CoDGS: Training-Free Codec-Driven 3D Gaussian Splatting for Real-Time Free-Viewpoint Video Streaming

Jusuk Kim Hun Heo Jungmin You
Seoul National University
{jusukkim, greathunh, ujxlan409}@snu.ac.kr

Abstract

Online free-viewpoint video (FVV) streaming demands per-frame processing for immediate playback, yet existing methods require seconds of per-frame optimization, falling far short of the 33 ms budget needed for 30 fps playback. State-of-the-art online multi-view approaches heavily rely on iterative gradient-based optimization or learned inference. Our key observation is that modern video codecs already compute motion vectors during encoding; re-estimating this motion at the receiver is redundant. We present CoDGS*, a training-free framework that leverages these standard codec cues to perform closed-form updates of 3D Gaussians for real-time multi-view FVV streaming. Specifically,* CoDGS *(1) lifts 2D codec motion vectors to 3D using rendered depth to compute multi-scale, voxel-wise rigid motion fields, (2) assigns stable and coherent motion to each Gaussian via a quality-aware selection mechanism and multi-view consensus, and (3) seamlessly handles emerging content by delegating it to the codec's Group of Pictures (GOP) keyframe structure, completely bypassing the instability of heuristic Gaussian spawning.* CoDGS *sustains real-time 30 fps on PanopticSports and 20 fps on N3DV, 100–350 × faster than prior online methods, making it, to our knowledge, the first training-free approach to achieve real-time full-scene FVV streaming on commodity hardware.*

1. Introduction

Free-viewpoint video (FVV) enables immersive, interactive experiences central to virtual and augmented reality, telepresence, and 3D conferencing. Unlike pre-recorded volumetric assets, *online* FVV streaming allows immediate playback without preloading the full sequence. This imposes a strict latency requirement: to sustain smooth playback at 30 fps, each incoming frame must be reconstructed and made renderable within a 33 ms budget. While 3D Gaussian Splatting (3DGS) [8] has made real-time rendering practical—solidifying its role as a compelling representation for dynamic scenes—the per-frame reconstruction needed for online streaming remains orders of magnitude away from this tight budget.

Methods relevant to dynamic 3DGS streaming fall into two categories. *Offline* methods [10, 17, 19] achieve high quality but require the entire sequence and full-model transmission, precluding streaming. *Online multi-view* methods [2, 3, 14, 18] enable progressive transmission, but share a fundamental bottleneck: per-frame gradient-based optimization that takes seconds per frame, far exceeding the 33 ms budget. A notable exception is feed-forward portrait reconstruction, VoluMe [4], which achieves real-time rates but is confined to single-view webcam capture via learned networks, and cannot handle multi-view full-scene FVV. Moreover, the reported costs of these methods typically measure only the per-frame update, excluding stream decoding and novel-view rendering that any end-to-end streaming system must inherently absorb. Across both tracks, the root cause of the latency is uniform: existing methods recover scene motion through expensive per-frame optimization, learned deformation, or neural inference.

Yet, this motion estimation step is largely redundant. In any practical FVV pipeline, multi-view video is already encoded and transmitted via standard video codecs [12, 13, 16]; every streaming method must therefore decode the incoming bitstream before reconstruction can begin. During encoding, the codec performs block-matching to compute motion vectors—quantities that inherently provide temporal correspondence. Existing methods discard this readily available information and re-estimate motion from scratch. Our core insight is that the codec has already solved the motion-matching problem, and its solution arrives for free as a byproduct of decoding. While codec motion vectors describe 2D image-plane displacement and contain noise, we can elegantly resolve this by lifting them into 3D using the geometry already encoded in the 3DGS representation, enforcing multi-view consistency. The challenge is thus no longer to *estimate* motion, but to reliably *lift and filter* it into a coherent 3D update—a geometric problem with a closed-form solution.

Table 1. Key differences between related work and our method.

Method	Motion Source	Update Mechanism	Latency	Learning
3DGStream [14]	gradient	per-frame opt.	∼5s	Yes
QUEEN [3]	gradient	per-frame opt.	∼5s	Yes
HiCoM [2]	gradient	per-frame opt.	∼2s	Yes
IGS [18]	optical flow	transformer inf.	∼2.67s	Yes (pretrained)
DeltaStream [9]	codec MV	closed-form (PCL)	real-time	No
Ours	**codec MV**	**closed-form (3DGS)**	**∼33ms**	**No**

Building on this insight, we present CoDGS, a training-free framework that turns codec cues into closed-form 3DGS updates, mirroring the codec's structure through three components: **(a)** depth-guided 3D lifting that back-projects codec MVs and solves voxel-wise rigid motion in closed form via Kabsch; **(b)** a multi-scale, quality-aware selection that—via multi-view consensus and per-voxel fit statistics—updates each Gaussian only with coherent motion, else leaving it static; and **(c)** GOP-delegated emerging content, where periodic keyframes reconstruct fresh geometry and reset drift instead of unstable Gaussian spawning. No stage uses gradient descent or network inference.

We summarize our contributions:

- The first **training-free, codec-driven framework for online 3DGS streaming**. CoDGS directly exploits standard video codec motion vectors for per-frame updates, eliminating optimization and neural inference entirely.
- A **closed-form 3D lifting and robust multi-scale selection mechanism**. We lift per-view 2D motion to 3D and apply a quality-aware veto system to assign optimal, noise-filtered 6-DOF rigid motion to each Gaussian.
- An elegant **GOP-delegated scene update strategy**. We demonstrate that relying on streaming infrastructure (periodic I-frames) to handle emerging content is far more robust and efficient than dynamic Gaussian spawning.
- **End-to-end real-time multi-view FVV streaming.** CoDGS runs the full decode–update–render loop at real-time ∼30 fps on PanopticSports and ∼20 fps on the higher-resolution N3DV—two to three orders of magnitude faster than prior online methods, and, to our knowledge, the first training-free approach to reach real-time full-scene multi-view streaming on commodity hardware.

