Clustered Error Correction with Grouped 4D Gaussian Splatting

TAEHO KANG, Seoul National University, Republic of Korea
JAEYEON PARK, Seoul National University, Republic of Korea
KYUNGJIN LEE, Seoul National University, Republic of Korea
YOUNGKI LEE, Seoul National University, Republic of Korea

Existing 4D Gaussian Splatting (4DGS) methods struggle to accurately reconstruct dynamic scenes, often failing to resolve ambiguous pixel correspondences and inadequate densification in dynamic regions. We address these issues by introducing a novel method composed of two key components: (1) Elliptical Error Clustering and Error Correcting Splat Addition that pinpoints dynamic areas to improve and initialize fitting splats, and (2) Grouped 4D Gaussian Splatting that improves consistency of mapping between splats and represented dynamic objects. Specifically, we classify rendering errors into missing-color and occlusion types, then apply targeted corrections via backprojection or foreground splitting guided by cross-view color consistency. Evaluations on Neural 3D Video and Technicolor datasets demonstrate that our approach significantly improves temporal consistency and achieves state-of-the-art perceptual rendering quality, improving 0.39dB of PSNR on the Technicolor Light Field dataset. Our visualization shows improved alignment between splats and dynamic objects, and the error correction method's capability to identify errors and properly initialize new splats. Our implementation details and source code are available at https://github.com/tho-kn/cem-4dgs.

CCS Concepts: • **Computing methodologies** → **Rasterization**.

Additional Key Words and Phrases: Dynamic novel view synthesis, 3D Gaussian Splatting

ACM Reference Format:
Taeho Kang, Jaeyeon Park, Kyungjin Lee, and Youngki Lee. 2025. Clustered Error Correction with Grouped 4D Gaussian Splatting. In *SIGGRAPH Asia 2025 Conference Papers (SA Conference Papers '25), December 15–18, 2025, Hong Kong, Hong Kong*. ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3757377.3763858

1 Introduction

As augmented and virtual reality gain more attention, immersive content creation becomes more important. One way to provide hyperrealistic content is to incorporate the real world into computer graphics. Traditional 3D representations, such as meshes or voxel grids, are effective for handcrafted 3D models; however, they have limitations for reconstructing complex real-world scenes from images. In recent years, NVS (Novel View Synthesis) has become a popular method for the hyperrealistic representation of scenes. NeRF (Neural Radiance Field) [Mildenhall et al. 2020] showed promising results in synthesizing views in extremely high quality, given camera transformation in the scene with neural representation. Later efforts improved its efficiency [Chen et al. 2022; Müller et al. 2022] and rendering quality [Barron et al. 2021]. 3DGS (3D Gaussian Splatting) [Kerbl et al. 2023] has gained attention due to its efficiency, allowing real-time rendering. The original 3DGS is limited to synthesizing novel views for a static scene. Many efforts expand the 3DGS methods into 4DGS for dynamic scenes prevalent in the real world [Bae et al. 2024a; Guo et al. 2024; Lee et al. 2024b; Li et al. 2024a; Wu et al. 2024; Yang et al. 2024b].

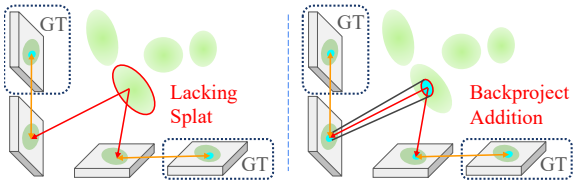

Fig. 1. Failure of densification results in a lack of splat for the blue colored area on the green surface. By backprojecting the elliptical error cluster's ground truth color, we effectively add a Gaussian splat that corrects pixel error.

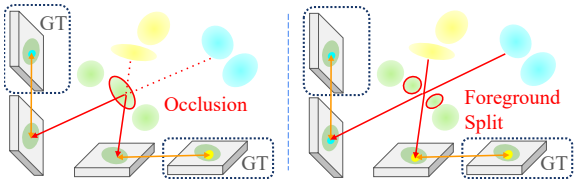

Fig. 2. Due to occluding splat, there is a green visual error where ground truth colors should be yellow and blue, respectively. Splitting and optimizing the occluding foreground splat to cover only the correct area makes occluded splats with ground-truth colors visible from both views.

Fig. 3. Looking at the image as a whole, the correspondence between stripes is apparent between different frames. However, focusing on a small area like splats, the correspondence between them becomes ambiguous, as shown in the red box in the same coordinate. A splat to represent part of a stripe can be mapped to its neighboring stripe with a dynamic transform during the optimization process.

Authors' Contact Information: Taeho Kang, Seoul National University, Seoul, Republic of Korea, taeho.kang@hcs.snu.ac.kr; Jaeyeon Park, Seoul National University, Seoul, Republic of Korea, jypark@snu.ac.kr; Kyungjin Lee, Seoul National University, Seoul, Republic of Korea, kjlee818@gmail.com; Youngki Lee, Seoul National University, Seoul, Republic of Korea, youngki.lee@gmail.com.

SA Conference Papers '25, Hong Kong, Hong Kong

ACM ISBN 979-8-4007-2137-3/25/12
https://doi.org/10.1145/3757377.3763858

Despite the effort, several challenges remain for dynamic scene NVS (Novel View Synthesis). The densification strategy is not practical for dynamic areas in 4D Gaussian Splatting. In particular, dynamic details are not effectively reconstructed because of the lack of splats. 3D Gaussian Splatting uses the average viewpoint gradient to determine which splats to densify. However, the extension to a 4D scene causes the appearance of the splat to vary over time, and the vanilla approach does not pinpoint the dynamic area to densify, as it is significantly impacted by temporal transform error and visibility changes. Furthermore, optimizing time-variant splats for details of a dynamic object is not trivial. Each splat covers a small area with one color, and there are multiple possible correspondences with similarly colored pixels near each other in adjacent frames, as shown in Fig. 3. Extending the dynamic splat temporally with the correct motion is challenging with this ambiguity. Previous efforts, such as Embedding-Based Deformable 3D Gaussian Splatting (E-D3DGS) [Lee et al. 2024b] and Spacetime Gaussians (STG) [Li et al. 2024a], have attempted to mitigate these issues by adapting multiview DSSIM loss for densification or uniformly sampling erroneous patches. However, these strategies lack precise localization of error regions and explicit modeling of splat correspondences across frames, which are crucial for detailed and temporally stable reconstruction. Consequently, dynamic scenes rendered by these methods still exhibit noticeable flickering and compromised detail reconstruction, particularly in highly dynamic areas or around fine textures.

To explicitly address these limitations, we propose a novel method that consists of two complementary components: (1) an elliptical clustering method that accurately identifies and localizes high-error pixel regions for precise injection of Gaussian splats, and (2) a group-based dynamic splat modeling approach that ensures temporal consistency by enforcing shared motion transforms across grouped splats. Our elliptical clustering enables accurate pinpointing of regions requiring correction, effectively reducing redundant splats and enhancing visual accuracy. Simultaneously, our group-based modeling approach resolves the ambiguity of splat correspondences, significantly reducing temporal jitter by ensuring consistent splat tracking across frames (see Fig. 8).

Specifically, we first detect and cluster rendering errors into elliptical shapes by comparing the rendered frames with ground-truth reference views. Each error cluster is further categorized into missing-color or occlusion errors by performing cross-view color consistency analysis. For missing-color errors, new splats are injected through targeted backprojection; occlusion errors are corrected by foreground splat splitting to resolve depth inaccuracies. These targeted correction strategies substantially improve spatial precision and perceptual quality, rendering dynamic scene details more faithfully.

Experiments on widely used benchmarks, including Neural 3D Video [Tianye Li 2022] and Technicolor [Sabater et al. 2017] datasets, demonstrate that our proposed method achieves state-of-the-art perceptual quality, significantly outperforming existing approaches in metrics such as PSNR, DSSIM, and LPIPS. Furthermore, our temporal modeling consistently enhances temporal stability and reduces visual artifacts, clearly advancing dynamic scene rendering. We will publicly release our implementation and detailed source code to support future research and applications. Our main contributions are the following:

- We introduce a pixel error clustering approach that accurately identifies the elliptical region to correct with splats.
- We propose a dynamic splats grouping that clarifies the correspondence between dynamic areas across different frames.
- We identify two types of errors in the details of 4DGS unresolved by densification and devise a multi-view driven method to pinpoint coordinates to introduce new splats.
- Our method achieves state-of-the-art visual quality in Neural 3D Video and Technicolor datasets.

2 Related Works

2.1 Novel View Synthesis for Static Scenes

Traditional approaches solve for precise geometries [Furukawa and Ponce 2006; Schönberger and Frahm 2016], the novel view rendering of meshes or point clouds generated by MVS has limited accuracy. For high-quality NVS, image-based rendering, given dense image samples, was a popular approach.

