From NeRFs to 3D Gaussians: A Survey on Dynamic Scene Modeling and Rendering

Yuwei Zhao, Kaiyuan Zhang, Yuxiang Liu, Yilin Zhang, and Keqin Zhang

Abstract—Dynamic scene modeling is a pivotal technology for emerging applications in the Metaverse, virtual reality, and free-viewpoint video. While Neural Radiance Fields (NeRF) have achieved photorealistic rendering, their application to dynamic scenes is hindered by slow training and inference speeds. Recently, 3D Gaussian Splatting (3DGS) has emerged as a disruptive alternative, offering real-time rendering capabilities through explicit splatting. In this survey, we provide a comprehensive review of the transition from implicit to explicit representations for dynamic scene modeling. We propose a structured taxonomy categorizing existing methods into four paradigms: Implicit Deformation Fields, Spacetime Neural Fields, Hybrid Representations, and Dynamic Gaussian Splatting. Furthermore, we conduct a systematic comparative analysis based on reconstruction quality, training efficiency, and storage footprint. Our analysis reveals that while implicit methods excel in storage efficiency (< 5 MB), explicit Gaussian-based methods demonstrate superior real-time performance (> 80 FPS). Finally, we discuss open challenges, including storage compression and integration with generative AI. We believe this survey serves as a timely roadmap for researchers navigating the rapid paradigm shift from implicit fields to explicit primitives in 4D vision. Our compiled resources are available at https://github.com/8arbatos/Academic-English-Group-Paper.

Index Terms—Dynamic Scene Modeling, Neural Radiance Fields, 3D Gaussian Splatting, View Synthesis, 4D Reconstruction.

1 Introduction

With the rapid advancement of virtual reality (VR), augmented reality (AR), and the Metaverse, the demand for photorealistic 3D content creation has surged exponentially. Dynamic scene modeling, which aims to reconstruct 3D geometry and appearance from 2D video streams, serves as a fundamental technology for these applications, enabling immersive telepresence, free-viewpoint video, and digital human avatars [1]. Unlike static scene reconstruction, modeling dynamic scenes from video presents a highly ill-posed inverse problem due to the entanglement of object motion, topology changes, and time-variant lighting conditions.

Traditionally, 3D reconstruction relied on Structure-from-Motion (SfM) and Multi-View Stereo (MVS) algorithms [2]. While these methods, such as COLMAP, provide robust camera pose estimation, they struggle to capture thin structures and view-dependent effects (e.g., reflections). More importantly, traditional pipelines typically assume a static world, making them brittle when applied to dynamic video sequences where geometry consistency is violated over time.

To address dynamic content before the deep learning era, non-rigid reconstruction methods were developed to warp a canonical template to live frames. Pioneering works like **DynamicFusion** [3] and **Fusion4D** [4] extended volumetric fusion techniques to handle real-time non-rigid deformation. However, these approaches typically relied on depth sensors (RGB-D) or complex hand-crafted regularizers to track motion. They often struggled to produce photorealistic textures or handle large topological changes (e.g., fluid or fracture) due to the limitations of explicit mesh or TSDF representations.

• *Y. Zhao, K. Zhang, Y. Liu, Y. Zhang, and K. Zhang are with the School of Computer Science and Technology, Ocean University of China, Qingdao 266100, China, and also with the School of MPs & EPs, Heriot-Watt University, Edinburgh EH14 4AS, UK.*

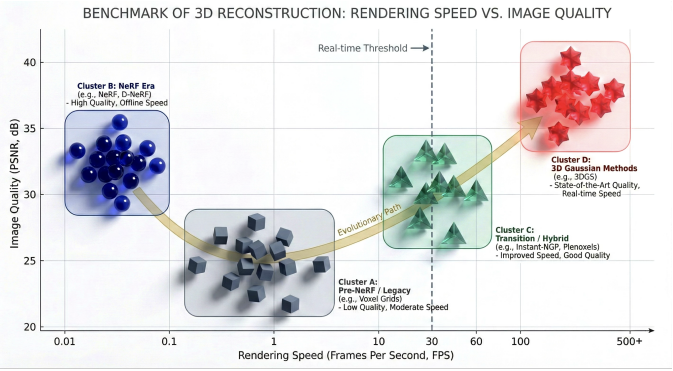

Fig. 1. Performance landscape of video-based 3D reconstruction methods. We visualize the trade-off between rendering speed (FPS, log scale) and reconstruction quality (PSNR). Implicit methods achieve high quality but slow inference. Explicit methods (3DGS) achieve real-time performance while maintaining high fidelity.

The field witnessed a paradigm shift with the introduction of **Neural Radiance Fields (NeRF)** [5]. By representing scenes as implicit continuous functions parameterized by Multi-Layer Perceptrons (MLPs), NeRF achieved unprecedented rendering quality. Following the original NeRF, foundational extensions were quickly proposed to enhance robustness: **NeRF++** [6] introduced inverted sphere parameterization to model unbounded scenes, and **Mip-NeRF** [7] employed conical frustums to address aliasing artifacts, which was further refined by **Mip-NeRF 360** [8] for large-scale environments. Additionally, **Ref-NeRF** [9] significantly improved the rendering of glossy surfaces by restructuring the view-dependent parameterization. Building on these static foundations, subsequent works such as D-NeRF [10] and Nerfies [11] extended the implicit paradigm to dynamic domains by introducing learnable deformation fields to handle non-rigid motion.

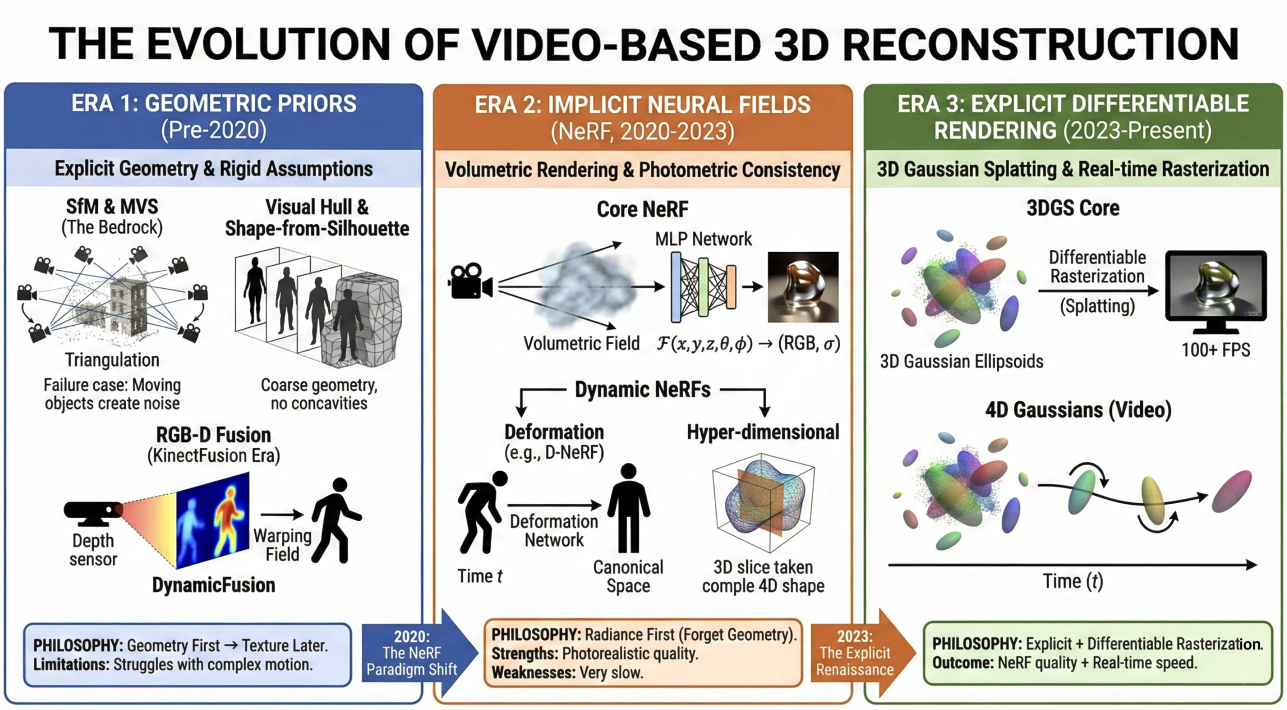

Fig. 2. **Chronological evolution of dynamic scene modeling.** The timeline illustrates the paradigm shift from **Implicit Neural Fields** (e.g., D-NeRF, DyNeRF), which prioritize storage efficiency but suffer from slow inference, to **Hybrid Representations** (e.g., HexPlane) that utilize explicit spatial structures for acceleration, and finally to the emerging **Explicit Dynamic 3DGS** (e.g., 4D-GS). This trajectory highlights the community's move from continuous MLP-based functions to discrete, rasterizable primitives to achieve real-time rendering capabilities.

