GauFRe: Gaussian Deformation Fields for Real-time Dynamic Novel View Synthesis

Yiqing Liang[‡], Numair Khan, Zhengqin Li, Thu Nguyen-Phuoc,
Douglas Lanman, James Tompkin[‡], Lei Xiao
Meta [‡]Brown University

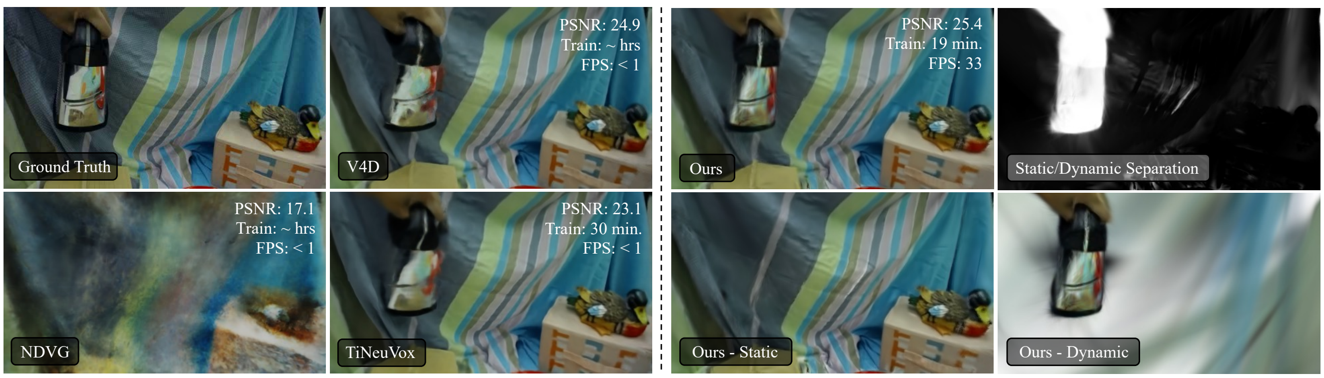


Figure 1. GauFRe reconstructs dynamic scenes from casually-captured monocular video inputs. Our representation renders in real-time ($>$ 30FPS) while achieving high rendering performance. GauFRe also decomposes static/dynamic regions without extra supervision.

Abstract

We propose a method for dynamic scene reconstruction using deformable 3D Gaussians that is tailored for monocular video. Building upon the efficiency of Gaussian splatting, our approach extends the representation to accommodate dynamic elements via a deformable set of Gaussians residing in a canonical space, and a time-dependent deformation field defined by a multi-layer perceptron (MLP). Moreover, under the assumption that most natural scenes have large regions that remain static, we allow the MLP to focus its representational power by additionally including a static Gaussian point cloud. The concatenated dynamic and static point clouds form the input for the Gaussian Splatting rasterizer, enabling real-time rendering. The differentiable pipeline is optimized end-to-end with a self-supervised rendering loss. Our method achieves results that are comparable to state-of-the-art dynamic neural radiance field methods while allowing much faster optimization and rendering.

Project Webpage: this url.

1. Introduction

High-quality 3D reconstruction of dynamic scenes from RGB images is a persistent challenge in computer vision. The challenge is especially great from monocular camera video: the setting is ill-posed as constraints on the surface geometry must be formed by simultaneously solving for an estimate of the scene's motion over time. Structure from motion provides an estimate of rigid motion for static scenes, but real-world scenes have motions that extend beyond rigid or piecewise rigid to continual deformation, such as on human subjects. Given this challenge, one relaxation of the problem is to consider novel view synthesis instead, where we reconstruct the appearance of the scene to allow applications in editing to re-pose or re-time the scene.

Inverse graphics approaches using an image rendering loss have recently created optimization-based reconstruction methods that can achieve high quality for static or dynamic scenes with many cameras. These often use neural networks (or multi-layer perceptrons; MLPs) as a function to predict the values of physical properties in a field, such as the density and radiance volumes within the influential neural radiance field (NeRF) technique [20]. Optimizing these MLPs with gradient descent is robust, often leading to good solutions without tricky regularization [35]. However, neural networks are time-consuming to optimize via gradient descent, and volume rendering requires many samples of the network to create an image. Faster optimization and subsequent rendering can be achieved with the help