2. Related Work

2.1. Dynamic 3D Gaussian Splatting

3DGS [8] represents scenes as explicit anisotropic Gaussians, enabling real-time photorealistic rendering through hardware rasterization. Extensions to dynamic scenes follow two paradigms: *time-conditioned* methods [10, 17] embed temporal variation directly into Gaussian primitives, while *deformation-field* methods [6, 19] maintain canonical Gaussians transformed by a learned deformation field. Both paradigms achieve high reconstruction quality, but share a fundamental streaming incompatibility: they require the full video sequence before reconstruction begins and transmit the entire model at once, precluding per-frame progressive delivery.

To enable motion-driven updates without full-sequence optimization, a parallel line of work attaches Gaussians to compact control structures. SC-GS [6] and ComGS [1] use sparse control points; H3D-DGS [5] decomposes motion into optically observable and unobservable components; IGS [18] lifts multi-view optical-flow features onto anchor points via a generalizable transformer, reducing per-frame cost to ∼2.67s. These methods improve over full re-optimization, but still depend on learned motion estimation.

2.2. Streamable Gaussian Representations

Online methods decouple scene acquisition from playback by updating the Gaussian model incrementally as each frame arrives. 3DGStream [14] introduced a hash-based neural transformation cache with adaptive Gaussian addition; QUEEN [3] unifies residual learning with quantization-sparsity compression; HiCoM [2] applies hierarchical coherent motion; ComGS [1] achieves extreme compression via shared motion over 200 keypoints; 4DGCPro [20] adds progressive multi-bitrate transmission. Feed-forward alternatives such as VoluMe [4] predict live Gaussian splats from monocular webcam frames for 3D video calls, but they target human-centric single-view reconstruction rather than multi-view FVV streaming and do not exploit codec-derived inter-frame motion.

In contrast to feed-forward human reconstruction, general multi-view FVV methods still rely on per-frame gradient-based optimization, whose cost—ranging from ∼2s to ∼10s per frame—exceeds the 33 ms budget for 30 fps playback by one to two orders of magnitude. This latency arises because these methods infer or optimize per-frame Gaussian updates from image observations.

2.3. Codec-Driven and Compressed 3D Representations

Exploiting compressed-domain signals for 3D reconstruction is a nascent but growing direction. DeltaStream [9]

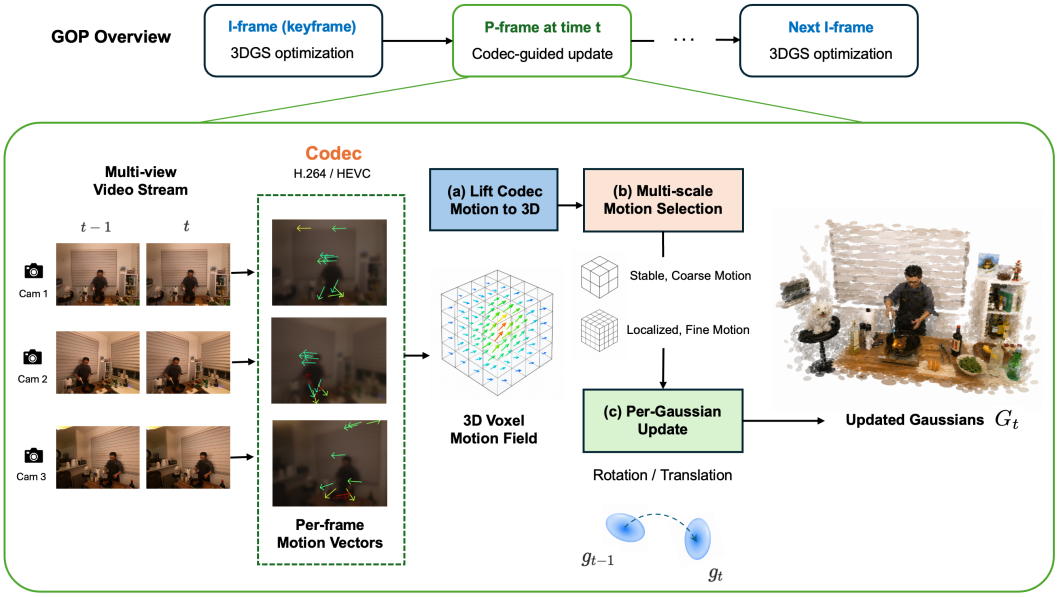

Figure 1. **Overview of the proposed codec-driven 3D Gaussian streaming framework.** *(Top)* The pipeline mirrors the video codec's Group of Pictures (GOP) structure. I-frames undergo full 3DGS optimization to anchor the scene and naturally incorporate newly emerging content. *(Bottom)* For intermediate P-frames, we bypass computationally expensive per-frame optimization. Instead, 2D motion vectors are extracted directly from the H.264/HEVC bitstream and (a) lifted to 3D and aggregated into a voxel motion field using rendered depths. To balance stability and detail, (b) a multi-scale selection mechanism filters noise and assigns the most reliable rigid transformation to each local region. Finally, (c) the position and rotation of each Gaussian are updated rigidly, yielding the new representation G_t in real-time.

demonstrates that codec motion vectors can drive real-time point-cloud updates via geometric lifting, establishing the core viability of codec-driven 3D updates. V^3 [15] repurposes hardware video codecs as a compression channel for Gaussian attributes, achieving mobile-friendly streaming, but uses the codec solely for compression rather than motion estimation.

No prior work consumes video codec motion vectors to directly drive 3D Gaussian motion for multi-view streaming. The primary challenge lies in the inherent noise and 2D image-plane nature of raw codec MVs. Our method fills this gap by directly lifting 2D codec motion vectors to 3D surface correspondences using rendered depths, and regularizing this noisy data through a multi-scale rigid motion formulation backed by multi-view consensus. By further delegating topological changes (emerging content) to the codec's Group of Pictures (GOP) structure, we enable robust, sub-33 ms Gaussian updates without relying on any learned priors or per-frame optimization.

