This WACV paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

TED-4DGS: Temporally Activated and Embedding-based Deformation for 4DGS Compression

Cheng-Yuan Ho[1], He-Bi Yang[1], Jui-Chiu Chiang[2], Yu-Lun Liu[1], Wen-Hsiao Peng[1]
[1]National Yang Ming Chiao Tung University, Taiwan
[2]National Chung Cheng University, Taiwan
{kelvinhe0218.cs12, mrrrimge32.cs13}@nycu.edu.tw, rachel@ccu.edu.tw
yulunliu@cs.nycu.edu.tw, wpeng@cs.nycu.edu.tw

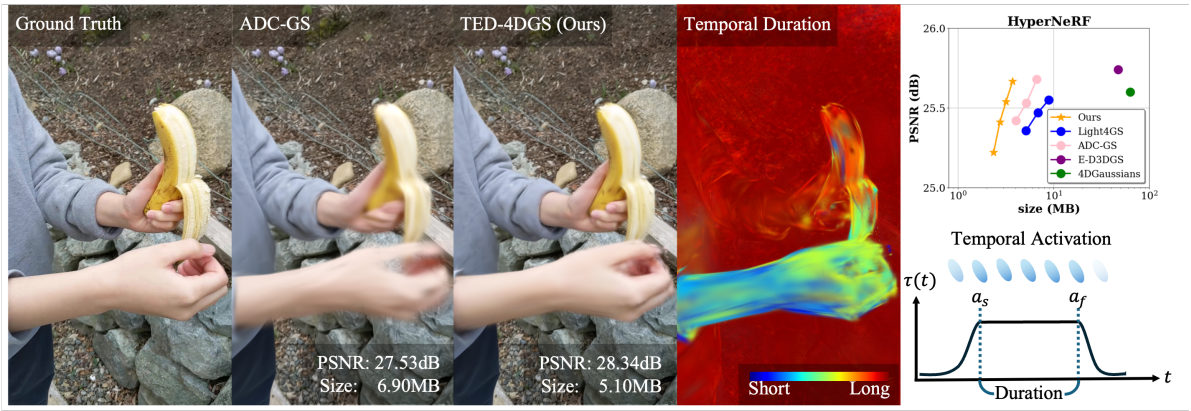


Figure 1. **Overview of TED-4DGS.** *Left*: Qualitative comparison on a banana scene. Our TED-4DGS reconstructs the scene with superior rendering quality compared to ADC-GS. It achieves a 26% file size reduction while closely matching the ground-truth view. *Centre*: Temporal duration map. Static background regions reuse long-duration Gaussian primitives, whereas occluded parts of the hand and banana are represented by short-duration primitives, demonstrating the effectiveness of temporal activation. *Right-top*: Rate-distortion comparison on the HyperNeRF [36] benchmark. Our TED-4DGS attains higher PSNR with smaller file sizes than prior methods. *Right-bottom*:Illustration of the learnable temporal-activation function, which activates a Gaussian primitive from its appearance (a_s) to disappearance (a_f).

Abstract

Building on the success of 3D Gaussian Splatting (3DGS) in static 3D scene representation, its extension to dynamic scenes–commonly referred to as 4DGS or dynamic 3DGS–has attracted increasing attention. However, designing more compact, efficient deformation schemes together with rate-distortion-optimized compression strategies for dynamic 3DGS representations remains an underexplored area. Prior methods either rely on space-time 4DGS with overspecified, short-lived Gaussian primitives or on canonical 3DGS with deformation that lacks explicit temporal control. To address this, we present ***TED-4DGS****, a temporally activated and embedding-based deformation scheme for rate-distortion-optimized 4DGS compression that unifies the strengths of both families. TED-4DGS is built on a sparse anchor-based 3DGS representation. Each canonical anchor is assigned with learnable temporal-activation parameters to specify its appearance and disappearance transitions over time, while a lightweight per-anchor temporal embedding queries a shared deformation bank to produce anchor-specific deformation. For rate-distortion compression, we incorporate an implicit neural representation (INR)-based hyperprior to model anchor attribute distributions, along with a channel-wise autoregressive model to capture intra-anchor correlations. With these novel elements, our scheme achieves the state-of-the-art rate-distortion performance on several commonly used real-world datasets. To the best of our knowledge, this work represents one of the first attempts to pursue a rate-distortion-optimized compression framework for dynamic 3DGS representations.*

1. Introduction

Reconstructing dynamic scenes from multi-view videos has long been a central challenge in 3D vision, with applications ranging from novel view synthesis and free-viewpoint rendering to dynamic scene understanding. Building on the success of 3D Gaussian Splatting (3DGS) in static 3D scene representation, its extension to dynamic scenes–commonly referred to as 4DGS or dynamic 3DGS–has attracted increasing attention. Analogous to the evolution from 2D images to 2D videos, leveraging temporal redundancy for more efficient representation calls for principled design in both temporal modeling and compression strategies. This work represents one of the first attempts to pursue a rate-distortion-optimized compression framework for dynamic 3DGS representations. Unlike previous methods that primarily aim to reduce memory footprint or accelerate rendering for dynamic 3DGS, our approach integrates entropy coding into the compression of Gaussian primitives for storage and transmission purposes.

Existing works on dynamic 3DGS representations generally fall into two categories: the first represents the dynamic scene using explicit space-time 4DGS, while the second models deformation fields over canonical 3D Gaussian primitives (i.e. canonical 3DGS with deformation). Space-time 4DGS [8, 25, 39, 44] augments the spatial attributes of 3D Gaussian primitives with temporal parameters that specify their deformation and visibility across time. In contrast, canonical 3DGS with deformation [2, 15, 17, 27, 40] reconstructs dynamic scenes by learning deformation fields that warp canonical 3D Gaussian primitives over time. Since these methods define Gaussian primitives in a canonical space shared across time, they typically require significantly fewer Gaussian primitives compared to space-time 4DGS. Consequently, canonical 3DGS with deformation is often adopted as the core representation in rate-distortion-optimized compression frameworks [15, 27].

However, canonical 3DGS with deformation methods face a common challenge in handling occlusion and disocclusion. Since each Gaussian primitive persists throughout the entire sequence duration, the absence of temporal parameters that explicitly define its active duration may lead to peculiar deformation that is necessary, in some cases, to relocate non-contributing Gaussian primitives at specific time instances. The irregular deformation results in training instability and poses a challenge in rate-distortion compression.