However, implicit methods suffer from prohibitive computational costs due to the extensive ray-marching sampling required during both training and inference, limiting their deployment in real-time applications. Prior to 3DGS, methods like **DVGO** [12] and **Plenoxels** [13] attempted to accelerate training using explicit voxel grids, while **NeuS** [14] and **VolSDF** [15] focused on improving geometric fidelity using Signed Distance Functions (SDF).

Recently, **3D Gaussian Splatting (3DGS)** [16] has emerged as a compelling alternative, marking a return to explicit volumetric representations. By combining the differentiability of deep learning with the efficiency of rasterization-based rendering, 3DGS enables real-time rendering (100+ FPS) and fast training speeds. This breakthrough has triggered a new wave of research focused on extending Gaussian primitives to 4D spatiotemporal modeling [17], [18], aiming to combine the efficiency of explicit representations with the flexibility of neural fields.

Despite the explosion of research papers in this domain, a systematic comparison between the established implicit (NeRF-based) paradigms and the emerging explicit (Gaussian-based) approaches is lacking. To fill this gap, this paper provides a structured survey of dynamic scene modeling. Our main contributions are summarized as follows:

- **Structured Taxonomy:** We categorize dynamic modeling methods into four distinct streams: Implicit Deformation Fields (e.g., D-NeRF), Spacetime Neural Fields (e.g., DyNeRF), Hybrid Representations (e.g., HexPlane), and Dynamic Gaussian Splatting (e.g., 4D-GS). This classification clarifies the evolution from continuous functions to discrete primitives.
- **Comprehensive Comparison:** We provide an in-depth comparative analysis of these methods, evaluating them not just on visual quality (PSNR/SSIM), but critically on computational efficiency (Training Time, FPS) and storage costs. We highlight the trade-off between the compactness of implicit methods and the speed of explicit methods.
- **Future Insights:** We identify critical open challenges, such as the "baked-in" lighting limitation and the storage bottleneck of dynamic Gaussians, and propose future directions involving generative AI integration and inverse rendering.

2 Preliminaries

In this section, we first provide a unified mathematical formulation for the problem of video-based 3D reconstruction. We then detail the fundamental principles of the two dominant paradigms: the implicit neural representation introduced by NeRF and the explicit differentiable representation proposed by 3D Gaussian Splatting.

2.1 Mathematical Formulation

Video-based 3D reconstruction can be fundamentally formulated as an inverse rendering problem. Given a sequence of N observed images $\mathcal{I} = \{I_1, I_2, \ldots, I_N\}$ captured from a video stream, along with their corresponding camera poses $\mathcal{P} = \{P_1, P_2, \ldots, P_N\}$ (typically estimated via Structure-from-Motion, SfM), the goal is to optimize a 3D scene representation $\mathcal{S}$ such that it can synthesize novel views consistent with the input observations.

Formally, we seek to minimize the photometric reconstruction error between the rendered images and the ground truth images:

$$\mathcal{S}^* = \arg\min_{\mathcal{S}} \sum_{k=1}^{N} \mathcal{L}\left(\mathcal{R}(\mathcal{S}, P_k), I_k\right) \tag{1}$$

where $\mathcal{R}(\cdot)$ denotes the differentiable rendering function that projects the 3D representation $\mathcal{S}$ onto the 2D image plane given a camera pose P_k, and $\mathcal{L}$ represents the loss function (typically ℓ_1, ℓ_2, or D-SSIM) measuring the discrepancy between the synthesized and observed pixels.

2.2 Fundamentals of Neural Radiance Fields

Neural Radiance Fields (NeRF) represent a scene implicitly as a continuous volumetric function parameterized by a Multi-Layer Perceptron (MLP).

2.2.1 Scene Representation

The core function, denoted as F_Θ, maps a continuous 5D coordinate input—consisting of a 3D spatial location $\mathbf{x} = (x, y, z)$ and a 2D viewing direction $\mathbf{d} = (\theta, \phi)$—to a volume density σ and a view-dependent emitted color $\mathbf{c} = (r, g, b)$:

$$F_\Theta : (\mathbf{x}, \mathbf{d}) \rightarrow (\mathbf{c}, \sigma) \tag{2}$$

By conditioning the color on the viewing direction $\mathbf{d}$, NeRF is able to model complex view-dependent effects such as specular reflections.

2.2.2 Volume Rendering Equation

To synthesize an image, NeRF employs classical volume rendering techniques. The color $C(\mathbf{r})$ of a camera ray $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$ is computed by integrating the radiance along the ray from near bound t_n to far bound t_f:

$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}(t))\mathbf{c}(\mathbf{r}(t), \mathbf{d})\, dt \tag{3}$$

where $T(t)$ denotes the accumulated transmittance, representing the probability that the ray travels from t_n to t without hitting any other particles:

$$T(t) = \exp\left(-\int_{t_n}^{t} \sigma(\mathbf{r}(s))\, ds\right) \tag{4}$$

2.2.3 Discretization

In practice, the continuous integral is approximated using stratified sampling. The ray is divided into N bins, and the integral is reduced to a finite sum:

$$\begin{aligned} \hat{C}(\mathbf{r}) &= \sum_{i=1}^{N} T_i(1 - \exp(-\sigma_i\delta_i))\mathbf{c}_i\,, \\ \text{where } T_i &= \exp\left(-\sum_{j=1}^{i-1} \sigma_j\delta_j\right) \end{aligned} \tag{5}$$

Here, δ_i is the distance between adjacent samples. This differentiable rendering pipeline allows the MLP parameters Θ to be optimized via stochastic gradient descent.

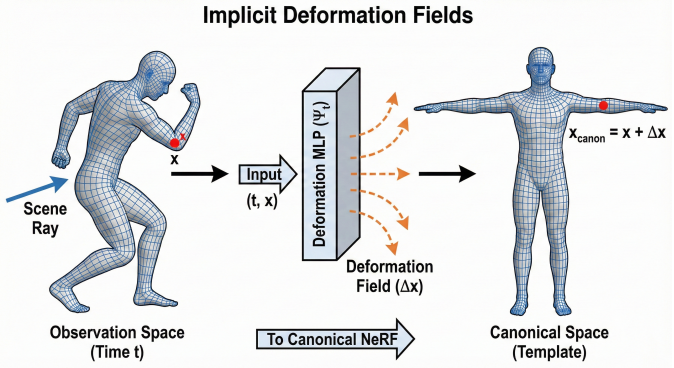

Fig. 3. Overview of Implicit Deformation Fields. Dynamic scenes are modeled by learning a deformation field that maps points from the observation space to a canonical space, where a static NeRF represents the scene's appearance and geometry.

2.3 Fundamentals of 3D Gaussian Splatting

To address the computational inefficiency of volumetric ray-marching in NeRF, 3D Gaussian Splatting (3DGS) introduces an explicit representation based on anisotropic 3D Gaussians, enabling real-time rendering via rasterization.

2.3.1 3D Gaussian Representation

The scene is represented by a set of 3D Gaussians. Each primitive is defined by a center position μ, a covariance matrix Σ, an opacity α, and spherical harmonics (SH) coefficients for view-dependent color. The influence of a Gaussian at a point $\mathbf{x}$ is defined as:

$$G(\mathbf{x}) = \exp\left(-\frac{1}{2}(\mathbf{x} - \mu)^T\Sigma^{-1}(\mathbf{x} - \mu)\right) \tag{6}$$

To ensure the covariance matrix Σ remains positive semi-definite during optimization, it is factorized into a scaling matrix S and a rotation matrix R:

$$\Sigma = RSS^TR^T \tag{7}$$

2.3.2 Projection and Splatting

Unlike NeRF's ray-marching, 3DGS projects the 3D Gaussians directly onto the 2D image plane. Given a viewing transformation W, the covariance matrix Σ' in camera coordinates is approximated as:

$$\Sigma' = JW\Sigma W^TJ^T \tag{8}$$

where J is the Jacobian of the affine approximation of the projective transformation. This step effectively "splats" the 3D ellipsoids into 2D ellipses.

