on a variety of synthetic and real world datasets. While these existing test scenes have movement in them, often the movement represented is rather limited. Typically, videos feature a single moving object, like a 3D printer or a deforming human face, while much of the surrounding environment remains relatively static. Thus, we set out to create and provide additional synthetic scenes that will better reflect and showcase specific challenging situations posed by dynamic scenarios. Our new datasets include various types of motion, occlusions, and number of objects in the scene to provide a rigorous testing ground for evaluating model performance.

2. Background

2.1. Prior Dynamic NeRF Datasets

2.1.1 D-NeRF Dataset

The D-NeRF paper used a dataset of 8 different dynamic scenes. 7 of the scenes featured a single subject, often a humanoid. Other non-humanoid subjects included a LEGO model and a T-Rex. The one scene with multiple moving objects was the 3 bouncing balls scene, which had a red, blue, and green ball moving up and down as the camera rotated.

2.1.2 HyperNeRF Dataset

The HyperNeRF paper used a dataset of 17 scenes. Unlike D-NeRF, HyperNeRF used real world video and reconstructed the camera positions using COLMAP. The HyperNeRF dataset offers more variety than D-NeRF, with more complicated movements and changes in object topology. However, similar to D-NeRF, these videos often focus on a singular subject (such as a 3D printer, banana, or hand) which represents the majority of the dynamic movement in the frame.

2.2. D-NeRF

D-NeRF [4] extends the NeRF framework to handle scenes with temporal changes, enabling the reconstruction of dynamic scenes. Unlike traditional NeRF, which assumes static scenes, D-NeRF incorporates a temporal dimension into the NeRF model by training two MLPs: a Canonical Network and a Deformation Network. The Canonical Network encodes a fixed canonical representation of a scene, while the Deformation Network maps the

Figure 1. 3 frames from an example scene in the D-NeRF dataset. This kind of scene with a single object and limited movement is representative of many test cases in this dataset.

canonical representation into the deformed scene at a particular time. This dual-network approach allows D-NeRF to model complex deformations over time. However, due to the assumption of a fixed canonical representation, D-NeRF may struggle with certain types of deformations, potentially limiting its performance in highly complex dynamic scenes.

D-NeRF is trained on a sparse set of synthetic images of a dynamic scene captured with a monocular camera. For example, one dataset introduced in the D-NeRF paper captures a 3D animated man standing up from a crouching position, as illustrated in figure 1. This scenario exemplifies D-NeRF's capability to reconstruct detailed and temporally coherent dynamic scenes. The dataset, however has minimal movement occurring across the 150 training images.

2.3. HyperNeRF

Similar to D-NeRF, HyperNeRF [3] trains a network that encodes a canonical scene and a deformation field model. The authors argue that while this kind of deformable NeRF can capture any motion that preserves topology, it is not able to easily handle motion that changes the topology of the scene. To address this, HyperNeRF also trains a third slicing MLP which outputs an additional vector intended to represent a slice of a higher dimensional space. Both the inputs from the slicing model and deformation model are combined and input into the canonical scene model to get the final outputs. The method and dataset use real-world 15fps monocular video clips (waving a mobile phone in front of a moving scene) that are 30-60s long and registered using COLMAP.

2.4. TiNeuVox

TiNeuVox [1] integrates time-encoded voxel grids into the NeRF framework, enabling efficient modeling of dynamic scenes. Unlike D-NeRF, which uses MLPs to capture temporal changes, TiNeuVox represents a scene with a 4D voxel grid, where the additional dimension encodes time. This grid-based representation allows TiNeuVox to use the spatial coherence of voxels, potentially providing higher fidelity and more efficient reconstructions than D-NeRF or HyperNeRF.

TiNeuVox employs a dual-network architecture comprising a Static Network and a Dynamic Network. The Static Network encodes the time-invariant features of the scene into a 3D voxel grid, while the Dynamic Network captures the temporal variations and encodes them into the 4D voxel grid. This division of labor helps in efficiently handling the complexities of dynamic scenes.

Another key change in TiNeuVox is its multi-distance interpolation method, which allows the model to interpolate and extrapolate the scene's appearance at different time points more accurately. This method leverages distance-weighted interpolation to smoothly blend voxel values over time, enhancing the temporal consistency of the reconstructed scenes.