To address this challenge, we propose **TED-4DGS**, a temporally activated and embedding-based deformation framework for 4DGS compression. We build TED-4DGS upon a sparse anchor-based 3DGS representation [8]. It features a per-anchor embedding-based deformation field, where each anchor is assigned a learnable temporal feature that serves as a query to a shared global deformation bank. To model occlusion and temporal visibility, we extend canonical 3D anchors into 4D by introducing temporal activation parameters, which specify the appearance and disappearance transitions of each anchor over time. Furthermore, inspired by recent advances in 3DGS rate–distortion compression [6, 46], we incorporate an implicit neural representation (INR)–based hyperprior to model anchor attribute distributions, along with a channel-wise autoregressive model to capture intra-anchor correlations. In addition, we observe that cross-camera color inconsistencies across multi-view captures can severely destabilize training and degrade rendering quality. To mitigate this issue, we introduce a lightweight Color Correction Module (CCM) that compensates for camera-specific color bias during training, leading to more stable optimization and color-consistent renderings across views.

Table 1. Comparative analysis of dynamic 3DGS approaches.

Model	Temporal Activation	RD Comp.	Motion Model
Space-time 4DGS			
4DGS [44]	✓	×	Linear
STG [25]	✓	×	Polynomial
4DScaffoldGS [8]	✓	×	Linear
FreeTimeGS [39]	✓	×	Linear
Canonical 3DGS + Deformation			
4DGaussians [40]	×	×	Coord.-based
E-D3DGS [2]	×	×	Embedding-based
Light4GS [27]	×	✓	Coord.-based
ADC-GS [15]	×	✓	Embedding-based
Ours (TED-4DGS)	✓	✓	Embedding-based

To sum up, our contributions include:

- We design a per-anchor embedding-based deformation network that leverages temporal features to query a shared global deformation bank, effectively capturing anchor-specific deformation.
- We extend static 3D anchors into 4D by introducing temporal activation parameters, promoting stable deformation and improved visibility modeling.
- We develop a rate-distortion-optimized compression framework that incorporates an INR-based hyperprior and a channel-wise autoregressive model for efficient attribute coding.

With these novel elements, our scheme achieves the state-of-the-art rate-distortion performance on several commonly used real-world datasets.

2. Related Work

2.1. Dynamic 3DGS Representations

Dynamic 3DGS representations can be broadly classified into two types: (1) space-time 4DGS and (2) canonical 3DGS with deformation, as illustrated in Tab. 1.

Space-time 4DGS is a natural extension of 3DGS. It augments the spatial attributes of 3D Gaussian primitives with temporal parameters that specify their motion and visibil-

ity across time. Methods in this category often feature simple parametric motion modeling, such as linear motion [8, 35, 39, 44] or polynomial motion [22, 25]. As such, recent methods in this category, e.g. 4DScaffoldGS [8] and FreeTimeGS [39], favor temporally short-lived Gaussian primitives, in order to model complex motion and enhance rendering fidelity. For instance, 4DScaffoldGS [8] introduces a temporal coverage-aware anchor growing scheme to promote short-lived anchors, while FreeTimeGS [39] applies a regularization strategy to penalize Gaussian primitives with extended lifespans across time. Although showing promising rendering fidelity, they significantly increase the number of Gaussian primitives or anchors, leading to higher storage demands.

In contrast, the notion of canonical 3DGS with deformation [9–11, 13, 16, 18–21, 28, 31, 37, 41, 42] aims to reconstruct dynamic scenes by deforming a set of canonical 3D Gaussian primitives that persist throughout the entire scene duration. Deformation signaling can be either coordinate-based or embedding-based. Coordinate-based methods model the deformation field as a space-time 4D function, which is typically cast as learning a dense 4D grid. To reduce memory consumption, 4DGaussians [40] decomposes the 4D grid with a hexplane representation [5], while Grid4D [17] employs 3D hash grids [34]. Despite these efforts, coordinate-based methods still incur considerable storage overhead. Instead of learning grid points, E-D3DGS [2], a representative embedding-based method, learns a per-Gaussian embedding for each Gaussian primitive, along with a number of temporal embeddings, one per time instance. These embeddings are concatenated or multiplied to arrive at the deformation for a given Gaussian primitive. When coupled with a sparse canonical 3DGS representation, embedding-based methods usually have much reduced parameter count, as compared to coordinate-based methods. Although both strategies are capable of modeling complex motion, the absence of temporal parameters that explicitly define the active duration of each Gaussian primitive may lead to peculiar motion that is necessary, in some cases, to relocate non-contributing Gaussian primitives at specific time instances.

Drawing inspiration from both types of dynamic 3DGS representations, this work integrates the temporal activation capability of space-time 4DGS with the strengths of embedding-based deformation in modeling complex motion. This enables the temporal duration of each canonical 3DGS Gaussian primitive to be defined explicitly, allowing non-contributing Gaussian primitives to be excluded from rendering without resorting to unnatural motion for their relocation. Consequently, our approach achieves memory and storage efficiency by modeling complex yet natural motion with a reduced set of Gaussian primitives.

2.2. Rate-Distortion-Based 3DGS Compression

Rate-distortion-optimized 3DGS compression is attracting growing interest due to increasing demands for efficient storage and transmission. The key challenge, which goes beyond merely reducing the parameter count of the 3DGS representation [1, 12, 23], lies in minimizing the entropy rate of these parameters to achieve compression efficiency.

Recent work [6, 7, 26, 38, 46] in 3DGS compression converges on anchor-based representations, e.g. ScaffoldGS [8, 29], paired with joint rate-distortion optimization via the hyperprior [4] and context models [14, 32, 33]. For rate-distortion optimization, HAC [6] introduces a hash grid-based hyperprior to capture spatial correlations among anchor attributes. To further exploit both inter-anchor and intra-anchor correlations, CAT-3DGS [46] employs a triplane-based hyperprior framework that features both spatial and channel-wise autoregressive modeling.

Rate-distortion-optimized compression for dynamic 3DGS scenes remains largely underexplored. Most early attempts [15, 27] build on canonical 3DGS representations with deformation, with a particular focus on how deformation is encoded. Light4GS [27] encodes hexplanes used for deformation modeling as images by adopting implicit neural representation (INR)-based image coding. ADC-GS [15] encodes embedding-based deformation with an autoregressive model applied along the channel dimension.

2.3. Comparison with Prior Work

As shown in Table 1, our work differs from previous work on space-time 4DGS by incorporating rate-distortion optimized compression and embedding-based deformation. The capability to model complex motion via embedding-based deformation enables the representation of dynamic scenes using fewer Gaussian primitives, thereby facilitating rate-distortion optimized compression.

In contrast to Light4GS [27] and ADC-GS [15], which rely on canonical 3DGS representations with deformation, our method introduces temporal activation to more effectively manage occlusion and disocclusion without introducing artificial deformation artifacts.

Last but not least, we replace the grid-based hyperprior, e.g. [6, 46], with a lightweight implicit neural representation (INR)-based hyperprior, which eliminates the signaling overhead associated with grid structures such as triplanes or hash tables.