2.3.3 Tile-based Blending

The final pixel color is computed using α-blending of the N sorted Gaussians overlapping the pixel:

$$C = \sum_{i\in\mathcal{N}} c_i\alpha_i \prod_{j=1}^{i-1}(1 - \alpha_j) \tag{9}$$

Here, α_i is obtained by evaluating the 2D Gaussian multiplied by the learned opacity. This tile-based rasterization approach allows for highly efficient backpropagation and real-time rendering speeds (> 100 FPS), making it particularly suitable for high-resolution video reconstruction.

3 Taxonomy of Dynamic Scene Modeling

3.1 Implicit Deformation Fields

This category of methods addresses dynamic scene modeling by decoupling the scene's geometry into two components: a static structure and a time-variant deformation.

3.1.1 Core Assumption and Architecture

The fundamental assumption of this paradigm is the existence of a time-independent **Canonical Space**. The dynamic scene observed at any time instance t is modeled as a warped version of this canonical configuration [10]. This approach simplifies the learning objective by separating motion from appearance.

Architecturally, these frameworks typically employ two joint Multilayer Perceptrons (MLPs):

Deformation Network: This network learns the mapping from the observation space to the canonical space. It takes the current 3D position $\mathbf{x}$ and a temporal indicator (time t or a latent code ω_i) as input, and outputs a displacement vector $\Delta\mathbf{x}$ (or a dense vector field).

Canonical Network: This is a standard static NeRF that represents the scene in its "rest pose". It takes the corrected position $\mathbf{x}_{canon} = \mathbf{x} + \Delta\mathbf{x}$ and viewing direction $\mathbf{d}$ to predict color $\mathbf{c}$ and density σ.

Mathematically, this process can be formulated as:

$$(\mathbf{x}_{canon}, \Delta\mathbf{x}) = \mathcal{D}(\mathbf{x}, t), \quad (\mathbf{c}, \sigma) = \mathcal{F}_{\text{NeRF}}(\mathbf{x}_{canon}, \mathbf{d}) \quad (10)$$

where $\mathcal{D}$ represents the deformation network and $\mathcal{F}_{\text{NeRF}}$ represents the canonical radiance field [10].

3.1.2 Representative Frameworks

The evolution of implicit deformation fields can be traced through general-purpose deformations to specialized domain-specific solutions.

General Non-Rigid Deformation: Pumarola et al. established the baseline with **D-NeRF**, introducing the translation field $\mathbf{x}(t) \rightarrow \mathbf{x}_{canon} + \Delta\mathbf{x}$. It optimizes the scene using a sparse set of monocular images, successfully reconstructing objects like a moving mechanical arm or a jumping human [10]. Addressing the challenges of casual selfie videos (e.g., camera movement), Park et al. introduced **Nerfies**, which employs learned latent deformation codes ω_i and **elastic regularization** to impose rigidity priors, preventing unrealistic distortions in background regions [11]. Furthermore, **NR-NeRF** [19] demonstrated robust non-rigid reconstruction from monocular video. To handle high-resolution dynamics efficiently, **HyperReel** [20] introduced a memory-efficient 6-DoF tensor representation, significantly improving rendering speed compared to pure MLP-based deformations.

Handling Topological Changes: A critical limitation of continuous deformation fields is their inability to model topological changes (e.g., a mouth opening). Park et al. proposed **HyperNeRF** to address this by lifting the scene into a higher-dimensional "hyperspace". Instead of simply deforming points, HyperNeRF learns a **slicing surface** in this ambient hyperspace, effectively modeling topological variations as continuous deformations of the slicing manifold [21].

Human-Centric Extensions: Reconstructing dynamic humans presents unique challenges due to complex articulated motions. Pure deformation fields often fail to capture the kinematic constraints of the human body. To address this, **Neural Body** [22] incorporates a parametric 3D human model (SMPL) as a geometric prior. It anchors latent codes to the vertices of the deformable mesh, allowing the network to diffuse information across valid poses. Similarly, **HumanNeRF** [23] and **Animatable NeRF** [24] introduce skeleton-driven deformation fields, where the warping is explicitly conditioned on the transformation matrices of human bones, enabling high-fidelity re-posing and free-viewpoint rendering of performers.

3.1.3 Summary

Pros: These methods excel at reconstructing non-rigid, organic objects (such as human faces and bodies) with photorealistic quality. By incorporating domain-specific priors (e.g., SMPL [22]), they can achieve robust performance even under severe self-occlusion.

Cons: The primary drawback is the computational cost; training is extremely slow due to the MLP queries required for ray marching.

Furthermore, without explicit topology handling (like HyperNeRF) or skeletal priors, pure deformation fields struggle with large, discontinuous motions.

3.2 Spacetime Neural Fields

Unlike deformation-based methods that warp a static template, Spacetime Neural Fields treat time as an intrinsic dimension of the scene. These methods directly learn a continuous function over the spatiotemporal domain, effectively slicing the 4D volume to retrieve the state of the scene at any specific moment.

3.2.1 Core Concepts and Architecture

The fundamental shift in this paradigm is the expansion of the input domain. While a standard NeRF maps a 5D coordinate $(\mathbf{x}, \mathbf{d})$ to color and density, Spacetime Neural Fields extend this to a **Higher-dimensional Input** by incorporating time t. The mapping function evolves into $F_\Theta : (\mathbf{x}, \mathbf{d}, t) \rightarrow (\mathbf{c}, \sigma)$. To handle complex dynamics, two primary strategies are employed:

Latent Code Modulation: Instead of using a raw scalar t, methods like DyNeRF learn a set of compact, time-variant *latent codes* $\mathbf{z}_t$ for each frame. These codes are fed into the MLP alongside spatial coordinates, acting as a switch that modulates the network's activations to represent topological changes, varying illumination, and transient objects.

Physical Consistency (Scene Flow): To enforce temporal coherence, some approaches incorporate physical constraints. A notable example is **NSFF (Neural Scene Flow Fields)** [25], which predicts not only the radiance and density but also the 3D *Scene Flow* vector $\mathbf{f}$. By enforcing consistency between the backward/forward flow and the geometric positions across frames, the model ensures that the dynamic representation adheres to physical motion laws.

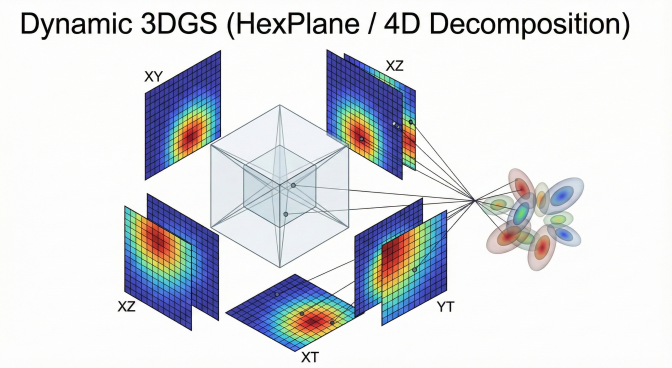

Fig. 4. Overview of Dynamic Gaussian Splatting. Dynamic scenes are modeled by extending 3D Gaussians with temporal parameterization, either via deformation from a canonical set or direct 4D spatiotemporal representation.

Mathematically, the general formulation for this category can be expressed as:

$$(\mathbf{c}, \sigma) = F_\Theta(\mathbf{x}, \mathbf{d}, \mathbf{z}_t) \quad \text{or} \quad (\mathbf{c}, \sigma, \mathbf{f}) = F_\Theta(\mathbf{x}, t) \tag{11}$$

where $\mathbf{z}_t$ represents the learned latent code and $\mathbf{f}$ denotes the predicted scene flow vector.

3.2.2 Representative Frameworks

In the following, we discuss two representative frameworks that address dynamic modeling under distinct input constraints: **DyNeRF** for synchronized multi-view videos and **Video-NeRF** for casual monocular captures.

DyNeRF (Dynamic NeRF): Li et al. proposed DyNeRF to address the challenge of reconstructing dynamic scenes from multi-view videos [26]. Instead of a deformation field, DyNeRF conditions the radiance field on a learnable latent embedding $\mathbf{z}_t$. This allows the model to capture complex dynamics that are difficult to model with continuous deformations, such as flame, pouring liquid (topology changes), and view-dependent lighting effects.

Video-NeRF: Focusing on the challenging monocular setting, Video-NeRF learns a direct mapping from spacetime coordinates $(\mathbf{x}, t)$ to an irradiance field (omitting view dependence to handle single-video ambiguity). To prevent the "motion-appearance ambiguity" (where geometry changes are incorrectly explained as texture changes), it incorporates depth supervision from pre-trained monocular estimators and introduces a "static scene loss" to constrain unobserved regions by propagating information across time [27].

