Dynamic 6-DoF Volumetric Video Generation: Software Toolkit and Dataset

Mufeng Zhu*, Yuan-Chun Sun†, Na Li*, Jin Zhou‡, Songqing Chen‡, Cheng-Hsin Hsu†, and Yao Liu*
*Rutgers University, Piscataway, NJ, USA
†National Tsing Hua University, Hsinchu, Taiwan
‡George Mason University, Fairfax, VA, USA

***Abstract*—Volumetric video streaming has become increasingly popular in recent years due to its support of 6 degrees-of-freedom (6-DoF) exploration. There is, however, a shortage of dynamic 6-DoF content suitable for comparing the performance among heterogeneous volumetric video representations. This paper introduces a software toolkit for creating both a dataset of dynamic 6-DoF content in point clouds and a dataset for training and testing neural-based representations such as neural radiance fields (NeRF). Starting with freely available 3D assets online, our software toolkit uses the Blender Python API to generate training and testing datasets for neural-based dynamic volumetric model training. The created datasets are compliant with existing neural-based model training and rendering frameworks. The software can also construct point cloud sequences derived from synthetic dynamic 3D meshes. This further facilitates comparing point clouds and neural-based methods for volumetric video representation. We release the software toolkit along with a rich set of sequence datasets generated in compliance with the permissions granted by the original 3D asset creators. With our toolkit and dataset, we aim to facilitate research from the multimedia systems community to support practical volumetric streaming. Our software toolkit and dataset are available at: `https://6-dof-dynamic-content-software.github.io/`.**

***Index Terms*—Software, Dataset, Point Cloud, Neural Radiance Field, Volumetric Content**

I. Introduction

Volumetric video is an emerging media form that allows users to experience and navigate a scene in 6 degrees-of-freedom (6-DoF)–from any perspective at any position. It has been widely used in immersive applications such as virtual reality (VR), augmented reality (AR), and mixed reality (MR). One notable representation of volumetric videos is the point cloud. A point cloud is typically generated through active or passive capturing methods, such as 3D scanning via LiDAR or photogrammetry, containing geometry and color attributes. However, raw point clouds from depth cameras can contain millions of points with noise. Consequently, many of the studies focus on point cloud registration and compression. Point cloud registration aligns multiple point clouds to create an accurate representation of a 3D scene. For example, the Iterative Closest Point [1] algorithm is a feature-based registration method that iteratively refines the transformation to minimize the distance between corresponding points in two point clouds. Meanwhile, due to the large size of point cloud sequences, point cloud compression (PCC) plays an important role in dynamic point cloud streaming. There are two main categories of PCC methods [2]: geometry-based PCC (G-PCC) focuses on compressing the geometry information, including spatial coordinates and connectivity; and video-based PCC (V-PCC) [3] projects point clouds on to 2D planes and uses existing 2D video codec for compression.

Besides point clouds, an emerging approach is to use neural-based methods for 3D representation. Given a set of images of the same scene taken with different camera poses, these methods train neural models that can predict novel views of the scene with high fidelity. For example, neural radiance fields (NeRF) [4] uses a single multi-layer perceptron (MLP) for scene representation. However, the original NeRF model suffers from very long training time, e.g., 1 to 2 days. Following NeRF, many enhanced neural radiance filed models [5]–[8] have been developed and published, achieving faster training and rendering speed, with Instant-NGP notably can finish training within seconds. Despite NeRF's significant progress in static scene representation, development in dynamic scenes still faces challenges. D-NeRF [9] and DyNeRF [10] are two early proposed dynamic NeRF models. D-NeRF takes 48 hours to complete training a synthetic model, while DyNeRF requires 8 GPUs running for a week to train a single scene captured from the real world. Recently, K-Planes [11] uses 6 planes to represent dynamic scenes while achieving faster training times of 1.8 hours on the DyNeRF dataset and 52 minutes on the D-NeRF dataset. K-Planes also has a compact model size of 3–50 MB, which is suitable for today's access network bandwidth for transmission and streaming. Despite the improved training time, the rendering time of dynamic models remains a constraint. For example, ReRF [12] reaches up to 20 fps, which still fails to meet the requirement for real-time volumetric video streaming (e.g., 30 fps).