Since it uses a voxel representation TiNeuVox benefits from faster training times compared to purely MLP-based models like D-NeRF. The TiNeuVox paper claims it achieves better or similar rendering quality to D-NeRF and HyperNeRF while requiring less storage and training time. However, the voxel grid representation can be limiting in terms of resolution, and handling very fine details might require significant computational resources.

3. Approach

3.1. Synthetic Data Generation

Seven animated scenes were synthetically generated in Blender version 2.92 using a modified version of the script provided in the Blender files provided by Mildenhall et al.'s paper [2]. Each of the animations are 200 frames long, and as we update the time step, the camera rotates about the center of the scene giving us different views of the moving objects. Since we have used an 80%-10%-10% split for our training-testing-validation sets, we have around 160 frames in the training set, and 20 each for the test and validation set.

Matching the formats of the existing datasets, for HyperNeRF and D-NeRF (also used for TiNeuVox) we output camera transforms data and corresponding timestamps for each synthetic image generated.

We prepare the D-NeRF and HyperNeRF datasets slightly differently. HyperNeRF needs absolute timestamps—of any range. In our case we use 0 to 199. D-NeRF on the other hand, requires that the timestamps range from 0.0 to 1.0 regardless of the real timestamp in the original animation in each of the data splits. To prevent D-NeRF from having a bad starting condition causing a completely black or white render, the original start and end frames are duplicated and included in all of the data splits (train, test, val).

Please refer to the appendix to download the full dataset, and to see videos of each of the animated scenes.

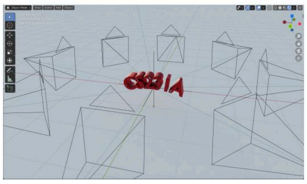

Figure 2. Blender Crumble Scene Set Up (Before The Crumbling)

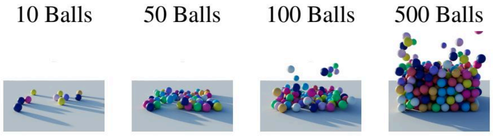

Figure 3. All bouncing ball datasets, frame $t = 199$

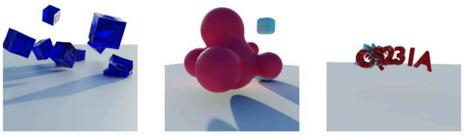

Figure 4. Jello $(t = 53)$, Metaballs $(t = 21)$, Crumble $(t = 5)$

3.2. Datasets Generated

3.2.1 Bouncing Balls

A key question we had was how well these models reconstruct scenes as the number of moving objects increased. To understand this, we created four different bouncing ball datasets with 10, 50, 100, and 500 colorful balls generated over the course of 200 frames. These datasets also provide test cases of many objects appearing in the scene after the first frame, and some leaving as they bounce above the area the camera focuses on. While other prior datasets have also used bouncing balls (3 slow moving balls in D-NeRF), the examples we create contain far more balls than those datasets. These ball datasets were generated with a particle and physics system through the Molecular Plus add-on [5] for Blender.

For the midterm report, these scenes contained the balls within a wireframe box with no ground plane present. We found during testing that this detail made it significantly harder for HyperNeRF and D-NeRF to render descent outputs. We speculate that this was due to the occlusions of the thin edges of the box which caused smearing in the depth reconstruction.

Since we wanted this dataset to focus more on the challenges of reconstructing the balls themselves rather than the challenges of reconstructing the thin wireframe, all of the bouncing ball datasets had their wireframe boxes removed and a ground plane added.

3.2.2 Jello Cubes

The Jello Cubes image set consists of 12 blue jello-like cubes falling from above onto a ground plane. These cubes deform heavily upon collision with the ground plane and other cubes, but become generally cubic when settled on the ground in the final frames. This dataset provides more examples of objects appearing in the scene after the first frame, but also gives us a test set to see how well the three models work with deformations that do not radically change the shape of the object. This example was modeled with built in Blender features and did not require any additional add-ons.

