Dynamic Scene Reconstruction

Quang Minh Dinh - 301449800

1. Introduction

Photorealistic image-based reconstruction is an important problem that has been pursued for a long time by the computer vision and computer graphics communities due to its wide range of applications in 3D content creation and entertainment. Recent neural rendering techniques, initiated by Neural Radiance Fields (NeRF) [15] and 3D Gaussian Splatting (3DGS) [10], introduced a breakthrough in generating high-quality multi-view renderings for static scenes. Along with the success of static 3D reconstruction techniques, a natural follow-up research direction was to also account for time and model dynamic scenes. Specifically, given either a single-view monocular video or multiple multi-view videos input, the expected output of dynamic 3D scene reconstruction is an implicit or explicit representation that can effectively capture the change of objects, people, lighting, surfaces, or any other environmental conditions over time. As most of the real-world environments are dynamic, simulating the changes of 3D scenes allows us to have many highly impactful applications in different domains, such as capturing moving animations in video games or movies, training robotic agents for urban planning, and simulating real-world scenes and events for virtual environments during AR or VR game development.

Compared to static scene reconstruction, dynamic scene reconstruction is a much more difficult problem, as different methods need to capture both the spatial and temporal dynamics within the scene. To handle the spatial and temporal continuities within dynamic scenes, many methods aimed to directly learn parametric 6D plenoptic functions using neural models [1, 7, 12, 24]. These methods did not explicitly model the motions and changes of the scene's structure over time and relied on the neural models to learn by themselves. On the other hand, other approaches explicitly modeled the scene's underlying motions by conditioning their 3D representations on a time variable, using either different types of temporal functions or a deformation field [14, 18, 22, 28]. Regardless of the direction, most methods were only able to model dynamic scenes for only a few seconds. They generally had trouble with generating high-quality renderings at real-time speed and had to trade off one to boost the other [13, 28, 29].

Besides the difficulty posed by the complexity of the dynamic 3D scene reconstruction task, the weaknesses of many methods came from the underlying static reconstruction method that they extended. The early NeRF-based architectures [2, 3, 15] were notorious for their low rendering speed, which many subsequent works tried to mitigate [6, 16, 25, 30, 31]. Most 3DGS-based models utilized a rasterization pipeline to speed up the running time, which made it non-trivial for them to render complex lighting effects [8]. Recently, Radiant Foam [8] was introduced as a method that could render high-quality scenes and complex lighting conditions at real-time speed using a ray tracing pipeline and a Voronoi-based representation.

This project aims to extend Radiant Foam with additional temporal variables to capture dynamic scenes. The contributions can be summarized as follows:

- A new set of temporal functions are introduced to model the changes of point densities, motions, and attributes over time.
- A temporal perturbation mechanism is proposed to avoid structure collapse and increase the model's robustness.

2. Related Works

2.1. Static Scene Reconstruction

In recent years, image-based novel view synthesis has observed a huge breakthrough, which started with the introduction of NeRF [15]. NeRF utilized a volumetric radiance field as the implicit 3D representation and leveraged a coordinate-based MLP to render high-quality scene images at different viewing angles with the use of volumetric rendering. Subsequent works attempted to alleviate the major weakness of NeRF, which is its low rendering speed, by improving the input encoding strategy [16], the sampling strategy [1, 17], utilizing grid structures [5, 25], or incorporating other localized neural representations [3, 19, 30]. Most of these methods employed a complex multi-stage training paradigm to maintain NeRF's rendering quality at higher speed [8]. Later, 3DGS [10] was introduced, which employed 3D Gaussians as the point-based representation, which carries the colour, opacity, and position information in the scene. 3DGS leveraged a differentiable rasterization pipeline, which allowed it to render photorealistic multi-view images at high speed. However, most 3DGS-based methods suffered from floater artifacts and flicker ef-

fects [9, 23, 28, 32] due to the difficulty of rendering complex lighting effects with rasterization [8]. Given that most dynamic reconstruction works extended existing static reconstruction methods with additional temporal factors, they also inherited the weaknesses of the underlying static reconstruction methods. Recently, Radiant Foam [8] was introduced as a ray tracing method that could capture high-quality static scenes at real-time speed. Radiant Foam divided the 3D space into a dense Voronoi tessellation, where each point belonged to exactly one Voronoi cell. During ray tracing, the rays accumulated the radiance of all the passing cells to get the final pixel color. Radiant Foam is a good candidate for dynamic reconstruction as it can render photorealistic images with complex lighting conditions at high speed.