NeRF [Mildenhall et al. 2020] is the pioneering work that initiated this trend, using a multilayer perceptron (MLP) to learn the radiance field and volumetric rendering to synthesize images from any viewpoint. Subsequent work has aimed to boost the quality and efficiency of differentiable-volume rendering. One direction focuses on improving the sampling strategy to reduce the number of point queries [Müller et al. 2022; Neff et al. 2021; Piala and Clark 2021]. Another group incorporates explicit and localized neural representations to enhance model compactness or reduce rendering time [Barron et al. 2023; Chen et al. 2022; Hu et al. 2023; Liu et al. 2020; Reiser et al. 2021; Suhail et al. 2022; Yu et al. 2022, 2021].Using image-based priors or depth information has proven effective in NeRF-based models. For example, Roessle et al. [Roessle et al. 2022] and NerfingMVS [Wei et al. 2021] use a pre-trained network to estimate the dense depth from the sparse SfM points and supervise the NeRF model with the estimated depth. MonoPatchNeRF[Wu et al. 2025] integrates patch-based monocular priors for more precise handling of depth and normal maps.

As another significant stream of novel view synthesis, 3D Gaussian Splatting [Kerbl et al. 2023] (3DGS) represents a scene using a 3D Gaussian collection. 3D coordinates, color, size, density, orientation, and spherical harmonics characterize each Gaussian in 3DGS. These parameters are optimized using an efficient differentiable rasterization algorithm, starting from a set of Structure-from-Motion (SfM) points to represent a given 3D scene best.

2.2 Novel View Synthesis for Dynamic Scenes

Synthesizing dynamic scenes introduces additional complexities, notably the need to capture temporal coherence across frames accurately. Dynamic extensions of NeRF [Mildenhall et al. 2020] (e.g., Nerfies [Park et al. 2021a], HyperNeRF [Park et al. 2021b], D-NeRF [Pumarola et al. 2021]) use deformation networks and latent space embeddings to represent time-varying geometry and appearance, or combines NeRF with time-conditioned latent codes to represent dynamic scenes. [Li et al. 2022, 2021]. More recent approaches like K-Planes [Fridovich-Keil et al. 2023], HexPlane [Cao

and Johnson 2023], and Tensor4D [Shao et al. 2023] factorize the 4D spacetime domain into 2D feature planes for a more compact model size.

Efforts have also been made to extend 3D Gaussian Splatting (3DGS) into dynamic scenes [Wu et al. 2024; Yang et al. 2024b]. Some approaches focus on multi-view images from the dynamic monocular or stereo camera [Huang et al. 2024; Liang et al. 2025; Lin et al. [n. d.]; Lu et al. 2024], such as the D-NeRF [Pumarola et al. 2021] and HyperNeRF [Park et al. 2021b] datasets. SC-GS [Huang et al. 2024] and HiMoR [Liang et al. 2025] use interpolation of multiple control points to represent dynamicity. Others focus on compressing the 4D Gaussian Splatting representation for storage efficiency [Lee et al. 2024a]. Multi-camera setups have been widely explored. Early approach fixes the Gaussians' number and properties and optimizes their position and orientation [Luiten et al. 2024]. 4DGaussians [Wu et al. 2024] and D3DGS [Yang et al. 2024a] reconstruct scenes using 3D Gaussians with a deformation field, while E-D3DGS [Bae et al. 2024b] further defines deformations as functions of per-Gaussian and temporal embeddings. ST-4DGS [Li et al. 2024b] enhances the deformation-based approach by incorporating temporal shape regularization and temporal-aware density control. The other direction is to use a Gaussian distribution in the 4D space and slice it in the time axis to get the 3D distribution [Duan et al. 2024; Yang et al. 2024b]. STG [Li et al. 2024a] proposed modeling dynamicity with temporal opacity, parametric motion/rotation, and splatted feature rendering. SWinGS [Shaw et al. 2024] uses sliding-window optimization with adaptively sampled windows. Ex4DGS [Lee et al. 2024b] explicitly extracts dynamic Gaussian splats and models its motion with a keyframe interpolation-based dynamic transform. Concurrent work uses sparse 4D grid anchors [Cho et al. 2024] and disentangles temporal components from a 4D Gaussian distribution [Feng et al. 2025].

Some prior works [Li et al. 2024b; Luiten et al. 2024] use local rigidity loss for effective temporal modeling. In our experiment with similar loss, the higher weights degraded the visual quality, while the lower weights did not produce a consistent object-aligned motion. Our method improves the quality of 4D Gaussian Splatting by modeling more accurate correspondences of dynamic objects across frames using shared dynamic properties for splats and splitting the group carefully.

2.3 Gaussian Splatting Densification Strategies

Densification in 3DGS refers to the process of adding Gaussians to enhance the scene representation. One of the challenges with 3DGS is that it can excessively increase the number of Gaussians without explicitly constraining the real geometric structure, leading to numerous redundant Gaussians and significant memory consumption.

Recent methods aim to optimize densification in 3DGS by reducing redundancy. For instance, some approaches transform 3D Gaussians into a more compact format by minimizing the number of Gaussians without compromising visual quality [Fan et al. 2024; Morgenstern et al. 2025]. Another approach employs a progressive propagation strategy to guide the densification of 3D Gaussians, leveraging priors from existing reconstructed geometries and patch-matching techniques to produce new Gaussians [Cheng et al. 2024]. Some approaches consider pixel error for densification. One accumulates pixel errors that correspond to splats instead of viewpoint gradient [Rota Bulò et al. 2025]. Pixel-GS rescales the gradient used for densification using the number of pixels each splat covers [Zhang et al. 2025]. However, these methods are restricted to static scenes and do not apply to dynamic objects. Our proposed method complements the limitations of naive 3DGS densification for the 4D Gaussian Splatting setup, and improves the temporal fidelity of dynamic 3DGS methods while maintaining efficiency.

For dynamic scenes, most methods use densification similar to 3DGS with a gradient from L1 and DSSIM loss [Lee et al. 2024b; Li et al. 2024a; Wu et al. 2024; Yang et al. 2024b]. STG [Li et al. 2024a] additionally samples new Gaussians along the rays of patches of pixels with large errors to improve faraway static object quality. E-D3DGS [Bae et al. 2024b] periodically adds a multi-view DSSIM loss to induce Gaussian densification. They only indirectly determine where more Gaussian splats are needed to improve quality with simple approaches. Our method pinpoints where and what Gaussian splats should be added to correct the error in the rendered frame, and expands it temporally following the existing splat group's motion.

3 Preliminary

3.1 3D Gaussian Splatting

3D Gaussian splatting represents scenes using sparse 3D Gaussian distributions, allowing for efficient and high-quality rendering of static 3D scenes. Each Gaussian is defined by a covariance matrix $\Sigma \in \mathbb{R}^{3\times3}$ and a mean $\mu \in \mathbb{R}^3$. The Gaussian function is given by:

$$\mathbf{G}(\mathbf{x}) = e^{-\frac{1}{2}(\mathbf{x}-\mu)^\top \Sigma^{-1}(\mathbf{x}-\mu)}, \quad where \quad \Sigma = RSS^\top R^\top. \tag{1}$$

To ensure the positive semi-definiteness of Σ and simplify the learning process, Σ is decomposed into a scaling matrix S and a rotation matrix R. Rendering from a specific viewpoint involves transforming Σ into the camera coordinates using the viewing transformation matrix W and the Jacobian J of the affine approximation of the projective transformation:

$$\Sigma' = JW\Sigma W^\top J^\top. \tag{2}$$

Each 3D Gaussian is parameterized by position $\mathbf{x} \in \mathbb{R}^3$, color defined by spherical harmonics coefficients $\mathbf{c} \in \mathbb{R}^k$ (where k means the number of SH coefficients), rotation $\mathbf{r} \in \mathbb{R}^4$ represented by quaternion, scale $\mathbf{s} \in \mathbb{R}^3$, and opacity $o \in \mathbb{R}$. The color C of each pixel($\mathbf{p}$) is determined using point-based alpha blending, influenced by the Gaussians overlapping that pixel:

$$\mathbf{C}(\mathbf{p}) = \sum_{i=1}^{N} \mathbf{T}_i \alpha_i \mathbf{c}_i, \quad \text{where} \quad \alpha_i = \sigma_i \mathbf{G}_i(\mathbf{x}),\ \mathbf{T}_i = \prod_{j=1}^{i-1}(1-\alpha_j) \tag{3}$$

where α_i represents the density of each Gaussian, $\mathbf{T}_i$ is the transmittence, and $\mathbf{c}_i$ is the color of gaussians overlapped with the pixel.

3.2 Explicit 4D Gaussian Splatting

We choose Fully Explicit 4D Gaussian Splatting [Lee et al. 2024b] as a baseline, explicitly distinguishing scenes between static and dynamic splats.

Static splats are modeled with a simple linear displacement $\mathbf{d} \in \mathbb{R}^3$ in addition to the vanilla 3D Gaussian Splats. The position of the static splat at time t is:

$$\mathbf{x_s}(t) = \mathbf{x_s}(0) + t * \mathbf{d} \tag{4}$$

After optimization, splats for the dynamic areas have high displacement. They are periodically extracted and promoted to dynamic splats.