3. Preliminary: ScaffoldGS

To construct an efficient 4DGS representation, our TED-4DGS leverages ScaffoldGS [30]–an anchor-based 3DGS representation–as its core representation framework. ScaffoldGS [30] is a compact and storage-efficient representation initially designed for static scenes. It introduces anchor

points (referred hereafter to as anchors) located on a predefined voxel grid. Each anchor encapsulates information for a fixed number K of Gaussian primitives. Their structural and color attributes, including the scale $\{\boldsymbol{s}_i\}_{i=0}^{K-1}$, rotation $\{\boldsymbol{r}_i\}_{i=0}^{K-1}$, color $\{\boldsymbol{c}_i\}_{i=0}^{K-1}$, and opacity $\{\alpha_i\}_{i=0}^{K-1}$, are decoded from the anchor's feature $\boldsymbol{f}$. Moreover, the anchor's position $\boldsymbol{x}$ and those $\{\boldsymbol{\mu}_i\}_{i=0}^{K-1}$ of its associated Gaussian primitives are related by $\{\boldsymbol{\mu}_i\}_{i=0}^{K-1} = \boldsymbol{x}+\boldsymbol{l}\cdot\{\boldsymbol{O}_i\}_{i=0}^{K-1}$, where $\{\boldsymbol{O}_i\}_{i=0}^{K-1}$ are learned offsets, which are scaled by a learned vector $\boldsymbol{l}$. Due to its storage-friendly design, ScaffoldGS has been widely adopted in many RD-optimized compression frameworks [6, 26, 38, 46].

4. Proposed Method: TED-4DGS

Based on ScaffoldGS [30], this work introduces a temporally activated and embedding-based deformation scheme for 4DGS compression.

First, we adopt a per-anchor embedding approach to modeling deformation. Anchors are classified into static and dynamic types via a learnable, binary temporal mask M_t. Each dynamic anchor is equipped with a temporal feature $\boldsymbol{\phi} \in \mathbb{R}^d$ that serves as a soft query to a global deformation bank $\boldsymbol{Z} \in \mathbb{R}^{\frac{F}{2}\times D}$ for F frames, in order to retrieve its deformation. This design supports flexible motion modeling while exploiting anchor spatial sparsity to minimize the parameter count for deformation signaling.

Second, we assign each canonical anchor an explicit temporal-activation parameter $\boldsymbol{\tau}$ that defines its active duration. This facilitates more accurate modeling of occlusion and dis-occlusion, and prevents the generation of unnatural deformation caused by relocating non-contributing anchors during scene rendering at specific time instances

Lastly, we adopt an INR-based hyperprior framework to model the coding distributions of anchor attributes, including the anchor's feature $\boldsymbol{f}$, scaling $\boldsymbol{l}$, offsets $\{\boldsymbol{O}_i\}_{i=0}^{K-1}$, temporal feature $\boldsymbol{\phi}$, and temporal activation $\boldsymbol{\tau}$.

4.1. System Overview

Fig. 2 illustrates our TED-4DGS framework. The representation of a dynamic scene begins with the generation of anchor points, each characterized by a position and a set of spatial and temporal attributes. The spatial attributes, adopted from ScaffoldGS [30], capture the geometry and appearance of Gaussian primitives in canonical space, while the temporal attributes encode deformation (; Section 4.2). To generate an anchor-specific deformation latent $\boldsymbol{z}_t^a$ for time instance t, the temporal feature $\boldsymbol{\phi}$ of a dynamic anchor is multiplied by a global deformation vector $z^{(t)}$, which is obtained via temporal interpolation between deformation vectors corresponding to uniformly sampled time instances in the global deformation bank $\boldsymbol{Z}$. The resulting latent $\boldsymbol{z}_t^a$ is fed into the deformation decoder F_{deform} to predict an anchor displacement $\Delta\boldsymbol{x}$ and a feature residual $\Delta\boldsymbol{f}$. These are used to update $\boldsymbol{x}$ and $\boldsymbol{f}$ as the deformed anchor position $\boldsymbol{x}'$ and feature $\boldsymbol{f}'$, respectively. Finally, $\boldsymbol{f}'$ is used to decode the scale $\{\boldsymbol{s}_i\}_{i=0}^{K-1}$ and rotation $\{\boldsymbol{r}_i\}_{i=0}^{K-1}$ for each Gaussian primitive, whereas the color $\{\boldsymbol{c}_i\}_{i=0}^{K-1}$ and opacity $\{\alpha_i\}_{i=0}^{K-1}$ are treated as time-invariant and are decoded from the canonical feature $\boldsymbol{f}$.

The rendering process utilizes both dynamic and static anchors. Dynamic anchors are temporally deformed by decoding their respective temporal attributes. With the deformed dynamic anchors and static anchors, the remaining procedure largely follows ScaffoldGS, except that each Gaussian primitive's opacity is additionally modulated by the temporal activation of its associated anchor to account for temporal visibility.

The entropy encoding (Section 4.4) of spatial and temporal features for each anchor leverages an INR-based hyperprior. It applies positional encoding to the anchor's position, with the resulting representation used to derive quantization step sizes and Gaussian parameters (i.e. means and variances) for entropy encoding associated attributes. Notably, for coding the anchor feature $\boldsymbol{f}$, which carries much structural and color information, we further incorporate a channel-wise autoregressive model to exploit intra correlations among its components. These attributes are jointly compressed in an rate-distortion-optimized manner. Additionally, an offset mask $\boldsymbol{M}_o$ is learned to suppress less critical Gaussian primitives.

The final compressed bitstream include (a) anchors' positions $\boldsymbol{x}$ and attributes, (b) the global deformation bank, (c) the network weights of the deformation decoder, Scaffold MLP decoder, hyperprior decoder, and channel-wise autoregressive model, and (d) the binary offset and temporal masks. The anchors' positions are stored in 16-bit floating-point format (FP16), while the network weights and the global deformation bank are in FP32. Both masks are entropy encoded.

4.2. Deformation Fields

To model dynamic motion, we introduce an embedding-based deformation field that predicts the temporal deformation of each anchor (Fig. 2c. The deformation is conditioned on both the per-anchor temporal feature $\boldsymbol{\phi} \in \mathbb{R}^d$ and a global deformation bank $\boldsymbol{Z} \in \mathbb{R}^{\frac{F}{2}\times D}$ for F frames. In video language, $\boldsymbol{Z}$ functions analogously to a set of optical flow maps while $\boldsymbol{\phi}$ serves as a pixel's location that determines where the relevant motion of that pixel should be extracted from these flow maps.