2.2. Dynamic Scene Reconstruction

Dynamic scene reconstruction is a difficult task due to the intricacy of learning the underlying spatial and temporal correlations within the scene. Early works attempted to extend NeRF and learn a parametric 6D plenoptic function by adding a time-conditioned latent code [12], separating static and moving components [11, 24], leveraging keyframe-based representations for a sampling prediction network [1], and factorizing dynamic scenes into 2D feature planes [4, 7]. These methods often struggled with coupling the parameters [29]. Other methods extended NeRF or 3DGS and explicitly modeled the continuous motions or deformations in the scene by conditioning the underlying structure on a time variable [13, 29] or leveraging a deformation field [27, 28]. Compared to the earlier methods with a 6D plenoptic function, this line of work performed better in learning rigid motions, and they were flexible enough to account for topological shifts in the videos [28]. 4D Gaussian Splatting [29] directly incorporated time into the formulation of 3D Gaussians, enabling high-quality dynamic generation at real-time speed. However, they relied on prior assumptions of the data, such as the availability of depth information. Spacetime Gaussian [13] introduced different temporal radial basis functions to encode time in the point densities, motions, and attributes within the scene and was able to render high-quality dynamic scenes at a higher speed than 4D Gaussian Splatting. However, they were bounded by the limitations of the rasterization pipeline in capturing complex lighting conditions. In this project, I borrow the ideas behind Spacetime Gaussian with significant modifications to the temporal radial basis functions.

Compared to dynamic multi-view reconstruction, dynamic monocular reconstruction is a harder task where the model can only rely on a single-view monocular video, with one captured viewpoint at each time step. Most multi-view reconstruction methods failed on the monocular tasks, and many dynamic monocular reconstruction works had to rely on additional supervision from geometric, motion, or scene flow priors [13]. In this project, the proposed method will be evaluated on both the monocular and multi-view settings.

3. Approach

In this project, I leveraged the Radiant Foam codebase and implemented my own set of temporal radial basis functions to incorporate time into the architecture and my own dataloaders for the two datasets D-NeRF [18] and Neural 3D Video [12] to feed the necessary inputs and outputs to the training pipeline. I also made modifications to the entire rendering pipeline where necessary to adopt the changes introduced by the new architecture and data loaders. All the changed files will be listed in the README.md documentation of the GitHub repository that will be provided. In the following sections, I will first provide a high-level explanation of the rendering pipeline utilized by Radiant Foam, followed by the changes that I made to incorporate time. Two additional implementations that I added to the codebase to increase the training performace, which are the temporal perturbation and the Structure Similarity Index Measure (SSIM) loss [26], will be outlined in the following section. Finally, I will give reasonings about how my project involves rendering and image generation.

3.1. Volume Rendering Pipeline

Given a ray cast from a viewpoint to the scene, the volume rendering technique allows all points in the scene to contribute to the final color of the ray, often in the forms of density and radiance. Specifically, between the segment $(q_{\min}, q_{\max})$ of the viewing ray, the color $\mathbf{c_r}$ of ray $\mathbf{r}$ is:

$$\mathbf{c_r} = \int_{q_{\min}}^{q_{\max}} T(q) \cdot \sigma(\mathbf{r}(q)) \cdot \mathbf{c}(\mathbf{r}(q)) dt \tag{1}$$

$$T(q) = \exp\left(-\int_{q_{\min}}^{q} \sigma(\mathbf{r}(u)) du\right) \tag{2}$$

where $\mathbf{r}(q)$ indicates the point at distance q along ray $\mathbf{r}$, $\sigma(\mathbf{r}(q))$ is the point density, and $\mathbf{c}(\mathbf{r}(q))$ is the point radiance.