3. Method

3.1. Overview

We propose a training-free framework that maintains a dynamic 3D Gaussian Splatting (3DGS) representation of a multi-view video stream and updates it online, one frame at a time, using only the metadata from a standard video codec (H.264/HEVC). Instead of relying on computationally expensive optical-flow networks, our method directly reuses the motion vectors (MVs) that are already computed during the codec's block-based motion estimation and recovered as a byproduct of decoding.

To guarantee real-time, low-latency processing, our framework assumes a low-delay Group of Pictures (GOP) structure composed exclusively of intra-coded keyframes (I-frames) and forward-predicted frames (P-frames), explicitly avoiding bidirectional B-frames that require buffering future frames. We mirror this structure with two components:

- **Keyframe reconstruction (I-frame).** At each keyframe we obtain a per-frame 3DGS G from the decoded multi-view images using any existing static reconstruction method. This anchors the representation and absorbs new scene content (Sec. 3.7).
- **Motion update (P-frame).** For each subsequent P-frame we update G in place using the codec MVs, without optical flow and without any per-frame optimization.

The per-P-frame update is summarized as follows. Given the current Gaussians G_{t-1}, we (i) render a depth map from G_{t-1} for each camera, (ii) lift the 2D codec MVs of frame t into a set of 3D point correspondences using these depths (Sec. 3.3), (iii) fit a compact, voxel-wise rigid motion field to the correspondences in closed form (Sec. 3.4–3.5), and (iv) apply the field to transform the Gaussians, yielding G_t (Sec. 3.6). Every step is training-free and admits a closed-form solution, so the representation can be propagated at real-time rates. Algorithm 1 summarizes one P-frame update; an I-frame instead replaces G with a fresh reconstruction.

Algorithm 1 Codec-MV update for one P-frame t

Require: previous Gaussians G_{t-1}; codec MVs per view; cameras $\{\pi_c\}$; scales $\mathcal{S}$; floor $\tau_{\text{obs}}(\cdot)$; consensus K; selection rule
Ensure: updated Gaussians G_t

for each camera c **do**
  $D^c_{t-1} \leftarrow \textsc{RenderDepth}(G_{t-1}, c)$
end for
$\mathcal{P} \leftarrow \varnothing$ ▷ 3D correspondences, tagged by camera
for each camera c, each MV with pixels $(\mathbf{u}_s, \mathbf{u}_d)$ **do**
  $d \leftarrow D^c_{t-1}(\mathbf{u}_s)$ ▷ Eq. (4)
  $\mathbf{p}_s \leftarrow \pi_c^{-1}(\mathbf{u}_s, d)$; $\mathbf{p}_d \leftarrow \pi_c^{-1}(\mathbf{u}_d, d)$
  if $\|\mathbf{p}_d - \mathbf{p}_s\| \le \tau_{\text{mot}}$ **then**
    $\mathcal{P} \leftarrow \mathcal{P} \cup \{(\mathbf{p}_s, \mathbf{p}_d, c)\}$
  end if
end for
for each scale $s \in \mathcal{S}$, each occupied voxel $\mathcal{V}$ at scale s **do**
  if $\mathcal{V}$ is valid **then** ▷ Eq. (9)
    $(R_{\mathcal{V}}, \mathbf{t}_{\mathcal{V}}) \leftarrow \textsc{KAbsch}(\mathcal{V})$ ▷ Eq. (7)
    compute $\rho_{\mathcal{V}}$, $\eta_{\mathcal{V}}$ ▷ Eq. (10)
  end if
end for
for each Gaussian $\mathbf{g} \in G_{t-1}$ **do**
  $s^\star \leftarrow \textsc{Select}\big(\mathcal{S}(\mathbf{g})\big)$ ▷ Eq. (11)
  apply $(R^\star, \mathbf{t}^\star)$ of $\mathcal{V}_{s^\star}(\mathbf{g})$ ▷ Eq. (12)
end for
return G_t ▷ scale, opacity, color carried over

3.2. Preliminaries

3D Gaussian Splatting. A scene is represented by a set of anisotropic 3D Gaussians. Each Gaussian G carries a center $\mathbf{g} \in \mathbb{R}^3$, an orientation expressed as a unit quaternion $\mathbf{q}$, a scale, an opacity, and a view- dependent color. An image is formed by projecting the Gaussians to the image plane and alpha-compositing them in depth order. We assume a pre-trained 3DGS for each keyframe is available and focus on propagating it across P-frames; our method does not modify the Gaussian appearance parameters.

Codec motion vectors. During inter-frame encoding the codec partitions a frame into blocks and, for each block b, finds the best-matching region in the previously decoded reference frame. The block's backward motion vector $\mathbf{m}_b \in \mathbb{R}^2$ is the displacement from its position in the current frame to that predictor in the reference frame, and is transmitted together with a residual $\mathbf{r}$. Decoding P-frame t of camera c therefore reconstructs the current image I^c_t from the motion-compensated reference I^c_{t-1} plus the residual,

$$I^c_t(\mathbf{u}_d) \;=\; I^c_{t-1}(\mathbf{u}_s) + \mathbf{r}(\mathbf{u}_d), \qquad \mathbf{u}_s \;=\; \mathbf{u}_d + \mathbf{m}_b, \quad (1)$$

where $\mathbf{u}_d \in b$ is a pixel of the current frame t and $\mathbf{u}_s$ its source in the reference frame $t-1$: since the vector is backward, adding it to the destination $\mathbf{u}_d$ gives the source $\mathbf{u}_s$ from which the content is copied. We extract the set of motion vectors $\{\mathbf{m}_b\}$ and residuals $\mathbf{r}$ directly from the bitstream without performing any additional motion estimation. Since real-time streaming cannot reference future frames without introducing unacceptable latency, our method excludes bidirectional B-frames and relies exclusively on P-frames.

Problem statement. Given the previous Gaussians G_{t-1} and the codec MVs of frame t, we seek G_t that reproduces the scene at time t, computed within a real-time per-frame budget (e.g. 33 ms for 30 fps).