The deformation inference for a dynamic anchor at time t starts by interpolating temporally between deformation vectors for various time instances in the global deformation bank $\boldsymbol{Z}$. The outcome $\boldsymbol{z}^{(t)} = \text{interp}(\boldsymbol{Z}, t)$ is a time-specific $\boldsymbol{z}^{(t)} \in \mathbb{R}^D$, which (similar to a temporally-interpolated optical flow map in the context of video) encapsulates defor-

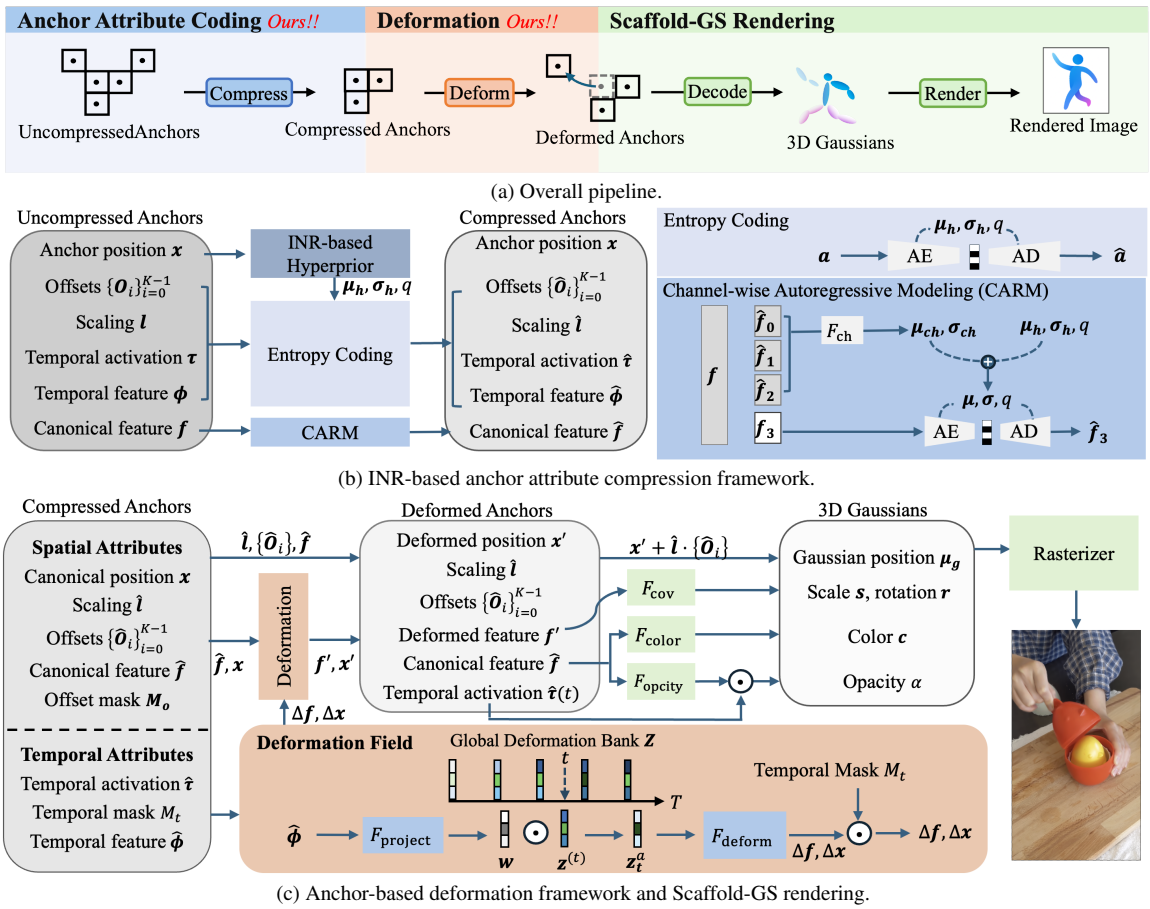


Figure 2. System overview of our TED-4DGS framework.

mation information for all dynamic anchors at time t and serves as a shared query target. To perform the query, the temporal feature ϕ of a dynamic anchor is projected by $\boldsymbol{w} = F_{\text{project}}(\phi)$, which is multiplied with $\boldsymbol{z}^{(t)}$ to yield an anchor-specific temporal latent $\boldsymbol{z}_t^a = \boldsymbol{w} \cdot \boldsymbol{z}^{(t)}$. This latent $\boldsymbol{z}_t^a$ further undergoes a deformation decoder F_{deform} to arrive at an positional displacement $\Delta \boldsymbol{x}$ and a feature residual $\Delta \boldsymbol{f}$:

$$\Delta \boldsymbol{f},\ \Delta \boldsymbol{x} = F_{\text{deform}}\big(F_{\text{project}}(\phi)\ \cdot\ \text{interp}(\boldsymbol{Z}, t)\big), \quad (1)$$

We then update the anchor's position and feature as $\boldsymbol{x}' = \boldsymbol{x} + \Delta \boldsymbol{x}$ and $\boldsymbol{f}' = \boldsymbol{f} + \Delta \boldsymbol{f}$.

We note that the deformation vectors stored in $\boldsymbol{Z}$ are learnable parameters that must be explicitly signaled in the bitstream. To balance deformation accuracy and bitrate, we signal a deformation vector for every other frame.

4.3. Progressive 3D-to-4D Anchor Transformation

A key challenge in deformation-based dynamic scene representations lies in handling occlusion and disocclusion, as canonical anchors persist throughout the entire sequence duration. Ideally, when an object is occluded by another, its corresponding anchors should remain behind those of the occluder. However, as shown in Fig. 6, a trivial yet improper solution is to deform occluded anchors such that they fall outside the viewing frustum. This leads to a highly irregular deformation field, hindering compression efficiency.

To address this issue, we extend the static 3D anchor representation into a 4D formulation by introducing a learnable temporal activation parameter $\tau = [(a_s, b_s), (a_f, b_f)]$, which explicitly specifies each anchor's active duration as

$$\tau(t) = \begin{cases} \exp\Big[-\big((t - a_s)/b_s\big)^2\Big], & t < a_s, \\ 1, & a_s \le t \le a_f, \\ \exp\Big[-\big((t - a_f)/b_f\big)^2\Big], & t > a_f, \end{cases} \quad (2)$$

where (a_s, b_s) and (a_f, b_f) define the smooth transitions for appearance and disappearance, respectively. The specification of $\tau(t)$ allows each anchor and its associated Gaussian primitives to activate or deactivate gradually over time. During rendering, the time-aware opacity α_t for each Gaussian primitive at time t is computed by modulating its static opacity α with the temporal activation $\tau(t)$:

$$\alpha_t = \alpha \cdot \tau(t), \quad (3)$$

which enables dynamic visibility modeling over time.