Radiant Foam expressed the density field σ and radiance field $\mathbf{c}$ as constant values within each foam cell and simplified the volume rendering equation as the sum of M ray segments:

$$\mathbf{c_r} = \sum_{m=1}^{M} T_m \cdot (1 - \exp(-\sigma_m \delta_m)) \cdot \mathbf{c}_m dt \tag{3}$$

$$T_m = \prod_{j=1}^{m} \exp(-\sigma_j \delta_j) \tag{4}$$

where δ_m denoted the width of segment m.

3.2. Temporal Modeling

Radiant Foam leveraged spherical harmonics to compute the point radiances and utilized learnable parameters to represent the point positions, densities, and spherical harmonic coefficients for training. To model dynamic scenes, the goal of this project was to represent the three attributes with temporal functions.

Temporal Density. Inspired by Spacetime Gaussian [13], I used a temporal radial basis function to represent the density $\sigma_i(t)$ of point i at time t. My temporal density function was different from Spacetime Splatting, which had empirically shown to be better for training dynamic Radiant Foam:

$$\sigma_i(t) = \sigma_i^s \exp\left(-\frac{|t/T - \mu_i^\tau|^2}{\exp(2s_i^\tau)}\right) \tag{5}$$

where T was the duration of the scene, μ_i^τ denoted the temporal center, s_i^τ was a learnable temperature, and σ_i^s was the time-independent spatial density, which I used the original learnable density parameters in the Radiant Foam codebase to represent. μ_i^τ represented the timestamp that point i was most visible, and s_i^τ was the duration where the temporal density was high.

Temporal Position. Similar to Spacetime Gaussian, I employed a radial basis polynomial function to model the position $\mu_i(t)$ of point i at time t:

$$\mu_i(t) = \mu_i^s + \sum_{k=1}^{n_p} b_{i,k} \left(t/T - \mu_i^\tau\right)^k \tag{6}$$

where $\{b_{i,k}\}_{k=1}^{n_p}$ were the learnable polynomial coefficients, with $n_p = 5$ as the polynomial degree. I used $n_p = 5$ instead of $n_p = 3$ like Spacetime Gaussian as I found empirically that the training performance was not good for $n_p = 3$. μ_i^s denoted the time-independent canonical position, which was represented using the original learnable position parameters in the Radiant Foam codebase.

Temporal Spherical Harmonics. Inspired by Yang et al. [29], I generated the temporal spherical harmonics by merging the spatial spherical harmonics with a 1D Fourier sequence. Since the exact calculation of the spherical harmonics was implemented using CUDA programming in the Radiant Foam's codebase, I implemented the temporal spherical harmonic $sh_i(t)$ of point i at time t by modifying the spatial spherical harmonic coefficients:

$$sh_i^j(t) = \cos\left(\frac{2\pi j}{T}t\right) sh_i^{s,j} \tag{7}$$

where $sh_i^j(t)$ was the j^{th} coefficient of the temporal spherical harmonic $sh_i(t)$, and $sh_i^{s,j}$ was the j^{th} coefficient of the spatial spherical harmonic sh_i^s of point i. j was also the order of the Fourier series. Let $a_i^{s,j}$ be the j^{th} term of the spatial spherical harmonic sh_i^s:

(a) Rendered image at iteration 1500.

(b) Rendered image at iteration 2000.

Figure 1. Structure collapse during training. Smooth structures learned at earlier iterations broke down in later iterations.[1]

$$sh_i^s = \sum_{j=1}^{l} a_i^{s,j} sh_i^{s,j} \tag{8}$$

where l is the number of spherical harmonic terms. Eq. (7) directly leads to the formulation of the temporal spherical harmonic $sh_i(t)$:

$$\begin{aligned} sh_i(t) &= \sum_{j=1}^{l} a_i^{s,j} sh_i^j(t) \\ &= \sum_{j=1}^{l} \cos\left(\frac{2\pi j}{T}t\right) a_i^{s,j} sh_i^{s,j} \end{aligned} \tag{9}$$

[1] All the illustrations shown in this report were the generated images for the first images of the test sets, which were the representatives of the entire test sets.

Figure 2. Color gradient dependency problem. *(left)* the generated image during training. *(middle)* the ground-truth image. *(right)* the error map between the generated image and the ground-truth image, which was calculated using an L1 distance metric. The model focuses on reconstructing regions with high color intensities and ignore other regions.