Dynamic splats have four dynamic properties in addition to 3DGS properties: position $\mathbf{x_d} \in \mathbb{R}^{3\times K}$, rotation $\mathbf{r_d} \in \mathbb{R}^{4\times K}$ replacing static counterparts, and time-variant opacity weight center $\mathbf{o_c} \in \mathbb{R}^2$ and variance $\mathbf{o_v} \in \mathbb{R}^2$, where K is the number of keyframes. The position and rotation are explicitly given as values for the keyframes every 10 frames, and the cubic Hermite spline (CHip) and slerp [Shoemake 1985] interpolation are used for position and rotation in the intermediate frames. Dynamic splat's opacity is a multiplication of opacity o and opacity weight $o_w(t)$ parameterized by $\mathbf{o_c}$ and $\mathbf{o_v}$. The piecewise function, composed of two Gaussian segments and a constant segment, models the temporal behavior of a dynamic object: the Gaussian segments capture its appearance and disappearance with individual centers and variances. In contrast, the constant segment between Gaussian segments represents its stable existence. The supplementary material provides further details of this formulation.

4 Method

4.1 Overview

Our overall framework is shown in Fig. 4. Our method consists of three main components: 4D Gaussian Splatting with shared dynamic transform for a group of splats (Sec. 4.2), clustering erroneous pixels to extract effective additional splat candidates (Sec. 4.3), and finally multi-view comparison to add splats by foreground splitting or backproject addition (Sec. 4.4).

We train a Grouped 4D Gaussian Splatting model, following the progressive learning approach of Ex4DGS [Lee et al. 2024b], starting from the first few frames and gradually expanding the model's timespan to encompass the full timespan. Group splitting, 3DGS-like densification, and pruning are done on this stage. In the second step, our pipeline focuses on correcting the errors of the trained Grouped 4D Gaussian Splatting model, with Elliptical Error Clustering and Error-Correcting Splat Addition triggered every few hundred optimization steps. Dynamic and static parameters of groups and splats are still optimized; however, the splat's group membership is frozen in this stage.

4.2 Grouped 4D Gaussian Splatting

4.2.1 Grouped Temporal Modeling. We decompose the transform of each splat into two components: a shared dynamic transform at the group level and a relative static transform at the splat level. Specifically, we assign a group-level position and rotation over time, denoted by $\mathbf{x}_\mathcal{G}(t)$ and $\mathbf{r}_\mathcal{G}(t)$. Each splat has position $\mathbf{x}$, rotation $\mathbf{r}$ relative to the group transform, and a displacement vector $\mathbf{d}$, defined in the global frame.

The group motion is represented using K keyframes: $\mathbf{x}_\mathcal{G} \in \mathbb{R}^{3\times K}$ and $\mathbf{r}_\mathcal{G} \in \mathbb{R}^{4\times K}$, which are interpolated across time to produce continuous motion. The final position and orientation of a splat $g \in \mathcal{G}$ at time t are:

$$\mathbf{x}(t) = \mathbf{x}_\mathcal{G}(t) + \mathbf{R}_\mathcal{G}(t)\mathbf{x} + t \cdot \mathbf{d}, \tag{5}$$

$$\mathbf{R}(t) = \mathbf{R}_\mathcal{G}(t)\mathbf{R}, \tag{6}$$

where $\mathbf{R}_\mathcal{G}(t)$ and $\mathbf{R}$ are the rotation matrices corresponding to quaternions $\mathbf{r}_\mathcal{G}(t)$ and $\mathbf{r}$, respectively.

4.2.2 Dynamic and Static Splats. Static splats are implemented by fixing their group transform to the global frame, so their relative transform $\mathbf{x}$, $\mathbf{r}$ is effectively global. In this case, the overall parameterization reduces exactly to Ex4DGS's static formulation with linear motion via $\mathbf{d}$. Dynamic splats additionally have temporal opacity parameters o_c (center) and o_v (variance), which control visibility across time.

4.2.3 Graph-based Dynamic Grouping. We now have to determine which subsets of splats show consistent global motion and should share a group-level transformation. At initialization, all splats are treated as static. As training progresses, we identify splats that move independently from their group. These are detected based on their displacement vectors $\mathbf{d}$, which capture motion deviating from the shared group trajectory. We train splats with the regularization term similar to Ex4DGS [Lee et al. 2024b], splats with consistent movements are discouraged from having large displacements. Thus, splats exhibiting a large displacement $\mathbf{d}$ are identified as candidates for a new group.

We construct an undirected graph over these large displacement splats within a group. An edge connects splats i and j when they satisfy two conditions: spatial overlap, and their learned displacements are coherent beyond a threshold τ_d, measured by cosine similarity. Denoting by $\mathbf{x}_i(t) \in \mathbb{R}^3$ the center of splat i at a selected largest mean loss timestamp for group splitting, and by $s(t)$ its effective size, the distance where opacity falls below threshold with Gaussian function along its major axis, the conditions for an edge are:

$$\|\mathbf{x}_i(t) - \mathbf{x}_j(t)\|_2 < s_i + s_j \quad \text{and} \quad \frac{\mathbf{d}_i^\top \mathbf{d}_j}{\|\mathbf{d}_i\|\|\mathbf{d}_j\|} \geq \tau_d, \tag{7}$$

Upon identification of such connected components, a new dynamic group is formed. The dynamic transform of the split group is initialized by selecting one representative splat from the component. Its relative transform and displacement are merged, and its final transform at keyframes is set as the initial keyframe positions and rotations of the newly formed dynamic group. Subsequently, the inverse of the selected splat's relative transform is applied to the positions and rotations of all splats within this new group to appropriately re-align them relative to the new group's dynamic transform. Furthermore, the chosen splat's displacement is subtracted from the displacements of the splats in the new group, as this motion is now absorbed by the group's shared transform.

4.3 Elliptical Error Clustering

Our method improves reconstruction fidelity by first clustering erroneous pixels in the image space into elliptical regions. This stage is critical for accurate and efficient Gaussian splat updates, as elliptical error regions can be directly mapped into additional

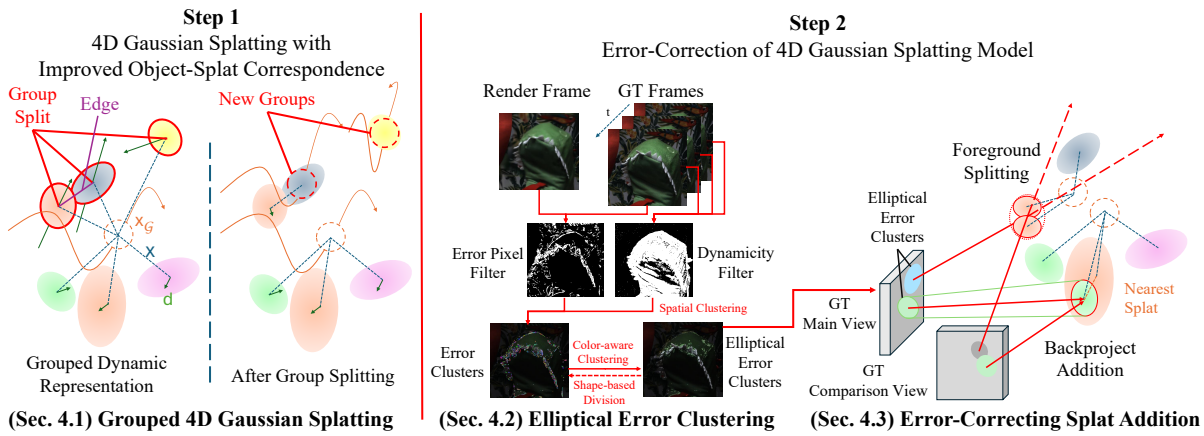

Fig. 4. Overall framework of our method. We aim to improve the rendering quality by introducing dynamic splats that correct observed pixel errors in the training set. Grouped 4DGS uses shared dynamic transforms for a group of dynamic splats to improve temporally consistent correspondence between splats and dynamic scene objects. (Sec. 4.2) Elliptical Error Clustering extracts error pixels and clusters pixels into an elliptical shape, ready for backprojection to initialize a Gaussian splat. (Sec. 4.3) Error-Correcting Splat Addition corrects the pixel error of clusters caused by occlusion and lack of splats, by projecting the erroneous point's current depth to another view, comparing the ground-truth color, and applying foreground splitting or backproject addition. (Sec. 4.4).

ellipsoid-shaped Gaussian splats in world space. This section details our clustering strategy, and our splat addition method is introduced in the following section (Sec 4.4). Our method consists of two steps: i) identifying erroneous pixels using pixel dynamicity and mean RGB error, ii) a recursive clustering and fitting to split them into Gaussian splat candidates.

