Per-Gaussian Embedding-Based Deformation for Deformable 3D Gaussian Splatting

Jeongmin Bae[1]⋆, Seoha Kim[1]⋆, Youngsik Yun[1],
Hahyun Lee[2], Gun Bang[2], and Youngjung Uh[1]

[1] Yonsei University, Seoul 03722, Korea
{jaymin.bae, hailey07, bbangsik, yj.uh}@yonsei.ac.kr
[2] Electronics and Telecommunications Research Institute, Daejeon 34129, Korea
{hanilee, gbang}@etri.re.kr

Fig. 1: **Overview.** (a) Existing deformable 3D Gaussian Splatting methods show blurry results in complex dynamic scenes, even with deformation fields using finer feature grids. (b) Our model solves the problem by employing per-Gaussian latent embeddings to predict deformations for each Gaussian and achieves a clearer results.

Abstract. As 3D Gaussian Splatting (3DGS) provides fast and high-quality novel view synthesis, it is a natural extension to deform a canonical 3DGS to multiple frames for representing a dynamic scene. However, previous works fail to accurately reconstruct complex dynamic scenes. We attribute the failure to the design of the deformation field, which is built as a coordinate-based function. This approach is problematic because 3DGS is a mixture of multiple fields centered at the Gaussians, not just a single coordinate-based framework. To resolve this problem, we define the deformation as a function of per-Gaussian embeddings and temporal embeddings. Moreover, we decompose deformations as coarse and fine deformations to model slow and fast movements, respectively. Also, we introduce a local smoothness regularization for per-Gaussian embedding to improve the details in dynamic regions.
  Project page: https://jeongminb.github.io/e-d3dgs/

Keywords: Deformable gaussian splatting · 4D scene reconstruction · Novel view synthesis

⋆ Authors contributed equally to this work.

1   Introduction

Dynamic scene reconstruction from multi-view input videos is an important task in computer vision, as it can be extended to various applications and industries such as mixed reality, content production, *etc.* Neural Radiance Fields (NeRF) [19], which enable photorealistic novel view synthesis from multi-view inputs, can represent dynamic scenes by modeling the scene with an additional time input [12, 22]. However, typical NeRFs require querying multilayer perceptron (MLP) for hundreds of points per camera ray, which limits rendering speed.

On the other hand, the recently emerging 3D Gaussian Splatting (3DGS) [10] has the advantage of real-time rendering compared to NeRFs using a differentiable rasterizer for 3D Gaussian primitives. 3DGS directly optimizes the parameters of 3D Gaussians (position, opacity, anisotropic covariance, and spherical harmonics coefficients) and renders them via projection and α-blending. Since 3DGS has the characteristics of continuous volumetric radiance fields, some recent studies [3, 8, 14, 29, 31, 32] represent dynamic scenes by defining a canonical 3DGS and deforming it to individual frames as deformable NeRFs [31] do. Specifically, they model the deformation as a function of 4D (x, y, z, t) coordinates with MLPs or grids to predict the change in the 3D Gaussian parameters.

However, since 3DGS is a mixutre of multiple volumetric fields, it is not appropriate to model the deformation of Gaussian parameters with a single coordinate-based network to represent dynamic scenes. In addition, existing field-based approaches are constrained by the resolution of the grid which models the deformation field, the capacity of the model, or the frequencies of the input. As shown in Figure 1, existing study does not properly represent complex dynamic scenes, and even introducing an additional feature grid that is twice the maximum resolution has only a slight improvement in performance (See Appendix for more results). We alleviate this problem by introducing a novel dynamic representation to deform each Gaussian.

In this paper, we model the deformation of Gaussians at frames as 1) a function of a product space of per-Gaussian embeddings and temporal embeddings. We expect this rational design to bring quality improvement by precisely modeling different deformations of different Gaussians. Additionally, 2) We decompose temporal variations of the parameters into coarse and fine components, namely coarse-fine deformation. The coarse deformation represents large or slow movements in the scene, while fine deformation learns the fast or detailed movements that coarse deformation does not cover. Finally, we propose 3) a local smoothness regularization for per-Gaussian embedding to ensure the deformations of neighboring Gaussians are similar.

In our experiments, we observe that our per-Gaussian embeddings, coarse-fine deformation, and regularization improve the deformation quality. Our approach outperforms baselines in capturing fine details in dynamic regions and excels even under challenging camera settings. Additionally, our method also achieves fast rendering speed and relatively low capacity.

2   Related Work

In this section, we review methods for dynamic scene reconstruction that deform 3D canonical space and methods for reconstructing dynamic scenes utilizing dynamic 3D Gaussians. Afterward, we review methods that use embeddings and spatial relationships of Gaussians.

Deforming 3D Canonical Space D-NeRF [22] reconstructs dynamic scenes by deforming ray samples over time, using the deformation network that takes 3D coordinates and timestamps of the sample as inputs. Nerfies [20] and HyperNeRF [21] use per-frame trainable deformation codes instead of time conditions to deform the canonical space. Instead of deforming from the canonical frame to the entire frames, HyperReel [1] deforms the ray sample of the keyframe to represent the intermediate frame. 4DGaussians [29] and D3DGS [31] reconstruct the dynamic scene with a deformation network which inputs the center position of the canonical 3D Gaussians and timestamps. MoDGS [15] learns the mapping between canonical space and space at specific timestamp through invertible MLP. In contrast, we demonstrate a novel deformation representation as a function of a product space of per-Gaussian latent embeddings and temporal embeddings.