3.3. Temporal Perturbation

When the model was trained with discrete frame times, there was often a stability problem where the scene structure that the model learned in the earlier iterations suddenly collapsed in later iterations, as can be seen in Fig. 1. Color jitters appeared in many regions in the rendered image, and local structures were no longer maintained. It often took the model a few hundred iterations just to recover the original structure, which slowed down the training significantly.

This structure collapse phenomenon might have happened because the model was not able to learn the temporal continuity with the discrete frame times data, hence, I added randomness to the training frame time to make it continuous. Specifically, for each training frame, the frame time t' was sampled from a Gaussian distribution, where the mean is the true frame time t and the standard deviation is the average frame duration Δt of the scene divided by 4, which was found empirically:

$$t' \sim \mathcal{N}(t, \Delta t/4) \tag{10}$$

The training frame time t' was clipped by $\Delta t/2$, so that it would not be too far off from the true time t:

$$\begin{cases} t' = t + \Delta t/2 & \text{if} \quad t' > t + \Delta t/2 \\ t' = t - \Delta t/2 & \text{if} \quad t' < t - \Delta t/2 \end{cases} \tag{11}$$

3.4. Color Gradient Dependency

Another problem that the model encountered during training was the color gradient dependency problem, where it relied too heavily on color gradients for learning temporal correspondences. The result was that the model focused on reconstructing the regions with high color intensities in the scene and ignored the other regions with lower color intensities, as can be seen in Fig. 2.

To mitigate this issue, I employed the Structure Similarity Index Measure (SSIM) loss [26] in addition to Radiant Foam's losses. SSIM is a perceptual loss that takes into account luminance, contrast and most importantly, the local structure within the images:

$$\mathcal{L}_{\text{SSIM}}(x, y) = 1 - \frac{(2\mu_x\mu_y + c_1)\,(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)\,(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{12}$$

where μ and σ are the pixel sample mean and variance respectively. The SSIM loss ensures the structure integrity when training and makes the model less prone to learning only the high color intensity regions.

The final training objective of the model was as follows:

$$\mathcal{L}(\text{pred}, \text{gt}) = \mathcal{L}_{RF}(\text{pred}, \text{gt}) + \lambda \mathcal{L}_{\text{SSIM}}(\text{pred}, \text{gt}) \tag{13}$$

where $\mathcal{L}_{RF}$ is the original training objective of Radiant Foam and λ is the weight of the SSIM loss, which I set to 0.25.

3.5. Relations to the Course

This project is heavily related to rendering and image generation. The volumetric rendering technique is utilized in the project to render the final video frame results from the temporal point positions, densities, and spherical harmonic coefficients.

4. Results & Analysis

4.1. Datasets

The method introduced in this project was evaluated on both the monocular and multi-view scenarios. For the multi-view setup, I employed the Neural 3D Video dataset [12], which contains six real-world scenes, each lasting around 10 seconds. The evaluation was conducted on the coffee martini scenario. Due to the resource constraint, I extracted each view video at 30 FPS and only used the first 30 frames for training and evaluation. The frames from camera 0 were set

aside for testing, and all the frames from the other cameras were utilized for training. I used COLMAP [20, 21] to generate the initial point positions and downsampled the frame images to four times for both training and evaluation.

For the monocular evaluation, the evaluation was conducted on the bouncing balls scenario of the D-NeRF [18] dataset. D-NeRF contains eight monocular videos of synthetic scenes with 50-200 frames, and only one single image from a viewpoint is accessible at each time step. D-NeRF was used by many of the dynamic reconstruction works to evaluate their methods [18, 28, 29], and it is a well-known dataset for monocular scene reconstruction. A subset of the real data from the Neural 3D Video dataset was also utilized to evaluate the model's performance on monocular view reconstruction. Specifically, I used the first 150 frames from camera 0 for the training and 150 frames from camera 7 for the evaluation. The example testing frames and generation results are illustrated in Sec. 4.4.