3.2.3 Summary

Pros: These methods are well-suited for large-scale scenes and complex dynamics involving drastic topological changes (e.g., fluid dynamics, fire) where defining a canonical template is infeasible.

Cons: The addition of the time dimension significantly expands the sampling space, leading to the "curse of dimensionality". Without sufficient data (e.g., multi-view rigs) or strong regularization (e.g., depth/flow priors), these models are prone to overfitting or producing temporally incoherent artifacts.

3.3 Dynamic Gaussian Splatting

This category represents the current state-of-the-art (SOTA), marking a shift from implicit neural fields back to explicit, rasterizable primitives. By extending 3D Gaussian Splatting (3DGS) to the temporal domain, these methods aim to combine the photorealism of NeRF with the real-time rendering capabilities of rasterization.

3.3.1 Core Concepts and Architecture

The fundamental primitive in this paradigm is the **3D Gaussian**, characterized by a center position μ, a covariance matrix Σ, an opacity α, and spherical harmonics (SH) for color. To ensure physical validity, the covariance is factorized into scaling S and rotation R matrices: $\Sigma = RSS^TR^T$.

To handle dynamic scenes without incurring the prohibitive memory cost of storing per-frame Gaussians, the core strategy is **Temporal Parameterization**. The attributes of Gaussians are modeled as functions of time t. The influence of a dynamic Gaussian at time t is formulated as:

$$G(\mathbf{x}, t) = \exp\left(-\frac{1}{2}(\mathbf{x} - \mu(t))^T\Sigma(t)^{-1}(\mathbf{x} - \mu(t))\right) \tag{12}$$

where $\mu(t)$ and $\Sigma(t)$ denote time-variant position and covariance. Two primary streams exist to model these functions:

Deformation-based Approach: Similar to D-NeRF, this stream maintains a set of static *Canonical Gaussians*. A learnable deformation network predicts offsets for position, rotation, and scaling $(\delta x, \delta r, \delta s)$ to warp canonical primitives to the current timestamp.

4D Spacetime Primitives: Alternatively, the scene can be represented holistically by querying a high-dimensional feature space. Techniques like **HexPlane decomposition** factorize the 4D spatiotemporal volume into multi-resolution 2D planes (e.g., xy, xt, yt) to efficiently encode temporal dynamics.

3.3.2 Representative Frameworks

In the following, we discuss two representative frameworks that implement these strategies: **Deformable 3DGS** for the deformation-based stream and **4D-GS** for the spatiotemporal representation.

Deformable 3DGS: Yang et al. proposed a framework that learns 3D Gaussians in a canonical space coupled with a deformation field. Unlike NeRF-based deformations that only modify positions, Deformable 3DGS explicitly predicts offsets for position, rotation, and scaling $(\delta x, \delta r, \delta s)$ to ensure the ellipsoids align correctly with surface motion. It also introduces an "annealing smooth training" mechanism to mitigate temporal jitter caused by inaccurate poses in real-world datasets [28].

4D-GS: Wu et al. introduced a holistic representation combining 3D Gaussians with 4D neural voxels. To solve the memory bottleneck, they utilize a **HexPlane** structure to encode spatial-temporal features. A lightweight multi-head decoder then predicts the Gaussian deformations from these features. This design enables real-time rendering (e.g., 82 FPS at 800×800 resolution) while maintaining compact storage comparable to static methods [17].

3.3.3 Specialized and Robust Dynamic Gaussians

Beyond general scenes, significant efforts have been directed towards human-centric reconstruction. **GauHuman** [29] and **HUGS** [30] integrate SMPL priors into the 3DGS pipeline to achieve real-time rendering of articulated humans (60+ FPS). **GaussianAvatar** [31] further enables drivable avatars from monocular video, while **GPS-Gaussian** [32] introduces a generalizable pixel-wise splatting approach for robust human synthesis.

Specific to head avatars, **NerFace** [33] pioneered dynamic facial modeling with NeRFs. Recent Gaussian-based works like **FlashAvatar** [34], **InstaAvatar** [35], and **Mono-GaussianAvatar** [36] achieve high-fidelity head reconstruction with extreme efficiency.

To enhance physical plausibility and controllability, **PhysGaussian** [37] incorporates unified simulation and rendering, allowing 3D Gaussians to respond to physical forces. **Gaussian-Flow** [38] and **CoGS** [39] explicitly model scene flow and controllability to address the limitations of purely deformation-based approaches. Furthermore, **SC-GS** [40] utilizes sparse control points for editable dynamics, while **Spacetime Gaussian** [41] employs feature splatting to enhance temporal consistency.

Finally, addressing the sampling artifacts in multi-scale rendering, **Mip-Splatting** [42] introduces 3D smoothing filters to enable alias-free rendering. To improve geometric stability, **Scaffold-GS** [43] utilizes anchor points to guide the distribution of Gaussians. In the domain of robotics, 3DGS has been successfully adapted for Simultaneous Localization and Mapping (SLAM). Systems like **GS-SLAM** [44] and **Splat-SLAM** [45] demonstrate that dynamic Gaussians can serve as a unified representation for both tracking and mapping.

3.3.4 Summary

Pros: These methods inherit the efficiency of the rasterization pipeline, achieving real-time rendering speeds ($>$ 30 FPS) and fast training convergence (often within minutes), which addresses the primary limitation of implicit NeRFs [17].

Cons: They can be memory-intensive (high VRAM usage) when the number of primitives grows large. Furthermore, modeling large, complex motions from monocular video remains challenging, potentially leading to overfitting or artifacts in unseen views [28].

3.4 Hybrid and Accelerated Representations

While implicit deformation fields offer high fidelity and spacetime neural fields handle complex topology, both suffer from slow training and inference speeds due to the heavy reliance on large MLPs. Hybrid representations address this bottleneck by combining **explicit spatial data structures** (to store learnable features) with **lightweight implicit decoders** (to interpret them). This paradigm serves as a crucial bridge between pure NeRFs and the fully explicit 3DGS.

3.4.1 Core Strategy: Spatial Acceleration

The core insight of this category is to replace the "deep" computation of a large MLP with a "shallow" lookup operation. Instead of mapping a coordinate $\mathbf{x}$ directly to color/density via a deep network, hybrid methods first query a learnable feature vector $\mathbf{f}$ from a spatial structure (e.g., Voxel Grid, Octree, or Hash Table) and then pass this feature to a tiny MLP:

$$\mathbf{f} = \text{Query}(\mathcal{S}, \mathbf{x}), \quad (\mathbf{c}, \sigma) = \text{MLP}_{\text{tiny}}(\mathbf{f}) \tag{13}$$

where $\mathcal{S}$ represents the explicit data structure. This design significantly reduces the floating-point operations (FLOPs) required per sample.

3.4.2 Representative Frameworks

In the following, we discuss four milestone frameworks that utilize distinct spatial data structures—ranging from hash grids and tensor decomposition to sparse voxels and point clouds—to achieve computational efficiency.

Instant-NGP: Müller et al. revolutionized the field with *Instant Neural Graphics Primitives*, which introduced **Multiresolution Hash Encoding**. By mapping spatial coordinates to a set of learnable feature vectors stored in hash tables at multiple resolution levels, Instant-NGP reduces the training time of radiance fields from hours to seconds. Although originally static, its hash encoding backbone has become the standard component for accelerating dynamic methods (including many Spacetime NeRFs and Dynamic 3DGS variants) [46].

TensoRF & K-Planes: To handle the high memory footprint of dense voxel grids (which scale cubically $O(N^3)$), these methods employ **Tensor Decomposition**. TensoRF factorizes the 3D scene tensor into vector-matrix outer products. Extending this to dynamic scenes, **K-Planes** decomposes the 4D spatiotemporal volume into six planar planes (e.g., xy, xt, yt). This decomposition allows for explicit modeling of time while maintaining a low memory footprint, effectively "unrolling" the complex dynamic scene into manageable 2D feature maps [47], [48]. Similarly, **Tensor4D** [49] employs tensor decomposition to model 4D dynamics efficiently. For streaming applications, **StreamRF** [50] proposes a transformation-aware approach to handle long dynamic video sequences with on-the-fly training.

Plenoxels: Taking the "hybrid" concept to its limit, Fridovich-Keil et al. proposed *Plenoxels*, which eliminates the neural network entirely. It optimizes a sparse voxel grid where each voxel stores spherical harmonics coefficients directly. This work demonstrated that the photorealism of NeRF stems primarily from the differentiable volumetric rendering formulation rather than the neural network itself [13].