3.3. Lifting Codec Motion to 3D

The codec MVs describe motion in the 2D image plane; to move 3D Gaussians, we lift them to 3D space using the geometry already encoded in our representation. Rendering G_{t-1} from camera c yields a depth map D^c_{t-1}, which assigns a metric depth to every pixel.

For camera c with intrinsics K_c and camera-to-world rotation/translation $(R_c, \mathbf{t}_c)$, the back-projection π_c^{-1} of a pixel $\mathbf{u}$ with depth d is

$$\pi_c^{-1}(\mathbf{u}, d) \;=\; R_c\left(d\,K_c^{-1}\,\tilde{\mathbf{u}}\right) + \mathbf{t}_c, \quad (2)$$

where $\tilde{\mathbf{u}}$ is the homogeneous pixel coordinate. Given a motion vector linking a destination pixel $\mathbf{u}_d$ to its source $\mathbf{u}_s$ (Eq. 1), we retrieve the depth d of the moving content from the previous frame at the source location:

$$d \;=\; D^c_{t-1}(\mathbf{u}_s), \quad (3)$$

Assuming depth varies negligibly over a single frame interval, we associate the same surface point with both endpoints, establishing the 3D correspondence:

$$\mathbf{p}_s = \pi_c^{-1}(\mathbf{u}_s, d), \qquad \mathbf{p}_d = \pi_c^{-1}(\mathbf{u}_d, d). \quad (4)$$

The pair $(\mathbf{p}_s, \mathbf{p}_d)$ is a 3D correspondence: the surface point was at $\mathbf{p}_s$ in frame $t-1$ and is at $\mathbf{p}_d$ in frame t.

We aggregate correspondences from *all* cameras into a single set $\mathcal{P} = \{(\mathbf{p}^i_s, \mathbf{p}^i_d, c^i)\}$. Pooling multiple views is critical: a 3D motion consistent across cameras is well-constrained, while correspondences arising from occlusions or unreliable depth estimates typically disagree across views and are suppressed during the subsequent voxel-based aggregation. We discard correspondences whose implied displacement exceeds a threshold, $\|\mathbf{p}^i_d - \mathbf{p}^i_s\| > \tau_{\text{mot}}$, removing spurious matches from depth discontinuities.

3.4. Voxel-wise Rigid Motion

Individual correspondences are noisy, and a dense per-point displacement field would be both unstable and expensive to apply. Instead, we summarize the motion as a piecewise-rigid field over a regular voxel grid, which is robust to per-point noise and yields a transform that can be shared by all Gaussians in a local region.

We assign each correspondence to the voxel $\mathcal{V}_s(\mathbf{p}_d)$ containing its current position, i.e. the cell with integer index $\lfloor \mathbf{p}_d/s \rfloor$, where s is the voxel size. For a voxel $\mathcal{V}$ that contains at least τ_{obs} correspondences, we fit the rigid transform $(R_{\mathcal{V}}, \mathbf{t}_{\mathcal{V}})$ that best maps source points to destinations:

$$R_{\mathcal{V}},\, \mathbf{t}_{\mathcal{V}} \;=\; \arg\min_{R\in SO(3),\, \mathbf{t}\in\mathbb{R}^3} \sum_{i\in\mathcal{V}} \left\| R\,\mathbf{p}_s^i + \mathbf{t} - \mathbf{p}_d^i \right\|^2. \quad (5)$$

This orthogonal Procrustes problem has a closed-form solution given by the Kabsch algorithm. Using the centroids $\bar{\mathbf{p}}_s, \bar{\mathbf{p}}_d$ of the points in $\mathcal{V}$ and the cross-covariance

$$H = \sum_{i\in\mathcal{V}} (\mathbf{p}_s^i - \bar{\mathbf{p}}_s)(\mathbf{p}_d^i - \bar{\mathbf{p}}_d)^\top = U\Sigma V^\top, \quad (6)$$

the solution is

$$R_{\mathcal{V}} = V\,\mathrm{diag}\big(1, 1, \det(VU^\top)\big)\,U^\top, \qquad \mathbf{t}_{\mathcal{V}} = \bar{\mathbf{p}}_d - R_{\mathcal{V}}\,\bar{\mathbf{p}}_s, \quad (7)$$

where the determinant term enforces a proper rotation. Voxels with fewer than three correspondences cannot fully constrain a rotation and fall back to a pure translation, $\mathbf{t}_{\mathcal{V}} = \bar{\mathbf{p}}_d - \bar{\mathbf{p}}_s$. We treat all correspondences within a voxel equally; weighting them (e.g. by codec residual magnitude) yielded no measurable benefit, as the codec's motion compensation already aligns the matched regions sufficiently.

3.5. Multi-Scale Motion with Quality Selection

A single voxel size forces a fixed trade-off: large voxels pool many observations and provide stable but coarse motion, while small voxels capture fine, localized motion but are easily corrupted when too few observations fall inside them. We resolve this by computing rigid fields at several scales $\mathcal{S} = \{s_1 > s_2 > \cdots > s_L\}$ and selecting, per Gaussian, the scale that is both sufficiently observed and best fit.