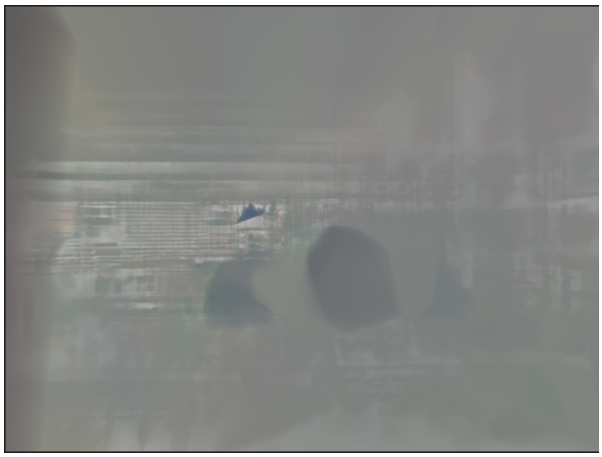
4.2. Training Details

Similar to Radiant Foam, I employed an Adam optimizer and directly optimized the learnable parameters using spherical harmonics of degree three. For the spatial point positions, I used an initial learning rate of $2e^{-4}$ and decayed it using a cosine annealing scheduler to a final learning rate of $5e^{-6}$. The initial learning rate for the temporal point position polynomial coefficients was $4e^{-4}$, and it was decayed to $9e^{-6}$ using a similar scheduler. The initial learning rates for the point densities and spherical harmonic coefficients were $1e^{-1}$ and $5e^{-3}$, and their final learning rates were $1e^{-2}$ and $5e^{-4}$ respectively. I used $5e^{-4}$ and $3e^{-2}$ as the initial learning rate for the temporal center and temperature, and $5e^{-5}$ and $3e^{-3}$ as their final learning rate respectively. The model was trained for 50000 iterations, where the first 2000 iterations were for warm-up training. After the first 2000 iterations, the total number of Voronoi cells gradually increased until iteration 31000.

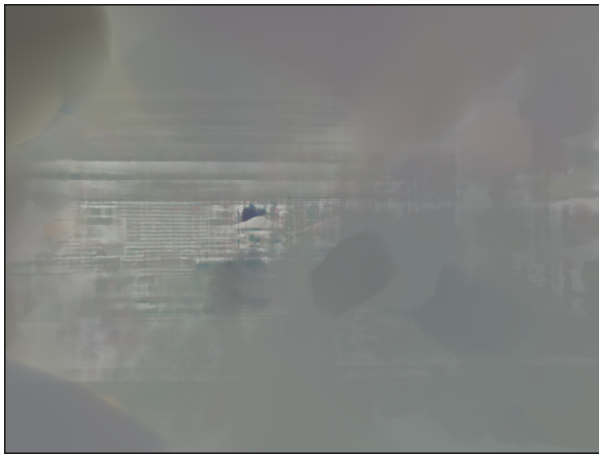
A core operation of Radiant Foam is the Delaunay triangulation to construct the Voronoi cells. The Delaunay triangulation was implemented using an Axis-Aligned Bounding Box (AABB) tree, which was rebuilt incrementally with the assumption of fixed point ordering after a certain number of iterations in the Radiant Foam's codebase. The original Radiant Foam codebase did a full rebuild of the AABB tree after each densification and near the end of the training. In this project, as the neighbors of each point differed depending on the time frame, I did an incremental rebuild of the AABB tree after each iteration and a complete rebuild of the AABB tree after every 1000 iterations.

4.3. Points of Comparisons

As most of the training results were not complete due to the resource constraint and the original Radiant Foam model could not be directly applied to dynamic scene reconstruction, I did not have a baseline or comparisons to external methods with completed training results. The trained models were mostly ranked based on how similar the generated renderings were to the ground-truth images.



(a) Rendered image at iteration 1500.



(b) Rendered image at iteration 2000.