3.2.3 Metaballs

The Metaballs image set consists of 5 spherical pink metaballs each of different sizes, and 2 blue cubic metaballs slowly shifting position, merging, and spreading out over the course of 200 frames. This set provides a slower paced fusion deformation with changing topology and is intended as a more challenging case for shape reconstruction. Like the Jello Cubes this was created with native Blender 2.92 features.

3.2.4 CS231A Crumble

The CS231A Crumble image set consists of 3D text that spells out "CS231A" in red lettering. Over the course of 200 frames, the 3D letters explode into approximately 500 fractured shards with a light blue shading for the inner faces. This scene used Cell Fracture to split the 3D text, and simple rigid body physics to let the shards fall somewhat realistically. This represents an extreme case of objects drastically changing their appearance over the scene.

3.3. Evaluation Metrics

To evaluate the results of our dataset on each of the three models, we examine things qualitatively by providing a grid of images comparing the models on each of the scenes. Quantitatively, we use similar metrics that are standard in other dynamic NeRF papers including mean squared error (MSE), peak signal-to-noise ratio (PSNR), structural similarity (SSIM), and perceptual similarity (LPIPS).

3.4. Hardware Setup

We used 3 T4 n1-standard-4 Google Cloud Instances (4 vCPU, 2 core, 15 GB memory) to perform the training needed for our 3 models. Overall for a single dataset, our D-NeRF runs would roughly take 8 hours, TiNeuVox would converge in under 2 hours, and HyperNeRF would converge in roughly 6 hours.

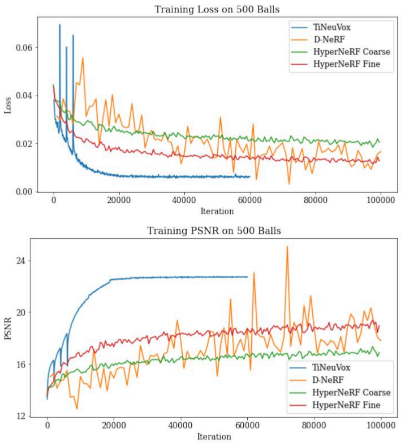

Figure 5. 500 Balls Training Iterations vs Loss and PSNR. HyperNeRF's codebase naturally tracks two different losses and PSNRs so both are reported here. Since HyperNeRF splits the PSNR we can't compare these directly with those of D-NeRF and HyperNeRF.

4. Experiments and Results

4.1. Training Setup

Images from all sets were preprocessed to have a white background. Image resolutions were reduced by a factor of two from their original resolution of 800x800. We used the default configurations and hyperparameters for each model.

Each of the models were trained for as many iterations as possible given our time constraints and compute credits. For HyperNeRF and TiNeuVox, we were able to run each of them to convergence as seen in Figure 4. D-NeRF was only run for 100,000 iterations (about 8 hours), while the original paper ran it for 800,000 iterations. Regardless of our attempts, D-NeRF did not converge.

4.2. Initial attempts

As mentioned previously, our ball datasets were initially set up with a box with thin wireframe edges and no ground plane in our midterm report. However, we found that sometimes the models would completely fail when working with this configuration, and return either an empty white or black image or a reconstruction with artifacts and heavy blurring. Both HyperNeRF and D-NeRF performed poorly even on an image set containing a singular static ball positioned in the middle of the bottom of the wireframe box.

Attempting to debug this issue, we noticed during our testing for the midterm report that most models performed better on the jello cubes dataset. This dataset differed from our original ball datasets in that it had a concrete ground plane and had no thin obstructors (like the edges of the box). As a result, the final dataset we evaluated on removed the thin box edges and added back this ground plane for the ball dataset. After doing this, the reconstructions improved. We speculate that the thin box edges posed a challenge to the NeRF model in general. We also speculate that adding in the ground plane could have helped provide additional information to the model, like shadows or some kind of depth reference for the scene. Ultimately we made this dataset change because while these results were interesting in what they revealed as potential challenges for even static NeRFs, they weren't really related to the challenges of dynamic scenes which were the focus of our project. Our final dataset better reflects the challenges associated with difficult object movement rather tricky geometry.