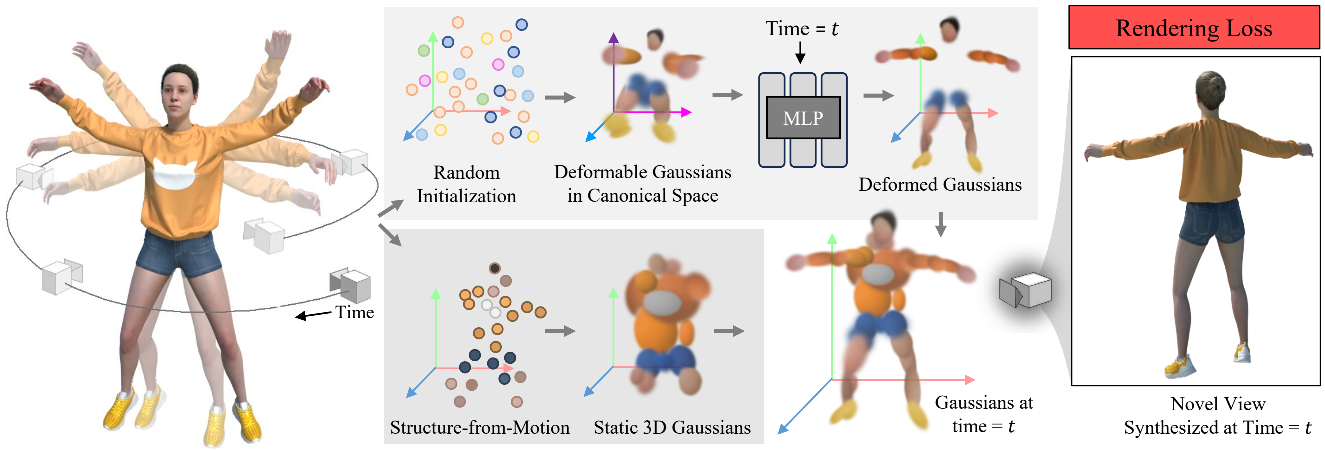


Figure 2. **An overview of our dynamic scene representation.** At each time frame t, our method reconstructs the scene as a combination of static and deformable anisotropic 3D Gaussians. The features of the deformable Gaussians are optimized in a canonical space and warped into frame t using a deformation field. The static Gaussians are optimized in world space. We represent the deformation field using a multi-layer perceptron (MLP) parameterized by time t.

of spatial indexing data structures, such as voxel grids [6], octrees [41], and multi-scale hash tables [21, 31], or with proxy geometries such as planes [2, 7]. As they lack the self-regularizing properties of neural networks, these may require additional regularization. Other proxies are possible: following point-based graphics [40, 44], composing a scene of many isotropic Gaussians is convenient as they are differentiable everywhere [27], can be splatted in closed form [19, 30], and can be z-sorted efficiently under small-Gaussian assumptions without ray marching [13]. Careful efficient implementation [12] leads to real-time rendering at high resolutions, and overall produces compelling results.

Extending this idea to parameterize Gaussians by time for dynamic scenes is natural, with the idea that each represents a moving and deforming particle or blob/area of space tracked through time—the Lagrangian interpretation in the analogy to fluid flow. This can work well in settings with sufficient constraints upon the motion of the flow, e.g., in 360° multi-camera settings [18]. For the underconstrained monocular video setting where constraints are sparse, it is challenging to directly optimize the positions and covariances of Gaussians as accurate prediction of both geometry and motion are required, leading to low-quality output.

Instead, for monocular video, we propose a Eulerian perspective on motion by modeling a field over time that can be sampled to predict Gaussian deformation. This deformation happens from a canonical Gaussian space. Rather than use a fixed spatial data structure for this field, which can be memory expensive, we use an MLP to represent the field. This MLP is less sensitive to incorrect initialization and can optimize more easily given the sparse constraints than directly optimizing Gaussians. Beyond that, most regions of real-world dynamic scene are quasi-static. As such, we split the scene into its static and dynamic components with a separate non-deforming set of Gaussians that are initialized around structure-from-motion-derived 3D points to ease the separation. Using reconstruction losses on the input video, we optimize the position, covariance, opacity, and appearance parameters of static Gaussians directly, and optimize the position of the dynamic Gaussian clouds through the deformation MLP. In evaluation, this approach achieves comparable quality to non-Gaussian-based neural scene representations, while being fast to optimize (20 minutes rather than hours) and providing real-time rendering for novel view synthesis.

We contribute:

1. A dynamic scene representation of canonical Gaussians deformed by a field represented by an MLP.
2. A static Gaussian cloud that represents quasi-static regions and allows explicit separation of dynamic regions.
3. An experimental validation of this approach on synthetic and real-world datasets against eight baselines.

2. Related Work

3D and 4D Scene Reconstruction Given the success of 3D voxel grids as an explicit representation for static scenes [3, 6, 11, 31, 41], a straightforward approach is to extend them into a fourth dimension for time to handle dynamic content. Unfortunately, the memory requirements of such a 4D grid quickly become prohibitive even for short sequences. As a result, a number of methods propose structures and techniques that reduce the memory complexity while still fundamentally being four-dimensional grids. Park *et al.* [24] extend Muller *et al.*'s multi-level spatial hash grid [21] to 4D, and additionally allow for the separate learning of static and dynamic features. This latter capability allows the model to focus the representational power of the 4D hash grid on dynamic regions. Another common approach factorizes the spatio-temporal grid into low-dimensional components. Jang and Kim [10] propose rank-one vectors or low-rank matrices, providing a 4D counterpart of the 3D tensorial radiance fields of Chen *et al.* [3]. Shao *et al.* [29] hierarchically decompose the scene into three time-conditioned volumes, each represented by three orthogonal planes of feature vectors. An even more compact HexPlanes solution by Cao and Johnson [2] use six planes, each spanning two of the four spatio-temporal axes. A similar decomposition is presented by Fridovich-Keil *et al.* [7] as part of a general representation that uses $\binom{d}{2}$ planes to factorize any d-dimensional space.

Motion Reconstruction While 4D, none of these approaches explicitly account for motion, e.g., they do not define correspondence over time. To do so, Tretschk *et al.* [32] discover deformation fields that align scenes under reprojection. Park *et al.* [23] deform a canonical field by a higher-dimensional time-varying field to accommodate topology changes. Fang *et al.* [5] demonstrate a hybrid representation that defines a 3D canonical space explicitly as a voxel grid, then queries it using an implicit deformation field for 4D spatio-temporal points. Guo *et al.* [9] propose a similar approach but use additional features interpolated from an explicit time-independent deformation grid to deform into the canonical frame. Some works also attempt to split static and dynamic scene parts to improve quality [16, 17, 34]. Unlike these works, our approach uses Gaussian clouds instead of MLPs or spatial data structures, including a static cloud and a dynamic cloud deformed from a canonical frame.

Point-based Rendering Optimization-based point graphics are also popular for reconstruction [1], including spherical proxy geometries [15], splatting-based approaches [12, 40], methods for computing derivates of points rendered to single pixels [28], and methods for view-varying optimization of points [14]. Point-based approaches can also be fast, accelerating rendering and so optimization too [36, 42]. Such approaches are also adaptable to dynamic scenes [25], such as the dynamic point fields method of Prokudin *et al.* In contrast, our approach uses Gaussians as the base primitive, from which dynamic scene regions are deformed around fixed static regions.