Dynamic 3D Gaussians To extend the fast rendering speed of 3D Gaussian Splatting [10] into dynamic scene reconstructions. 4DGaussians [29] decodes features from multi-resolution HexPlanes [2] for temporal deformation of 3D Gaussians. While D3DGS [31] uses an implicit function that processes the time and location of the Gaussian. 4DGS [30] decomposes the 4D Gaussians into a time-conditioned 3D Gaussians and a marginal 1D Gaussians. STG [13] represents changes in 3D Gaussian over time through a temporal opacity and a polynomial function for each Gaussian.

Our method uses deformable 3D Gaussians as 4DGaussians [29] and D3DGS [31] do, but does not necessitate the separated feature field to obtain the input feature of the deformation decoder. Our approach uses embeddings allocated to each Gaussian and a temporal embedding shared within a specific frame.

Latent Embedding on Novel View Synthesis Some studies incorporate latent embeddings to represent different states of the static and dynamic scene. NeRF-W [18] and Block-NeRF [25] employ per-image embeddings to capture different appearances of a scene, representing the scenes from in-the-wild image collections. DyNeRF and MixVoxels [12, 27] employ a temporal embedding for each frame to represent dynamic scenes. Nerfies [20] and HyperNeRF [21] incorporate both per-frame appearance and deformation embeddings. Sync-NeRF [11] introduces time offset to calibrate the misaligned temporal embeddings on dynamic scenes from unsynchronized videos. We introduce per-Gaussian latent embedding to encode the changes over time of each Gaussian and use temporal embeddings to represent different states in each frame of the scene.

Considering Spatial Relationships of Gaussians Scaffold-GS [16] recon-
struct 3D scenes by synthesizing Gaussians from anchors, utilizing that the neigh-
boring Gaussians have similar properties. SAGS [26] creates a graph based on
k-nearest neighbors (KNN) so that each Gaussian is optimized while consider-
ing its neighboring Gaussians. In dynamic scene reconstruction, SC-GS [9] and
GaussianPrediction [34] deform Gaussians by combining the deformations of key
point Gaussians. Dynamic 3D Gaussians [17] utilizes regularization to encourage
that Gaussians and their neighboring Gaussians deform with local rigidity. Sim-
ilarly, we propose a local smoothness regularization that encourages neighboring
Gaussians to have similar embeddings, resulting in similar deformations.

3   Method

In this section, we first provide a brief overview of 3D Gaussian Splatting (Section
3.1). Next, we introduce our overall framework, embedding-based deformation
for Gaussians (Section 3.2) and coarse-fine deformation scheme consisting of
coarse and fine deformation functions (Section 3.3). Finally, we present a local
smoothness regularization for per-Gaussian embeddings to achieve better details
on dynamic regions (Section 3.4).

3.1   Preliminary: 3D Gaussian Splatting

3D Gaussian splatting [10] optimizes a set of anisotropic 3D Gaussians through
differentiable tile rasterization to reconstruct a static 3D scene. By its efficient
rasterization, the optimized model enables real-time rendering of high-quality
images. Each 3D Gaussian kernel $G_i(x)$ at the point x consist with position $\mathbf{x}_i$,
rotation R_i, and scale S_i:

$$G_i(x) = e^{-\frac{1}{2}(x-\mathbf{x}_i)^T \Sigma_i^{-1}(x-\mathbf{x}_i)}, \qquad \text{where} \quad \Sigma_i = R_i S_i S_i^T R_i^T. \qquad (1)$$

To projecting 3D Gaussians to 2D for rendering, covariance matrix Σ' are cal-
culated by viewing transform W and the Jacobian J of the affine approximation
of the projective transfomation [35] as follows:

$$\Sigma' = JW\Sigma W^T J^T. \qquad (2)$$

Blending $\mathcal{N}$ depth-ordered projected points that overlap the pixel, the Gaussian
kernel $G_i(x)$ is multiplied by the opacity of the Gaussian σ_i and calculates the
pixel color C with the color of the Gaussian c_i:

$$C = \sum_{i \in \mathcal{N}} c_i \alpha_i \prod_{j=1}^{i-1}(1 - \alpha_j), \qquad \text{where} \quad \alpha_i = \sigma_i G_i(x). \qquad (3)$$

The color of Gaussian c_i is determined using the SH coefficient with view direc-
tion.
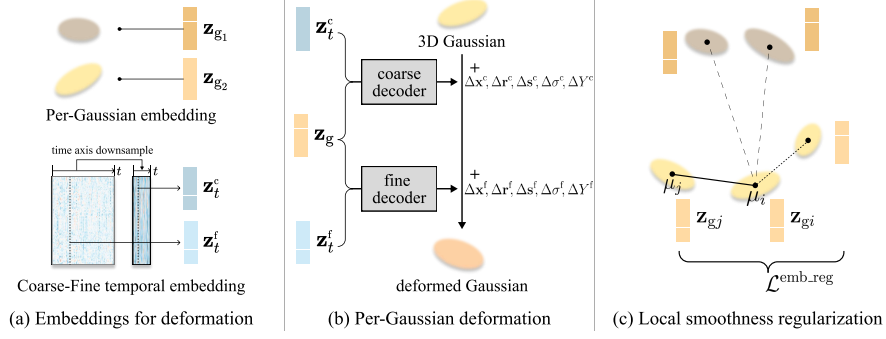
Fig. 2: **Framework.** Existing coordinate-based network methods struggle to represent complex dynamic scenes. To this end, we define per-Gaussian deformation. (a) Firstly, we assign a latent embedding for each Gaussian. Additionally, we introduce coarse and fine temporal embeddings to represent the slow and fast state of the dynamic scene. (b) By employing two decoders that take per-Gaussian latent embeddings along with coarse and fine temporal embeddings as input, we estimate slow or large changes and fast or detailed changes to model the final deformation, respectively. (c) Finally, we introduce a local smoothness regularization so that the embeddings of neighboring Gaussians are similar.