4.3.1 Erroneous Pixel Identification. In every few hundred training iterations, we randomly select a viewpoint and compare the rendered result to the ground truth image to identify erroneous regions. These regions are defined using two key criteria: pixel dynamicity and mean RGB error. To focus on dynamic regions, we compute each pixel's dynamicity as the maximum L1 distance of its ground truth RGB values across the previous, current, and next frames. Pixels exceeding the dynamicity threshold τ_D are retained. Among these dynamic pixels, we further isolate error-prone areas by applying both an absolute RGB error threshold τ_a and a relative threshold τ_r, which selects the top percentile of high-error pixels. This filtering process yields a refined set of erroneous pixels suitable for clustering and correction.

4.3.2 Recursive Pixel Clustering. Filtered pixels are first split into clusters with DBSCAN based on spatial locality. Then, recursive clustering splits them until they are close enough to an ellipse shape. The recursive process consists of: a) pixel clustering by position and color similarity, b) ellipse fitting of clustered pixels. If the ellipse fitting fails, recursion occurs with each cluster split by the first step. This recursion continues until all clusters are a valid ellipse fit.

Initially, we apply spatial clustering using the DBSCAN (Density-Based Spatial Clustering of Applications with Noise) algorithm [Ester et al. 1996], which groups pixels based on local density and can discover clusters of arbitrary shape without requiring the number of clusters to be specified in advance. After obtaining the initial set, we further refine it using color information. These initial clusters go through the recursion process below.

First, we combine spatial and color information to group the selected pixels into coherent error clusters. Intuitively, nearby pixels with similar colors are more likely to belong to the same object undergoing consistent motion. We use the DBSCAN for spatial-color clustering, which considers both pixel coordinates and the RGB values (scaled by a constant). This ensures that the final clusters are both spatially coherent and color-consistent, which is essential for fitting elliptical Gaussian splats.

Secondly, for each error cluster obtained from the previous stage, we fit an ellipse to assess its suitability for Gaussian splat correction. Specifically, we compute the minimum-area bounding rectangle of the cluster and evaluate how well the cluster fills an ellipse inscribed within this rectangle. A cluster is accepted as a final elliptical error cluster if its filled area ratio exceeds a predefined threshold, indicating a strong resemblance to an ellipse.

Clusters that fail this test are split into two using K-means clustering on the spatial coordinates. These clusters are recursively fed back into DBSCAN spatial-color clustering and re-evaluated for ellipse fitting to achieve more compact and dense groupings. This iterative process continues until all clusters are accepted. The final set of accepted clusters is parameterized as an ellipse, with their representative color selected from their center pixel for simplicity, as clusters are already color-homogeneous.

4.4 Error-Correcting Splat Addition

This section details our approach to correcting elliptical error clusters by adding or refining existing splats, as depicted in Fig. 1, 2. This process targets rendering inaccuracies where the underlying object surface is largely well-reconstructed, but quality suffers from either missing color information or over-reconstructing occlusions. We take advantage of cross-view comparison to diagnose two primary error types. We sample depths from its center and nearby pixels on the rendered depth map (i.e., alpha-blended depth) for each elliptical cluster, back-projecting them to 3D points. Each 3D point is then

projected onto a *comparison view* (from a different camera but the same frame as the *main view* where the error cluster was identified). By comparing the ground-truth pixel colors at the projected 3D location across the main and comparison views, we can distinguish the type of error and devise a targeted fixing strategy.

4.4.1 Types of Errors. Lacking Splats (Missing Color) error occurs when the 3D surface is adequately reconstructed, but correct color information is absent. The defining characteristic is when the ground truth color from the main view largely matches that from the comparison view. This suggests surface consistency across views but inaccurate coloring. *Occlusion (Over-reconstruction)* arises when foreground splats are excessively reconstructed, incorrectly obscuring an object that should be visible. Here, the ground truth color from the main view differs from the comparison view's. This discrepancy implies that the 3D position is not accurately represented, as different background content should be visible from distinct viewpoints.

4.4.2 Identifying Error Types. To apply the appropriate correction, we extract k depth samples $\{z_j\}_{j=1}^{k}$ from an $n \times n$ kernel around the ellipse center on the rendered depth map, back-projecting them to 3D points $\{\mathbf{p}_j\}_{j=1}^{k}$. Each $\mathbf{p}_j$ is then projected to the comparison view for its corresponding ground truth color $\mathbf{c}_{\text{comp, j}}$. Let $\mathbf{c}_{\text{main, j}}$ be the ground truth color at the error cluster's ellipse center pixel in the main view.

We apply backproject addition if the following condition, indicating high color similarity across views (thus, a lacking splat error), is met:

$$\min_{j=1,\ldots,k} \left\|\mathbf{c}_{\text{main, j}} - \mathbf{c}_{\text{comp, j}}\right\|_{\infty} < \delta_{\text{rgb}}, \tag{8}$$

Here, $\|\cdot\|_{\infty}$ indicates the maximum element-wise difference. Otherwise, foreground splitting is performed, addressing an occlusion error.

4.4.3 Correcting Error. When the color similarity condition is satisfied, a new splat is initialized to correct the missing color at the 3D point $\mathbf{p}_k$ (corresponding to depth z_k) that minimizes $\left\|\mathbf{c}_{\text{main, j}} - \mathbf{c}_{\text{comp, j}}\right\|_{\infty}$. This new splat is attached to the nearest existing splat's group, leveraging the grouped dynamic transform for temporal generalization. Its rotation is initialized to align two axes with the ellipse cluster's projected axes and the other to be perpendicular to it. Its scale is set to create a disk-like shape, covering the ellipse along two axes with a very small scale along the third, ensuring minimal interference with other views at initialization while effectively correcting the error. The splat's opacity is set to $1 - \min_{j=1,\ldots,k} \left\|\mathbf{c}_{\text{main, j}} - \mathbf{c}_{\text{comp, j}}\right\|_{\infty}$, allowing other splats to contribute to the remaining difference, and its temporal opacity weight is initialized to be maximal at the selected view's timestamp.

If the color similarity condition is not met, indicating an occlusion error, the splat nearest to the 3D point at the current depth is split. This process is analogous to the densification strategy employed in traditional 3D Gaussian Splatting, effectively refining the problematic region by dividing an existing splat to represent the scene's geometry better and alleviate over-reconstruction.

This addition requires the back-projected 3D point to be visible from the comparison view, reducing the possibility of additional floater overfitting to a specific view. The process is performed based on individual frames and depends on the generalizability of Grouped Temporal Modeling across time. However, each splat has an independent opacity timespan. Splats that fail to generalize across neighboring frames using grouped motion naturally shrink in temporal extent or are split into new groups.

Table 1. Quantitative comparisons on the Technicolor Light Field dataset.

Method	PSNR ↑	DSSIM_1 ↓	DSSIM_2 ↓	LPIPS ↓	Size ↓
DyNeRF [Li et al. 2022]	31.80	N/A	0.021	0.140	**30 MB**
HyperReel [Attal et al. 2023]	32.73	0.047	N/A	0.109	60 MB
4DGS [Yang et al. 2024b]	29.54	0.065	0.032	0.149	N/A
4DGaussians [Wu et al. 2024]	30.79	0.079	0.040	0.178	N/A
STG [Li et al. 2024a]	33.56	**0.040**	0.019	**0.084**	**55 MB**
SWinGS [Shaw et al. 2024]	**33.65**	**0.033**	N/A	0.117	N/A
E-D3DGS [Bae et al. 2024a]	33.24	0.047	N/A	0.100	77 MB
Ex4DGS [Lee et al. 2024b]	33.62	0.042	0.019	0.088	144 MB
Ours	**34.04**	**0.040**	**0.018**	**0.081**	177 MB

5 Experiments

We conducted experiments on the Neural 3D Video and Technicolor dataset, a commonly used benchmark dataset for dynamic novel-view synthesis from multiple cameras. In a similar setup listed in the table, we compared our method with other methods.

We use representative perceptual quality metrics for quantitative evaluation, PSNR, DSSIM, and LPIPS. Some works use a different setup to measure DSSIM, using *scipy*'s function with *data_range* value 1.0 and 2.0 as pointed out in prior works [Attal et al. 2023; Fridovich-Keil et al. 2023; Lee et al. 2024b; Li et al. 2024a]. We show them as DSSIM_1 and DSSIM_2 in tables respectively. The model size is for all 50 frames in the Technicolor Light Field dataset and 300 frames in the Neural 3D Video dataset.

5.1 Comparison to State-of-the-arts

5.1.1 Technicolor Light Field Dataset. The Technicolor Light Field dataset contains scenes with dynamic objects covering a large portion of the video, compared to the Neural 3D Video dataset. Our method significantly outperforms previous approaches in this dataset. Following previous works [Attal et al. 2023; Li et al. 2024a], we use 5 scenes from the dataset (Painter, Fabien, Theater, Train, Birthday) at an original image resolution of 2024 by 1088. The dataset is captured with 4 by 4 camera grid. We select 50 frames from each scene and choose a camera at the second row, second column as a test set, following convention from HyperReel [Attal et al. 2023].