Voxel validity: observation floor and multi-view consensus. Because the number of pixel observations on a surface patch scales with its area, we set a per-scale floor proportional to the squared voxel size:

$$\tau_{\text{obs}}(s) \;=\; \max\Big(\tau_{\min},\; \tau_0\,(s/s_1)^2\Big), \quad (8)$$

so that finer scales are admitted with proportionally fewer observations rather than being held to a rigid coarse-scale threshold. In addition, we require that a voxel's correspondences originate from at least K distinct cameras. This multi-view consensus rejects motion supported by a single view and we find $K=2$ sufficient to eliminate the spurious single-view transforms that would otherwise corrupt the affected Gaussians. A voxel is *valid* at scale s only if it passes both the observation floor and the consensus test:

$$|\mathcal{V}_s| \ge \tau_{\text{obs}}(s) \qquad \text{and} \qquad \big|\{\, c^i : i \in \mathcal{V}_s \,\}\big| \ge K. \quad (9)$$

Fit quality. For a fitted voxel, we measure how well its rigid transform explains its own correspondences. The per-correspondence residual is $r_i = \|R_{\mathcal{V}}\mathbf{p}_s^i + \mathbf{t}_{\mathcal{V}} - \mathbf{p}_d^i\|$. We summarize a voxel using two complementary statistics: its RMS fit residual and its inlier ratio,

$$\rho_{\mathcal{V}} = \sqrt{\tfrac{1}{|\mathcal{V}|}\sum_{i\in\mathcal{V}} r_i^2}, \qquad \eta_{\mathcal{V}} = \frac{1}{|\mathcal{V}|}\sum_{i\in\mathcal{V}} \mathbb{1}[\,r_i < \tau_{\text{in}}\,], \quad (10)$$

where τ_{in} is a fixed tolerance. A low $\rho_{\mathcal{V}}$ or a high $\eta_{\mathcal{V}}$ indicates coherent, rigidly moving content, whereas a voxel straddling multiple motions or corrupted by noise scores poorly on both metrics.

Per-Gaussian selection. For a Gaussian at position $\mathbf{g}$, let $\mathcal{V}_s(\mathbf{g})$ be its containing voxel at scale s, and let $\mathcal{S}(\mathbf{g}) = \{\, s \in \mathcal{S} : \mathcal{V}_s(\mathbf{g}) \text{ is valid} \,\}$ be its set of valid scales (Eq. 9). We assign the Gaussian the transform of the voxel chosen by one of three fixed rules—*finest-first* (FF), *fit-residual* (FR), and *inlier-ratio* (IR):

$$s^\star(\mathbf{g}) = \begin{cases} \min \mathcal{S}(\mathbf{g}) & (\text{FF}), \\ \arg\min\limits_{s\in\mathcal{S}(\mathbf{g})} \rho_{\mathcal{V}_s(\mathbf{g})} & (\text{FR}), \\ \arg\max\limits_{s\in\mathcal{S}(\mathbf{g})} \eta_{\mathcal{V}_s(\mathbf{g})} & (\text{IR}). \end{cases} \quad (11)$$

If $\mathcal{S}(\mathbf{g}) = \varnothing$, the Gaussian remains unchanged. FF always takes the smallest valid voxel, whereas FR and IR defer to whichever valid scale is most internally consistent. The latter two behave almost identically in practice—since the outlier rate is low after consensus filtering—and both improve upon FF by a margin that *grows* as finer scales are added. This confirms that a quality veto becomes highly valuable precisely when fine, noise-prone scales are introduced.

The floor and the veto are thus complementary: the floor prevents trusting an under-sampled fine voxel, while the veto selects the most coherent motion among adequately sampled candidates. Consequently, fine scales are utilized only where reliable, naturally falling back to coarser, stable motion elsewhere. The number of scales acts as a direct quality-speed control: a single scale provides the fastest update, while adding scales increases accuracy at a modest cost to runtime (Sec. 4).

3.6. Gaussian Update

The selected per-voxel transform is applied rigidly to each Gaussian. A Gaussian with center $\mathbf{g}$ and orientation $\mathbf{q}$ is updated by the transform $(R^\star, \mathbf{t}^\star)$ of its selected voxel:

$$\mathbf{g}' = R^\star \mathbf{g} + \mathbf{t}^\star, \qquad \mathbf{q}' = \mathbf{q}(R^\star) \otimes \mathbf{q}, \tag{12}$$

where $\mathbf{q}(R^\star)$ is the quaternion of $R^\star$ and $\otimes$ denotes quaternion multiplication (followed by renormalization). Because the transform is a rigid body motion, scale, opacity, and color are preserved, which keeps the appearance of moving content consistent across frames. The update is cumulative: the resulting G_t becomes the input to the next P-frame, so motion accumulates over the GOP until the next keyframe.

3.7. Handling Emerging Content

Rigid motion propagation cannot account for newly appearing content, such as disocclusions. Rather than dynamically spawning new Gaussians from P-frame residuals, we delegate emerging content entirely to the GOP structure. Modern video codecs typically route drastic appearance changes to intra-coded blocks rather than inter-frame residuals, meaning P-frame residuals carry little coherent 3D structure signal. Therefore, we avoid the complexity and instability of on-the-fly generation. Instead, the periodic keyframe reconstruction (Sec. 3.1) naturally incorporates new geometry and resets accumulated drift, allowing intermediate P-frames to focus strictly on highly efficient rigid motion propagation.

4. Experiments

4.1. Experimental Setup

Datasets. We evaluate our method on two datasets, N3DV [8] and PanopticSports. The N3DV dataset comprises six indoor scenes, each captured with up to 20 views at a resolution of 2704×2028 and 30 fps, with 300 frames per view; we downsample all frames by a factor of 0.5 for both training and evaluation. PanopticSports is reconstructed from the Panoptic Studio dataset [7] by Dynamic 3D Gaussians [11], and consists of six dynamic sports scenes, each recorded by 31 cameras at 640×360, 30 fps, and 150 frames. We use 27 cameras for training and reserve the remaining four (indices 0, 10, 15, and 30) for testing.

Baselines. We compare CoDGS against two representative online multi-view streaming methods, 3DGStream [14] and QUEEN [3]. For a fair comparison, we evaluate all methods under a unified end-to-end protocol that times the full decode–update–render loop, rather than the per-frame update alone. We additionally wrap the GPU operations of both baseline models with `torch.cuda.synchronize` before and after timing to obtain accurate CUDA execution times.

Metrics. We assess reconstruction quality on the held-out test views using PSNR and SSIM, and efficiency using the end-to-end per-frame latency, decomposed into stream decoding, Gaussian update, and novel-view rendering. A method is real-time if this loop fits within the 33 ms budget.