3.2 Embedding-Based Deformation for Gaussians

Deformable NeRFs consist of a deformation field that predicts displacement $\Delta\mathbf{x}$ for a given coordinate $\mathbf{x}$ from the canonical space to each target frame, and a radiance field that maps color and density from a given coordinate in the canonical space $(\mathbf{x} + \Delta\mathbf{x})$. Existing deformable Gaussian methods employ the same approach for predicting the deformation of Gaussians, i.e., utilizing a deformation field based on coordinates.

Unlike previous methods, we start from the design of 3DGS: the 3D scene is represented as a mixture of Gaussians that have individual radiance fields. Accordingly, the deformation should be defined for each Gaussian. Based on this intuition, we introduce a function $\mathcal{F}_\theta$ that produces deformation from learnable embeddings $\mathbf{z}_\mathrm{g} \in \mathbb{R}^{32}$ belonging to individual Gaussians (Figure 2a), and typical temporal embeddings $\mathbf{z}_t \in \mathbb{R}^{256}$ for different frames:

$$\mathcal{F}_\theta : (\mathbf{z}_\mathrm{g}, \mathbf{z}_t) \to (\Delta\mathbf{x}, \Delta\mathbf{r}, \Delta\mathbf{s}, \Delta\sigma, \Delta Y), \tag{4}$$

where $\mathbf{r}$ is a rotation quaternion, $\mathbf{s}$ is a vector for scaling, σ is an opacity, and Y is SH coefficients for modeling view-dependent color. We implement $\mathcal{F}_\theta$ as a shallow multi-layer perceptron (MLP) followed by an MLP head for each parameter. As a result, the Gaussian parameters at frame t are determined by adding $\mathcal{F}_\theta(\mathbf{z}_\mathrm{g}, \mathbf{z}_t)$ to the canonical Gaussian parameters (Figure 2c).

We jointly optimize the per-Gaussian embeddings $\mathbf{z}_\mathrm{g}$, the deformation function $\mathcal{F}_\theta$, and the canonical Gaussian parameters to minimize the rendering loss. We use the L1 and periodic DSSIM as the rendering loss between the rendered image and the ground truth image.

3.3 Coarse-Fine Deformation

Different parts of a scene may have coarse and fine motions [5]. E.g., a hand swiftly stirs a pan (fine) while a body slowly moves from left to right (coarse). Based on this intuition, we introduce a coarse-fine deformation that produces a summation of coarse and fine deformations.

Coarse-fine deformation consists of two functions with the same architecture: one for coarse and one for fine deformation (Figure 2c). The functions receive different temporal embeddings as follows:

Following typical temporal embeddings, we start from a 1D feature grid $Z \in \mathbb{R}^{N \times 256}$ for N frames and use an embedding $\mathbf{z}_t^{\mathrm{f}} = \mathrm{interp}(Z, t)$ for fine deformation. For coarse deformation, we linearly downsample Z by a factor of 5 to remove high-frequencies responsible for fast and detailed deformation. Then we compute $\mathbf{z}_t^{\mathrm{c}}$ as a linear interpolation of embeddings at enclosing grid points (Figure 2b).

As a result, coarse deformation $\mathcal{F}_{\theta_{\mathrm{c}}}(\mathbf{z}_{\mathrm{g}}, \mathbf{z}_t^{\mathrm{c}})$ is responsible for representing large or slow movements in the scene, while fine deformation $\mathcal{F}_{\theta_{\mathrm{f}}}(\mathbf{z}_{\mathrm{g}}, \mathbf{z}_t^{\mathrm{f}})$ learns the fast or detailed movements that coarse deformation does not cover. This improves the deformation quality. Refer to the Ablation study section for more details.

3.4 Local Smoothness Regularization

Neighboring Gaussians constructing dynamic objects tend to exhibit locally similar deformation. Inspired by [17], we introduce a local smoothness regularization for per-Gaussian embedding $\mathbf{z}_{\mathrm{g}}$ (Figure 2d) to encourage similar deformations between nearby Gaussians i and j:

$$\mathcal{L}^{\mathrm{emb\_reg}} = \frac{1}{k|\mathcal{S}|} \sum_{i \in \mathcal{S}} \sum_{j \in \mathrm{KNN}_{i;k}} (w_{i,j} \|\mathbf{z}_{\mathrm{g}i} - \mathbf{z}_{\mathrm{g}j}\|_2),$$

where $w_{i,j} = \exp(-\lambda_w \|\mu_j - \mu_i\|_2^2)$ is the weighting factor and μ is the Gaussian center. We set λ_w to 2000 and k to 20 following [17]. To reduce the computational cost, we obtain sets of k-nearest-neighbors only when the densification occurs.