Point-NeRF: While grid-based methods offer speed, they struggle with empty space inefficiency. Xu et al. proposed *Point-NeRF*, which combines accurate 3D point clouds (from MVS) with neural radiance fields. Instead of querying a dense grid, Point-NeRF stores neural features on discrete points. During ray-marching, features are aggregated from nearby points to predict radiance. This approach effectively skips empty space and serves as a significant precursor to 3DGS, demonstrating the potential of point-based neural rendering [51].

Advanced Optimization: To address aliasing artifacts in grid-based methods, **Zip-NeRF** [52] combines the speed of grids with the anti-aliasing properties of multi-sampling, setting a new benchmark for static quality. For dynamic

robustness, **RoDynRF** [53] introduces robust loss functions to handle occlusions and outliers in monocular video, significantly improving stability in uncontrolled environments.

3.4.3 Summary

Pros: These methods offer a superior balance between speed and quality. Training is orders of magnitude faster than pure MLP-based NeRFs, and inference is often real-time or near real-time.

Cons: They typically consume more memory (VRAM) than compact pure-implicit models. Additionally, implementing efficient CUDA kernels for hash lookups or sparse grids is technically more complex than standard MLPs.

4 Datasets and Metrics

4.1 Common Datasets

To evaluate the performance of dynamic scene modeling algorithms, various datasets have been proposed, ranging from controlled synthetic environments to complex real-world captures. We categorize the mainstream datasets into synthetic and real-world collections.

4.1.1 Synthetic Datasets

D-NeRF Dataset — Pumarola et al. introduced this dataset as a standard benchmark for monocular dynamic view synthesis. It consists of 8 synthetic scenes generated using physically-based rendering. The scenes exhibit diverse motion patterns, categorized into:

- **Articulated Motion:** *T-Rex, Lego, Tractor*.
- **Non-rigid / Deformable Motion:** *Hell Warrior, Mutant, Hook, Stand Up, Jumping Jacks*.
- **Topological Changes:** *Bouncing Balls*.

Since it provides perfect camera poses and ground truth depth, it serves as the primary testbed for validating deformation accuracy and rendering quality without the interference of pose errors errors[1].

4.1.2 Real-world Datasets

Neural 3D Video (N3V) Dataset — Proposed by Li et al. (DyNeRF), this dataset focuses on high-fidelity reconstruction from synchronized **multi-view** videos (typically 18-21 cameras). Recorded at 2.7K resolution and 30 FPS, it features challenging indoor scenarios with complex illumination and volumetric effects. Representative scenes include *Flame Salmon* (fire/smoke), *Coffee Martini* (liquid/reflections), *Cook Spinach* (steam), and *Cut Roasted Beef* (texture changes). This dataset is widely used to evaluate methods targeting immersive video and photo-realistic rendering of complex materials[2].

1. D-NeRF dataset available at: https://www.albertpumarola.com/research/D-NeRF/index.html
2. N3D dataset available at: https://github.com/facebookresearch/Neural_3D_Video

TABLE 1
Summary of Common Datasets for Dynamic Scene Modeling

Dataset	Type	Cameras	Key Challenges
D-NeRF [10]	Synthetic	Monocular	Large deformations, Articulated motion
N3V (DyNeRF) [26]	Real	Multi-view	Fire, Smoke, Reflections, High-res
HyperNeRF [21]	Real	Monocular	Topological changes, Casual capture
Technicolor	Real	Multi-view	Studio quality, Human performance

HyperNeRF Dataset — To address the limitation of deformation fields in handling topological changes, Park et al. collected this dataset using casually captured **monocular** videos. It emphasizes "topologically varying" objects where standard deformation fields fail. Key scenes include *3D Printer* (appearing/disappearing materials), *Broom* (bristles spreading), *Chicken* (elastic deformation), and *Peel Banana* (separation of geometry). It also provides a "Validation Rig" subset for quantitative evaluation[3].

Technicolor Light Field Dataset — Originally designed for light field research, this dataset contains high-quality multi-view video sequences captured in a studio setting with a 4x4 camera rig. Scenes such as *Fabien*, *Painter*, and *Birthday* involve human performances and are frequently used to benchmark hybrid and explicit representations for free-viewpoint video[4].

4.2 Evaluation Metrics

To quantitatively assess the performance of dynamic scene modeling methods, the community relies on a set of standard metrics covering reconstruction quality, perceptual fidelity, and computational efficiency.

4.2.1 Reconstruction Quality Metrics

The quality of novel view synthesis is typically assessed using standard image quality assessment (IQA) metrics. These metrics evaluate the pixel-level fidelity and perceptual similarity between the rendered images and ground truth references.

Peak Signal-to-Noise Ratio (PSNR): PSNR is the most widely used metric for measuring pixel-level reconstruction accuracy. It calculates the ratio between the maximum possible power of a signal and the power of corrupting noise, expressed in decibels (dB):

$$\text{PSNR} = 10 \cdot \log_{10} \left(\frac{\text{MAX}_I^2}{\text{MSE}} \right) \tag{14}$$

where MAX_I is the maximum pixel value (e.g., 255) and MSE is the Mean Squared Error between the rendered and ground truth images. While high PSNR indicates low pixel

3. HyperNeRF dataset available at: https://hypernerf.github.io
4. Technicolor dataset available at: https://www.interdigital.com/data_sets/light-field-dataset

TABLE 2
Quantitative Comparison of State-of-the-Art Methods. We report average metrics across standard test sets. **Class** denotes the method category: Imp. (Implicit), Hyb. (Hybrid), and Exp. (Explicit). Best results are highlighted in **bold**, and second best are underlined.

Method	Class	Year	Rendering Quality			Efficiency		Storage
			PSNR ↑	SSIM ↑	LPIPS ↓	FPS ↑	Train Time ↓	Size (MB)
Dataset: D-NeRF Synthetic (Monocular, 800 × 800)								
D-NeRF [10]	Imp.	2021	30.31	0.96	0.039	< 1	$\sim$ 2 days	$<$ **5**
TiNeuVox [54]	Hyb.	2022	32.67	0.97	0.040	1.5	28 min	48
HexPlane [55]	Hyb.	2023	31.04	0.97	0.040	2.5	11 min	38
K-Planes [48]	Hyb.	2023	31.61	0.97	0.038	0.9	52 min	418
Deformable 3DGS [28]	Exp.	2024	33.07	0.98	**0.018**	45	$<$ **10 min**	$\sim$ 20
4D-GS [17]	Exp.	2024	**34.05**	**0.98**	0.020	**82**	8 min	18
Dataset: HyperNeRF / N3V (Real-world, various resolutions)								
Nerfies [11]	Imp.	2021	22.20	0.80	0.193	< 1	$\sim$ 16 hours	$<$ **10**
HyperNeRF [21]	Imp.	2021	22.40	0.81	0.185	< 1	$\sim$ 32 hours	$<$ **10**
DyNeRF [26]	Imp.	2022	29.58	0.92	0.098	< 1	$\sim$ 1 week	28
HexPlane [55]	Hyb.	2023	31.70	**0.95**	0.075	0.2	12 hours	250
4D-GS [17]	Exp.	2024	31.15	0.94	**0.049**	**30**	**40 min**	90

error, it often favors smooth/blurry results over sharp but slightly misaligned textures.

Structural Similarity Index (SSIM): Unlike PSNR, SSIM measures the structural similarity between two images, which correlates better with the human visual system's sensitivity to structural information. It considers luminance, contrast, and structure:

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{15}$$

where μ and σ represent the mean and variance of image windows. SSIM values range from -1 to 1, with 1 indicating perfect similarity.

Learned Perceptual Image Patch Similarity (LPIPS): Standard metrics (PSNR/SSIM) often fail to capture high-frequency details and perceptual quality. LPIPS measures the distance between image patches in the deep feature space of a pre-trained network (e.g., VGG or AlexNet). Lower LPIPS scores indicate better perceptual quality, aligning more closely with human judgment, especially for dynamic scenes where slight motion misalignments penalize PSNR heavily but look plausible to humans.

4.2.2 *Efficiency Metrics*

In addition to visual quality, the computational efficiency of a method determines its practicality for real-time applications (such as VR/AR) and its scalability for large datasets.

Frames Per Second (FPS): This metric evaluates the inference/rendering speed. For real-time applications (VR/AR), a method typically needs to achieve $\geq$ 30 FPS. Explicit methods like 3DGS often reach 100+ FPS, while implicit NeRFs may struggle at < 1 FPS.

Training Time: This measures the duration required to optimize the scene representation from scratch. It is a critical factor for scalability. Recent Gaussian-based methods have reduced training time from days (D-NeRF) to minutes (4D-GS).