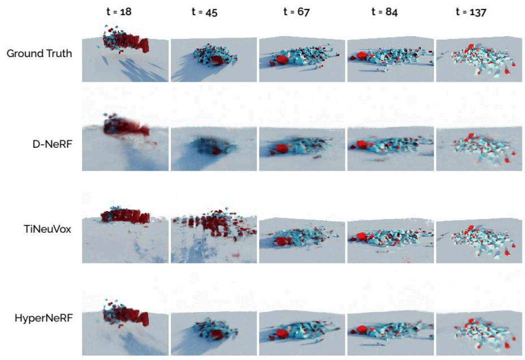

Figure 6. Crumble Renderings

4.3. Final Results

To do our comparisons, we have chosen five images from our testing set spanning the full time interval. These correspond to the following timestamps: $t = 18, 45, 7, 84, 137, 178$. These frames were used for the images across each of the figures and metrics tables in this section.

4.3.1 Effect of Movement Speed on Reconstruction

Figure 6 displays the renderings for different frames of the crumble dataset. Figure 7 displays their evaluation metrics. The frames at the beginning of the crumbling video have more movements than the later frames in which the pieces rest.

D-NeRF appears blurry, likely due to its training loss not converging. Despite that, D-NeRF has a better visual reconstruction and better metric results than TiNeuVox in earlier frames. TiNeuVox's renderings have many artifacts during

frame	Model	MSE	PSNR	SSIM	LPIPS
18	D-NeRF	0.013	18.704	0.763	0.362
	TiNeuVox	0.041	13.892	0.717	0.358
	HyperNerf	**0.003**	**25.757**	**0.875**	**0.269**
45	D-NeRF	0.005	23.177	0.844	0.301
	TiNeuVox	0.047	13.260	0.617	0.438
	HyperNerf	**0.002**	**26.861**	**0.906**	**0.243**
67	D-NeRF	0.006	22.232	0.812	0.260
	TiNeuVox	0.008	20.791	0.793	0.214
	HyperNerf	**0.003**	**25.091**	**0.898**	**0.200**
84	D-NeRF	0.005	23.055	0.842	0.232
	TiNeuVox	0.007	21.565	0.845	**0.191**
	HyperNerf	**0.004**	**24.309**	**0.890**	0.253
137	D-NeRF	0.005	23.402	0.846	0.235
	TiNeuVox	**0.002**	**27.937**	**0.950**	**0.070**
	HyperNerf	0.004	23.961	0.867	0.255

Figure 7. Crumble at Different Frames Metrics

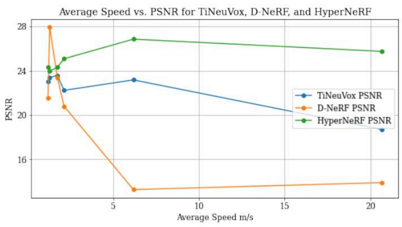

Figure 8. Crumble Average Speed vs PSNR

high movement frames, but it outperforms the other methods when the scene settles during the later frames. TiNeuVox has significantly worse metric scores than the other models during the first frame and significantly better metric scores during the final frame. HyperNeRF significantly outperformed the other models in all categories in frame 18 and 45 when there was the most movement.

Figure 8 plots the average speed of all the objects between the current and previous frame against the PSNR of the current frame. The frames with lower average speeds had relatively consistent PSNR scores for all three models. At higher average speeds, the models differed significantly in PSNR values. HyperNeRF had similar PSNR at fast speeds as its slow speeds. Surprisingly, HyperNeRF had slightly higher PSNR scores at its faster speeds evaluated. Both D-NeRF and TiNeuVox's PSNR dipped at higher speeds, with D-NeRF's PSNR value being reduced by over a third compared to the slower speeds.

4.3.2 Effect of Number of Moving Objects on Reconstruction

We tested the effects of increasing the number of moving objects had on reconstruction by comparing the models' reconstructions of datasets containing 10, 50, 100, and 500 balls We picked a time relatively early on in the video ($t = 18$) where the balls are still moving and haven't yet settled, as they do at the end of the video. As seen in 10 for the three models, MSE, PSNR, and LPIPS increase with the number of moving objects in the scene. Likewise, SSIM decreases as the number of moving objects decreases. This translates qualitatively to blurrier images with more artifacts.