Note that unlike previous approaches that directly constrain physical properties such as rigirity or rotation, We implicitly induce locally similar deformation by ensuring that per-Gaussian embeddings are locally smooth. Our regularization allows better capture of textures and details of dynamic objects.

4 Experiment

In this section, we first describe the criterion for selection of baselines, and evaluation metrics. We then demonstrate the effectiveness of our method through comparisons with various baselines and datasets (Section 4.1-4.2). Finally, we conduct analysis and ablations of our method (Section 4.3).

Table 1: Average performance in the test view on Neural 3D Video Dataset
The computational cost was measured based on `flame_salmon_1` on the A6000. [1]flame salmon scene includes only the first segment, comprising 300 frames. [2]reported time and DyNeRF is trained on 8 GPUs and tested only on flame salmon. [3]trained with 90 frames. [4]trained with 50 frames.

model	metric			computational cost		
	PSNR↑	SSIM↑	LPIPS↓	Training time↓	FPS↑	Model size↓
DyNeRF[12] [12]	29.58	-	0.083	1344 hours	0.01	56 MB
NeRFPlayer[13] [24]	30.69	-	0.111	6 hours	0.05	1654 MB
MixVoxels [27]	30.30	0.918	0.127	1 hours 40 mins	0.93	512 MB
K-Planes [6]	30.86	0.939	0.096	1 hours 40 mins	0.13	309 MB
HyperReel[4] [1]	30.37	0.921	0.106	9 hours 20 mins	1.0	1362 MB
4DGS [30]	31.19	0.940	0.051	9 hours 30 mins	33.7	8700 MB
4DGaussians [29]	30.71	0.935	0.056	50 mins	51.9	59 MB
Ours	**31.31**	**0.945**	**0.037**	1 hours 52 mins	74.5	35 MB

Baselines We choose the start-of-the-art method as a baseline in each dataset. We compared against DyNeRF, NeRFPlayer, MixVoxels, K-Planes, HyperReel, Nerfies, HyperNeRF, and TiNeuVox on the NeRF baseline. In detail, we use the version of NeRFPlayer TensoRF VM, HyperNeRF DF, Mixvoxels-L, K-Planes hybrid. We compared with 4DGaussians, 4DGS, and D3DGS based on the Gaussian baseline. Meanwhile, we have not included STG in our comparison due to its requirement for per-frame Structure from Motion (SfM) points, which makes conducting a fair comparison challenging. Also, STG is not a deformable 3D Gaussian approach. We followed the official code and configuration, except increasing the training iterations for the Technicolor dataset to 1.5 times that of the 4DGaussians, in comparison to the Neural 3D Video dataset.

Metrics We report the quality of rendered images using PSNR, SSIM, and LPIPS. Peak Signal-to-Noise Ratio (PSNR) quantifies pixel color error between the rendered video and the ground truth. We utilize SSIM [28] to account for the perceived similarity of the rendered image. Additionally, we measure higher-level perceptual similarity using Learned Perceptual Image Patch Similarity (LPIPS) [33] with an AlexNet Backbone. Higher PSNR and SSIM values and lower LPIPS values indicate better visual quality.

4.1   Effectiveness on Dynamic Region

Neural 3D Video Dataset [12] includes 20 multi-view videos, with each scene consisting of either 300 frames, except for the `flame_salmon` scene, which comprises 1200 frames. These scenes encompass a relatively long duration and various movements, with some featuring multiple objects in motion. We utilized Neural 3D Video dataset to observe the capability to capture dynamic areas. Total six scenes (`coffee_martini, cook_spinach, cut_roasted_beef, flame_salmon, flame_steak, sear_steak`) are evaluated in Figure 3 and Table 1. The `flame_salmon` scene divided into four segments, each containing 300 frames.
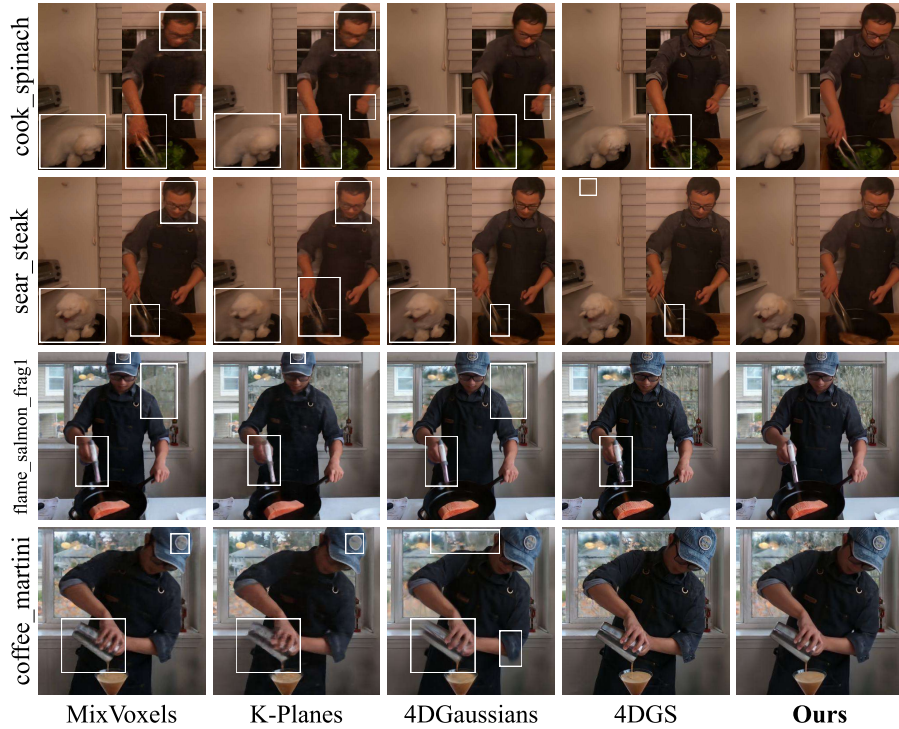
Fig. 3: Qualitative comparisons on the Neural 3D Video Dataset.