To facilitate further research in addressing the practical challenges of volumetric video streaming using neural-based representations and point clouds, datasets of dynamic 6-DoF content are needed. Today, a number of dynamic point cloud sequence datasets are available, including `8iVFBV2` [13], `vsenseVVDB2` [14], and Panoptic Studio [15]. More recently, the Dynamic 3D point cloud [16] dataset offers point clouds with accurate motion vectors as ground truth, derived from synthetic 3D models. Existing datasets for neural-based representations primarily focus on static scenes, e.g., the original NeRF dataset [4] and the LLFF dataset [17]. However, datasets for dynamic scenes are limited. D-NeRF, DyNeRF, and Immersive video datasets [18] are frequently used to

evaluate dynamic neural-based representation methods. ReRF introduces an outside-looking-in dynamic dataset from the real world, which only includes three dynamic scenes. To the best of our knowledge, none of these dynamic datasets enables fair comparisons between point cloud and neural-based representations. This prevents the multimedia systems community from evaluating different design choices on the 3D representation in volumetric streaming systems.

In this paper, we introduce a software toolkit for creating dynamic 6-DoF content in both the point cloud and neural-based representations. We aim to bridge the gap between the research in representations of dynamic 6-DoF content and the research in addressing system-level challenges in volumetric video streaming. Our software toolkit can enable the generation of training and testing datasets for neural-based research in volumetric video streaming by rendering animated synthetic scenes and point clouds in Blender[1]. Our software toolkit can also generate dynamic point cloud sequences from synthetic dynamic scenes by directly sampling points on the 3D mesh representation, compatible with relevant point cloud research. This further facilitates comparing point clouds and neural-based methods for volumetric video representation and streaming. As a showcase, we generate and release a dataset of dynamic 6-DoF content using our software toolkit, available at: `https://6-dof-dynamic-content-software.github.io/`. Overall, our software toolkit and dataset serve the following purposes:

- Our software toolkit can generate training and testing datasets based on animated synthetic scenes for neural-based model learning.
- Our software toolkit can generate datasets for training and testing neural-based methods given existing dynamic point cloud sequences.
- Finally, our software can generate dynamic point cloud sequences derived from animated synthetic scenes, e.g., in 3D meshes.

II. Related Work

While our software toolkit and dataset are the first of their kind, prior datasets have been designed for either point cloud or neural-based representations, as the following survey shows.

A. Existing Dynamic Point Cloud Datasets

The **8iVFBV2** [13] dataset, known as 8i Voxelized Full Bodies, comprises four dynamic point cloud sequences captured from the real world. Each sequence was recorded by 42 RGB cameras at 30 fps with a duration of 10 seconds. The cameras are configured in 14 clusters, each functioning as a local RGBD camera to capture depth information. This dataset has been used to evaluate various research problems, including point cloud compression [3], [19], interpolation [20], and bitrate allocation [21]. **VsenseVVDB2** [14] includes four real-world dynamic point cloud sequences. Additionally, this dataset provides compressed point clouds using V-PCC, featuring various combinations of compression parameters. It also includes 3D meshes of the recorded sequences. This dataset has been widely used for visual quality assessment in point clouds [22]–[24]. **Dynamic 3D point cloud dataset** [16] comprises nine dynamic point cloud sequences generated from scenes with diverse animations. Each scene includes ground truth for motion estimation vectors, enabling the evaluations of algorithms like point cloud registration and error concealment. This dataset offers valuable resources for advancing research in dynamic 3D point clouds and associated algorithms.