Implementation details. For all experiments, we treat only the first frame of each scene as a keyframe and encode the entire remaining sequence as P-frames, so that the codec yields single-reference motion vectors throughout (B-frames disabled, H.264). Accordingly, vanilla 3DGS [8] is applied only to the first frame—producing 150,000 initial points for N3DV and 200,000 for PanopticSports—and every method, including the baselines, updates this representation from the second frame onward. The voxel scales in $\mathcal{S}$ and the associated observation floor $(\tau_{\min}, \tau_0)$ are the main hyperparameters of our method. Since N3DV scenes are largely static and visually uniform, a small number of coarse scales suffices, whereas the more dynamic and varied motion in PanopticSports benefits from a wider range of finer scales; in both cases we searched over a range of voxel sizes and their corresponding observation floors that perform robustly across the sweep. The remaining thresholds τ_{mot} and τ_{in} are fixed across all scenes. All experiments are conducted on a single NVIDIA RTX 3090 GPU.

4.2. Comparisons

Tables 2 and 3 report per-scene PSNR, SSIM, and average per-frame processing time on N3DV and PanopticSports. The dominant difference is update cost: the optimization-based baselines re-fit the representation at every frame and need several seconds per frame, whereas CoDGS propagates the keyframe Gaussians with a single closed-form rigid update and completes each frame in tens of milliseconds.

Runtime. On N3DV, whose higher rendering resolution yields denser Gaussian sets and thus larger per-frame cost, our lightest configuration updates each frame in roughly 47–53 ms, i.e. around 20 fps. The lower-resolution PanopticSports sequences run in 33–36 ms, reaching near-30 fps interactive rates. In contrast, 3DGStream requires about 11 s per frame on N3DV and 4 s on PanopticSports, and QUEEN about 18 s and 7 s, respectively. CoDGS is therefore more than two orders of magnitude faster—over $200\times$ and $350\times$ faster than 3DGStream and QUEEN on N3DV, and over $100\times$ and $200\times$ faster on PanopticSports—while using no per-frame optimization.

Table 2. Quantitative comparison on N3DV (15 frames).

	sear_steak			*cook_spinach*			*cut_roasted_beef*			*flame_steak*			*flame_salmon_1*			*coffee_martini*		
Method	PSNR↑	SSIM↑	Time↓	PSNR↑	SSIM↑	Time↓	PSNR↑	SSIM↑	Time↓	PSNR↑	SSIM↑	Time↓	PSNR↑	SSIM↑	Time↓	PSNR↑	SSIM↑	Time↓
3DGStream [14]	23.61	0.8963	11.48	21.87	0.8668	10.81	25.37	0.8986	11.29	21.38	0.8724	11.06	23.75	0.8594	11.29	22.45	0.8471	11.49
QUEEN [3]	21.27	0.8512	18.68	19.61	0.8080	18.66	23.35	0.8663	18.07	20.39	0.8347	19.11	19.22	0.7772	18.56	19.03	0.7749	17.05
Ours (Light)	23.19	0.8743	**0.050**	21.31	0.8376	**0.047**	24.88	0.8786	**0.052**	21.03	0.8497	**0.049**	23.23	0.8428	**0.053**	21.69	0.8198	**0.052**
Ours (Medium)	23.22	0.8747	0.054	21.31	0.8378	0.057	24.89	0.8787	0.057	21.06	0.8503	0.054	23.26	0.8430	0.060	21.77	0.8202	0.059
Ours (Heavy)	23.25	0.8757	0.065	21.33	0.8383	0.063	24.92	0.8794	0.068	21.07	0.8509	0.064	23.27	0.8433	0.071	21.84	0.8209	0.069

Table 3. Quantitative comparison on PanopticSports (15 frames).

	basketball			*boxes*			*football*			*juggle*			*softball*			*tennis*		
Method	PSNR↑	SSIM↑	Time↓	PSNR↑	SSIM↑	Time↓	PSNR↑	SSIM↑	Time↓	PSNR↑	SSIM↑	Time↓	PSNR↑	SSIM↑	Time↓	PSNR↑	SSIM↑	Time↓
3DGStream [14]	21.39	0.7332	3.84	22.26	0.7570	3.96	22.54	0.7361	4.04	21.70	0.7548	4.03	22.45	0.7444	3.96	22.66	0.7475	3.90
QUEEN [3]	25.10	0.8575	6.90	25.73	0.8727	7.25	26.59	0.8737	7.35	24.64	0.8752	7.14	25.70	0.8731	7.17	26.32	0.8737	7.10
Ours (Light)	19.37	0.6506	**0.034**	20.78	0.6883	**0.035**	20.80	0.6560	**0.036**	20.77	0.6908	**0.035**	20.93	0.6741	**0.033**	21.14	0.6789	**0.035**
Ours (Medium)	19.47	0.6533	0.040	20.86	0.6887	0.039	20.81	0.6557	0.040	20.74	0.6885	0.041	20.89	0.6742	0.039	21.23	0.6788	0.039
Ours (Heavy)	19.49	0.6526	0.044	20.85	0.6886	0.046	20.79	0.6555	0.045	20.78	0.6874	0.044	20.89	0.6742	0.044	21.13	0.6767	0.044

Reconstruction quality. Because the baselines optimize every frame, they attain higher fidelity. CoDGS nonetheless stays competitive using only codec-derived motion: on N3DV it is within 0.3–0.6 dB PSNR of 3DGStream and exceeds QUEEN on every scene (e.g. 23.25 vs. 21.27 dB on *sear_steak*), widening to 1–2 dB below 3DGStream on the faster-moving PanopticSports. Figure 2 reflects this: 3DGStream is sharp and QUEEN is blurry, while CoDGS tracks the dynamic content—the moving pan and the subject's pose—but renders fine details such as the face less faithfully. CoDGS thus trades a modest amount of high-frequency fidelity for a two-to-three order-of-magnitude latency reduction.