Contemporaneous Work Finally, we note contemporaneous works using dynamic Gaussian representations, all published or in preprint within the last three months. Liuten *et al.* [18] consider the 360° multi-camera case that is constrained in spacetime, and take a Lagrangian tracking approach. Yang and Yang et al. [39] consider a more flexible approach where 4D Gaussian position and color are directly optimized over time. Zielonka *et al.* [43] approach the problem of driveable human avatars from multi-camera capture using Gaussians for tracking.

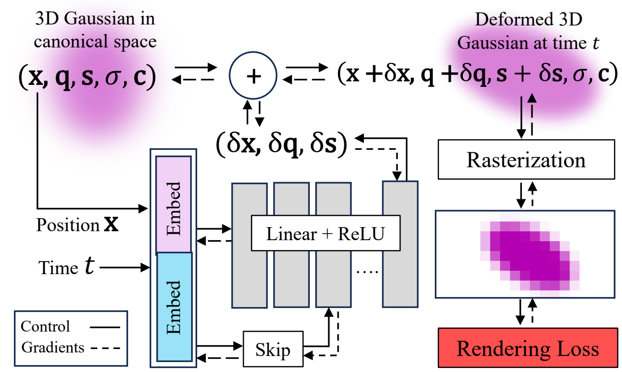


Figure 3. **Optimization architecture.** The deformation field's domain is an embedding space of time t and Gaussian position $\mathbf{x}$. Gradients do not flow to $\mathbf{x}$ from the MLP to prevent entanglement.

The methods of Wu *et al.* [33] and Yang and Gao *et al.* [38] are closest to our approach as both use a deformation field parameterized by an MLP. Wu *et al.* represent the Gaussians via HexPlanes and hashed coordinates, which is fast to render but does not produce as high a quality of reconstruction as our method and does not separate static and dynamic scene regions. Yang and Gao *et al.* [38] use an explicit rather than spatial representation for the Gaussians, but also do not separate static and dynamic regions.

3. Method

3.1. 3D Gaussian Splatting

Following Kerbl *et al.* [12], we start by representing the scene as a set of n points $\{\mathbf{x}_i \in \mathbb{R}^3, i = 1, ..., n\}$. Each point is associated with features $(\Sigma_i, \sigma_i, \mathbf{c}_i)$ that define the local radiance field as an anisotropic Gaussian distribution centered at $\mathbf{x}_i$ with covariance Σ_i, density σ_i, and view-dependent color $\mathbf{c}_i$ represented by 2^{nd}-order spherical harmonics. Given a set of multi-view images of the scene, we can penalize a rendering loss to optimize the set of Gaussians $\{\mathbf{G}_i = (\mathbf{x}_i, \Sigma_i, \sigma_i, \mathbf{c}_i)\}$ to represent the scene's global radiance field for tasks like novel view synthesis.

To ensure Σ_i represents a valid positive semi-definite covariance matrix in the optimization, it is factored into a rotation matrix $\mathbf{R}_i \in \mathbb{R}^{3\times3}$ and scaling vector $\mathbf{s}_i \in \mathbb{R}^3$. An exponential activation is applied to $\mathbf{s}_i$ to prevent negative values while retaining differentiability over the domain. Thus,

$$\Sigma_i = \mathbf{R}_i \mathrm{Exp}(\mathbf{s}_i) \mathrm{Exp}(\mathbf{s}_i{}^T) \mathbf{R}_i{}^T . \tag{1}$$

In practice, $\mathbf{R}_i$ is inferred from a unit-length quaternion

$\mathbf{q}_i \in \mathbb{R}^4$ that provides better convergence behavior. The initial position $\mathbf{x}_i$ of the Gaussians is provided by a 3D point cloud obtained with a structure-from-motion algorithm. As the optimization proceeds, the Gaussians are periodically cloned, split, and pruned to achieve a suitable trade-off between rendering quality and computational resources.

In addition to the Gaussian scene representation, Kerbl *et al*. demonstrate how the many continuous radiance distributions can be efficiently rendered on graphics hardware. Given a target camera view transformation $\mathbf{V}$ and projection matrix $\mathbf{K}$, each $\mathbf{G}_i$ is reduced to a Gaussian distribution in screen space with projected mean $\mathbf{u}_i = \mathbf{K}\mathbf{V}\mathbf{x}_i \in \mathbb{R}^2$ and 2D covariance defined by the Jacobian $\mathbf{J}$ of $\mathbf{K}$ as

$$\Sigma_i' = \mathbf{J}\mathbf{V}\Sigma_i\mathbf{V}^T\mathbf{J}^T \tag{2}$$

The 2D Gaussians are then rasterized using Zwicker *et al*.'s Elliptical Weighted Average (EWA) splatting [44].

3.2. Deformable Gaussian Fields

To model a dynamic scene, we assume that it is equivalent to a static scene that is deformed from a canonical point set $\{G_i = (\mathbf{x}_i, \mathbf{s}_i, \mathbf{q}_i, \sigma_i, \mathbf{c}_i)\}_{i \in N}$ via a deformation field parameterized by an MLP Φ:

$$\Phi : (\mathbf{x}_i, t) \rightarrow (\delta\mathbf{x}_i^t, \delta\mathbf{s}_i^t, \delta\mathbf{q}_i^t), \tag{3}$$

where density σ_i does not change over time, and neither does the view-dependent appearance $\mathbf{c}_i$ of a Gaussian—e.g., only a Gaussian's position $\mathbf{x}$, scale $\mathbf{s}$, and rotation via $\mathbf{q}$ can changes to describe the scene, which is equivalent to an affine transform. Allowing the deformation field to vary σ_i and $\mathbf{c}_i$ provides too few constraints on the underlying scene motion, as Gaussians can appear or disappear, or change their appearance, to represent motions.