B. Existing Neural-Based Datasets

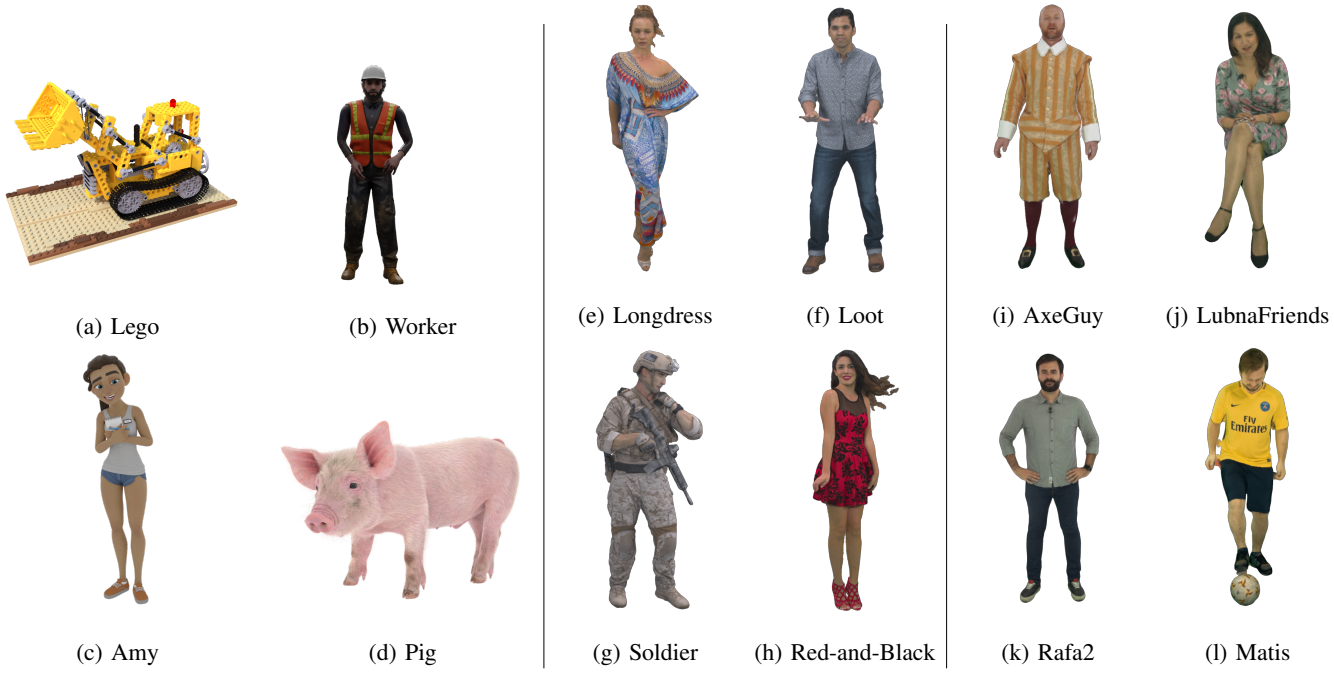
A number of static datasets exist for training and evaluating neural-based approaches. NeRF offered the first dataset generated from synthetic scenes. Additionally, LLFF and Mip-NeRF360 datasets provided scenes captured from the real world and are widely used in neural-based approaches [6]–[8]. More recently, OMMO [25] provided a new large-scale outdoor multimodal dataset captured by a drone.

Existing datasets of dynamic scenes for training neural-based models, however, are limited. **D-NeRF** [9] presents eight synthetic dynamic scenes with animations in varying duration, spanning from 50 to 200 frames per scene. The training dataset is derived from a monocular camera that renders views from different viewpoints at each step. The datasets include 100 to 200 rendered views with a resolution of 800x800 pixels. The dataset has been split into training/validation/test sets and used in the evaluations of K-planes [11] and DyNeRF [10]. **DyNeRF** offers plenoptic video datasets containing videos captured from six real-world dynamic scenes. Each scene comprises 15 to 21 10-second videos at 30 fps, captured by 21 static GoPro Black Hero 7 cameras in forward-facing directions. Despite the cameras being time-synchronized, some videos still exhibit apparent offsets. Their intrinsic and extrinsic parameters are obtained through COLMAP [26]. The central camera is used for testing while the rest of the camera views are used for training. The DyNeRF dataset has been used to evaluate multiple works such as NeRFplayer [27] and K-planes. **Immersive video** [18] provides a light field video dataset with 15 dynamic real-world scenes. Captured with 46 4K time-synchronized fisheye cameras on a hemispherical surface, these videos offer inside-looking-out directions. All recordings maintain a 30 fps frame rate, and raw video data are provided. This dataset has been used in the evaluations of DyNeRF and NeRFplayer.

III. Our Software Toolkit and Dataset

Our software and dataset generation are based on Blender, a software capable of creating, editing, animating, and rendering 3D meshes. It supports multiple cameras in each scene and automatically synchronizes all cameras when capturing, which is challenging in real-world setups. In this section, we describe the dynamic 3D contents we work with and the procedure for generating datasets for both point clouds and neural-based representations. The entire dataset can be generated simply via command line execution of Python scripts.