5 Comparative Analysis

In this section, we provide a systematic comparison of the representative methods discussed in Section III. We evaluate them based on quantitative reconstruction quality, computational efficiency, and storage requirements.

5.1 Quantitative Benchmarks

Table 2 summarizes the performance of Implicit (NeRF-based), Hybrid, and Explicit (Gaussian-based) frameworks across standard benchmarks. The comparison is divided into synthetic scenes (D-NeRF dataset) and complex real-world scenes (HyperNeRF and N3V datasets).

5.1.1 *Quality vs. Efficiency Trade-off*

The benchmarks reveal a clear paradigm shift. Implicit methods (e.g., D-NeRF, HyperNeRF) produce high-fidelity results with extremely compact storage (typically < 10 MB) due to the spectral bias of MLPs. However, their inference speed is prohibitively slow (< 1 FPS) for interactive applications. Hybrid methods (e.g., HexPlane, TiNeuVox) significantly reduce training time to minutes by utilizing explicit spatial structures, yet their rendering speed remains limited by the ray-marching/accumulation step.

Explicit Gaussian-based methods (Deformable 3DGS, 4D-GS) break this bottleneck. As shown in Table 2, **4D-GS** achieves real-time performance (82 FPS on synthetic data, 30 FPS on high-res real data) while matching or surpassing the PSNR/SSIM of offline implicit methods. This efficiency gain comes from replacing costly volumetric sampling with rasterization. However, this comes at the cost of increased storage requirements (often $2\times$ to $10\times$ larger than implicit models) to store millions of Gaussian primitives.

5.2 Qualitative Visual Comparisons

While quantitative metrics provide a numerical ranking, they often fail to capture perceptual artifacts such as ghosting, over-smoothing, or geometric distortion. In Figure 5, we provide a visual comparison of representative methods across three challenging scenarios: topological changes, fine geometric structures, and rapid motion.

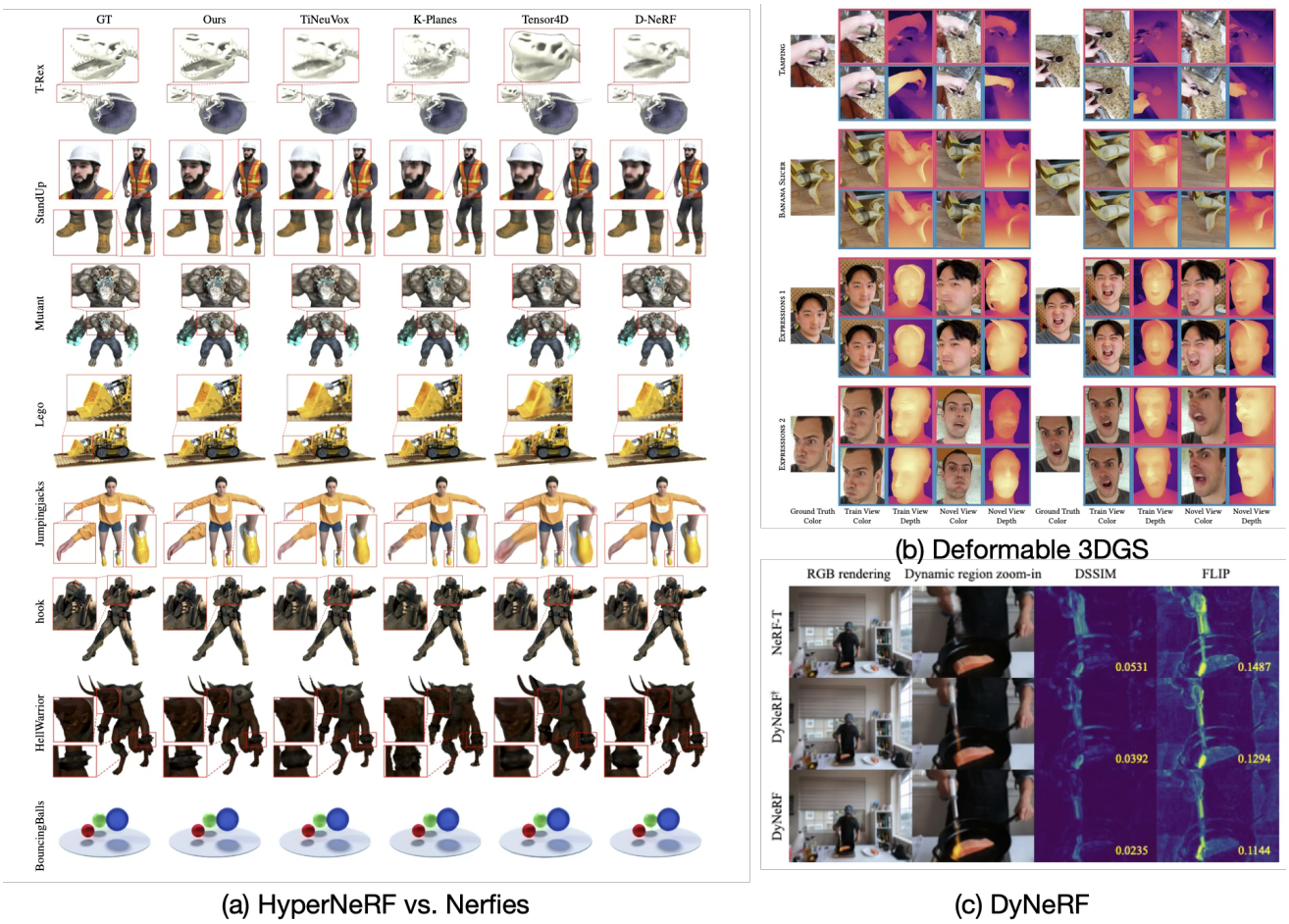

Fig. 5. **Qualitative comparison on challenging scenarios. Left:** Handling topological changes. Deformation fields (Nerfies) cause stretching artifacts, while HyperNeRF handles discontinuities correctly [21]. **Top Right:** Reconstructing fine details. Implicit methods (TiNeuVox) tend to smooth out thin structures, whereas explicit Deformable 3DGS preserves sharp geometry [28]. **Bottom Right:** Handling rapid motion. Basic temporal conditioning leads to ghosting (NeRF-T), while latent-conditioned methods (DyNeRF) recover sharp dynamics [26].

5.2.1 *Topological Adaptability*

Implicit deformation fields (e.g., Nerfies) rely on continuous mapping functions, which inherently struggle with discontinuous motions. As shown in the **left panel** of Figure 5, when representing a splitting object (e.g., opening a mouth or slicing dough), deformation-based methods tend to produce unnatural stretching artifacts. In contrast, **HyperNeRF** successfully handles these topology changes by lifting the deformation into a higher-dimensional hyperspace, allowing for discontinuous slicing [21].

5.2.2 *High-Frequency Details*

Explicit representations generally exhibit superior capability in preserving high-frequency details compared to implicit or hybrid fields. As illustrated in the **top right** of Figure 5 (T-Rex scene), grid-based methods like TiNeuVox often result in over-smoothed geometry due to the limited resolution of the voxel grid or the spectral bias of MLPs. **Deformable 3DGS**, leveraging discrete Gaussian primitives, maintains sharp edges and intricate structures (e.g., the ribs of the skeleton) without blurring [28].

5.2.3 *Motion Blur and Ghosting*

Modeling rapid motion from monocular or sparse video remains a significant challenge. Naive temporal extensions (e.g., NeRF-T) often fail to disentangle geometry from motion blur, leading to "ghosting" artifacts where multiple timestamps appear overlaid. As shown in the **bottom right** of Figure 5, methods utilizing compact latent codes (like **DyNeRF**) demonstrate better temporal consistency, effectively reconstructing sharp appearances even in the presence of fast-moving elements like flames or fluids [26].

5.2.4 *Artifact Analysis*

While 3DGS excels in rendering speed, it is prone to specific artifacts not typically seen in NeRFs. Due to the discrete nature of primitives, "floaters" (randomly placed opaque Gaussians) often appear in near-camera regions. Furthermore, in areas with sparse observations, optimization can lead to "needle-like" elongated Gaussians, which degrade visual quality when viewed from novel angles. Recent regularization techniques [42], [43] attempt to mitigate these issues, but they remain an open problem for unconstrained dynamic scenes.

5.3 Efficiency Analysis

Beyond visual fidelity, the practical deployment of dynamic scene modeling hinges on computational efficiency. We analyze the trade-offs across three key dimensions: training convergence, rendering latency, and storage overhead.