The deformed position requires no activation to apply:

$$\mathbf{x}_i^t = \mathbf{x}_i + \delta\mathbf{x}_i^t \tag{4}$$

For S, we could predict a pre- or post-exponentiated delta:

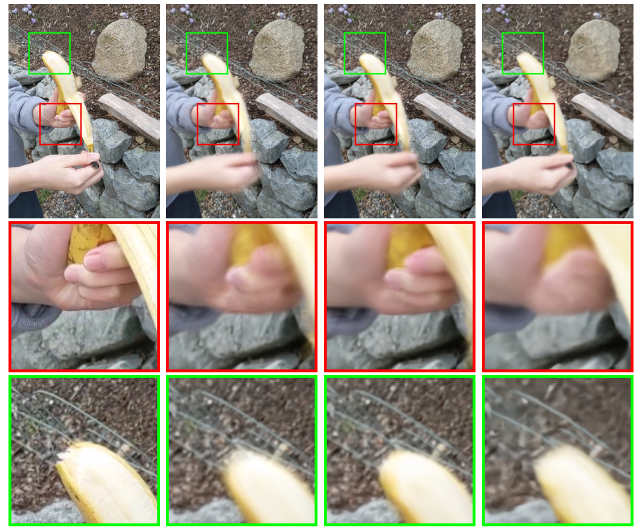
$$\mathrm{Exp}(\mathbf{s}_i^t) = \mathrm{Exp}(\mathbf{s}_i + \delta\mathbf{s}_i^t) \quad \text{or} \quad \mathrm{Exp}(\mathbf{s}_i^t) = \mathrm{Exp}(\mathbf{s}_i) + \delta\mathbf{s}_i^t \tag{5}$$

Pre- presents a log-linear estimation problem, which is simpler than an exponential one, and allows the optimization to still estimate negative values. Empirically, this improved optimized representation quality substantially (Fig. 4).

We must also be careful with quaternion deformation:

$$||\mathbf{q}_i^t|| = ||\mathbf{q}_i + \delta\mathbf{q}_i^t|| \quad \text{or} \quad ||\mathbf{q}_i^t|| = ||\mathbf{q}_i|| + \delta\mathbf{q}_i^t \tag{6}$$

Only unit quaternions represent rotations, so the right-hand variant will introduce additional unwanted transformations into the deformation field. Further, $\mathbf{q}_i + \delta\mathbf{q}_i^t$ represents a rotation that is half-way between $\mathbf{q}_i$ and $\delta\mathbf{q}_i^t$. While not strictly a delta, it is fast and sufficient for small rotations.



Input | Post-activ. | Ours | TiNeuVox [4]

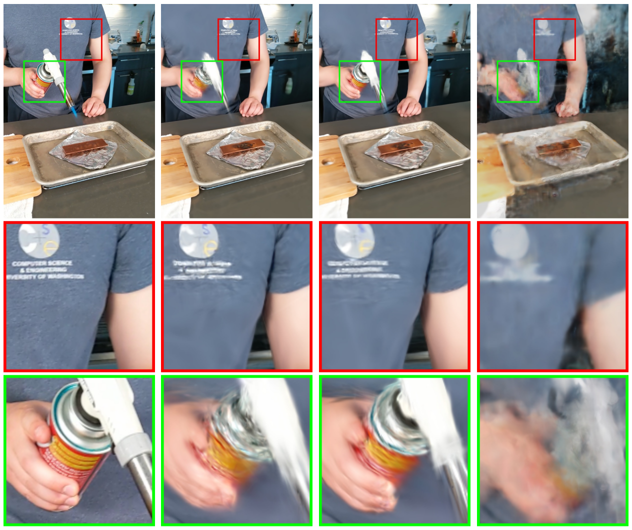
Figure 4. **Estimating pre-activated scale significantly improves quality on dynamic objects.** *Left to right:* Ground truth, estimating post-activation scale difference, estimating pre-activation scale difference, and the results of prior TiNeuVox work [4] that uses an MLP to directly predict the new scale for a particular timestep.

3.3. Static-Dynamic Decomposition

Many dynamic scenes contain significant static world regions that do not need to be deformed. Further, real-world sequences contain small image noise or camera pose errors. A fully-dynamic model will be forced to spend MLP capacity describing deformations in irrelevant regions. In contrast, if these regions can be ignored, the MLP can increase deformation fidelity and overall improve image quality. This issue compounds with the number of required Gaussians and their cloning and splitting as too many Gaussians can overfit to represent noise; similarly, unnecessarily cloning and splitting wastes machine memory. For instance, for a fully-dynamic model, some of our tested sequences lead to out-of-memory crashes.

To combat this, we use a separate static Gaussian point cloud $\{\mathbf{G}_j = (\mathbf{x}_j, \Sigma_j, \sigma_j, \mathbf{c}_j)\}_{j \in N_r}$ that leads to higher-quality overall dynamic view synthesis (Fig. 5). In random initialization settings, half of the point cloud is assigned as static and half as dynamic. During rendering, we concatenate $\{\mathbf{G}_i^t\}_{i \in N}$ and $\{\mathbf{G}_j\}_{j \in N_r}$ as input to the rasterizer. During optimization, the two point cloud are densified and pruned separately and can capture the appearance of static and dynamic parts without explicit supervision (Fig. 6).

However, this process may need some help. Consider the case when Kerbl *et al*. [12] initialize the Gaussian point cloud by sampling a precomputed 3D point cloud from structure from motion. Dynamic objects will not appear in this point cloud, and so it will take a long time to optimize



Input Dyn. only Ours NDVG [9]

Figure 5. **Separate deformable and quasi-static regions improves quality in dynamic parts.** *Left to right:* Ground truth, deformable set, separate static and deformable sets, and the results of NDVG [9] that uses deformable grids without separation.