[1]https://www.blender.org/



(a) Lego (b) Worker (e) Longdress (f) Loot (i) AxeGuy (j) LubnaFriends

(c) Amy (d) Pig (g) Soldier (h) Red-and-Black (k) Rafa2 (l) Matis

Fig. 1: Front-facing images all 12 dynamic 3D contents. (a)-(d): Our `Synthetic Dynamic Scenes` sequences; (e)-(h): `8iVFBV2` sequences [13]; (i)-(l): `vsenseVVDB2` sequences [14]. From these contents, our software toolkit can generate datasets compatible with training neural-based representations. Our software toolkit can also generate dynamic point cloud sequences from the `Synthetic Dynamic Scenes` dataset for point cloud research.

A. Dynamic Scenes Descriptions

Figure 1 shows the front-facing images of 12 3D contents that we use for demonstrating the use of our software toolkit. These 12 contents are from three distinct sources.

- We collected `Synthetic Dynamic Scenes` from 3D assets that are freely available online. These include four animated Blender 3D models: Lego [4], Worker from the Dynamic Point Cloud dataset [16], Amy[2], and Pig[3]. Animation for "Lego" includes raising and lowering the bulldozer's bucket, created by manipulating the built-in control panel in the original `.blend` file. The "Worker" model is animated to dance. The "Amy" model includes animations of a girl talking, blinking, and swaying her body. The "Pig" model is designed with animations including shaking head, shaking ears, and wagging tail. All the models are animated with 60 frames in total.
- `8iVFBV2` [13] contains four dynamic point cloud sequences: Longdress, Loot, Soldier, and RedandBlack in raw `.ply` files.
- `vsenseVVDB2` [14] contains four dynamic point cloud sequences – AxeGuy, LubnaFriends, Rafa2, and Matis, also in raw `.ply` files. Both `8iVFBV2` and `vsenseVVDB2` share the exact spatial resolution of 1024x1024x1024 voxels. All point cloud sequences have 300 frames in total.

For our `Synthetic Dynamic Scenes`, we describe the procedure for generating datasets for training/testing neural-based representations in Section III.C. We also describe how we generate point clouds from 3D meshes in Blender in Section III.D. Furthermore, for `8iVFBV2` and `vsenseVVDB2` that are in raw point cloud formats, we describe in Section III.B how we render point clouds in Blender so that we reuse the pipeline in Section III.C to generate training/testing datasets for neural-based representations.

[2]https://www.blender.org/download/demo-files/

[3]https://blendermarket.com/products/piggy-animations-vfx-grace

B. Point Cloud Rendering in Blender

Blender does not inherently support the rendering of points, e.g., `.ply` files. To address this problem, we adopt a strategy of generating faces for the points. Generating meshes from point clouds is well supported in Open3D [28], an open-source library used to process meshes and point clouds, using Poisson Surface Reconstruction [29]. However, we found that noise in the point clouds seriously affects the generated meshes, as shown in Figure 2. We, therefore, directly assign faces for each point and then render all these faces in Blender as follows.

We first import point clouds as vertices with geometry positions and colors into Blender. Then, using the geometry node in Blender, we create a cube centered on each vertex. To achieve this, we first create a cube mesh using the geometry node of type "Cube Mesh" to build an instance. Then, we add geometry nodes of type "Instance on Points" and "Realize Instances" to the "Cube Mesh" node to assign point geometry to the cube mesh instance. Subsequently, we create a new material, linking the color attribute of the point cloud to the material. This is achieved by adding the "Color Attribute" node and making it the base color of the "Diffuse BSDF" node. After this, we add the "Set Material" node to the "Cube Mesh" node we created before, assigning color attributes to the

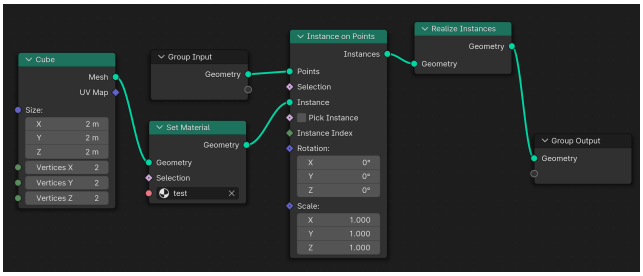
Fig. 2: Comparison between Poisson Surface Reconstruction [29] and our method. Left: rendering results from meshes generated from Poisson Surface Reconstruction in Open3D [28]. Right: rendering results using our method, closely resembling the original content.

cube. Given that the rendered image in Blender is influenced by default lighting and shadows, to render authentic RGB colors of points, we set the output color management type to "Standard" and render output channel to "Diffuse Color". With this method, point clouds can be rendered in Blender successfully. The complete node maps are shown in Figure 3.