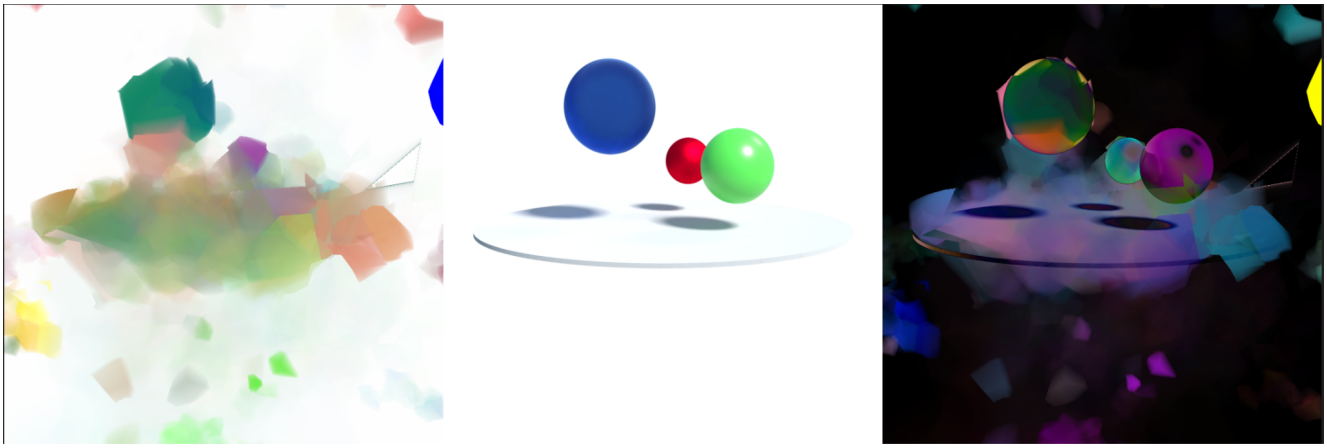
Figure 3. Effects of Temporal Perturbations. Smooth structures learned at earlier iterations were maintained in later iterations.

4.4. Analysis

Effects of Temporal Perturbations. Adding perturbations to the frame time had a strong effect on stabilizing the training, and structure collapse no longer happened frequently. This effect was maintained throughout the experiments with temporal perturbations, which showed that the model became more robust in capturing continuous frame times.

Effects of SSIM Loss. The effects of the SSIM loss were high. As can be seen in Fig. 4, the model changed its focus to reconstructing the entire scene uniformly instead of just the regions with high color intensity. The model also converged faster. The rendering shown in Fig. 4 was generated

Figure 4. Effects of SSIM Loss. *(left)* the generated test image. *(middle)* the ground-truth test image. *(right)* the error map between the generated image and the ground-truth image, which was calculated using an L1 distance metric. The model reconstructed the entire scene uniformly.



(a) Qualitative results on the D-NeRF's test set.

(b) Qualitative results on the Neural 3D Video's test set.

Figure 5. The model's performances on the monocular scene reconstruction task. *(left)* the generated test image. *(middle)* the ground-truth test image. *(right)* the error map between the generated image and the ground-truth image, which was calculated using an L1 distance metric.

at iteration 500 with the SSIM loss, but normally it would take around 12000-16000 iterations to have a rendering with similar qualities without the SSIM loss.

Qualitative Results on Monocular Scene Reconstruction. My experiments showed that the model did not perform well on the monocular scene reconstruction task, as can be seen in Fig. 5. On the synthetic data of the D-NeRF dataset, the model was not able to converge and rendered images with incorrect colors. Although many learnt point positions seemed to be somewhat correct, the other point

Figure 6. Qualitative results for the multi-view setup on the Neural 3D Video's test set. *(left)* the generated test image. *(middle)* the ground-truth test image. *(right)* the error map between the generated image and the ground-truth image, which was calculated using an L1 distance metric.

positions were far off from the ground-truth image. On the subsamples of the Neural 3D Video datasets that I used for the monocular pipeline, although the qualities of the rendering in many parts were good, the model seems to have been overfitted to the training data as it was not able to provide renderings from the correct viewing angles. This result might have happened because a single-view video was not able to cover a wide enough range of viewing angles for the model to generalize.

Qualitative Results on Multi-view Scene Reconstruction. The multi-view reconstruction required significantly higher computing resources and training time because of the large amount of training data and that the AABB tree had to be repeatedly rebuilt during training. Although I did not have enough resources to complete the training, the intermediate rendering showed that the model was able to generalize to the correct viewing angle, as can be seen in Fig. 6. Given more training time, it should be able to reconstruct the scene fully.