To ensure stable learning, we adopt a progressive training strategy. During the initial 20,000 iterations, all anchors are treated as static 3D anchors without temporal activation modeling, encouraging them to remain consistently active across frames. Subsequently, the temporal activation parameters a_s, a_f are initialized for each anchor based on its earliest and latest time instances that an anchor is visible in the viewing frustum. These temporal parameters are then optimized jointly with other anchor attributes.

As a side note, we adopt the anchor pruning strategy of ScaffoldGS [30], removing anchors whose associated Gaussian primitives exhibit negligible collective opacity. By incorporating temporal activation, our time-aware opacity enables more effective removal of anchors that contribute minimally over time.

4.4. INR-based Compression for Anchor Attributes

To encode anchors' attributes in a rate-distortion-optimized fashion, we propose an INR-based hyperprior to predict the distribution parameters for entropy coding and a channel-wise autoregressive model to further exploit intra-anchor correlations for its feature compression.

The INR-based hyperprior is implemented as a multi-layer perceptron that learns prior distributions over anchor attributes, including the feature $\boldsymbol{f}$, offsets $\{\boldsymbol{O}_i\}_{i=0}^{K-1}$, scaling $\boldsymbol{l}$, temporal feature $\boldsymbol{\phi}$ and temporal activation $\boldsymbol{\tau}$. Given the positional embedding of the anchor's position $\boldsymbol{x}$, the network outputs the corresponding distribution parameters: means $\boldsymbol{\mu}_h$, variances $\boldsymbol{\sigma}_h$, and quantization step sizes q. For each attribute type, a distinct quantization step size is predicted and shared across its components. The probability of a quantized attribute $\hat{\boldsymbol{a}}$ is evaluated as

$$p(\hat{\boldsymbol{a}} \mid \boldsymbol{x}) = \int_{\hat{\boldsymbol{a}}-\frac{q}{2}}^{\hat{\boldsymbol{a}}+\frac{q}{2}} \mathcal{N}\big(\boldsymbol{\mu}_h,\ \boldsymbol{\sigma}_h\big)\, d\boldsymbol{a}, \tag{4}$$

which amounts to the likelihood of the quantized value under the learned Gaussian prior. Since an INR hyperprior is grid-free, we transmit only a few network weights, eliminating the extra bits that triplanes or hash tables require and keeping the model lightweight.

Following [46], we adopt a channel-wise autoregressive model to exploit intra-feature dependencies for coding anchor features $\boldsymbol{f}$, which typically comprise a substantial portion of the bitstream (Fig. 2b).

4.5. Training Objectives

The training objective of TED-4DGS, as defined in Eq. (5), comprises several terms with distinct roles. The distortion loss $\mathcal{L}_{\text{distortion}}$ and rate loss $\mathcal{L}_{\text{rate}}$ form the classical rate-distortion pair, where $\mathcal{L}_{\text{distortion}}$ combines L1 and SSIM losses and $\mathcal{L}_{\text{rate}}$ denotes the average bitrate per anchor. To promote sparsity, we adopt the offset mask loss $\mathcal{L}_{\text{offset mask}}$

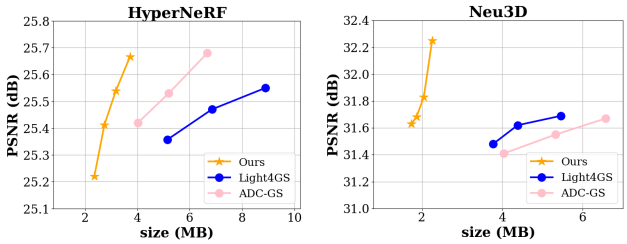


Figure 3. Rate-distortion comparison of our TED-4DGS, Light4GS [27] and ADC-GS [15].

and temporal mask loss $\mathcal{L}_{\text{temp mask}}$ from [23]. These losses suppress redundant spatial offsets and identify static anchors, respectively. The weight of $\mathcal{L}_{\text{offset mask}}$ is scaled proportionally to the rate-control factor λ_{rate}, so that lowering the target bitrate encourages more aggressive pruning of anchors and reduce the model size. Lastly, the regulization losses $\mathcal{L}_{\text{vol}}$ and $\mathcal{L}_{\text{tv}}$ encourage scale consistency and enforce temporal smoothness on the deformation vectors in the deformation bank $\boldsymbol{Z}$. These are adopted from ScaffoldGS [30] and E-D3DGS [2]. The complete objective is thus:

$$\begin{aligned}\mathcal{L} = \mathcal{L}_{\text{distortion}} + \lambda_{\text{rate}}\big(\mathcal{L}_{\text{rate}} + \lambda_{\text{offset mask}}\,\mathcal{L}_{\text{offset mask}}\big) \\ + \lambda_{\text{temp mask}}\,\mathcal{L}_{\text{temp mask}} + \lambda_{\text{vol}}\,\mathcal{L}_{\text{vol}} + \lambda_{\text{tv}}\,\mathcal{L}_{\text{tv}}.\end{aligned} \tag{5}$$

5. Experiments

Implementation Details. Our method is implemented in PyTorch and trained on an NVIDIA RTX 3090 GPU. We evaluate a range of rate-control factors λ_{rate}, choosing $\lambda_{\text{rate}} = \{0.001, 0.002, 0.004, 0.008\}$ for the HyperNeRF dataset and $\lambda_{\text{rate}} = \{0.002, 0.004, 0.008, 0.016\}$ for the Neu3D dataset. In our experiments, the low-rate configuration corresponds to the highest λ_{rate}, which yields the lowest bitrate, whereas the high-rate configuration uses the lowest λ_{rate} to achieve the maximum bitrate. The details of training process and the choice of hyperparameters are provided in the supplementary document.

Baselines. We compare TED-4DGS with three groups of baseline methods: (1) rate-distortion-optimized compression approaches, including Light4GS [27] and ADC-GS [15], and non-rate-distortion-optimized approaches, including (2) deformation-based methods–e.g. 4DGaussians [40] and E-D3DGS [2], and (3) space-time 4DGS methods–e.g. 4DGS [44], STG [25] and FreeTimeGS [39].

Datasets. We follow the common test protocol [15] to evaluate our TED-4DGS on two real-world datasets: Neural 3D Video (Neu3D) [24] and HyperNeRF [36]. Specifically, on the Neu3D dataset, which includes multiple synchronized multi-view videos captured by 18–21 cameras per scene, the results are reported for the *cook spinach*, *cut roasted beef*, *flame salmon*, *flame steak*, and *sear steak* sequences. On the HyperNeRF dataset, which includes videos captured

Table 2. Quantitative results on Neu3D. The best and 2nd best results are highlighted in red and yellow cells.