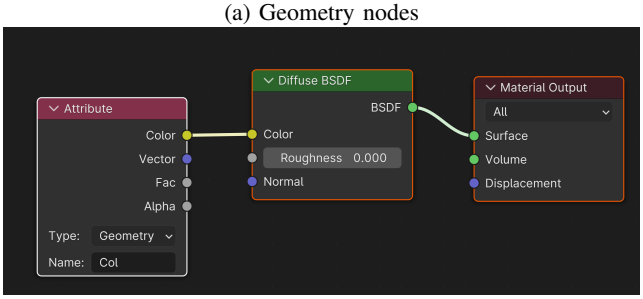
C. Training and Testing Dataset Generation

One advantage of using Blender is that we can set virtual cameras anywhere with any timestamp in the Blender world coordinates and obtain accurate camera parameters. The precision of camera parameters is very important in NeRF-like models. Without them, the trained model may have low visual quality. To generate training and test datasets, we first place the model's center at the origin (0, 0, 0). For `.blend` Blender meshes, we had to translate some meshes along the Z axis to make it approximately located in the origin. For `8iVFBV2` and `vsenseVVDB2` datasets, the point cloud positions span a range of [0, 1024]. Given that the default unit in Blender is 1 meter, i.e., the point cloud has a dimension of 1024 meters, it is difficult to capture the full body using a basic camera. Thus, we use a scale factor to resize the point clouds, bringing them to a more manageable size of approximately 2.3 meters to 2.5 meters in height within Blender's world coordinates. We recalculate the center of each point cloud by setting its origin to the center of volume and placing it at (0, 0, 0). Finally, to appropriately align the scaled and translated model, we apply a rotation to ensure its up vector is +Z, and the XY plane aligns parallel to the ground plane. Detailed transformation parameters are provided in our dataset.

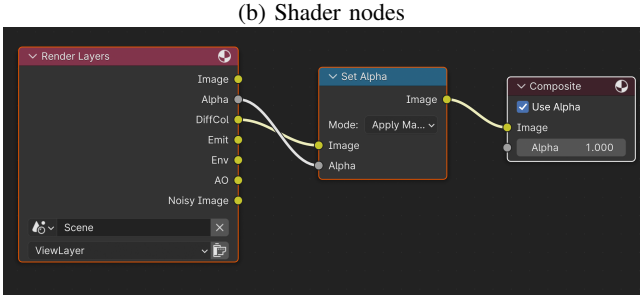
After properly initializing meshes, we follow the original NeRF [4] work to generate training and testing images of the dynamic 6-DoF contents. We place virtual cameras at different positions by rotating around the origin with randomly generated Euler angles, starting from position (0, 4.0, 0.5). To ensure consistent camera orientation and distance from the origin, we use Blender's "Track To" camera constraint, directing the camera to look at the origin throughout its



(a) Geometry nodes



(b) Shader nodes



(c) Composite nodes

Fig. 3: Node maps for rendering point clouds in Blender.

rotation. To maintain synchronization across all cameras, we import and render point clouds frame by frame and set the seed to a fixed value for different frames to yield the same set of Euler angles.

For Blender meshes, we render all the frames (full animation) at one position first, then move to the next position and render all the frames again. Generally, we set different random seeds for training data and testing data. The total number of views for both the training and testing datasets, along with the desired output resolution, is adjustable. As an example, we configure the software toolkit to generate 80 views for training and 20 views for testing, each with a resolution of 800x800 pixels, saving the rendered images as `.png` files. The overall pipeline of training/testing dataset generation from point clouds in `.ply` files is shown in Figure 4.

We are not allowed to redistribute the `.blend` files and the `8iVFBV2` and `vsenseVVDB2` datasets, so we only release the generated training and testing data in the `Synthetic Dynamic Scenes` dataset. We provide codes for training and testing dataset generation, which can be adapted for any 3D meshes/point clouds besides the 12 ones described in this paper. Interested users can download other 3D meshes, point clouds and large realistic scenes with complex background supported by Blender, and use our software toolkit to generate the datasets themselves.