5.3.1 Training Efficiency

Implicit methods (e.g., D-NeRF, DyNeRF) typically require the longest training times (ranging from hours to days) due to the dense sampling required by volumetric ray-marching during backpropagation. The global nature of MLPs also makes it difficult to update local regions independently. Hybrid representations (e.g., TiNeuVox, HexPlane) dramatically accelerate this process (to minutes) by using explicit feature grids that allow for local updates and faster convergence. Explicit methods (e.g., 4D-GS) further push the boundary, often converging within minutes (e.g., 8 minutes for 4D-GS [17]). This speedup is attributed to the rasterization-based pipeline, which avoids expensive volumetric sampling and allows gradients to propagate directly to discrete primitives.

5.3.2 Rendering Speed

Rendering speed is the primary bottleneck for implicit methods. Standard D-NeRF operates at < 1 FPS due to the necessity of evaluating a deep MLP hundreds of times per ray. Hybrid methods improve this by replacing MLP evaluations with feature lookups, achieving interactive rates (1–10 FPS) but often falling short of real-time requirements for high-resolution output. Explicit Gaussian-based methods demonstrate a decisive advantage here, achieving real-time performance ($30 \sim 100+$ FPS). This is achieved by projecting 3D primitives directly onto the 2D screen (splatting), bypassing the ray-marching loop entirely [17], [28].

5.3.3 Storage and Memory Footprint

Storage (Disk) — Implicit methods are the most storage-efficient, often compressing an entire dynamic sequence into a few megabytes (MB) of MLP weights (e.g., D-NeRF < 5 MB). Explicit methods, conversely, suffer from a "curse of storage" as they need to store millions of Gaussian parameters. A naive per-frame 3DGS could consume gigabytes of data. However, recent optimized frameworks like 4D-GS reduce this footprint significantly (e.g., ~ 18 MB) by using tensor decomposition (HexPlane) to share features across time, making them competitive with compact implicit models [17], [26].

Memory (VRAM) — During training and inference, explicit and hybrid methods typically consume more GPU memory (VRAM) to store feature grids or point clouds, whereas pure implicit methods have a lower peak memory footprint but higher computational load.

5.3.4 Summary of Trade-offs

For low-storage, offline applications: Implicit methods (D-NeRF) remain a strong candidate due to their extreme compactness.

For real-time applications: Explicit methods (4D-GS) are currently the only viable option for high-resolution (> 1080p) real-time rendering.

For balanced needs: Hybrid representations offer a middle ground, providing faster training than pure NeRFs with manageable storage costs.

6 Open Challenges and Future Directions

Despite the remarkable progress in dynamic scene modeling, several critical challenges remain unresolved. In this section, we identify three emerging research directions that are pivotal for the next generation of immersive media.

6.1 Storage Efficiency and Compression

While explicit methods like 3DGS achieve real-time rendering, they suffer from a high storage footprint compared to implicit MLPs. For instance, a naive dynamic 3DGS implementation can consume gigabytes of storage for a short sequence, as it may spawn millions of primitives. Although methods like **4D-GS** utilize HexPlane decomposition to reduce this to ~ 18 MB, this is still orders of magnitude larger than standard 2D video codecs (e.g., H.264/HEVC).

- **Challenge:** Developing standard compression algorithms for 4D primitives.
- **Future Direction:** Future work focuses on vector quantization and codebook-based learning. Recent attempts like **Compact3D** [56] and **EAGLES** [57] have successfully reduced the storage of Gaussian fields by over $10\times$ without significant quality loss through quantized attribute encoding. Additionally, distinct "Level of Detail" (LOD) strategies could be developed to stream dynamic 3D content efficiently over limited bandwidth networks. Furthermore, optimizing the rasterization pipeline for **WebGPU and Mobile Hardware** is crucial. Enabling high-fidelity 4D rendering on edge devices (VR headsets, smartphones) will be the key to mass adoption in the Metaverse.

6.2 Handling Long-Duration Videos

Most current benchmarks (e.g., D-NeRF, N3V) are limited to short clips (typically < 10 seconds). Scaling reconstruction to minute-long or hour-long videos presents severe bottlenecks. As noted in **DyNeRF**, training on a 10-second clip already involves billions of ray samples and significant computational resources.

- **Challenge:** The "Curse of Dimensionality" in the temporal axis. Global MLPs struggle to fit long sequences without forgetting (catastrophic forgetting), while explicit methods run out of VRAM.
- **Future Direction:** We envision a shift towards **Streaming Architectures**. Instead of optimizing a global model for the entire video, future systems might employ a "sliding window" approach or keyframe-based residue learning. This would allow the model to reconstruct infinite-length streams by dynamically loading and unloading local spatiotemporal blocks, enabling applications like live volumetric telepresence.

6.3 Integration with Generative Models

The rapid rise of video generation models (e.g., **Sora**, **Veo**) has democratized 2D content creation, but these models often lack multi-view consistency and explicit 3D geometry.

- **Challenge:** Bridging the gap between 2D generative priors and 3D/4D reconstruction pipelines.
- **Future Direction:** Dynamic Scene Modeling serves as the ideal "3D lifting" mechanism for generative AI. Pioneering works like **DreamFusion** [58] utilized SDS loss for NeRF generation. Recently, **DreamGaussian** [59] adapted this to explicit splatting for minute-level generation. **LGM** [60] further demonstrates the potential of training large-scale transformers to directly predict 3D Gaussians from single images.
  Text-to-4D: Combining text-to-video models with Dynamic 3DGS to generate 4D assets directly from prompts. Seminal works like **MAV3D** [61] pioneered this direction using NeRFs. More recently, **Align Your Gaussians** [62] and **Animate124** [63] leverage the explicit nature of Gaussians to animate static objects or generate 4D scenes from single images with higher fidelity and speed.
  4D Editing: Using generative diffusion models to stylize or edit reconstructed 4D scenes (e.g., changing the weather in a captured dynamic video) while maintaining temporal consistency.

6.4 Disentanglement and Relighting

A fundamental limitation of current SOTA methods (including DyNeRF and 3DGS) is the problem of **"baked-in" lighting**. Existing frameworks typically model the final radiance color c directly via Spherical Harmonics (SH) or latent codes, which entangles the object's surface material properties with the environmental lighting at the time of capture [17], [26].

- **Challenge:** The inability to modify lighting conditions after training. Since shadows and reflections are "frozen" into the geometry or texture, moving these objects into a new virtual environment (e.g., in a game engine or Metaverse) results in visual inconsistencies.
- **Future Direction:** The community is moving towards **Dynamic Inverse Rendering**. This involves decomposing the radiance field into physically based components: geometry, material (BRDF: albedo, roughness), and spatially-varying lighting. Recent works like **SuGaR** [64] have made strides in extracting explicit meshes from Gaussian clouds, which is a prerequisite for physical simulations and relighting. Integrating physics-based rendering (PBR) pipelines with dynamic 3DGS will be crucial for achieving editable, relightable 4D assets.

7 Conclusion

In this survey, we have presented a comprehensive taxonomy of dynamic scene modeling, tracing the evolution from implicit deformation fields to the latest explicit 4D Gaussian Splatting techniques. We analyzed the trade-offs between different paradigms:

Implicit methods (e.g., D-NeRF) excel in storage efficiency (< 5 MB) but suffer from slow training.

Hybrid representations (e.g., HexPlane) bridge the gap by leveraging spatial data structures for faster convergence.

Explicit methods (e.g., 4D-GS) have revolutionized the field by enabling real-time rendering (> 80 FPS) suitable for interactive applications.

However, as discussed in our analysis of open challenges, the journey is far from over. Future research must address the bottlenecks of storage compression, long-duration video processing, and the disentanglement of lighting from geometry. We believe that the convergence of explicit dynamic representations with generative AI priors will pave the way for the next generation of immersive, editable, and photorealistic 4D content creation.

References

[1] Apple Inc. Expression estimation for headsets using low-profile antenna and impedance characteristic sensing, 2023. Patent/Technical Report.

[2] Johannes L Schönberger, Jan-Michael Frahm, Marc Pollefeys, Paul-Edouard Sarlin, and S Liu. Colmap. https://colmap.github.io/index.html, 2024.

[3] Richard A. Newcombe, Dieter Fox, and Steven M. Seitz. Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 343–352, 2015.

[4] Mingsong Dou, Sameh Khamis, Yury Degtyarev, Philip Davidson, Sean Ryan Fanello, Adarsh Kowdle, Sergio Orts Escolano, Christoph Rhemann, David Kim, Jonathan Taylor, et al. Fusion4d: Real-time performance capture of challenging scenes. *ACM Transactions on Graphics (TOG)*, 35(4):1–13, 2016.

[5] 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. *arXiv preprint arXiv:2003.08934*, 2020.