Table 1 shows the quantitative comparison of our method on the Technicolor Light Field dataset [Sabater et al. 2017], with metrics for other methods [Attal et al. 2023; Lee et al. 2024b; Li et al. 2022, 2024a] with metrics from prior works [Lee et al. 2024b; Li et al. 2024a] for methods without official results. Our approach significantly improves over the state-of-the-art while being second in terms of DSSIM_1. These remarks reflect our method's strong capability to improve quality for highly dynamic and complex objects, such as in the Technicolor dataset.

5.1.2 Neural 3D Dataset. Neural 3D Video dataset [Tianye Li 2022] is a widely used dataset for multi-view novel view synthesis with 18

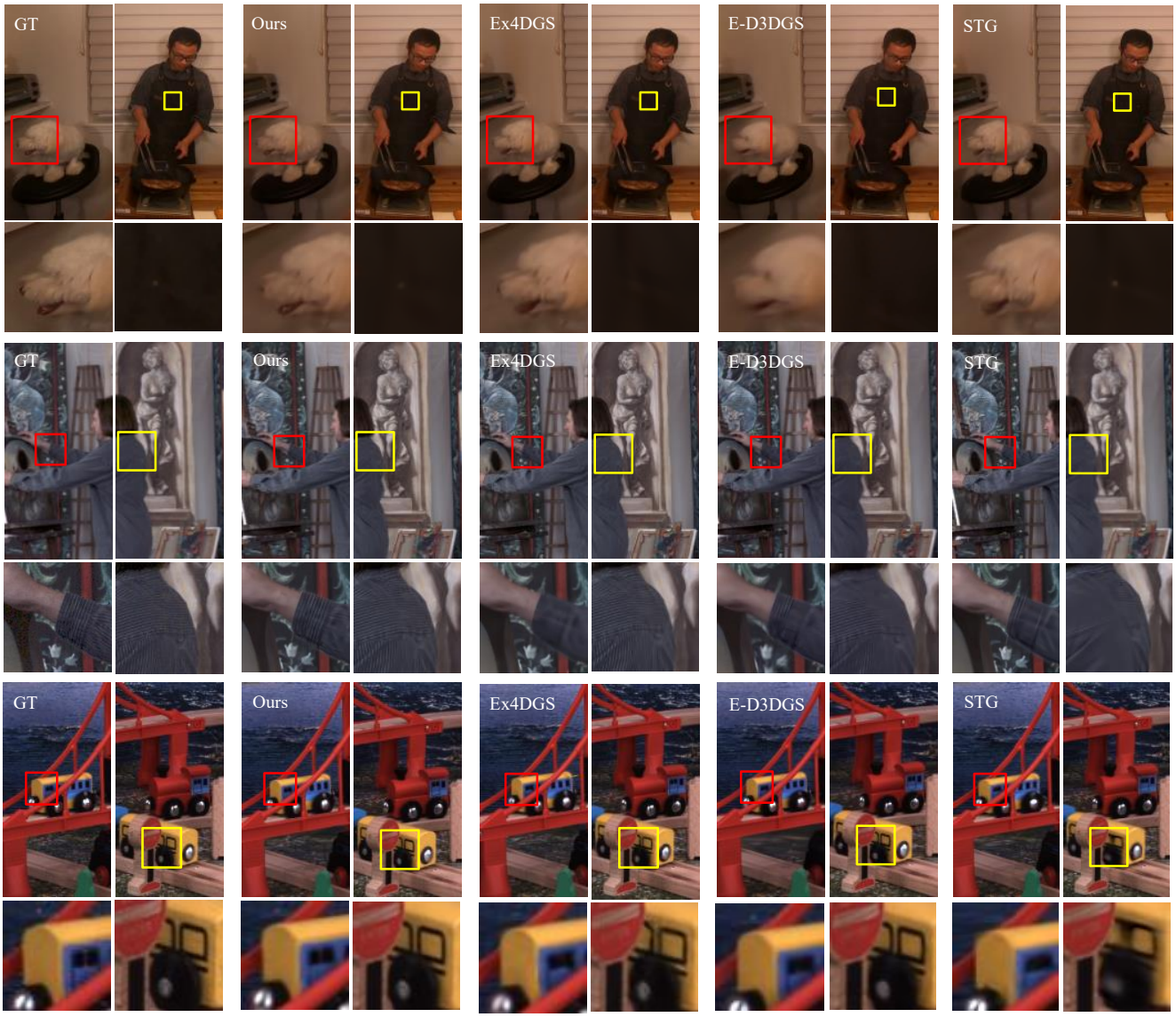

Fig. 5. Qualitative comparison of our method with state-of-the-art methods. From the top row, we show examples from the Sear Steak, Painter, and Train scenes from two datasets. The red and yellow box highlights the dynamic area with details, which is shown in a larger size below. For the Painter scene, brightness is adjusted from the ground-truth and rendering result for better visibility. The difference is pronounced in clearer teeth and a button, stripes in clothing, and sharper car window boundaries.

to 21 cameras. The dataset contains video of original resolution of 2704 by 2028, and downsampled to a half resolution as convention. We follow the recent convention [Lee et al. 2024b] for processing the dataset. Some prior works [Cao and Johnson 2023; Li et al. 2022; Lin et al. 2023; Lombardi et al. 2019; Mildenhall et al. 2019; Xu et al. 2024] show the evaluation results with different sets of scenes. Thus, we also show metrics for a commonly used subset of the dataset. The scene set used to compute the metric is denoted with a superscript for those evaluations. Per-scene metric is available in the supplementary material.

Table 2 compares our method to various approaches. Despite a small dynamic area to improve, our method achieves the best PSNR and outperforms baseline Ex4DGS [Lee et al. 2024b] in visual quality, while keeping the benefit from using sparse COLMAP point cloud input from only one frame, minimizing the pre-processing time and burden of per-frame COLMAP reconstruction size. Our method is also comparable to the state-of-the-art techniques in other metrics. The file size remains similar to the baseline because of reduced dynamic parameters despite additional splats.

Table 2. Quantitative comparisons on the Neural 3D Video Dataset.[1]Only the *Flame Salmon* scene. [2]Except for the *Coffee Martini* scene. [3]Only the *Cut Roasted Beef* scene.

Method	PSNR ↑	$DSSIM_1$ ↓	$DSSIM_2$ ↓	LPIPS ↓	Size ↓
Neural Volumes[1] [Lombardi et al. 2019]	22.80	N/A	0.062	0.295	N/A
LLFF[1] [Mildenhall et al. 2019]	23.24	0.076	N/A	0.235	N/A
DyNeRF[1] [Li et al. 2022]	29.58	N/A	0.020	0.083	28 MB
Ours[1]	29.49	0.040	0.022	0.067	125 MB
HexPlane[2] [Cao and Johnson 2023]	31.71	N/A	N/A	0.075	200 MB
Ours[2]	32.86	0.025	0.013	0.044	114 MB
Im4D[3] [Lin et al. 2023]	32.58	N/A	N/A	N/A	N/A
4K4D[3] [Xu et al. 2024]	32.86	N/A	N/A	N/A	N/A
Ours[3]	33.78	0.022	0.011	0.040	119 MB
NeRFPlayer [Song et al. 2023]	30.69	0.034	N/A	0.105	5130 MB
HyperReel [Attal et al. 2023]	31.10	0.036	N/A	0.096	360 MB
K-Planes [Fridovich-Keil et al. 2023]	31.63	0.041	N/A	0.110	311 MB
MixVoxels-L [Wang et al. 2023]	31.34	0.036	N/A	0.087	500 MB
MixVoxels-X [Wang et al. 2023]	31.73	0.032	N/A	0.078	500 MB
4DGS [Yang et al. 2024b]	32.01	N/A	**0.014**	0.055	6270 MB
4DGaussians [Wu et al. 2024]	31.15	N/A	0.016	0.049	34 MB
Rotor4DGS [Duan et al. 2024]	31.62	0.030	N/A	0.140	N/A
STG [Li et al. 2024a]	32.05	**0.026**	**0.014**	**0.044**	107 MB
SWinGS [Shaw et al. 2024]	31.10	0.030	N/A	0.096	N/A
E-D3DGS [Bae et al. 2024a]	31.31	**0.028**	N/A	**0.037**	66 MB
Ex4DGS [Lee et al. 2024b]	**32.11**	0.030	0.015	0.048	115 MB
Ours	**32.23**	**0.028**	0.015	0.047	115 MB

Table 3. Ablation study of our method regarding its visual quality and efficiency of splat addition. We conducted experiments with the Technicolor Light Field [Sabater et al. 2017] dataset.