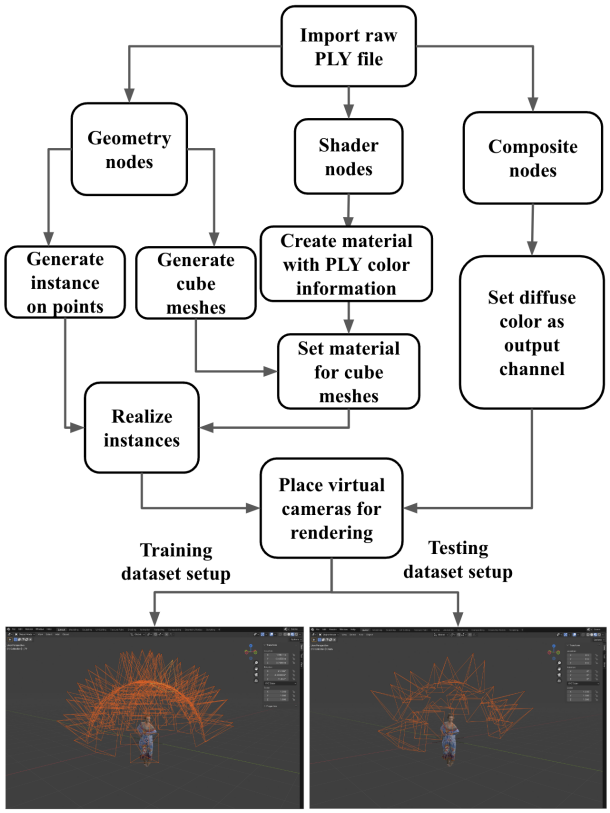


Fig. 4: Overall pipeline of generating neural-based training/testing datasets from point clouds in Blender.

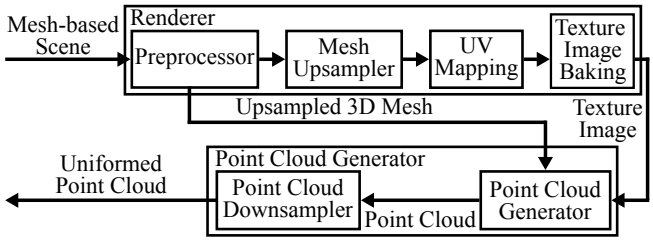


Fig. 5: Overall pipeline for generating point clouds from 3D meshes in Blender.

D. Dynamic Point Cloud Sequence Generation

To make the `Synthetic Dynamic Scenes` dataset useful for point cloud research, our software toolkit can generate dynamic point cloud sequences derived from synthetic scenes. The complete pipeline of generating point clouds using our software toolkit, shown in Figure 5, contains two components: *renderer* and *point cloud generator*.

First, we preprocess each mesh-based scene in the renderer to eliminate unnecessary components and add scene lights. Due to the limited number of vertices in the original Blender model, the exported point clouds from Blender are too sparse. To address this, we upsample meshes by evenly inserting vertices along each edge and connecting them to increase the number of meshes and vertices. Subsequently, we bake the color, texture, and shadows of each upsampled object into a new texture image to retrain the object's appearance with the effects of lights and shadows.

Fig. 6: Sample rendered point clouds generated from Blender 3D models in `Synthetic Dynamic Scenes` Lego, Worker, Amy, and Pig.

In the point cloud generator, the point cloud converter maps new texture images to the upsampled 3D meshes, using all vertices to form the point cloud. We duplicate the entire pipeline for each frame and save individual point clouds in `.ply` files with binary little endian 1.0 format, forming dynamic point cloud sequences. With our method, the generated point cloud of the synthetic scene with hundreds of objects, e.g., Lego, may have a very large number of points. Therefore, our software toolkit includes a point cloud downsampler, implemented based on voxel downsampling in Open3D [28], to create a uniformly downsampled point cloud. We release the generated point clouds of `Synthetic Dynamic Scenes` as illustrated in Figure 6.