Model	PSNR↑	$SSIM_2$↑	LPIPS↓	FPS↑	Size (MB)↓
4DGS [44]	32.01	0.986	0.055	114	1000
STG [25]	32.04	0.974	0.044	110	175
FreeTimeGS [39]	33.19	–	0.036	–	125
4DGaussians [40]	31.72	0.984	0.049	34	38
E-D3DGS [2]	31.20	0.974	0.030	42	40
Light4GS [27] (low rate)	31.48	–	0.064	40	3.77
Light4GS [27] (high rate)	31.69	–	0.053	37	5.46
ADC-GS [15] (low rate)	31.41	0.972	0.066	126	4.04
ADC-GS [15] (high rate)	31.67	0.981	0.061	110	6.57
Ours (low rate)	31.63	0.969	0.061	78	1.73
Ours (high rate)	32.25	0.972	0.051	73	2.26

Table 3. Quantitative results on HyperNeRF.

Model	PSNR↑	$SSIM_2$↑	LPIPS↓	FPS↑	Size (MB)↓
4DGaussians [40]	25.60	0.848	0.281	22	64
E-D3DGS [2]	25.74	0.816	0.231	26	47
Light4GS [27] (low rate)	25.35	–	–	28	5.15
Light4GS [27] (high rate)	25.55	–	–	27	8.87
ADC-GS [15] (low rate)	25.42	0.777	0.315	135	4.02
ADC-GS [15] (high rate)	25.68	0.825	0.252	101	6.67
Ours (low rate)	25.22	0.800	0.311	111	2.36
Ours (high rate)	25.67	0.808	0.287	96	3.72

using two phones rigidly mounted on a handheld stereo rig. [36], we report results for four dynamic scenes, *3D printer*, *banana*, *broom*, and *chicken*). We train (and test) on all frames at half resolution (536×960) in width and height according to [15, 40, 43].

Metrics. To compare the rate-distortion (RD) performance of the competing methods, we report peak-signal-to-noise ratio (PSNR), structural similarity index ($SSIM_2$), and perceptual quality measure LPIPS. For the rate-distortion-optimized approaches, we compute the average PSNR/$SSIM_2$/LPIPS and compressed file size across sequences for the low-rate and high-rate configurations. Likewise, we measure rendering speed in frames per second (FPS) under these configurations.

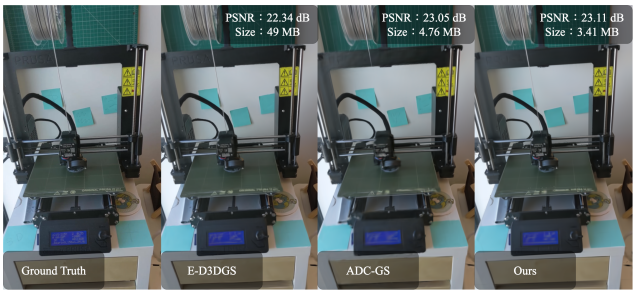
5.1. Rate-Distortion Comparison

In Fig. 3, Table 2 and Table 3, our TED-4DGS consistently outperforms the competing methods across both datasets, achieving the state-of-the-art RD performance.

Figs. 4 further presents the subjective quality comparison. On the *sear steak* scene (Neu3D), our TED-4DGS delivers similar or superior rendering quality while reducing the file size by over 14x relative to E-D3DGS [2] and over 18x compared to 4DGaussians [40].

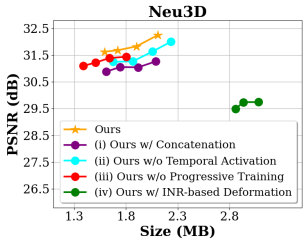
On the *3D printer* scene (HyperNeRF), our TED-4DGS achieves 23.1dB PSNR with a file size of 3.4MB, which amounts to a 28% bitrate reduction compared to the rate-distortion-optimized ADC-GS [15] at a comparable perceptual quality level. Compared to E-D3DGS [2], a non-rate-distortion-optimized approach, the reduction in file size exceeds 14x.

(a) *sear steak* (Neu3D)

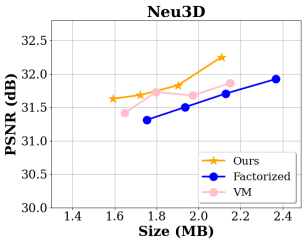


(b) *3D printer* (HyperNeRF)

Figure 4. Subjective quality comparisons.

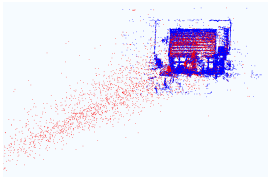


(a) Deformation Field Variants

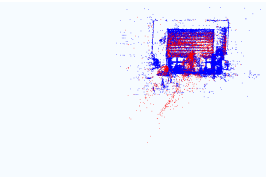


(b) Compression Variants

Figure 5. Rate-distortion comparisons on (a) deformation field variants and (b) compression variants.



(a) w/o Temporal Activation



(b) w/ Temporal Activation

Figure 6. Comparison of deformed point clouds in the *cook spinach* scene (Neu3D).

5.2. Ablation Experiments

We conduct ablation experiments on the Neu3D [24] dataset to validate (1) the query mechanism of the deformation bank, (2) the temporal activation, (3) the progressive training strat-

Table 4. Ablation study of deformation field variants on Neu3D at highest rate point.

Method	PSNR ↑	Size (MB)↓
Ours	32.25	2.11
(i) Ours w/ Concatenation	31.27	2.10
(ii) Ours w/o Temporal Activation	32.00	2.23
(iii) Ours w/o Progressive Training	31.44	1.80
(iv) Ours w/ INR-based Deformation	29.72	5.09

Table 5. Temporal duration versus motion complexity.

Scene	Motion Complexity	Temporal Duration ($\Delta\tau = a_f - a_s$) $\Delta\tau \leq 0.2$	$0.2 < \Delta\tau < 0.8$	$0.8 \leq \Delta\tau$
Flame Steak	Slow	3%	0%	97%
Banana	Medium	9%	6%	85%
Broom	Fast	18%	35%	47%

egy, and (4) the INR-based hyperprior. In addition, we include two baselines: (5) a pure ScaffoldGS compression without temporal modeling and (6) a naive INR-based deformation. Table 4 presents RD results at the highest rate point ($\lambda_{rate} = 0.002$) while Fig. 5 presents RD-optimized results in the form of RD curves.

Deformation Query. When querying the global deformation bank $\boldsymbol{Z}$ (Fig. 2 (e)), there are two query mechanisms: multiplication ($\boldsymbol{w} \cdot \boldsymbol{z}^{(t)}$) or concatenation (concat($\boldsymbol{\phi}, \boldsymbol{z}^{(t)}$)). Our approach adopts the multiplication strategy, which can be interpreted as retrieving relevant deformation from the global deformation vector $\boldsymbol{z}^{(t)}$. From Fig. 5a, this approach outperforms the concatenation design ((i) w/ concatenation) in terms of RD performance.