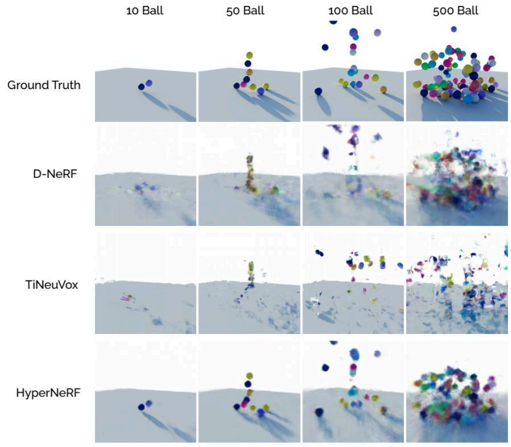

Figure 9. Bouncing Balls Renderings ($t = 18$)

# of Balls	Model	MSE	PSNR	SSIM	LPIPS
10	D-NeRF	0.002	27.693	0.927	0.251
	TiNeuVox	0.003	25.633	0.910	0.227
	HyperNerf	**0.0003**	**35.234**	**0.975**	**0.155**
50	D-NeRF	0.007	21.637	0.843	0.287
	TiNeuVox	0.010	20.214	0.820	0.297
	HyperNerf	**0.001**	**20.214**	**0.935**	**0.197**
100	D-NeRF	0.014	18.598	0.743	0.480
	TiNeuVox	0.025	16.060	0.699	0.828
	HyperNerf	**0.004**	**24.253**	**0.828**	**0.376**
500	D-NeRF	0.034	14.699	0.537	0.542
	TiNeuVox	0.067	11.737	0.412	0.605
	HyperNerf	**0.011**	**19.471**	**0.652**	**0.501**

Figure 10. Bouncing Balls Metrics ($t = 18$)

Comparing the three models qualitatively, we find that HyperNeRF provides the best reconstructions across all the datasets compared to TiNeuVox and D-Nerf (see 9). For all numbers of balls, HyperNeRF is able to reconstruct the scene with very few/small artifacts. Notably, HyperNeRF still struggles a bit with reconstructing the shadows in the 100 ball case. Both D-NeRF and TiNeuVox have noticeable artifacts like extreme blurring. They both struggle in general with reconstructing the moving ball dataset even with small numbers of moving objects. TiNeuVox and D-NeRF appear to render incomplete pieces of the balls as smaller blob shapes.

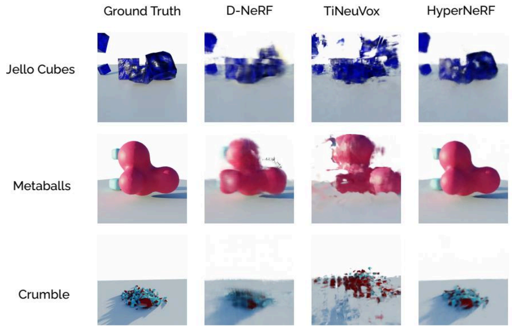

Figure 11. Deformation Renderings ($t = 18$)

Dataset	Model	MSE	PSNR	SSIM	LPIPS
Jello	D-NeRF	0.018	17.452	0.735	**0.330**
	TiNeuVox	0.064	11.962	0.642	0.467
	HyperNerf	**0.005**	**23.195**	**0.825**	0.427
Metaballs	D-NeRF	0.011	19.427	0.806	0.628
	TiNeuVox	0.050	12.941	0.628	0.460
	HyperNerf	**0.001**	**31.461**	**0.953**	**0.150**
Crumble	D-NeRF	0.005	23.177	0.844	0.301
	TiNeuVox	0.047	13.260	0.616	0.438
	HyperNerf	**0.002**	**26.861**	**0.906**	**0.243**

Figure 12. Deformation Image Metrics ($t = 18$)

4.3.3 Effect of Movement Types on Reconstruction

We tested different kinds of movement reconstruct by comparing the models' reconstructions of the Jello Cubes, Metaballs, and CS231A Crumble datasets. Figure 11 displays the renderings of the jello, metaballs, and crumble datasets on the 18th frame. Figure 12 displays their evaluation metrics. Again, we see that HyperNeRF significantly outperforms the other models for all of the metrics except for LPIPS on the Jello dataset. In particular with Metaballs, HyperNeRF's metrics were significantly better than the other models. The second closest, D-NeRF exhibits edge artifacts and misses the construction of the small cubes in the background. One potential reason for this is HyperNeRF's additional slicing MLP, which was intended to better cover cases with changing topology.