Speed–quality variants. The Light, Medium, and Heavy configurations expose this trade-off as a single knob. Adding voxel scales from Light to Heavy raises the per-frame budget (e.g. 50 → 65 ms on N3DV *sear_steak*) and brings small quality gains (23.19 → 23.25 dB). All three variants stay far below the baselines' per-frame cost; even the Heavy configuration remains interactive (≈15 fps on N3DV, ≈22 fps on PanopticSports), so the deployment target can select the operating point.

Figure 2. **Qualitative comparison on N3DV (held-out view).** 3DGStream is sharp and QUEEN is blurry, both at seconds per frame; CoDGS (tens of ms) captures the pan motion and pose but softens fine detail such as the face.

4.3. Ablation Study

Effectiveness of motion propagation. Figure 3 evaluates the effectiveness of our motion propagation against a static baseline. As the GOP progresses, the static 3DGS representation inherently drifts from the live scene, causing a steady decline in both PSNR and SSIM. By contrast, applying our codec-driven rigid motion (MV ONLY) effectively compensates for this temporal displacement. The right panels show a clear, sustained positive gain in both PSNR and SSIM throughout the 15-frame window. Notably, the ΔPSNR shows an increasing trend, suggesting that our motion propagation becomes increasingly vital as the distance from the I-frame increases and the static representation drifts further from the ground truth. This confirms that codec motion vectors provide a reliable signal for maintaining tracking stability in real-time.

Multi-scale selection and scale sensitivity. Figure 4 isolates the per-Gaussian selection rule and the size of the added second scale: from a single-scale baseline, we add one coarser voxel of size s and track ΔPSNR and ΔSSIM as s varies. The finest-first (FF) rule never improves on the single-scale result—when a fine voxel is under-sampled, the single-scale model safely keeps the Gaussian static,

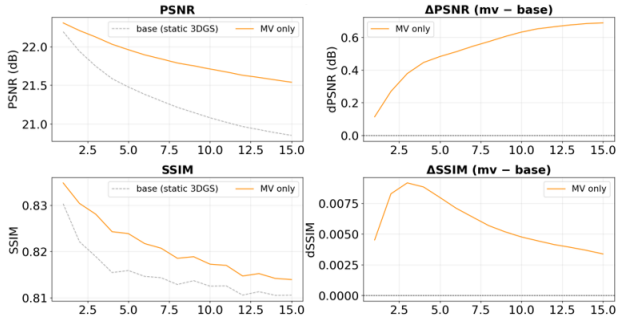

Figure 3. **Performance gain over static baseline.** MV ONLY vs. a static 3DGS baseline that freezes the keyframe geometry, over 15 frames. Left: raw PSNR/SSIM; right: ΔPSNR/ΔSSIM. The growing positive gap shows codec motion compensates for the drift the static baseline accumulates.

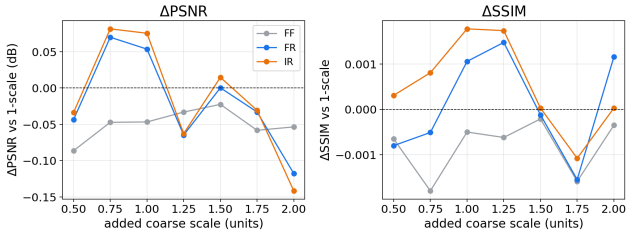

Figure 4. **Selection rule vs. added-scale size.** From a single-scale baseline we add one coarser voxel of size s (x-axis) and plot ΔPSNR (left)/ΔSSIM (right) for the three rules. FF never gains; FR/IR gain near $s \approx 1$ but reverse when s is too large.

whereas FF forces it to move with the coarse fallback. The quality-aware rules (FR, IR), in contrast, reach a net gain of up to $+0.08$ dB PSNR with simultaneous SSIM improvement near $s \approx 1$, since the fit-quality veto applies coarse motion only where it is internally coherent; the two track closely (IR marginally better on SSIM), consistent with the low post-consensus outlier rate. The gain is highly scale-sensitive, however, and reverses once s grows too large: at $s = 2$ the coarse voxel pools so many points that it scores artificially well on both $\rho_{\mathcal{V}}$ and $\eta_{\mathcal{V}}$ while averaging away local motion, so FR and IR are misled into this over-smoothed transform and fall even below FF. A moderate, fixed set of scales is therefore preferable to arbitrarily large ones.

The same behavior appears from an incremental view (Fig. 5), which adds one coarse scale ($1 \to 2$) and then one fine scale ($2 \to 3$, adding 0.5). Under the quality-aware rules both steps give positive gains— largest for the second scale and diminishing but still positive for the third— while FF degrades throughout. The quality veto thus acts as a robust filter, letting the framework exploit additional finer scales without being misled by their noise.

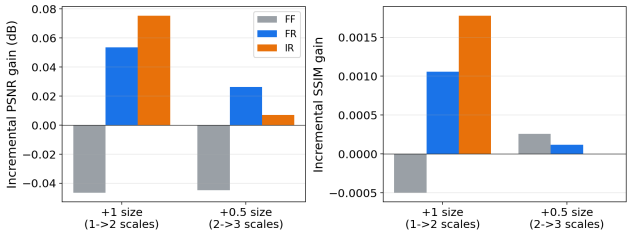

Figure 5. **Incremental gains from expanding the scale set.** Adding one coarse scale ($1 \to 2$, left) vs. one fine scale ($2 \to 3$, right). FF degrades; FR/IR give positive, diminishing gains.

5. Discussion and Limitations

While CODGS is training-free, our multi-scale motion field relies on a dataset-level calibration of the observation floor τ_{obs}, since the number of codec correspondences per voxel scales with camera density and image resolution. Denser rigs such as PanopticSports yield far more observations per voxel, so a single fixed threshold would admit noisy, weakly constrained voxels in high-density setups. We therefore calibrate τ_{obs} per dataset while keeping the same multi-scale voxel set across all sequences; adapting it automatically from the incoming camera geometry, without this manual step, is a compelling direction for future work.