4.5. Ablation Study

Beside the modeling approach using the temporal radial basis functions mentioned above, I also tried another method to add time to Radiant Foam for the monocular scene reconstruction task, which is to add a simple fully connected deformation network that took the canonical point position and density, as well as time t and returned the motion and density offset. This approach did not go well because even for a simple scene, the VRAM usage blew up by a lot, and I soon ran out of memory. The deformation network was not a main approach proposed in the project, it was a description of what I attempted to do in the project and did not work.

5. Limitation

Although this project proposed a method for incorporating time into Radiant Foam that seemed to work on multi-view scene reconstruction, the training was not complete, so follow-up works will benefit from completing the training and get different metric numbers to have a fairest comparison between dynamic Radiant Foam and other existing methods. To do that, a major weakness of this method that needs to be resolved is the training time. A few good directions for future works to mitigate this weakness are implementing a dynamic AABB tree to perform triangulations and finding a better way to represent the adjacency list without having to redo the triangulation at every step. Future works can also benefit from having a more suitable pruning and densification strategy for this method, which are potential in improving the model's performance. Since the training results showed that the approach with temporal radial basis functions did not work well with monocular video reconstruction, it is also a good direction to look for other approaches to incorporate time into Radiant Foam.

6. Conclusion

This project introduced a method to add time variables to Radiant Foam using temporal radial basis functions. A temporal perturbation method and an inclusion of the SSIM loss to the final training objective were also introduced to increase the training robustness and performance. Although the method did not perform well on monocular scene reconstruction, the preliminary training results on the multi-view scene reconstruction task showed positive signals about the model's convergence, which might benefit from a full training. A few key takeaways from this work are firstly, the temporal radial basis functions approach does not work well with monocular videos, especially when the viewing angles are limited. Secondly, dynamic reconstruction with Radiant Foam is very sensitive to pruning, so it might help the future works to evaluate their methods after doing just densification to confirm the effectiveness before starting to experiment with any type of pruning. Finally, for this type of reconstruction with Radiant Foam, the point density is

also very sensitive to change, so a lot of trials and errors are necessary to design a good temporal function.

7. Timeline

- **Week 1 (17/2 - 23/2):** Read Radiant Foam's codebase, replicated results. Implemented dataloaders for D-NeRF & Neural 3D Video.
- **Week 2 & 3 (24/2 - 9/3):** Pre-computed COLMAP sparse reconstructions. Added temporal modelling and ran experiments on monocular videos.
- **Week 4 (10/3 - 16/3):** Modified the training pipeline for multiviews videos.
- **Week 5 (17/3 - 23/3):** Fine-tuned multiviews pipeline, experimented with different temporal functions, losses...
- **Week 6 (24/3 - 30/3):** Implemented deformation network, experimented on the monocular pipeline.
- **Week 7 (31/3 - 6/3):** Conducted final evaluations, wrote report.

8. Code

https://github.com/quangminhdinh/CMPT469-Final

The documentation is provided in the README.md file.

References

[1] Benjamin Attal, Jia-Bin Huang, Christian Richardt, Michael Zollhoefer, Johannes Kopf, Matthew O'Toole, and Changil Kim. Hyperreel: High-fidelity 6-dof video with ray-conditioned sampling. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16610–16620, 2023. 1, 2

[2] Jonathan T. Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P. Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5835–5844, 2021. 1

[3] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Zip-nerf: Anti-aliased grid-based neural radiance fields. *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 19640–19648, 2023. 1

[4] Ang Cao and Justin Johnson. Hexplane: A fast representation for dynamic scenes. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 130–141, 2023. 2

[5] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. *ArXiv*, abs/2203.09517, 2022. 1

[6] Anpei Chen, Zexiang Xu, Xinyue Wei, Siyu Tang, Hao Su, and Andreas Geiger. Factor fields: A unified framework for neural fields and beyond. *ArXiv*, abs/2302.01226, 2023. 1

[7] Sara Fridovich-Keil, Giacomo Meanti, Frederik Warburg, Benjamin Recht, and Angjoo Kanazawa. K-planes: Explicit radiance fields in space, time, and appearance. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12479–12488, 2023. 1, 2

[8] Shrisudhan Govindarajan, Daniel Rebain, Kwang Moo Yi, and Andrea Tagliasacchi. Radiant foam: Real-time differentiable ray tracing, 2025. 1, 2