Method	PSNR↑	$DSSIM_1$↓	LPIPS↓	N Splats↓
Baseline	33.62	0.0418	0.0878	**426K**
Baseline Lower Threshold	33.61	0.0410	0.0845	593K
Without Group	33.81	0.0407	0.0840	502K
Pixel-wise Backprojection	33.76	0.0409	0.0807	600K
Patch-wise Backprojection	33.78	0.0411	0.0820	551K
Without Color-aware Clustering	33.81	0.0408	0.0813	561K
Without Shape-based Division	33.93	0.0404	0.0813	558K
Only Backproject Addition	33.89	0.0404	**0.0804**	561K
Only Foreground Splitting	33.84	0.0408	0.0821	551K
Without Error-based Correction	33.66	0.0415	0.0860	550K
Ours	**34.04**	**0.0401**	0.0809	565K

5.1.3 Qualitative Results. We qualitatively compare our method with state-of-the-art methods with publicly available models, E-D3DGS [Bae et al. 2024a], Ex4DGS [Lee et al. 2024b], and the STG [Li et al. 2024a] model trained with official code, since models are not available for most scenes. Fig. 5 and the supplementary video provide qualitative results of our method. Qualitative results show our method's capability to reconstruct dynamic parts in the scene in detail.

5.2 Analysis

5.2.1 Ablation Studies. We conduct experiments to show the impact of components in pipelines, with metrics summarized in Table 3. To show the effectiveness of our error-based correction, we summarize the perpetual quality metrics and a number of splats.

First, we show the baseline method [Lee et al. 2024b] performance, with original and a lower dynamic splat densification threshold. Vanilla densification with more splat budget, *Baseline Lower Threshold*, does not identify appropriate spots to add splats. The shows a clear advantage of our specialized method of adding splats to gain quality improvement from additional splats. Interestingly, using a lower densification threshold improved DSSIM and LPIPS, while PSNR remained relatively unchanged, indicating a high pixel error, despite being supervised to minimize it. However, the quality improvement does not match our approach, despite having more splats.

We further experimented with the importance of our groupings introduced in Sec. 4.2, keeping the error correction method while ablating the grouped dynamic representation. *Without Grouping*, our error correction method is significantly affected. Combined with this result, we demonstrate the impact of grouping on improving consistent splat-object correspondence in Sec. 5.2.3.

We experiment to show the importance of Elliptical Error Clustering in Sec. 4.3. *Pixel-wise Backprojection* uses our algorithm without clustering, creating error correction candidates per pixel instead. It results in significantly more splats. *Patch-wise Backprojection* uses ellipse fitting per error points within a patch of size 16 by 16, similar to the error sampling method by STG [Li et al. 2024a]. Patch-wise projection adds similar splats to our method. Neither approach yielded a significant improvement. *Without Color-aware Clustering* creates clusters only based on the proximity, with an initial DBSCAN pass and K-Means splitting to fit an elliptical shape, which can result in non-uniform colored error within the cluster. *Without Shape-based Division* does not split clusters based on their similarity to an ellipse, resulting in shapes incompatible with splats. These experiments show that our clustering tailored for splat addition is crucial to pinpoint where splat should be added.

Finally, we analyze the impact of our Error-Correcting Splat Addition approach from Sec. 4.4. Generally, combining the two approaches performs better than using one for the rest of the scenes. However, the importance of the two approaches varied. Interestingly, *Only Backproject Addition* had better LPIPS on average than the full method. For the Train scene, using *Only Foreground Splitting* had a similar result to the full method. LPIPS is better than the full method for the Birthday scene, but has worse PSNR. *Without Error-based Correction*, using only grouped dynamic transform, a slight improvement is observed compared to the baseline, resulting in more splats. With improved correspondence of splats with dynamic objects, dynamic object representation could be better; however, reduced dynamic freedom requires more splats.

5.2.2 Visualization of Clustering and Splat Addition. We visualize the error correction process with figures. In the Foreground Splitting case (Fig. 6), hair is inaccurately represented, occluding the background. Checking the color consistency of the center pixel of the ellipse error cluster with other views, if the color appears inconsistent, the splat in the foreground, for hair, is split. The split splats are marked with red color. Further optimization with additional splats can provide a more accurate fit for the hair. Not all candidates are split due to a visibility constraint from the comparison view; multiple attempts from different views gradually improve details.

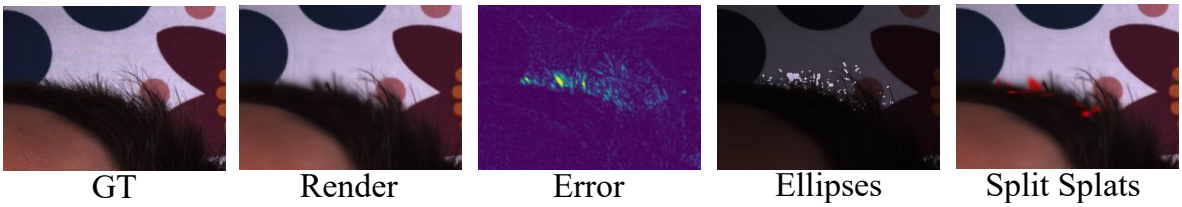

Fig. 6. Example of Foreground Splitting with our method in the Fabien scene. Hair has inconsistent details with the ground truth, occluding the background. Based on the error, ellipses are clustered for the correction candidate. Splats for hair are split (rendered with red color in the image).

Fig. 7. Example of Backproject Addition with our method in the Painter scene with brightness adjusted for visibility. The rendering result shows stripes missing on the clothes. Our method clusters those errors and generates ellipses. Finally, after applying our splat addition method, the dynamic object has a corrected surface. This can be further optimized to reduce errors.

For the Backproject Addition case (Fig. 7), the Grouped 4DGS exhibits a lack of splat on the surface of clothes, lacking stripes. The elliptical error clustering approach identifies erroneous areas and divides them into ellipses. The stripe is visible after adding splats based on the Elliptical Error Clusters. Repeating the error correction process and optimization, our method gradually corrects the dynamic object from the most erroneous area.

5.2.3 Splat-Object Correspondence Consistency. We demonstrate that the Grouped 4D Gaussian Splatting continuously tracks dynamic components in the scene by grouping the dynamic transform. Fig. 8 visualizes the splat and the correspondence between the displayed objects. For visualization, scenes are rendered with a random, unique color assigned for each dynamic splat to visualize how they move in the scene. Train and Fabien scene from the Technicolor Light Field dataset are visualized every three frames (0.1 sec).

Our method consistently assigns a splat to the same object across different frames with our grouped dynamic representation. The baseline per-splat method shows temporal jitter in this visualization. Multiple splats with the wrong dynamic transform and new splats appearing and disappearing in the area to represent one dynamic object indicate a failure to follow the dynamic transform of the real-world object correctly.

5.2.4 Runtime Measurements and Temporal Stability. We measured training time, rendering time, allocated memory during training using a NVIDIA RTX A6000 GPU for measurements, and tPSNR metric [Hasselgren et al. 2020] for the Technicolor Light Field dataset for comparison to the baseline in Table 4. The tPSNR metric is measured by computing PSNR with the ground-truth and the rendered image's residual between consecutive frames:

$$PSNR(\text{render}_t - \text{render}_{t-1}, \text{gt}_t - \text{gt}_{t-1}) \tag{9}$$

We trained for 10,000 additional epochs for the Error Correction stage in addition to the original epochs for the baseline. Additional epochs and splats are reflected in the proportional relevant time and memory. Following grouped motion may come with temporal instability when incorrect splats are added to groups. Group splitting and temporal opacity weight for individual splats effectively address this issue, resulting in improved temporal stability compared to the baseline.

Table 4. Comparison of training, rendering, memory, and temporal stability.

Metric	Baseline [Lee et al. 2024b]	Ours
Training (h)	1.90	3.21
Rendering (s/frame)	0.0140	0.0210
Memory (GB)	2.98	3.37
Temporal Stability (tPSNR)	37.43	37.60