Input Static Dynamic Separation

Figure 6. **Static/dynamic separation visualization.** The input video does not see one part of the static scene (white).

Gaussians into this region (if they ever reach it at all). As such, we randomly re-distribute the dynamic Gaussians in space. Further, some initial point clouds are dense given the video input ($> 1e5$) and using all points is memory expensive; random sampling only a subset mitigates this issue.

3.4. Implementation

Positional/Temporal Encoding We facilitate high-frequency deformation fields through PE encoding both the position $\mathbf{x}$ and time t inputs to the MLP by γ, where, for example, $L_{\mathbf{x}}$ is the respective encoding base for $\mathbf{x}$:

$$\begin{aligned}\gamma(\mathbf{x}) = (\sin(2^0\mathbf{x}), \cos(2^0\mathbf{x}), \sin(2^1\mathbf{x}),\\ \cos(2^1\mathbf{x}), \dots, \sin(2^{L_{\mathbf{x}}-1}\mathbf{x}), \cos(2^{L_{\mathbf{x}}-1}\mathbf{x}))\end{aligned} \tag{7}$$

We use $L_\mu = 10, L_t = 6$ for synthetic scenes and $L_\mu = 10, L_t = 6$ for real-world scenes.

Network architecture Our deformation MLP (Fig. 3) is inspired by Fang *et al*. [4]. Along with a time t embedding vector space, we also use a Gaussian position $\mathbf{x}$ embedding vector space. In the middle MLP layer, we add a skip connection such that the time embedding and Gaussian position $\mathbf{x}$ embedding are concatenated and passed into the second half of the MLP. Empirically, this improved performance. As the Gaussian position is input to the MLP and is also being optimized through a separate path, we manually stop the gradient from flowing back through the MLP to the Gaussian position. This prevents the deformation MLP from entangling the position representations.

We use MLPs with 8 layers of 256 neurons for synthetic scenes and 6 layers of 256 neurons for real-world scenes.

Optimizer We use Adam with $\beta = (0.9, 0.999)$ and eps $= 1e^{-15}$. The learning rate for the deformation MLP is 0.0007 for synthetic datasets and 0.001 for real-world datasets, with exponential scheduling that shrinks to $0.002\times$ the original learning rate until 30 K iterations. We densify both Gaussian point clouds until 20 K iterations, and keep optimizing both the Gaussians and the MLP until 40 K iterations for synthetic datasets and 60 K iterations for real-world datasets.

Warm-up Routine To stabilize optimization, the MLP linear layer initial weights and bias are set to $\sim \mathcal{N}(0, 1e^{-5})$ such that only small deformations are predicted at the beginning. During initial optimization, we only begin deforming Gaussians after 3 K iterations to provide a more stable start.

Losses We optimize using an image-based reconstruction loss only. For synthetic scenes, we use an L1 loss. For real-world scenes, in early optimization until 20 K iterations, we use an L2 loss; then, we switch to an L1 loss. This helps to increase reconstruction sharpness late in the optimization while allowing gross errors to be minimized quickly.

4. Experiments

Metrics We measure novel view synthesis performance using standard PSNR, SSIM, MS-SSIM and LPIPS metrics. We use both SSIM and MS-SSIM as each were used previously as standard on different datasets. We report optimization time and rendering time of methods on a single NVIDIA 3090 GPU.

Synthetic dataset: D-NeRF [26] This contains monocular exocentric 360° recordings of 8 dynamic synthetic objects with large motions and realistic materials. To fairly compare with peers, we train and render at half resolution (400×400) with a white background.

Real-world dataset: NeRF-DS [37] This is formed of real-world dynamic scene sequences containing specular objects captured with a handheld camera.

4.1. Comparisons

MLP + MLP deformation We report results from Nerfies [22], HyperNeRF[23] and NeRF-DS on the NeRF-DS dataset (Tab. 2). All three use volumes represented by MLPs to recover a scene, where the scene is then deformed via another MLP from a canonical space. While MLPs can achieve high quality, they take a long time to train. Our approach achieves higher reconstruction quality in a much shorter amount of time, and allows real-time rendering of the recovered representation.

Fast optimization Voxel grids allow faster optimization. For example, NDVG [9] uses a fixed voxel grid in a canonical space and deforms it with an MLP, leading to rapid optimization times. However, their quality suffers against our method, showing significantly worse metrics across D-NeRF and NeRF-DS dataset. In qualitative comparisons, we see lower image quality in both D-NeRF (Fig. 7) and NeRF-DS (Fig. 8). Another approach is TiNeuVox[4]. Again optimization time is fast, but quality is notably lower by eye and by metrics on D-NeRF and NeRF-DS. Plane-based methods also provide fast optimization. Both K-planes [7] and HexPlanes [2] show similar optimization times to our method on D-NeRF, with reduced image quality across all metrics.

Fast rendering If we only wished to render a scene quickly, we could spend longer in optimization. For instance, V4D [8] uses 3D voxel grids and a smaller deformation MLPs with volume rendering. Efficient implementation can allow inference at a few Hz. However, our representation can be rendered in real-time while showing better results on both D-NeRF and NeRF-DS data.

Contemporaneous Gaussian We run Wu *et al.* [33]'s 4DGaussians public code to generate results on D-NeRF. While their method is faster to optimize, both methods can be rendered in real time. Further, both methods produce broadly similar qualitative performance, with only a slight increase in metrics for our method. Our approach additionally separates static and dynamic regions, which may be useful additional information for downstream tasks.