A second limitation stems from our motion-only P-frame design. Propagating the scene purely by rigid motion lets small motion-estimation errors accumulate as drift over the GOP, so overall quality is bounded by the fidelity of the initial 3DGS keyframe. This is most visible in high-frequency, non-rigid regions—faces or loose clothing—where subtle deformations escape the rigid voxel model and appear as blur or ghosting until the next keyframe resets the geometry. Using the codec residual as a lightweight appearance correction, without reintroducing per-frame optimization, is a natural way to close this gap.

6. Conclusion

We presented CODGS, a training-free framework that updates an online dynamic 3D Gaussian representation directly from video-codec motion vectors, a dense motion signal the streaming client already decodes, eliminating the redundant per-frame motion re-estimation of prior methods. By lifting multi-view codec motion into a 3D voxel field and solving per-voxel rigid transforms in closed form, CODGS replaces optical-flow inference and per-frame optimization with a single closed-form update, reducing the per-frame cost from seconds to tens of milliseconds. On N3DV and PanopticSports it sustains real-time rates of roughly 20 and 30 fps, which is two to three orders of magnitude faster than optimization-based baselines reaching a regime those methods cannot.

References

[1] Jiacong Chen, Qingyu Mao, Youneng Bao, Xiandong Meng, Fanyang Meng, Ronggang Wang, and Yongsheng Liang. Motion matters: Compact Gaussian streaming for free-viewpoint video reconstruction. In *NeurIPS*, 2025. 2

[2] Qiankun Gao, Jiarui Meng, Chengxiang Wen, Jie Chen, and Jian Zhang. HiCoM: Hierarchical coherent motion for streamable dynamic scene with 3D Gaussian splatting. In *NeurIPS*, 2024. 1, 2

[3] Sharath Girish, Tianye Li, Amrita Mazumdar, Abhinav Shrivastava, David Luebke, and Shalini De Mello. QUEEN: QUantized Efficient ENcoding for streaming free-viewpoint videos. In *NeurIPS*, 2024. 1, 2, 6, 7

[4] Marti de La Gorce, Charlie Hewitt, Tibor Takacs, Robert Gerdisch, Zafiirah Hosenie, Givi Meishvili, Marek Kowalski (HE/HIM), Tom Cashman, and Antonio Criminisi. VoluMe - authentic 3D video calls from live gaussian splat prediction. In *ICCV*, 2025. 1, 2

[5] Bing He, Yunuo Chen, Guo Lu, Qi Wang, Qunshan Gu, Rong Xie, Li Song, and Wenjun Zhang. H3d-dgs: Exploring heterogeneous 3d motion representation for deformable 3d gaussian splatting. In *NeurIPS*, 2025. 2

[6] 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. In *CVPR*, 2024. 2

[7] Hanbyul Joo, Hao Liu, Lei Tan, Lin Gui, Bart Nabbe, Iain Matthews, Takeo Kanade, Shohei Nobuhara, and Yaser Sheikh. Panoptic studio: A massively multiview system for social motion capture. In *ICCV*, pages 3334–3342, 2015. 6

[8] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D Gaussian splatting for real-time radiance field rendering. *ACM TOG*, 42(4), 2023. 1, 2, 6

[9] Hojeong Lee, Yu Hong Kim, Sangwoo Ryu, James Won-Ki Hong, Sangtae Ha, and Seyeon Kim. DeltaStream: 2D-inferred delta encoding for live volumetric video streaming. In *ACM MobiSys*, 2025. 2

[10] Zhan Li, Zhang Chen, Zhong Li, and Yi Xu. Spacetime gaussian feature splatting for real-time dynamic view synthesis. In *CVPR*, pages 8508–8520, 2024. 1, 2

[11] Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3D Gaussians: Tracking by persistent dynamic view synthesis. In *3DV*, 2024. 6

[12] Debargha Mukherjee, Jim Bankoski, Adrian Grange, Jingning Han, John Koleszar, Paul Wilkins, Yaowu Xu, and Ronald Bultje. The latest open-source video codec vp9 – an overview and preliminary results. In *Picture Coding Symposium (PCS)*, pages 390–393, 2013. 1

[13] Gary J. Sullivan, Jens-Rainer Ohm, Woo-Jin Han, and Thomas Wiegand. Overview of the high efficiency video coding (hevc) standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 22(12):1649–1668, 2012. 1

[14] Jiakai Sun, Han Jiao, Guangyuan Li, Zhanjie Zhang, Lei Zhao, and Wei Xing. 3DGStream: On-the-fly training of 3d gaussians for efficient streaming of photo-realistic free-viewpoint videos. In *CVPR*, pages 20675–20685, 2024. 1, 2, 6, 7

[15] Penghao Wang, Zhirui Zhang, Liao Wang, Kaixin Yao, Siyuan Xie, Jingyi Yu, Minye Wu, and Lan Xu. V^3: Viewing volumetric videos on mobiles via streamable 2D dynamic Gaussians. *ACM TOG*, 43(6), 2024. 3

[16] T. Wiegand, G.J. Sullivan, G. Bjontegaard, and A. Luthra. Overview of the H.264/AVC video coding standard. *IEEE TCSVT*, 13(7):560–576, 2003. 1

[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. In *CVPR*, pages 20310–20320, 2024. 1, 2

[18] Jinbo Yan, Rui Peng, Zhiyan Wang, Luyang Tang, Jiayu Yang, Jie Liang, Jiahao Wu, and Ronggang Wang. Instant Gaussian stream: Fast and generalizable streaming of dynamic scene reconstruction via Gaussian splatting. In *CVPR*, 2025. 1, 2

[19] Ziyi Yang, Xinyu Gao, Wen Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. Deformable 3D Gaussians for high-fidelity monocular dynamic scene reconstruction. In *CVPR*, 2024. 1, 2

[20] Zihan Zheng, Zhenlong Wu, Houqiang Zhong, Yuan Tian, Ning Cao, Lan Xu, Jiangchao Yao, Xiaoyun Zhang, Qiang Hu, and Wenjun Zhang. 4DGCPro: Efficient hierarchical 4D Gaussian compression for progressive volumetric video streaming. In *NeurIPS*, 2025. 2