Temporal Activation. Fig. 5a shows that disabling temporal activation reduces our dynamic scene representation with deformation (Section 4.3). This consistently increases the bitrate across all rate points, while the rendering quality exhibits a slight degradation. Recall that our temporal activation approach facilitates time-aware opacity pruning, which discards anchors with limited relevance across time. Fig. 6 further confirms that in the absence of our temporal activation mechanism, a non-trivial yet improper solution may emerge, relocating non-contributing dynamic Gaussian primitives (red dots in Fig. 6) outside the viewing frustum.

We also analyze the active time span of Gaussians ($\Delta\tau = a_f - a_s$). Table 5 and Fig. 7 show that slow-motion scenes have over 95% of Gaussians active throughout the sequence, while higher motion complexity produces more short-duration Gaussians with a relatively uniform distribution. This demonstrates that temporal activation adaptively aligns anchor lifetimes with scene dynamics. Further results are provided in Supplementary Sec. E.

Progressive Learning. Fig. 5a further shows that our progressive training strategy (Section 4.3) leads to superior rate–distortion performance. Across all rate points, we observe a consistent reduction in bitrate, but this comes with

Figure 7. Comparison of rendered images and rendered temporal duration maps with different motion dynamics.

a noticeable drop in PSNR. The training process tends to prune anchors more aggressively when their deformation is not well learned in the early phase. This validates the effectiveness of using static 3D anchors to stabilize deformation learning prior to incorporating temporal activation modeling.

INR-based Hyperprior. Fig. 5b presents the rate-distortion comparison with a factorized prior [3] and a grid-based vector-matrix hyperprior [45]. The former treats attribute components as independent and identically distributed random variables. Our approach achieves a 20.0% BD-rate saving compared to FM and also provide superior RD performance than VM hyperprior.

INR-based Deformation. In Fig. 5a, we also include INR-based deformation as a baseline method, which extends Scaffold-GS [30] with a INR-based deformation network from D3DGS [43] to model anchor deformation $(\Delta\boldsymbol{x}, \Delta\boldsymbol{f})$ as a function of (x, y, z, t). The baseline performs notably worse than TED-4DGS in rate-distortion performance, highlighting the robustness of our deformation networks.

6. Conclusion

This work presents a novel rate-distortion-optimized compression framework for dynamic 3DGS. To efficiently represent anchor-wise temporal deformation, it features a compact embedding-based deformation network that leverages per-anchor temporal embeddings and a shared global deformation bank. Furthermore, a progressive 4D anchor learning strategy is introduced to ensure smooth deformation and explicit temporal visibility control via learnable temporal activation parameters. Finally, to achieve effective attribute coding, we combine an INR-based hyperprior and a channel-wise autoregressive model to predict the distribution for entropy coding. These components collectively enable TED-4DGS to achieve state-of-the-art rate-distortion performance across several real-world dynamic scene datasets.

7. Acknowledgement

This work is supported by MediaTek Advanced Research Center and National Science and Technology Council (NSTC), Taiwan, under Grants 113-2634-F-A49-007-, 112-2221-E-A49-092-MY3, and 114-2221-E-A49-035-MY3. We thank to National Center for High-performance Computing (NCHC) for providing computational and storage resources.

References

[1] Muhammad Salman Ali, Maryam Qamar, Sung-Ho Bae, and Enzo Tartaglione. Trimming the fat: Efficient compression of 3d gaussian splats through pruning, 2024. 3

[2] Jeongmin Bae, Seoha Kim, Youngsik Yun, Hahyun Lee, Gun Bang, and Youngjung Uh. Per-gaussian embedding-based deformation for deformable 3d gaussian splatting, 2024. 2, 3, 6, 7

[3] Johannes Ballé, Valero Laparra, and Eero P. Simoncelli. End-to-end optimized image compression, 2017. 8

[4] Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston. Variational image compression with a scale hyperprior, 2018. 3

[5] Ang Cao and Justin Johnson. Hexplane: A fast representation for dynamic scenes, 2023. 3

[6] Yihang Chen, Qianyi Wu, Weiyao Lin, Mehrtash Harandi, and Jianfei Cai. Hac: Hash-grid assisted context for 3d gaussian splatting compression, 2024. 2, 3, 4

[7] Yihang Chen, Qianyi Wu, Mengyao Li, Weiyao Lin, Mehrtash Harandi, and Jianfei Cai. Fast feedforward 3d gaussian splatting compression, 2025. 3

[8] Woong Oh Cho, In Cho, Seoha Kim, Jeongmin Bae, Youngjung Uh, and Seon Joo Kim. 4d scaffold gaussian splatting for memory efficient dynamic scene reconstruction, 2024. 2, 3

[9] Yuanxing Duan, Fangyin Wei, Qiyu Dai, Yuhang He, Wenzheng Chen, and Baoquan Chen. 4d-rotor gaussian splatting: Towards efficient novel view synthesis for dynamic scenes, 2024. 3

[10] Bardienus P. Duisterhof, Zhao Mandi, Yunchao Yao, Jia-Wei Liu, Jenny Seidenschwarz, Mike Zheng Shou, Deva Ramanan, Shuran Song, Stan Birchfield, Bowen Wen, and Jeffrey Ichnowski. Deformgs: Scene flow in highly deformable scenes for deformable object manipulation, 2024.

[11] Cheng-De Fan, Chen-Wei Chang, Yi-Ruei Liu, Jie-Ying Lee, Jiun-Long Huang, Yu-Chee Tseng, and Yu-Lun Liu. Spectromotion: Dynamic 3d reconstruction of specular scenes, 2025. 3

[12] Zhiwen Fan, Kevin Wang, Kairun Wen, Zehao Zhu, Dejia Xu, and Zhangyang Wang. Lightgaussian: Unbounded 3d gaussian compression with 15x reduction and 200+ fps, 2024. 3

[13] Tobias Fischer, Jonas Kulhanek, Samuel Rota Bulò, Lorenzo Porzi, Marc Pollefeys, and Peter Kontschieder. Dynamic 3d gaussian fields for urban areas, 2024. 3

[14] Dailan He, Ziming Yang, Weikun Peng, Rui Ma, Hongwei Qin, and Yan Wang. Elic: Efficient learned image compression with unevenly grouped space-channel contextual adaptive coding, 2022. 3

[15] He Huang, Qi Yang, Mufan Liu, Yiling Xu, and Zhu Li. Adc-gs: Anchor-driven deformable and compressed gaussian splatting for dynamic scene reconstruction, 2025. 2, 3, 6, 7

[16] Yi-Hua Huang, Yang-Tian Sun, Ziyi Yang, Xiaoyang Lyu, Yan-Pei Cao, and Xiaojuan Qi. Sc-gs: Sparse-controlled gaussian splatting for editable dynamic scenes, 2024. 3

[17] Xu Jiawei, Fan Zexin, Yang Jian, and Xie Jin. Grid4D: 4D decomposed hash encoding for high-fidelity dynamic gaussian splatting. *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. 2, 3

[18] Hanyang Kong, Xingyi Yang, and Xinchao Wang. Efficient gaussian splatting for monocular dynamic scene rendering via sparse time-variant attribute modeling, 2025. 3

[19] Agelos Kratimenos, Jiahui Lei, and Kostas Daniilidis. Dynmf: Neural motion factorization for real-time dynamic view synthesis with 3d gaussian splatting, 2024.