[6] Kai Zhang, Peng Song, and Thomas A Funkhouser. Nerf++: Analyzing and improving neural radiance fields. *arXiv preprint arXiv:2010.07492*, 2020.

[7] Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Ren Ng. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5855–5864, 2021.

[8] Jonathan T Barron et al. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *CVPR*, 2022.

[9] Dor Verbin et al. Ref-nerf: Structured view-dependent appearance for neural radiance fields. In *CVPR*, 2022.

[10] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. *arXiv preprint arXiv:2011.13961*, 2020.

[11] Keunhong Park, Utkarsh Sinha, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Steven M Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable neural radiance fields. *arXiv preprint arXiv:2011.12948*, 2021.

[12] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *CVPR*, 2022.

[13] Alex Yu, Sara Fridovich-Keil, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. *arXiv preprint arXiv:2112.05131*, 2021.

[14] Peng Wang et al. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. In *NeurIPS*, 2021.

[15] Lior Yariv et al. Volsdf: Volume rendering of neural implicit surfaces. In *NeurIPS*, 2021.

[16] Bernhard Kerbl, Georgios Wang, Fabrice Rousselle, and Vladlen Koltun. 3d gaussian splatting for real-time radiance field rendering. *arXiv preprint arXiv:2304.08914*, 2023.

[17] Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 4d gaussian splatting for real-time dynamic scene rendering. *arXiv preprint arXiv:2310.08528*, 2024.

[18] Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. *arXiv preprint arXiv:2308.09713*, 2023.

[19] Edgar Tretschk et al. Non-rigid neural radiance fields: Reconstruction and novel view synthesis of a dynamic scene from monocular video. In *ICCV*, 2021.

[20] Benjamin Attal et al. Hyperreel: High-fidelity 6-dof video with ray-conditioned sampling. In *CVPR*, 2023.

[21] 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. *arXiv preprint arXiv:2106.13228*, 2021.

[22] Sida Peng, Yuanqing Zhang, Yinghao Xu, Qianqian Wang, Qing Shuai, Hujun Bao, and Xiaowei Zhou. Neural body: Implicit neural representations with structured latent codes for novel view synthesis of dynamic humans. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9054–9063, 2021.

[23] Chung-Yi Weng, Brian Curless, Pratul P Srinivasan, Jonathan T Barron, and Ira Kemelmacher-Shlizerman. Humannerf: Free-viewpoint rendering of moving people from monocular video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16210–16220, 2022.

[24] Sida Peng, Junting Dong, Qianqian Wang, Shangzhan Zhang, Qing Shuai, Xiaowei Zhou, and Hujun Bao. Animatable neural radiance fields for modeling dynamic human bodies. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 14314–14323, 2021.

[25] Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. *arXiv preprint arXiv:2011.13084*, 2021.

[26] Tianye Li, Mira Slavcheva, Michael Zollhoefer, Simon Green, Christoph Lassner, Changil Kim, Tanner Schmidt, Steven Lovegrove, Michael Goesele, Richard Newcombe, and Zhaoyang Lv. Neural 3d video synthesis from multi-view video. *arXiv preprint arXiv:2103.02597*, 2021.

[27] Wenqi Xian, Jia-Bin Huang, Johannes Kopf, and Changil Kim. Space-time neural irradiance fields for free-viewpoint video. *arXiv preprint arXiv:2011.12950*, 2020.

[28] Ziyi Yang, Xinyu Gao, Wen Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. Deformable 3d gaussians for high-fidelity monocular dynamic scene reconstruction. *arXiv preprint arXiv:2309.13101*, 2023.

[29] Shoukang Hu and Yukara Kasai. Gauhuman: Articulated gaussian splatting from monocular human videos. In *CVPR*, 2024.

[30] Muhammed Kocabas Jiang et al. Hugs: Human gaussian splatting. In *CVPR*, 2024.

[31] Shenhan Qian et al. Gaussianavatar: Towards realistic human avatar modeling from monocular video using drivable 3d gaussians. In *CVPR*, 2024.

[32] Shunyao Zheng et al. Gps-gaussian: Generalizable pixel-wise 3d gaussian splatting for real-time human novel view synthesis. In *CVPR*, 2024.

[33] Guy Gafni et al. Nerface: Dynamic neural radiance fields for monocular face avatar reconstruction. In *CVPR*, 2021.

[34] Jun Xiang et al. Flashavatar: High-fidelity head avatar with efficient gaussian embedding. In *CVPR*, 2024.

[35] Yuxiao Zhang et al. Instaavatar: One-shot avatar from a single image. In *CVPR*, 2024.

[36] Yufan Chen et al. Monogaussianavatar: Monocular gaussian point-based head avatar. In *CVPR*, 2024.

[37] Tianyi Xie et al. Physgaussian: Physics-integrated 3d gaussians for generative dynamics. In *CVPR*, 2024.

[38] Youtian Lin et al. Gaussian-flow: 4d reconstruction with dynamic 3d gaussian splatting. In *CVPR*, 2024.

[39] Heng Yu et al. Cogs: Controllable gaussian splatting. *CVPR*, 2024.

[40] Yi-Hua Huang et al. Sc-gs: Sparse-controlled gaussian splatting for editable dynamic scenes. In *CVPR*, 2024.

[41] Zhan Li et al. Spacetime gaussian feature splatting for real-time dynamic view synthesis. In *CVPR*, 2024.

[42] Zehao Yu et al. Mip-splatting: Alias-free 3d gaussian splatting. In *CVPR*, 2024.

[43] Tao Lu et al. Scaffold-gs: Structured 3d gaussians for view-adaptive rendering. In *CVPR*, 2024.

[44] Chi Yan et al. Gs-slam: Dense visual slam with 3d gaussian splatting. In *CVPR*, 2024.

[45] Nikhil Keetha et al. Splat-slam: Globally consistent dense slam with 3d gaussians. In *CVPR*, 2024.

[46] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *arXiv preprint arXiv:2201.05989*, 2022.

[47] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. *arXiv preprint arXiv:2203.09517*, 2022.

[48] Sara Fridovich-Keil, Giacomo Meanti, Frederik Warburg, Benjamin Recht, and Angjoo Kanazawa. K-planes: Explicit radiance fields in space, time, and appearance. *arXiv preprint arXiv:2301.10241*, 2023.

[49] Ruixiang Shao et al. Tensor4d: Efficient neural 4d decomposition for high-fidelity dynamic reconstruction and rendering. In *CVPR*, 2023.

[50] Lingzhi Li et al. Streamrf: Streaming radiance fields for real-time photorealistic rendering of dynamic scenes. In *ICCV*, 2023.

[51] Qiangeng Xu, Zexiang Xu, Julien Philip, Sai Bi, Zhixin Shu, Kalyan Sunkavalli, and Ulrich Neumann. Point-nerf: Point-based neural radiance fields. *arXiv preprint arXiv:2201.08845*, 2022.

[52] Jonathan T Barron et al. Zip-nerf: Anti-aliased grid-based neural radiance fields. In *ICCV*, 2023.

[53] Yu-Lun Liu et al. Rodynrf: Robust dynamic neural radiance fields. In *CVPR*, 2023.

[54] 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. *arXiv preprint arXiv:2205.15285*, 2022.

[55] Ang Cao and Justin Johnson. Hexplane: A fast representation for dynamic scenes. *arXiv preprint arXiv:2301.09632*, 2023.

[56] Joo Chan Lee et al. Compact 3d gaussian representation for radiance field. In *CVPR*, 2024.

[57] Sharath Girish et al. Eagles: Efficient accelerated 3d gaussians with lightweight encodings. *arXiv preprint arXiv:2312.04564*, 2023.

[58] Ben Poole et al. Dreamfusion: Text-to-3d using 2d diffusion. In *ICLR*, 2023.

[59] Jiaxiang Tang et al. Dreamgaussian: Generative gaussian splatting for efficient 3d content creation. In *ICLR*, 2024.

[60] Jiaxiang Tang et al. Lgm: Large multi-view gaussian model for high-resolution 3d content creation. In *CVPR*, 2024.

[61] Uriel Singer et al. Mav3d: Make-a-video3d for dynamic 3d scene generation. In *ICLR*, 2023.

[62] Huan Ling et al. Align your gaussians: Text-to-4d with dynamic 3d gaussians and video diffusion models. In *CVPR*, 2024.

[63] Yuyang Zhao et al. Animate124: Animating one image to 4d dynamic scene. In *CVPR*, 2024.

[64] Antoine Guédon and Vincent Lepetit. Sugar: Surface-aligned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering. In *CVPR*, 2024.