Table 1 presents quantitative metrics on the average metrics across the test views of all scenes, computational and storage costs on the first fragment of `flame_salmon` scene. Refer to the Appendix for per-scene details. Our method demonstrates superior reconstruction quality, FPS, and model size across compared to baselines. As the table shows, NeRF baselines generally required longer training and rendering times. While 4DGS shows relatively high reconstruction performance, it demands longer training times and larger VRAM storage capacity compared to other baselines. 4DGaussians requires lower computational and storage cost but it displays low reconstruction quality in some scenes with rapid dynamics, as shown in the teaser and Figure 3.

Figure 3 reports the rendering quality. Our method successfully reconstructs the fine details in moving areas, outperforming baselines on average metrics across test views. Baselines show blurred dynamic areas or severe artifacts in low-light scenes such as `cook_spinach` and `flame_steak`. 4DGS exhibits the disappearance of some static areas. In 4DGaussians, a consistent over-smoothing occurs in dynamic areas. All baselines experienced reduced quality in reflective or thin dynamic areas like clamps, torches, and windows.

Table 2: Average performance in the test view on Technicolor Dataset

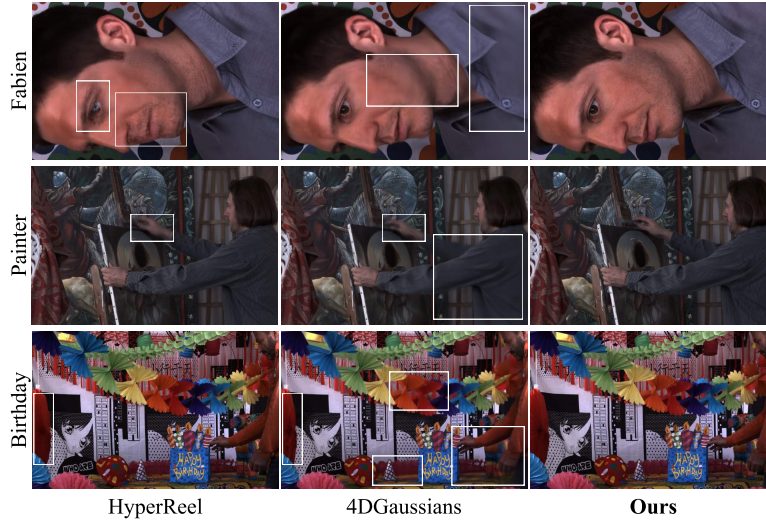| model	metric			computational cost		
	PSNR↑	SSIM↑	LPIPS↓	Training time↓	FPS↑	Model size↓
DyNeRF	31.80	-	0.140	-	0.02	0.6 MB
HyperReel	32.32	0.899	0.118	2 hours 45 mins	0.91	289 MB
4DGaussians	29.62	0.844	0.176	25 mins	34.8	51 MB
Ours	**33.24**	**0.907**	**0.100**	2 hours 55 mins	60.8	77 MB

Fig. 4: Qualitative comparisons on the Technicolor Dataset.

Technicolor Light Field Dataset [23] is a multi-view dataset captured with a time-synchronized 4×4 camera rig, containing intricate details. We train ours and the baselines on 50 frames of five commonly used scenes (`Birthday`, `Fabien`, `Painter`, `Theater`, `Trains`) using full-resolution videos at 2048×1088 pixels, with the second row and second column cameras used as test views.

Table 2 reports the average metrics across the test views of all scenes, computational and storage costs on `Painter` scene. HyperReel demonstrates overall high-quality results but struggles with relatively slow training times and FPS, and a larger model size. 4DGaussians exhibits fast training times and FPS but significantly underperforms in reconstructing fine details compared to other baselines. However, our method demonstrates superior reconstruction quality and faster FPS compared to the baselines.

As shown in Figure 4, HyperReel produces noisy artifacts due to incorrect predictions of the displacement vector. 4DGaussians fails to capture fine details in dynamic areas, exhibiting over-smoothing results. All baselines struggle to accurately reconstruct rapidly moving thin areas like fingers.