Ablation Table 4 shows image quality metrics for ablations of our method that remove components. The most significant of these is the removal of the static Gaussians, which improves image sharpness in static regions. We also ablate three other components: 1) the transition from L2 loss to L1, 2) the choice to pre-exponentiate the scale deformation instead of post-exponentiating it, and 3) the warmup of the Gaussian optimization before allowing deformation. Each contributes a smaller improvement to the final image quality of the method. Further, the addition of each component does not have an impact upon the final novel view render time.

	PSNR↑	SSIM↑	LPIPS↓	Optim. ↓	Render↓
NDVG [9]	30.5	0.965	0.054	25mins	1.7s
TiNeuVox [5]	32.9	0.972	0.041	20mins	1.9s
K-planes [7]	29.2	0.961	0.060	60mins	8s
Hexplane [2]	31.0	0.968	0.039	15mins	0.5s
V4D [8]	33.4	0.978	0.027	~hours	0.5s
4DGaussians [33]	32.9	0.977	0.024	15mins	0.01s
Ours	34.8	0.982	0.020	25mins	0.02s

Table 1. **Quantitative image quality results on the dataset from D-NeRF [26].** Our method produces the highest metrics in the synthetic D-NeRF dataset with accurate camera poses.

	PSNR↑	SSIM↑	LPIPS↓
Nerfies [22]	20.1	0.707	0.349
HyperNeRF [23]	23.0	0.854	0.181
NeRF-DS [37]	23.7	0.885	0.143
NDVG [9]	19.1	0.584	0.417
TiNeuVox [4]	21.7	0.818	0.219
V4D [8]	23.5	0.884	0.142
Ours	23.8	0.887	0.144

Table 2. **Quantitative image quality results on the dataset from *NeRF-DS [37]*.** Our approach on average shows improved PSNR, SSIM (here the multi-scale variant), and LPIPS metric performance than most methods, and comparable image quality metric performance to NeRF-DS [37] and V4D [8]. However, in contrast to these two methods that each take hours to optimize a representation for a scene and seconds to render each frame, our approach takes minutes to optimize per scene and can render frames in real time (Tab. 3).

5. Limitations and Conclusion

Currently, we use a single MLP to model the whole deformation field. Despite the existence of the static Gaussian points, modeling scenes with big or irregular motion is difficult for this network, especially given that the network capacity has to be constrained to prevent overfitting on the input images. This may produce blurriness or floating artifacts. A potential improvement is to aid the MLP with better knowledge of where deformations may happen, such as by using priors from optical flow estimation methods. Additionally, our dynamic/static separation is sensitive to the quality of the initial SfM point cloud for real-

Figure 7. Quantitative comparison of our approach and the baseline methods for test views from the synthetic DNeRF [26] dataset. All methods reproduce the rough geometry, but sufficient sampling is necessary to reproduce the fine detail. Our approach can efficiently spread Gaussians to both static and dynamic regions to maximize quality, producing the sharpest image of all compared methods.

world scenes. Its benefit is to better distribute static points, but dynamic points still must be optimized from a random initialization—some dynamic elements are sensitive to this.

Conclusion Gaussians parameterized by time are a powerful way to optimize and render dynamic scenes with high quality. For monocular settings, we show that a deformation field is a suitable representation that achieves high quality reconstructions without degrading render times beyond real time. Notably, we show that separating the Gaussian set into static and dynamic regions rather than just dynamic improves quality and can produce segmented scenes. In comparisons to other methods, we show improved quality by visual comparison, and by image-quality metrics like LPIPS on both synthetic and real world scenes.