IV. Sample Usage of Our Dataset

The dataset created using our software toolkit can be used to further research in neural-based representations of 3D scenes, especially for dynamic NeRF-like models. We have tested our datasets on three popular neural-based models, Nerfacto [7] and Instant-NGP [8] for static scene models, and K-Planes [11] for dynamic scene models. Because some existing dynamic NeRF works, such as D-NeRF [9] and DyNeRF [10], require long training time, we only considered K-Planes as the dynamic mode our experiments. With that said, our datasets are compatible with D-NeRF and DyNeRF. Figure 7 shows that our dataset works well with different neural-based models and their training and evaluation frameworks. Other dynamic NeRF-like models, including NeRFplayer [27] and ReRF [12], may require coordinate transformation and format conversion of `.json` files to align with the requirements of their models.

V. Conclusion

This paper introduces a software toolkit designed to generate datasets of dynamic scenes for volumetric videos. This includes the generation of training and test datasets from synthetic animated objects and dynamic point cloud sequences, enabling their use in neural-based 3D representations. The software toolkit also facilitates the generation of dynamic point cloud sequences derived from synthetic scenes. These datasets hold significant potential for diverse research fields, including dynamic NeRF-like neural-based representations and dynamic point cloud compression. We hope our software toolkit can enable more future research in the design, implementation, and evaluation of volumetric video streaming systems.

(a) NeRFstudio with Nerfacto model [7] (b) Instant-NGP [8] (c) NeRFstudio with K-Planes model [11]

Fig. 7: Our datasets work well with different NeRF-like models and their training and evaluation frameworks.

Acknowledgments. We appreciate constructive comments from anonymous referees. This work is partially supported by NSF under grants CNS-2007153, CNS-2200042, and CNS-2200048.

REFERENCES

[1] Z. Zhang, "Iterative point matching for registration of free-form curves and surfaces," *International journal of computer vision*, vol. 13, no. 2, pp. 119–152, 1994.

[2] D. Graziosi, O. Nakagami, S. Kuma, A. Zaghetto, T. Suzuki, and A. Tabatabai, "An overview of ongoing point cloud compression standardization activities: Video-based (v-pcc) and geometry-based (g-pcc)," *APSIPA Transactions on Signal and Information Processing*, vol. 9, p. e13, 2020.

[3] S. Schwarz, M. Preda, V. Baroncini, M. Budagavi, P. Cesar, P. A. Chou, R. A. Cohen, M. Krivokuća, S. Lasserre, Z. Li *et al.*, "Emerging mpeg standards for point cloud compression," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, no. 1, pp. 133–148, 2018.

[4] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," in *ECCV*, 2020.

[5] A. Chen, Z. Xu, A. Geiger, J. Yu, and H. Su, "Tensorf: Tensorial radiance fields," in *European Conference on Computer Vision (ECCV)*, 2022.

[6] J. T. Barron, B. Mildenhall, D. Verbin, P. P. Srinivasan, and P. Hedman, "Mip-nerf 360: Unbounded anti-aliased neural radiance fields," *CVPR*, 2022.

[7] M. Tancik, E. Weber, E. Ng, R. Li, B. Yi, J. Kerr, T. Wang, A. Kristoffersen, J. Austin, K. Salahi, A. Ahuja, D. McAllister, and A. Kanazawa, "Nerfstudio: A modular framework for neural radiance field development," in *ACM SIGGRAPH 2023 Conference Proceedings*, ser. SIGGRAPH '23, 2023.

[8] T. Müller, A. Evans, C. Schied, and A. Keller, "Instant neural graphics primitives with a multiresolution hash encoding," *ACM Trans. Graph.*, vol. 41, no. 4, pp. 102:1–102:15, Jul. 2022. [Online]. Available: https://doi.org/10.1145/3528223.3530127

[9] A. Pumarola, E. Corona, G. Pons-Moll, and F. Moreno-Noguer, "D-nerf: Neural radiance fields for dynamic scenes," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 10 318–10 327.

[10] T. Li, M. Slavcheva, M. Zollhoefer, S. Green, C. Lassner, C. Kim, T. Schmidt, S. Lovegrove, M. Goesele, R. Newcombe *et al.*, "Neural 3d video synthesis from multi-view video," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 5521–5531.

[11] S. Fridovich-Keil, G. Meanti, F. R. Warburg, B. Recht, and A. Kanazawa, "K-planes for radiance fields in space, time, and appearance," 2023.