4.2   Scenes with Challenging Camera Setting

Table 3: Average performance in the test view on Hypernerf Dataset

model	metric			computational cost		
	PSNR↑	SSIM↑	LPIPS↓	Training time↓	FPS↑	Model size↓
Nerfies [20]	22.23	-	0.170	∼ hours	< 1	-
HyperNeRF DS [21]	22.29	0.598	**0.153**	32 hours	< 1	15 MB
TiNeuVox [4]	24.20	0.616	0.393	3 hours 30 mins	1	48 MB
D3DGS [31]	22.40	0.598	0.275	16 mins	6.95	309 MB
4DGaussians	25.03	0.682	0.281	50 mins	96.3	60 MB
Ours	**25.43**	**0.697**	0.231	1 hours 15 mins	139.3	33 MB
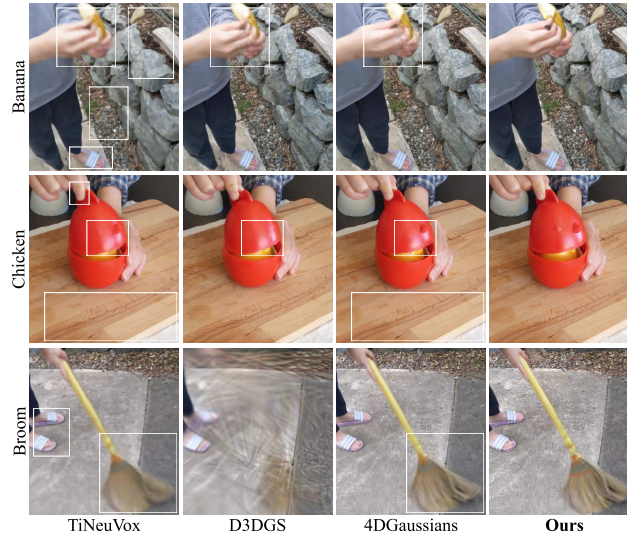
Fig. 5: Qualitative comparisons on the HyperNeRF Dataset.

HyperNeRF dataset includes videos captured using two phones rigidly mounted on a handheld rig. We train on all frames of four scenes (`3D Printer, Banana, Broom, Chicken`) at a resolution downsampled by half to 536×960. Due to memory constraints, D3DGS is trained on images downsampled by a quarter.

The table shows that our method outperforms the reconstruction performance with previous methods along with compact model size and faster FPS. Figure 5 shows that previous methods struggle to reconstruct fast-moving parts such as fingers and broom. Especially D3DGS deteriorates in `Broom` scene. Table 3 reports the average metrics across the test views of all scenes, computational and storage costs on `Broom` scene.

4.3   Analyses and Ablation Study

(a) canonical rendering   (b) coarse deform only   (c) fine deform only   (d) full rendering

Fig. 6: Deformation components. (a) The canonical space contains Gaussians to represent all target frames of the scene. (b) Applying coarse deformation to the canonical space roughly reflects the dynamics of the scene. (c) The rendering without coarse deformation and only with fine deformation looks similar to the canonical rendering, i.e., responsible for fine deformations. (d) Applying both coarse and fine deformation yields natural rendering results.

Frame 67   Frame 185

Fig. 7: Visualization of the magnitude of deformation. Coarse deformation (*blue*) captures large and slow changes, such as the movement of the head and torso, while fine deformation (*red*) is responsible for the fast and detailed movements of arms, tongs, shadows, *etc.*

Deformation components In Figure 6, we present an analysis of the coarse-fine deformation. To achieve this, we render a `flame_steak` scene by omitting each of our deformation components one by one. Our full rendering results from adding coarse and fine deformation to the canonical space (Figure 6d). When both are removed, rendering yields an image in canonical space (Figure 6a). Rendering with the coarse deformation, which handles large or slow changes in the scene, produces results similar to the full rendering (Figure 6b). On the other hand, fine deformation is responsible for fast or detailed changes in the scene, yielding rendering similar to canonical space (Figure 6c).

(a) Ours Full   (b) coarse deform only   (c) fine deform only   (d) Ours + **x** injection
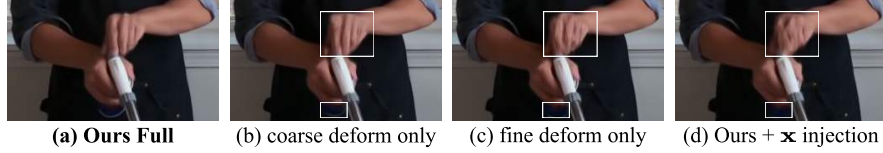
Fig. 8: Qualitative ablation results on coarse-fine deformation. (a) Our model achieves clear results with both coarse and fine decoders. (b-c) The quality of dynamic areas decreases if one is missing. (d) Additionally, introducing the coordinates of Gaussian as an additional input into our decoders results in a decrease in the quality of both static and dynamic regions.

Table 4: Quantitative ablation results on coarse-fine deformation.

| Method	PSNR↑	SSIM↑	LPIPS↓
Ours	**29.70**	**0.933**	**0.041**
coarse deformation only	29.48	0.931	0.044
fine deformation only	29.23	0.932	0.043
Ours + **x** injection	29.60	0.931	0.045

To examine the roles of the coarse and fine deformation in the coarse-fine deformation, we conduct a visualization on `flame_steak` scene. First, we compute the Euclidean norm of positional shifts between the current and subsequent frames. We then add the value to the DC components of the SH coefficients proportionally to the magnitude: blue for coarse deformation and red for fine deformation. For visual clarity, we render the original scene in grayscale. As illustrated in Figure 7, coarse deformation models slower changes such as body movement, while fine deformation models faster movements like cooking arms. Thus, we demonstrate that by downsampling the temporal embedding grid Z, we can effectively separate and model slow and fast deformations in the scene.