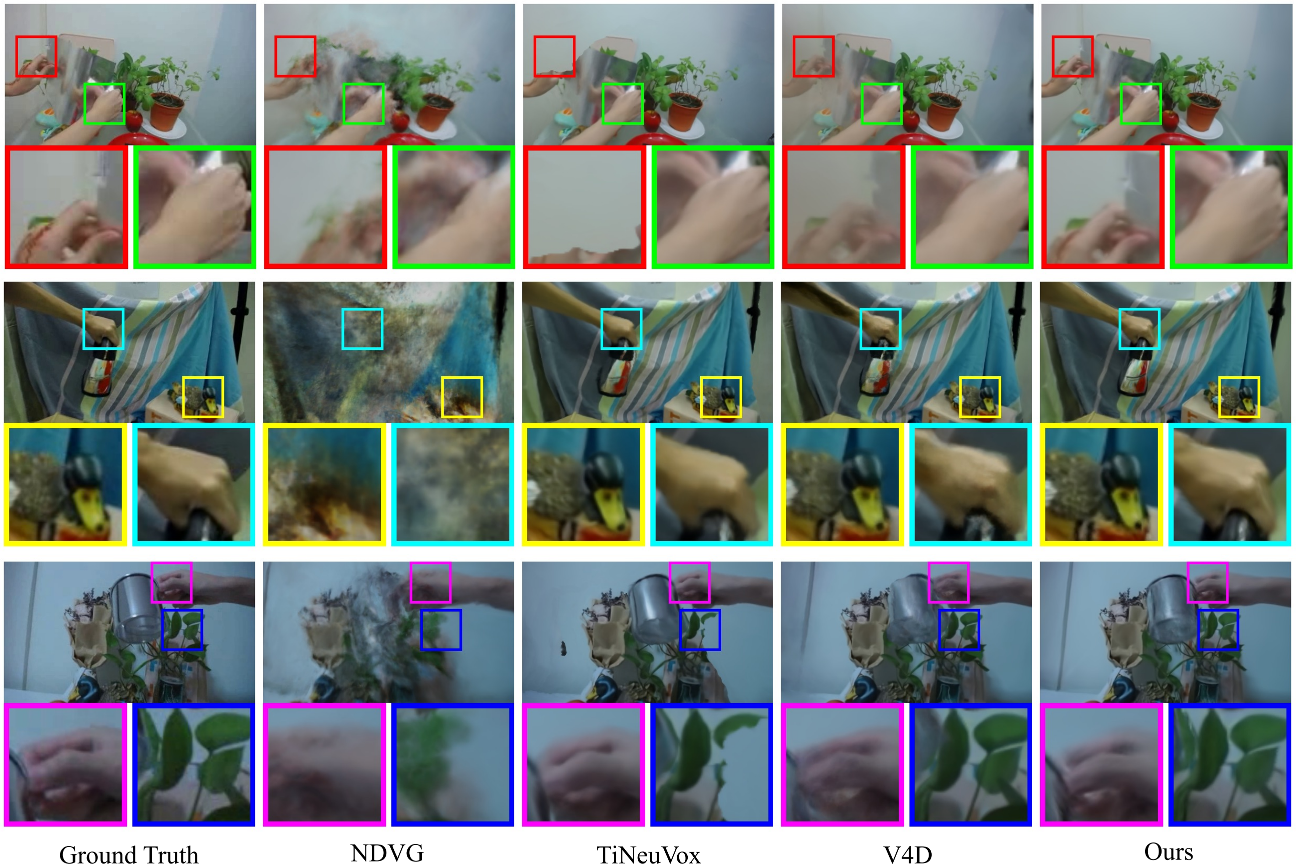


Figure 8. **Qualitative results on the NeRF-DS [37] monocular video dataset versus voxel- and sampled-space-based approaches show reduced quality compared to our method.** Sequences from top to bottom are *Sheet*, *Bell*, and *Cup*. Our approach better reproduces fine details for test views, including details on dynamic objects such as hands, and higher quality on static objects.

	PSNR↑	Optim.↓	Render↓
Nerfies [22]	20.1	∼hours	-
HyperNeRF [23]	23.0	∼hours	-
NeRF-DS [37]	23.7	∼hours	-
NDVG [9]	19.1	∼ 1hour	> 1s
TiNeuVox [4]	21.7	30mins	> 1s
V4D [8]	23.5	∼hours	> 1s
Ours	23.8	19mins	0.03s

Table 3. Quantitative comparison on the *NeRF-DS [37]* dataset. While some methods achieve comparable performance to ours, our method is much faster to optimize and to render.

	PSNR↑	MS-SSIM↑	LPIPS↓
No Static Gaussians	23.47	0.876	0.152
No L2 transition to L1	23.66	0.882	0.148
Scale post-exponentiate	23.77	0.882	0.154
No Gaussian deform warmup	23.78	0.884	0.146
Full	23.80	0.887	0.144

Table 4. **Model ablations using the *NeRF-DS [37]* dataset, ordered by increasing PSNR.** Our model with all components maximizes all three metrics, with the addition of static Gaussians increasing overall quality the most.

References

[1] Kara-Ali Aliev, Artem Sevastopolsky, Maria Kolos, Dmitry Ulyanov, and Victor Lempitsky. Neural point-based graphics. 2020. 3

[2] Ang Cao and Justin Johnson. Hexplane: A fast representation for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 130–141, 2023. 2, 3, 6

[3] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *European Conference on Computer Vision*, pages 333–350. Springer, 2022. 2, 3

[4] Jiemin Fang, Taoran Yi, Xinggang Wang, Lingxi Xie, Xiaopeng Zhang, Wenyu Liu, Matthias Nießner, and Qi Tian. Fast dynamic radiance fields with time-aware neural voxels. In *SIGGRAPH Asia 2022 Conference Papers*, 2022. 4, 5, 6, 8

[5] Jiemin Fang, Taoran Yi, Xinggang Wang, Lingxi Xie, Xiaopeng Zhang, Wenyu Liu, Matthias Nießner, and Qi Tian. Fast dynamic radiance fields with time-aware neural voxels. In *SIGGRAPH Asia 2022 Conference Papers*, pages 1–9, 2022. 3, 6

[6] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5501–5510, 2022. 2

[7] Sara Fridovich-Keil, Giacomo Meanti, Frederik Rahbæk Warburg, Benjamin Recht, and Angjoo Kanazawa. K-planes: Explicit radiance fields in space, time, and appearance. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12479–12488, 2023. 2, 3, 6

[8] Wanshui Gan, Hongbin Xu, Yi Huang, Shifeng Chen, and Naoto Yokoya. V4d: Voxel for 4d novel view synthesis. *IEEE Transactions on Visualization and Computer Graphics*, 2023. 6, 8

[9] Xiang Guo, Guanying Chen, Yuchao Dai, Xiaoqing Ye, Jiadai Sun, Xiao Tan, and Errui Ding. Neural deformable voxel grid for fast optimization of dynamic view synthesis. In *Proceedings of the Asian Conference on Computer Vision*, pages 3757–3775, 2022. 3, 5, 6, 8

[10] Hankyu Jang and Daeyoung Kim. D-tensorf: Tensorial radiance fields for dynamic scenes. *arXiv preprint arXiv:2212.02375*, 2022. 3

[11] Animesh Karnewar, Tobias Ritschel, Oliver Wang, and Niloy Mitra. Relu fields: The little non-linearity that could. In *ACM SIGGRAPH 2022 Conference Proceedings*, pages 1–9, 2022. 2

[12] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42 (4), 2023. 2, 3, 4

[13] Numair Khan, Min H. Kim, and James Tompkin. Differentiable diffusion for dense depth estimation from multi-view images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021. 2

[14] Georgios Kopanas, Julien Philip, Thomas Leimkühler, and George Drettakis. Point-based neural rendering with per-view optimization. *Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering)*, 40(4), 2021. 3