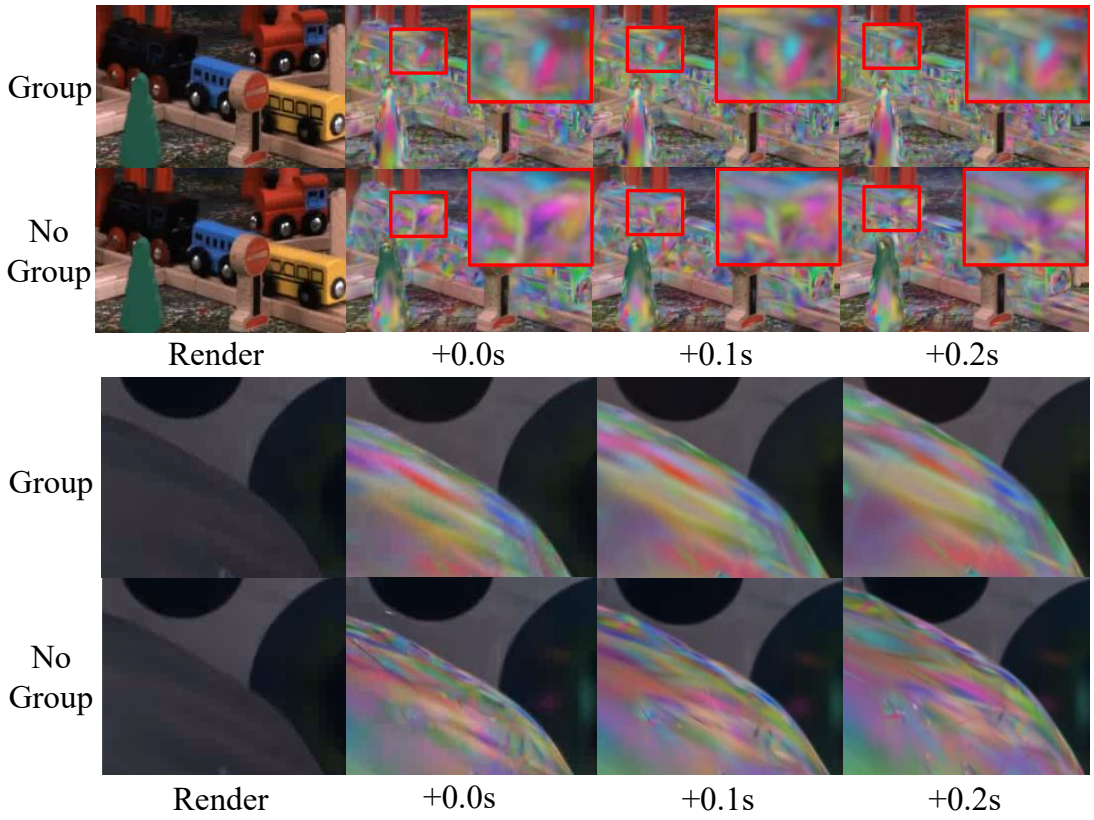

Fig. 8. Comparison of object-splat correspondence consistency of our group-based representation and baseline [Lee et al. 2024b], with 3 frame gaps (0.1 sec). In the first example in the Train scene, the red box focuses on a small area of the railroad car, featuring a mostly uniform color and small details. Comparison of splat alignment in different frames reveals that the grouped transform enables splats to follow the real object more faithfully. For the second example in Fabien's scene, clothes are represented with mostly the same splats with grouping, while the baseline exhibits consistently changing splats, particularly near the boundary area.

6 Conclusion and Limitations

We have presented a novel approach to enhance dynamic novel-view synthesis with 4D Gaussian Splatting by explicitly addressing the critical limitations of ambiguous temporal correspondences and ineffective splat densification. Our method introduces elliptical clustering to localize regions requiring additional splats precisely and employs group-based temporal modeling to enforce stable, coherent splat correspondences across frames. Through targeted error corrections—backprojection-based addition for missing-color errors and foreground splitting for occlusion errors—our method substantially improves perceptual rendering quality and temporal consistency. Experimental evaluations on widely used benchmarks demonstrate significant improvements, up to 0.39dB of PSNR over state-of-the-art methods, particularly in reducing visual artifacts and temporal jitter. We believe our contributions open promising directions for future research, including further optimization of splat efficiency and enhanced handling of complex dynamic scenes.

Our methods have limitations. The proposed formulation is best suited for rigid or geometrically contiguous deformations. Our method assumes a fixed color per splat and keyframed motion, following the baseline approach [Lee et al. 2024b], which limits the ability to model significant appearance changes. As shared by existing 3DGS and 4DGS approaches, translucent objects and volumetric effects (e.g., flames) remain challenging. Yet, our error-driven splat addition with group-level motion modeling provides a flexible foundation that can be adapted to more advanced representations, such as those with learned appearance or volumetric support.

Acknowledgments

This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No. RS-2024-00463802, No. RS-2023-00218601).

References

Benjamin Attal, Jia-Bin Huang, Christian Richardt, Michael Zollhoefer, Johannes Kopf, Matthew O'Toole, and Changil Kim. 2023. HyperReel: High-Fidelity 6-DoF Video with Ray-Conditioned Sampling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Jeongmin Bae, Seoha Kim, Youngsik Yun, Hahyun Lee, Gun Bang, and Youngjung Uh. 2024a. Per-Gaussian Embedding-Based Deformation for Deformable 3D Gaussian Splatting. In *European Conference on Computer Vision (ECCV)*.

Jeongmin Bae, Seoha Kim, Youngsik Yun, Hahyun Lee, Gun Bang, and Youngjung Uh. 2024b. Per-Gaussian Embedding-Based Deformation for Deformable 3D Gaussian

Splatting. In *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer.

Jonathan T. Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P. Srinivasan. 2021. Mip-NeRF: A Multiscale Representation for Anti-Aliasing Neural Radiance Fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. 2023. Zip-NeRF: Anti-Aliased Grid-Based Neural Radiance Fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

Ang Cao and Justin Johnson. 2023. HexPlane: A Fast Representation for Dynamic Scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 12770–12779. doi:10.1109/CVPR52729.2023.01238

Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. 2022. TensoRF: Tensorial Radiance Fields. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 333–350. doi:10.1007/978-3-031-19803-8_20

Kai Cheng, Xiaoxiao Long, Kaizhi Yang, Yao Yao, Wei Yin, Yuexin Ma, Wenping Wang, and Xuejin Chen. 2024. GaussianPro: 3D Gaussian Splatting with Progressive Propagation. *arXiv preprint arXiv:2402.14650* (2024).

Woong Oh Cho, In Cho, Seoha Kim, Jeongmin Bae, Youngjung Uh, and Seon Joo Kim. 2024. 4D Scaffold Gaussian Splatting for Memory Efficient Dynamic Scene Reconstruction. arXiv:2411.17044 [cs.CV] https://arxiv.org/abs/2411.17044

Yuanxing Duan, Fangyin Wei, Qiyu Dai, Yuhang He, Wenzheng Chen, and Baoquan Chen. 2024. 4D-Rotor Gaussian Splatting: Towards Efficient Novel View Synthesis for Dynamic Scenes. In *Proc. SIGGRAPH*.

Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining* (Portland, Oregon) *(KDD'96)*. AAAI Press, 226–231.

Zhiwen Fan, Kevin Wang, Kairun Wen, Zehao Zhu, Dejia Xu, and Zhangyang Wang. 2024. LightGaussian: Unbounded 3D Gaussian Compression with 15x Reduction and 200+ FPS. arXiv:2311.17245 [cs.CV]

Hao Feng, Hao Sun, and Wei Xie. 2025. Disentangled 4D Gaussian Splatting: Towards Faster and More Efficient Dynamic Scene Rendering. arXiv:2503.22159 [cs.GR] https://arxiv.org/abs/2503.22159

Sara Fridovich-Keil, Giacomo Meanti, Frederik Warburg, Benjamin Recht, and Angjoo Kanazawa. 2023. K-Planes: Explicit Radiance Fields in Space, Time, and Appearance. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 12759–12769. doi:10.1109/CVPR52729.2023.01237

Yasutaka Furukawa and J. Ponce. 2006. Carved Visual Hulls for Image-Based Modeling. *International Journal of Computer Vision - IJCV* 81, 564–577. doi:10.1007/11744023_44

Zhiyang Guo, Wengang Zhou, Li Li, Min Wang, and Houqiang Li. 2024. Motion-aware 3D Gaussian Splatting for Efficient Dynamic Scene Reconstruction. *IEEE Transactions on Circuits and Systems for Video Technology* (2024).

Jon Hasselgren, J. Munkberg, Marco Salvi, A. Patney, and Aaron Lefohn. 2020. Neural Temporal Adaptive Sampling and Denoising. *Computer Graphics Forum* 39 (07 2020), 147–155. doi:10.1111/cgf.13919

Wenbo Hu, Yuling Wang, Lin Ma, Bangbang Yang, Lin Gao, Xiao Liu, and Yuewen Ma. 2023. Tri-MipRF: Tri-Mip Representation for Efficient Anti-Aliasing Neural Radiance Fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

Yi-Hua Huang, Yang-Tian Sun, Ziyi Yang, Xiaoyang Lyu, Yan-Pei Cao, and Xiaojuan Qi. 2024. SC-GS: Sparse-Controlled Gaussian Splatting for Editable Dynamic Scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 2023. 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/

Junoh Lee, ChangYeon Won, Hyunjun Jung, Inhwan Bae, and Hae-Gon Jeon. 2024b. Fully Explicit Dynamic Guassian Splatting. In *Proceedings of the Neural Information Processing Systems*.

Joo Chan Lee, Daniel Rho, Xiangyu Sun, Jong Hwan Ko, and Eunbyung Park. 2024a. Compact 3D Gaussian Representation for Radiance Field. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 21719–21728.

Deqi Li, Shi-Sheng Huang, Zhiyuan Lu, Xinran Duan, and Hua Huang. 2024b. ST-4DGS: Spatial-Temporally Consistent 4D Gaussian Splatting for Efficient Dynamic Scene Rendering. In *ACM SIGGRAPH 2024 Conference Papers* (Denver, CO, USA). Association for Computing Machinery, New York, NY, USA.