Ablation study We report the results of an ablation study on the deformation decoder in Figure 8 and Table 4. First, our full method (using both coarse and fine decoders) produces clear rendering results and models dynamic variations well (Figure 8a). Training only with the coarse or fine decoder leads to blurred dynamic areas and a failure to accurately capture detailed motion (Figure 8b-c). Additionally, we demonstrate experiments where the Gaussian center coordinates **x** are injected into the input of each decoder. As shown in Figure 8d, including the Gaussian coordinates degrades the quality of deformation, supporting our argument that coordinate dependency should be removed from the deformation function.

Furthermore, we report the results of an ablation study on the local smoothness regularization for per-Gaussian embeddings. As shown in Figure 9, our regularization improves the details and texture quality of moving objects. In Table 5, we show a performance comparison between the proposed regularization and

Table 5: Quantitative ablation results on local smoothness regularization. We compare the performance of applying our regularization with the physically-based regularization of Dynamic 3D Gaussians [17]. Our regularization better captures details of dynamic objects.

Method	PSNR↑	SSIM↑	LPIPS↓
Ours w/o embedding reg	32.26	0.951	0.037
+ our embedding reg	**32.34**	**0.952**	**0.036**
+ physically-based reg	32.08	0.950	0.036

Ours full   Ours w/o reg   Ours full   Ours w/o reg
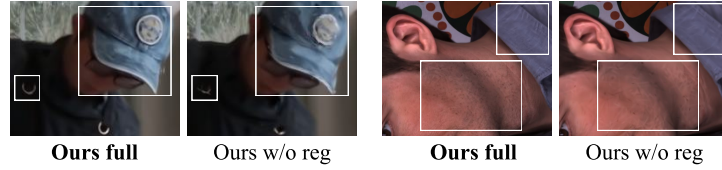
Fig. 9: Qualitative ablation results on local smoothness regularization.

the existing physically-based regularizations. To apply the method proposed by Luiten et al. [17] to ours, we make some modifications: 1) Like our method, we find the set of k-nearest-neighbours only when the densification occurs to reduce the computational cost. 2) For long-term local-isometry loss, we use the time of video frame used in previous training step instead of using the time of the first frame. Our regularization is simple and shows better performance compared to previous method.

5   Conclusion and Limitation

We propose a per-Gaussian deformation for 3DGS that takes per-Gaussian embeddings as input, instead of using the typical deformation fields from previous deformable 3DGS works, resulting in high performance. We enhance the reconstruction quality by decomposing the dynamic changes into coarse and fine deformation. However, our method learns inappropriate Gaussian deformation with casually captured monocular videos [7], like other baselines. We plan to address it in future work by introducing useful prior for monocular video settings.

4DGaussians   **Ours**   4DGaussians   **Ours**

Fig. 10: Limitation. Ours struggles with the casually captured monocular videos.

Acknowledgements

References

1. Attal, B., Huang, J.B., Richardt, C., Zollhoefer, M., Kopf, J., OâĂŹToole, M., Kim, C.: Hyperreel: High-fidelity 6-dof video with ray-conditioned sampling. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 16610–16620 (2023)
2. Cao, A., Johnson, J.: Hexplane: A fast representation for dynamic scenes. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 130–141 (2023)
3. Duisterhof, B.P., Mandi, Z., Yao, Y., Liu, J.W., Shou, M.Z., Song, S., Ichnowski, J.: Md-splatting: Learning metric deformation from 4d gaussians in highly deformable scenes. arXiv preprint arXiv:2312.00583 (2023)
4. Fang, J., Yi, T., Wang, X., Xie, L., Zhang, X., Liu, W., Nießner, M., Tian, Q.: Fast dynamic radiance fields with time-aware neural voxels. In: SIGGRAPH Asia 2022 Conference Papers (2022)
5. Feichtenhofer, C., Fan, H., Malik, J., He, K.: Slowfast networks for video recognition. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 6202–6211 (2019)
6. Fridovich-Keil, S., Meanti, G., Warburg, F.R., Recht, B., Kanazawa, A.: K-planes: Explicit radiance fields in space, time, and appearance. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 12479–12488 (2023)
7. Gao, H., Li, R., Tulsiani, S., Russell, B., Kanazawa, A.: Monocular dynamic view synthesis: A reality check. In: NeurIPS (2022)
8. Huang, Y.H., Sun, Y.T., Yang, Z., Lyu, X., Cao, Y.P., Qi, X.: Sc-gs: Sparse-controlled gaussian splatting for editable dynamic scenes. arXiv preprint arXiv:2312.14937 (2023)
9. Huang, Y.H., Sun, Y.T., Yang, Z., Lyu, X., Cao, Y.P., Qi, X.: Sc-gs: Sparse-controlled gaussian splatting for editable dynamic scenes. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 4220–4230 (June 2024)
10. Kerbl, B., Kopanas, G., Leimkühler, T., Drettakis, G.: 3d gaussian splatting for real-time radiance field rendering. ACM Transactions on Graphics **42**(4) (July 2023), https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/
11. Kim, S., Bae, J., Yun, Y., Lee, H., Bang, G., Uh, Y.: Sync-nerf: Generalizing dynamic nerfs to unsynchronized videos. arXiv preprint arXiv:2310.13356 (2023)
12. Li, T., Slavcheva, M., Zollhoefer, M., Green, S., Lassner, C., Kim, C., Schmidt, T., Lovegrove, S., Goesele, M., Newcombe, R., et al.: Neural 3d video synthesis from multi-view video. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2022)
13. Li, Z., Chen, Z., Li, Z., Xu, Y.: Spacetime gaussian feature splatting for real-time dynamic view synthesis. arXiv preprint arXiv:2312.16812 (2023)