[15] Christoph Lassner and Michael Zollh"ofer. *arXiv:2004.07484*, 2020. 3

[16] Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 3

[17] Yiqing Liang, Eliot Laidlaw, Alexander Meyerowitz, Srinath Sridhar, and James Tompkin. Semantic attention flow fields for monocular dynamic scene decomposition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 21797–21806, 2023. 3

[18] Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. In *3DV*, 2024. 2, 3

[19] Youssef A Mejjati, Isa Milefchik, Aaron Gokaslan, Oliver Wang, Kwang In Kim, and James Tompkin. GaussiGAN: Controllable image synthesis with 3d gaussians from unposed silhouettes. In *British Machine Vision Conference*, 2021. 2

[20] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020. 1

[21] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics (ToG)*, 41(4):1–15, 2022. 2

[22] Keunhong Park, Utkarsh Sinha, Jonathan T. Barron, Sofien Bouaziz, Dan B Goldman, Steven M. Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable neural radiance fields. *ICCV*, 2021. 6, 8

[23] Keunhong Park, Utkarsh Sinha, Peter Hedman, Jonathan T. Barron, Sofien Bouaziz, Dan B Goldman, Ricardo Martin-Brualla, and Steven M. Seitz. Hypernerf: A higher-dimensional representation for topologically varying neural radiance fields. *ACM Trans. Graph.*, 40(6), 2021. 3, 6, 8

[24] Sungheon Park, Minjung Son, Seokhwan Jang, Young Chun Ahn, Ji-Yeon Kim, and Nahyup Kang. Temporal interpolation is all you need for dynamic neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4212–4221, 2023. 2

[25] Sergey Prokudin, Qianli Ma, Maxime Raafat, Julien Valentin, and Siyu Tang. Dynamic point fields. *arXiv preprint arXiv:2304.02626*, 2023. 3

[26] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10318–10327, 2021. 5, 6, 7

[27] Helge Rhodin, Nadia Robertini, Christian Richardt, Hans-Peter Seidel, and Christian Theobalt. A versatile scene model with differentiable visibility applied to generative pose estimation. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015. 2

[28] Darius Rückert, Linus Franke, and Marc Stamminger. Adop: Approximate differentiable one-pixel point rendering. *arXiv preprint arXiv:2110.06635*, 2021. 3

[29] Ruizhi Shao, Zerong Zheng, Hanzhang Tu, Boning Liu, Hongwen Zhang, and Yebin Liu. Tensor4d: Efficient neural 4d decomposition for high-fidelity dynamic reconstruction and rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16632–16642, 2023. 3

[30] Carsten Stoll, Nils Hasler, Juergen Gall, Hans-Peter Seidel, and Christian Theobalt. Fast articulated motion tracking using a sums of gaussians body model. In *2011 International Conference on Computer Vision*, pages 951–958. IEEE, 2011. 2

[31] Cheng Sun, Min Sun, and H Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. 2022 ieee. In *CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5449–5459, 2021. 2

[32] Edgar Tretschk, Ayush Tewari, Vladislav Golyanik, Michael Zollhöfer, Christoph Lassner, and Christian Theobalt. Non-rigid neural radiance fields: Reconstruction and novel view synthesis of a dynamic scene from monocular video. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 12939–12950, 2021. 3

[33] Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Wang Xinggang. 4d gaussian splatting for real-time dynamic scene rendering. *arXiv preprint arXiv:2310.08528*, 2023. 3, 6

[34] Tianhao Wu, Fangcheng Zhong, Andrea Tagliasacchi, Forrester Cole, and Cengiz Oztireli. Dˆ 2nerf: Self-supervised decoupling of dynamic and static objects from a monocular video. *Advances in Neural Information Processing Systems*, 35:32653–32666, 2022. 3

[35] Yiheng Xie, Towaki Takikawa, Shunsuke Saito, Or Litany, Shiqin Yan, Numair Khan, Federico Tombari, James Tompkin, Vincent Sitzmann, and Srinath Sridhar. Neural fields in visual computing and beyond. *Computer Graphics Forum*, 2022. 1

[36] Qiangeng Xu, Zexiang Xu, Julien Philip, Sai Bi, Zhixin Shu, Kalyan Sunkavalli, and Ulrich Neumann. Point-nerf: Point-based neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5438–5448, 2022. 3

[37] Zhiwen Yan, Chen Li, and Gim Hee Lee. Nerf-ds: Neural radiance fields for dynamic specular objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8285–8295, 2023. 6, 8

[38] Ziyi Yang, Xinyu Gao, Wen Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. Deformable 3d gaussians for high-fidelity monocular dynamic scene reconstruction, 2023. 3

[39] Zeyu Yang, Hongye Yang, Zijie Pan, Xiatian Zhu, and Li Zhang. Real-time photorealistic dynamic scene representation and rendering with 4d gaussian splatting, 2023. 3

[40] Wang Yifan, Felice Serena, Shihao Wu, Cengiz Öztireli, and Olga Sorkine-Hornung. Differentiable surface splatting for point-based geometry processing. *ACM Transactions on Graphics (proceedings of ACM SIGGRAPH ASIA)*, 38(6), 2019. 2, 3

[41] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. Plenoctrees for real-time rendering of neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5752–5761, 2021. 2

[42] Qiang Zhang, Seung-Hwan Baek, Szymon Rusinkiewicz, and Felix Heide. Differentiable point-based radiance fields for efficient view synthesis. *arXiv preprint arXiv:2205.14330*, 2022. 3

[43] Wojciech Zielonka, Timur Bagautdinov, Shunsuke Saito, Michael Zollhöfer, Justus Thies, and Javier Romero. Drivable 3d gaussian avatars. 2023. 3

[44] Matthias Zwicker, Hanspeter Pfister, Jeroen Van Baar, and Markus Gross. Surface splatting. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 371–378, 2001. 2, 4