Across the board, TiNeuVox does the worst, exhibiting the most artifacts and the worst metrics. Especially on the Crumble dataset, TiNeuVox seems to include data from earlier time frames in its reconstruction, leading to significant corruption of the image.

5. Conclusion

Through our testing, we found that our new datasets were able to reveal potential limitations with existing Dynamic NeRF models. We found that with the exception of HyperNeRF, most methods struggled with fast movements, but had improved rendering qualities as speeds decreased. D-NeRF had the most blurry rendering artifacts while TiNeuVox had the most choppy rendering artifacts. HyperNeRF consistently outperformed the other models. Their additional slicing MLP does seem, based on the data, to let it reconstruct complex motions better.

While HyperNeRF wins on the majority of metrics and reconstructed images, there are still certain cases where the other models beat out HyperNeRF. This was infrequent for TiNeuVox and only occurred once for D-NeRF. This highlights the importance of a diverse evaluation set that is able to test a larger spectrum of movement possibilities.

We found that both a qualitative and quantitative approach is important for evaluating the performance of these models. There are often subtle visual differences, like the missing background cube for metaballs from the D-NeRF model that are hard to detect by inspection alone. Likewise the numerical metrics don't quite capture exactly what kinds of artifacts occur in the final renders.

Overall we find that there is great potential to make dynamic NeRF datasets more complex, diverse, and representative. We hope such developments can illuminate the characteristics of existing approaches and lead to better and more robust dynamic NeRF models in the future.

5.1. Future Work

While we found cases that challenged D-NeRF and TiNeuVox, for the most part HyperNeRF was able to reconstruct all of our test cases reasonably well. We think that future work developing even more challenging datasets for HyperNeRF could be useful in establishing more of the limitations of that method.

Another potential topic to investigate is training time. While we tried to train everything to convergence, it might be worthwhile for practical implementations to also investigate how the level of movement might affect how long it takes for a dynamic NeRF to train.

One test case which we rendered, but were unable to train on due to time/credits constraints was a dataset that tested twisting motions for objects. We think this could have provided additional information about how well these methods reconstructed particular kinds of movements. We didn't attempt to modify any of the existing methods in this project so it is unclear whether some of these problems could be resolved by tuning up some of the hyperparameters, like HyperNeRF's dimensionality. With more time and compute, we could further clarify how some of these parameters can affect the kind of motion that can be reconstructed.

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. In *SIGGRAPH Asia 2022 Conference Papers*, 2022. 1, 2

[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. In *ECCV*, 2020. 1, 2

[3] Keunhong Park, Utkarsh Sinha, Peter Hedman, Jonathan T. Barron, Sofien Bouaziz, Dan B Goldman, Ricardo Martin-Brualla, and Steven M. Seitz. Hypernerf: A higher-dimensional representation for topologically varying neural radiance fields. *ACM Trans. Graph.*, 40(6), dec 2021. 1, 2

[4] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. *arXiv preprint arXiv:2011.13961*, 2020. 1

[5] u3dreal. molecular-plus. 2021. 3

Appendix

We ran each of the model from their respective repos with very minor modifications. They can be found at:

- D-NeRF: https://github.com/albertpumarola/D-NeRF
- TiNeuVox: https://github.com/hustvl/TiNeuVox
- HyperNeRF: https://github.com/google/hypernerf

The dataset created for this paper and the code used to process the data can be found here: https://drive.google.com/drive/folders/1IP06-Vdd-sHY8h_WHqdiIT_r8HKg00HT?usp=sharing. Please refer to the subfolder titled "Videos" to view the ground truth animations.
Please see our final renders here:
https://drive.google.com/drive/folders/1YAelMjdB7ElVQuSC1onE7ZuGeBvMvgRk?usp=sharing.