[9] Miriam Jäger, Markus Hillemann, and Boris Jutzi. Featuregs: Eigenvalue-feature optimization in 3d gaussian splatting for geometrically accurate and artifact-reduced reconstruction. *ArXiv*, abs/2501.17655, 2025. 2

[10] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkuehler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics (TOG)*, 42:1 – 14, 2023. 1

[11] Lingzhi Li, Zhen Shen, Zhongshu Wang, Li Shen, and Ping Tan. Streaming radiance fields for 3d video synthesis. *ArXiv*, abs/2210.14831, 2022. 2

[12] Tianye Li, Mira Slavcheva, Michael Zollhoefer, Simon Green, Christoph Lassner, Changil Kim, Tanner Schmidt, Steven Lovegrove, Michael Goesele, Richard Newcombe, and Zhaoyang Lv. Neural 3d video synthesis from multi-view video, 2022. 1, 2, 4

[13] Zhan Li, Zhang Chen, Zhong Li, and Yinghao Xu. Spacetime gaussian feature splatting for real-time dynamic view synthesis. *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8508–8520, 2023. 1, 2, 3

[14] Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. *2024 International Conference on 3D Vision (3DV)*, pages 800–809, 2023. 1

[15] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf. *Communications of the ACM*, 65:99 – 106, 2020. 1

[16] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics*, 41 (4):1–15, 2022. 1

[17] Thomas Neff, Pascal Stadlbauer, Mathias Parger, Andreas Kurz, Joerg H. Mueller, Chakravarty Reddy Alla Chaitanya, Anton Kaplanyan, and Markus Steinberger. Donerf: Towards real-time rendering of compact neural radiance fields using depth oracle networks. *Computer Graphics Forum*, 40, 2021. 1

[18] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10313–10322, 2020. 1, 2, 5

[19] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 14315–14325, 2021. 1

[20] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 5

[21] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)*, 2016. 5

[22] Liangchen Song, Anpei Chen, Zhong Li, Z. Chen, Lele Chen, Junsong Yuan, Yi Xu, and Andreas Geiger. Nerfplayer: A streamable dynamic scene representation with decomposed neural radiance fields. *IEEE Transactions on Visualization and Computer Graphics*, 29:2732–2742, 2022. 1

[23] Hao Sun, Junping Qin, Lei Wang, Kai Yan, Zheng Liu, Xinglong Jia, and Xiaole Shi. 3dgs-hd: Elimination of unrealistic artifacts in 3d gaussian splatting. In *2024 6th International Conference on Data-driven Optimization of Complex Systems (DOCS)*, pages 696–702, 2024. 2

[24] Feng Wang, Sinan Tan, Xinghang Li, Zeyue Tian, and Huaping Liu. Mixed neural voxels for fast multi-view video synthesis. *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 19649–19659, 2022. 1, 2

[25] Peng Wang, Yuan Liu, Zhaoxi Chen, Lingjie Liu, Ziwei Liu, Taku Komura, Christian Theobalt, and Wenping Wang. F2-nerf: Fast neural radiance field training with free camera trajectories. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4150–4159, 2023. 1

[26] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4): 600–612, 2004. 2, 4

[27] Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 4d gaussian splatting for real-time dynamic scene rendering. *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 20310–20320, 2023. 2

[28] Ziyi Yang, Xinyu Gao, Wenming Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. Deformable 3d gaussians for high-fidelity monocular dynamic scene reconstruction. *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 20331–20341, 2023. 1, 2, 5

[29] Zeyu Yang, Zijie Pan, Xiatian Zhu, Li Zhang, Yu-Gang Jiang, and Philip H. S. Torr. 4d gaussian splatting: Modeling dynamic scenes with native 4d primitives. *ArXiv*, abs/2412.20720, 2024. 1, 2, 3, 5

[30] Alex Yu, Sara Fridovich-Keil, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5491–5500, 2021. 1

[31] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. Plenoctrees for real-time rendering of neural radiance fields. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5732–5741, 2021. 1

[32] Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. Mip-splatting: Alias-free 3d gaussian splatting. *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 19447–19456, 2023. 2