14. Liang, Y., Khan, N., Li, Z., Nguyen-Phuoc, T., Lanman, D., Tompkin, J., Xiao, L.: Gaufre: Gaussian deformation fields for real-time dynamic novel view synthesis (2023)
15. Liu, Q., Liu, Y., Wang, J., Lv, X., Wang, P., Wang, W., Hou, J.: Modgs: Dynamic gaussian splatting from causually-captured monocular videos (2024), `https://arxiv.org/abs/2406.00434`
16. Lu, T., Yu, M., Xu, L., Xiangli, Y., Wang, L., Lin, D., Dai, B.: Scaffold-gs: Structured 3d gaussians for view-adaptive rendering. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 20654–20664 (2024)
17. Luiten, J., Kopanas, G., Leibe, B., Ramanan, D.: Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. In: 3DV (2024)
18. Martin-Brualla, R., Radwan, N., Sajjadi, M.S.M., Barron, J.T., Dosovitskiy, A., Duckworth, D.: NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections. In: CVPR (2021)
19. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. Communications of the ACM **65**(1), 99–106 (2021)
20. Park, K., Sinha, U., Barron, J.T., Bouaziz, S., Goldman, D.B., Seitz, S.M., Martin-Brualla, R.: Nerfies: Deformable neural radiance fields. ICCV (2021)
21. Park, K., Sinha, U., Hedman, P., Barron, J.T., Bouaziz, S., Goldman, D.B., Martin-Brualla, R., Seitz, S.M.: Hypernerf: A higher-dimensional representation for topologically varying neural radiance fields. arXiv preprint arXiv:2106.13228 (2021)
22. Pumarola, A., Corona, E., Pons-Moll, G., Moreno-Noguer, F.: D-nerf: Neural radiance fields for dynamic scenes. arXiv preprint arXiv:2011.13961 (2020)
23. Sabater, N., Boisson, G., Vandame, B., Kerbiriou, P., Babon, F., Hog, M., Gendrot, R., Langlois, T., Bureller, O., Schubert, A., et al.: Dataset and pipeline for multi-view light-field video. In: Proceedings of the IEEE conference on computer vision and pattern recognition Workshops. pp. 30–40 (2017)
24. Song, L., Chen, A., Li, Z., Chen, Z., Chen, L., Yuan, J., Xu, Y., Geiger, A.: Nerfplayer: A streamable dynamic scene representation with decomposed neural radiance fields. IEEE Transactions on Visualization and Computer Graphics **29**(5), 2732–2742 (2023)
25. Tancik, M., Casser, V., Yan, X., Pradhan, S., Mildenhall, B., Srinivasan, P.P., Barron, J.T., Kretzschmar, H.: Block-nerf: Scalable large scene neural view synthesis (2022)
26. Ververas, E., Potamias, R.A., Song, J., Deng, J., Zafeiriou, S.: Sags: Structure-aware 3d gaussian splatting. arXiv:2404.19149 (2024)
27. Wang, F., Tan, S., Li, X., Tian, Z., Liu, H.: Mixed neural voxels for fast multi-view video synthesis. arXiv preprint arXiv:2212.00190 (2022)
28. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. IEEE transactions on image processing **13**(4), 600–612 (2004)
29. Wu, G., Yi, T., Fang, J., Xie, L., Zhang, X., Wei, W., Liu, W., Tian, Q., Xinggang, W.: 4d gaussian splatting for real-time dynamic scene rendering. arXiv preprint arXiv:2310.08528 (2023)
30. Yang, Z., Yang, H., Pan, Z., Zhang, L.: Real-time photorealistic dynamic scene representation and rendering with 4d gaussian splatting. In: International Conference on Learning Representations (ICLR) (2024)
31. Yang, Z., Gao, X., Zhou, W., Jiao, S., Zhang, Y., Jin, X.: Deformable 3d gaussians for high-fidelity monocular dynamic scene reconstruction. arXiv preprint arXiv:2309.13101 (2023)

32. Yu, H., Julin, J., Milacski, Z.Ã., Niinuma, K., Jeni, L.A.: Cogs: Controllable gaussian splatting (2023)
33. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 586–595 (2018)
34. Zhao, B., Li, Y., Sun, Z., Zeng, L., Shen, Y., Ma, R., Zhang, Y., Bao, H., Cui, Z.: Gaussianprediction: Dynamic 3d gaussian prediction for motion extrapolation and free view synthesis. In: Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers âĞŹ24. SIGGRAPH âĞŹ24, ACM (Jul 2024). `https://doi.org/10.1145/3641519.3657417`, `http://dx.doi.org/10.1145/3641519.3657417`
35. Zwicker, M., Pfister, H., Van Baar, J., Gross, M.: Surface splatting. In: Proceedings of the 28th annual conference on Computer graphics and interactive techniques. pp. 371–378 (2001)