[12] L. Wang, Q. Hu, Q. He, Z. Wang, J. Yu, T. Tuytelaars, L. Xu, and M. Wu, "Neural residual radiance fields for streamably free-viewpoint videos," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2023, pp. 76–87.

[13] E. d'Eon, B. Harrison, T. Myers, and P. A. Chou, "8i voxelized full bodies-a voxelized point cloud dataset," *ISO/IEC JTC1/SC29 Joint WG11/WG1 (MPEG/JPEG) input document WG11M40059/WG1M74006*, vol. 7, no. 8, p. 11, 2017.

[14] E. Zerman, C. Ozcinar, P. Gao, and A. Smolic, "Textured mesh vs coloured point cloud: A subjective study for volumetric video compression," in *Twelfth International Conference on Quality of Multimedia Experience (QoMEX)*, 2020.

[15] H. Joo, H. Liu, L. Tan, L. Gui, B. Nabbe, I. Matthews, T. Kanade, S. Nobuhara, and Y. Sheikh, "Panoptic studio: A massively multiview system for social motion capture," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 3334–3342.

[16] Y.-C. Sun, I.-C. Huang, Y. Shi, W. T. Ooi, C.-Y. Huang, and C.-H. Hsu, "A dynamic 3d point cloud dataset for immersive applications," in *Proceedings of the 14th Conference on ACM Multimedia Systems*, 2023, pp. 376–383.

[17] B. Mildenhall, P. P. Srinivasan, R. Ortiz-Cayon, N. K. Kalantari, R. Ramamoorthi, R. Ng, and A. Kar, "Local light field fusion: Practical view synthesis with prescriptive sampling guidelines," *ACM Transactions on Graphics (TOG)*, 2019.

[18] M. Broxton, J. Flynn, R. Overbeck, D. Erickson, P. Hedman, M. DuVall, J. Dourgarian, J. Busch, M. Whalen, and P. Debevec, "Immersive light field video with a layered mesh representation," vol. 39, no. 4, pp. 86:1–86:15, 2020.

[19] J. Wang, D. Ding, Z. Li, and Z. Ma, "Multiscale point cloud geometry compression," in *2021 Data Compression Conference (DCC)*. IEEE, 2021, pp. 73–82.

[20] A. Akhtar, Z. Li, G. Van der Auwera, and J. Chen, "Dynamic point cloud interpolation," in *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022, pp. 2574–2578.

[21] S. Wang, M. Zhu, N. Li, M. Xiao, and Y. Liu, "Vqba: Visual-quality-driven bit allocation for low-latency point cloud streaming," in *Proceedings of the 31st ACM International Conference on Multimedia*, 2023, pp. 9143–9151.

[22] Q. Liu, H. Su, Z. Duanmu, W. Liu, and Z. Wang, "Perceptual quality assessment of colored 3d point clouds," *IEEE Transactions on Visualization and Computer Graphics*, 2022.

[23] K. Cao, Y. Xu, and P. Cosman, "Visual quality of compressed mesh and point cloud sequences," *IEEE Access*, vol. 8, pp. 171 203–171 217, 2020.

[24] Y. Nehmé, F. Dupont, J.-P. Farrugia, P. Le Callet, and G. Lavoué, "Visual quality of 3d meshes with diffuse colors in virtual reality: Subjective and objective evaluation," *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 3, pp. 2202–2219, 2020.

[25] C. Lu, F. Yin, X. Chen, W. Liu, T. Chen, G. Yu, and J. Fan, "A large-scale outdoor multi-modal dataset and benchmark for novel view synthesis and implicit scene reconstruction," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 7557–7567.

[26] "COLMAP," https://colmap.github.io/.

[27] L. Song, A. Chen, Z. Li, Z. Chen, L. Chen, J. Yuan, Y. Xu, and A. Geiger, "Nerfplayer: A streamable dynamic scene representation with decomposed neural radiance fields," *IEEE Transactions on Visualization and Computer Graphics*, vol. 29, no. 5, pp. 2732–2742, 2023.

[28] Q.-Y. Zhou, J. Park, and V. Koltun, "Open3D: A modern library for 3D data processing," *arXiv:1801.09847*, 2018.

[29] M. Kazhdan, M. Bolitho, and M. Hoppe, "Poisson surface reconstruction," in *Proceedings of the fourth Eurographics symposium on Geometry processing*, vol. 7, no. 4, 2006.