[20] Sangwoon Kwak, Joonsoo Kim, Jun Young Jeong, Won-Sik Cheong, Jihyong Oh, and Munchurl Kim. Modec-gs: Global-to-local motion decomposition and temporal interval adjustment for compact dynamic 3d gaussian splatting, 2025.

[21] Isaac Labe, Noam Issachar, Itai Lang, and Sagie Benaim. Dgd: Dynamic 3d gaussians distillation, 2024. 3

[22] Junoh Lee, Chang-Yeon Won, Hyunjun Jung, Inhwan Bae, and Hae-Gon Jeon. Fully explicit dynamic gaussian splatting, 2024. 3

[23] Joo Chan Lee, Daniel Rho, Xiangyu Sun, Jong Hwan Ko, and Eunbyung Park. Compact 3d gaussian representation for radiance field, 2024. 3, 6

[24] Tianye Li, Mira Slavcheva, Michael Zollhoefer, Simon Green, Christoph Lassner, Changil Kim, Tanner Schmidt, Steven Lovegrove, Michael Goesele, Richard Newcombe, and Zhaoyang Lv. Neural 3d video synthesis from multi-view video, 2022. 6, 7

[25] Zhan Li, Zhang Chen, Zhong Li, and Yi Xu. Spacetime gaussian feature splatting for real-time dynamic view synthesis, 2024. 2, 3, 6, 7

[26] Lei Liu, Zhenghao Chen, Wei Jiang, Wei Wang, and Dong Xu. Hemgs: A hybrid entropy model for 3d gaussian splatting data compression, 2025. 3, 4

[27] Mufan Liu, Qi Yang, He Huang, Wenjie Huang, Zhenlong Yuan, Zhu Li, and Yiling Xu. Light4gs: Lightweight compact 4d gaussian splatting generation via context model, 2025. 2, 3, 6, 7

[28] Qingming Liu, Yuan Liu, Jiepeng Wang, Xianqiang Lyv, Peng Wang, Wenping Wang, and Junhui Hou. Modgs: Dynamic gaussian splatting from casually-captured monocular videos with depth priors, 2025. 3

[29] Xiangrui Liu, Xinju Wu, Pingping Zhang, Shiqi Wang, Zhu Li, and Sam Kwong. Compgs: Efficient 3d scene representation via compressed gaussian splatting, 2024. 3

[30] Tao Lu, Mulin Yu, Linning Xu, Yuanbo Xiangli, Limin Wang, Dahua Lin, and Bo Dai. Scaffold-gs: Structured 3d gaussians for view-adaptive rendering, 2023. 3, 4, 6, 8

[31] Hidenobu Matsuki, Gwangbin Bae, and Andrew J. Davison. 4dtam: Non-rigid tracking and mapping via dynamic surface gaussians, 2025. 3

[32] David Minnen and Saurabh Singh. Channel-wise autoregressive entropy models for learned image compression, 2020. 3

[33] David Minnen, Johannes Ballé, and George Toderici. Joint autoregressive and hierarchical priors for learned image compression, 2018. 3

[34] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics*, 41 (4):1–15, 2022. 3

[35] Seungjun Oh, Younggeun Lee, Hyejin Jeon, and Eunbyung Park. Hybrid 3d-4d gaussian splatting for fast dynamic scene representation, 2025. 3

[36] 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, 2021. 1, 6, 7

[37] Richard Shaw, Michal Nazarczuk, Jifei Song, Arthur Moreau, Sibi Catley-Chandar, Helisa Dhamo, and Eduardo Perez-Pellitero. Swings: Sliding windows for dynamic 3d gaussian splatting, 2024. 3

[38] Yufei Wang, Zhihao Li, Lanqing Guo, Wenhan Yang, Alex C. Kot, and Bihan Wen. Contextgs: Compact 3d gaussian splatting with anchor level context model, 2024. 3, 4

[39] Yifan Wang, Peishan Yang, Zhen Xu, Jiaming Sun, Zhanhua Zhang, Yong Chen, Hujun Bao, Sida Peng, and Xiaowei Zhou. Freetimegs: Free gaussian primitives at anytime and anywhere for dynamic scene reconstruction, 2025. 2, 3, 6, 7

[40] Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 4d gaussian splatting for real-time dynamic scene rendering, 2024. 2, 3, 6, 7

[41] Kun Yang, Yuxiang Liu, Zeyu Cui, Yu Liu, Maojun Zhang, Shen Yan, and Qing Wang. Ntr-gaussian: Nighttime dynamic thermal reconstruction with 4d gaussian splatting based on thermodynamics, 2025. 3

[42] Shuojue Yang, Qian Li, Daiyun Shen, Bingchen Gong, Qi Dou, and Yueming Jin. Deform3dgs: Flexible deformation for fast surgical scene reconstruction with gaussian splatting, 2024. 3

[43] Ziyi Yang, Xinyu Gao, Wen Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. Deformable 3d gaussians for high-fidelity monocular dynamic scene reconstruction, 2023. 7, 8

[44] Zeyu Yang, Hongye Yang, Zijie Pan, and Li Zhang. Real-time photorealistic dynamic scene representation and rendering with 4d gaussian splatting, 2024. 2, 3, 6, 7

[45] Yu-Ting Zhan, He bi Yang, Cheng-Yuan Ho, Jui-Chiu Chiang, and Wen-Hsiao Peng. Cat-3dgs pro: A new benchmark for efficient 3dgs compression, 2025. 8

[46] Yu-Ting Zhan, Cheng-Yuan Ho, Hebi Yang, Yi-Hsin Chen, Jui Chiu Chiang, Yu-Lun Liu, and Wen-Hsiao Peng. Cat-3dgs: A context-adaptive triplane approach to rate-distortion-optimized 3dgs compression, 2025. 2, 3, 4, 6