Tianye Li, Mira Slavcheva, Michael Zollhoefer, Simon Green, Christoph Lassner, Changil Kim, Tanner Schmidt, Steven Lovegrove, Michael Goesele, Richard Newcombe, and Zhaoyang Lv. 2022. Neural 3D Video Synthesis from Multi-view Video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 5521–5531. doi:10.1109/CVPR52688.2022.00547

Zhan Li, Zhang Chen, Zhong Li, and Yi Xu. 2024a. Spacetime Gaussian Feature Splatting for Real-Time Dynamic View Synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE.

Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. 2021. 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)*. 6498–6508. doi:10.1109/CVPR46437.2021.00644

Yiming Liang, Tianhan Xu, and Yuta Kikuchi. 2025. HiMoR: Monocular Deformable Gaussian Reconstruction with Hierarchical Motion Representation. arXiv:2504.06210 [cs.CV] https://arxiv.org/abs/2504.06210

Haotong Lin, Sida Peng, Zhen Xu, Tao Xie, Xingyi He, Hujun Bao, and Xiaowei Zhou. 2023. High-Fidelity and Real-Time Novel View Synthesis for Dynamic Scenes. In *SIGGRAPH Asia Conference Proceedings*.

Youtian Lin, Zuozhuo Dai, Siyu Zhu, and Yao Yao. [n. d.]. Gaussian-Flow: 4D Reconstruction with Dynamic 3D Gaussian Particle. In *Proceedings*.

Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. 2020. Neural Sparse Voxel Fields. In *Advances in Neural Information Processing Systems*.

Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. 2019. Neural Volumes: Learning Dynamic Renderable Volumes from Images. *ACM Trans. Graph.* 38, 4, Article 65 (July 2019), 14 pages.

Zhicheng Lu, Xiang Guo, Le Hui, Tianrui Chen, Min Yang, Xiao Tang, Feng Zhu, and Yuchao Dai. 2024. 3D Geometry-Aware Deformable Gaussian Splatting for Dynamic View Synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. 2024. Dynamic 3D Gaussians: Tracking by Persistent Dynamic View Synthesis. In *Proceedings of the IEEE International Conference on 3D Vision (3DV)*. IEEE.

Ben Mildenhall, Pratul P. Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. 2019. Local Light Field Fusion: Practical View Synthesis with Prescriptive Sampling Guidelines. *ACM Transactions on Graphics (TOG)* (2019).

Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. 2020. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *European Conference on Computer Vision (ECCV)*.

Wieland Morgenstern, Florian Barthel, Anna Hilsmann, and Peter Eisert. 2025. Compact 3D Scene Representation via Self-Organizing Gaussian Grids. In *Computer Vision – ECCV 2024*. Springer Nature Switzerland, Cham, 18–34. doi:10.1007/978-3-031-73013-9_2

Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. 2022. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics* 41, 4 (July 2022), 1–15. doi:10.1145/3528223.3530127

T. Neff, P. Stadlbauer, M. Parger, A. Kurz, J. H. Mueller, C. R. A. Chaitanya, A. Kaplanyan, and M. Steinberger. 2021. DONeRF: Towards Real-Time Rendering of Compact Neural Radiance Fields using Depth Oracle Networks. *Computer Graphics Forum* 40, 4 (2021). doi:10.1111/cgf.14340

Keunhong Park, Utkarsh Sinha, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Steven M Seitz, and Ricardo Martin-Brualla. 2021a. Nerfies: Deformable neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 5865–5874.

Keunhong Park, Utkarsh Sinha, Peter Hedman, Jonathan T. Barron, Sofien Bouaziz, Dan B. Goldman, Ricardo Martin-Brualla, and Steven M. Seitz. 2021b. HyperNeRF: A Higher-Dimensional Representation for Topologically Varying Neural Radiance Fields. In *ACM Transactions on Graphics (TOG)*, Vol. 40. 1–12. doi:10.1145/3478513.3480487

Martin Piala and Ronald Clark. 2021. TermiNeRF: Ray Termination Prediction for Efficient Neural Rendering. In *2021 International Conference on 3D Vision (3DV)*. IEEE, 1106–1114.

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

Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. 2021. KiloNeRF: Speeding up Neural Radiance Fields with Thousands of Tiny MLPs. arXiv:2103.13744 [cs.CV]

Barbara Roessle, Jonathan T. Barron, Ben Mildenhall, Pratul P. Srinivasan, and Matthias Nießner. 2022. Dense Depth Priors for Neural Radiance Fields from Sparse Input Views. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Samuel Rota Bulò, Lorenzo Porzi, and Peter Kontschieder. 2025. Revising Densification inGaussian Splatting. In *Computer Vision – ECCV 2024*, Aleš Leonardis, Elisa Ricci, Stefan Roth, Olga Russakovsky, Torsten Sattler, and Gül Varol (Eds.). Springer Nature Switzerland, Cham, 347–362.

Neus Sabater, Guillaume Boisson, Benoit Vandame, Paul Kerbiriou, Frederic Babon, Matthieu Hog, Tristan Langlois, Remy Gendrot, Olivier Bureller, Arno Schubert, and Valerie Allie. 2017. Dataset and Pipeline for Multi-View Light-Field Video. In *CVPR Workshops*.

Johannes Schönberger and Jan-Michael Frahm. 2016. Structure-from-Motion Revisited. doi:10.1109/CVPR.2016.445

Ruizhi Shao, Zerong Zheng, Hanzhang Tu, Boning Liu, Hongwen Zhang, and Yebin Liu. 2023. 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 (CVPR)*. 12780–12790. doi:10.1109/CVPR52729.2023.01239

Richard Shaw, Michal Nazarczuk, Jifei Song, Arthur Moreau, Sibi Catley-Chandar, Helisa Dhamo, and Eduardo Pérez-Pellitero. 2024. SWinGS: Sliding Windows for Dynamic 3D Gaussian Splatting. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

Ken Shoemake. 1985. Animating rotation with quaternion curves. In *Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '85)*. Association for Computing Machinery, New York, NY, USA, 245–254. doi:10.1145/325334.325242

Liangchen Song, Anpei Chen, Zhong Li, Zhang Chen, Lele Chen, Junsong Yuan, Yi Xu, and Andreas Geiger. 2023. NeRFPlayer: A Streamable Dynamic Scene Representation with Decomposed Neural Radiance Fields. *IEEE Transactions on Visualization and Computer Graphics* 29, 5 (2023), 2732–2742. doi:10.1109/TVCG.2023.3247082

Mohammed Suhail, Carlos Esteves, Leonid Sigal, and Ameesh Makadia. 2022. Light Field Neural Rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 8269–8279.

Michael Zollhooefer Simon Green Shristoph Lassner Changil Kim Tanner Schmidt Steven Lovegrove Michael Goesele Richard Newcombe et al. Tianye Li, Mira Slavcheva. 2022. Neural 3d video synthesis from multi-view video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Feng Wang, Sinan Tan, Xinghang Li, Zeyue Tian, Yafei Song, and Huaping Liu. 2023. Mixed Neural Voxels for Fast Multi-view Video Synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 19706–19716.

Yi Wei, Shaohui Liu, Yongming Rao, Wang Zhao, Jiwen Lu, and Jie Zhou. 2021. NerfingMVS: Guided Optimization of Neural Radiance Fields for Indoor Multi-view Stereo. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 2024. 4D Gaussian Splatting for Real-Time Dynamic Scene Rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE.

Yuqun Wu, Jae Yong Lee, Chuhang Zou, Shenlong Wang, and Derek Hoiem. 2025. MonoPatchNeRF: Improving Neural Radiance Fields with Patch-based Monocular Guidance. In *International Conference on 3D Vision (3DV)*.

Zhen Xu, Sida Peng, Haotong Lin, Guangzhao He, Jiaming Sun, Yujun Shen, Hujun Bao, and Xiaowei Zhou. 2024. 4K4D: Real-Time 4D View Synthesis at 4K Resolution. In *CVPR*.

Ziyi Yang, Xinyu Gao, Wen Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. 2024a. Deformable 3D Gaussians for High-Fidelity Monocular Dynamic Scene Reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE.

Zeyu Yang, Hongye Yang, Zijie Pan, Xiatian Zhu, and Li Zhang. 2024b. Real-time Photorealistic Dynamic Scene Representation and Rendering with 4D Gaussian Splatting. In *Proceedings of the International Conference on Learning Representations (ICLR)*. OpenReview.

Alex Yu, Sara Fridovich-Keil, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. 2022. Plenoxels: Radiance Fields without Neural Networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. 2021. PlenOctrees for Real-time Rendering of Neural Radiance Fields. arXiv:2103.14024 [cs.CV]

Zheng Zhang, Wenbo Hu, Yixing Lao, Tong He, and Hengshuang Zhao. 2025. Pixel-GS: Density Control with Pixel-Aware Gradient for 3D Gaussian Splatting. In *Computer Vision – ECCV 2024*, Aleš Leonardis, Elisa Ricci, Stefan Roth, Olga Russakovsky, Torsten Sattler, and Gül Varol (Eds.). Springer Nature Switzerland, Cham, 326–342.