

Motion-Guided Streaming Updates for Dynamic 3D Gaussian Splatting Scenes

Ziwen Yuan

CMU-CS-26-122

August 2026

Computer Science Department
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:

Fernando De la Torre, Chair
Louis-Philippe Morency, Reader

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science.

Keywords: Computer vision, dynamic scene reconstruction, free-viewpoint video, novel view synthesis, 3D Gaussian Splatting, streaming reconstruction, optical flow, motion-guided updates

Dedication

To my family.

Abstract

Free-viewpoint video has broad potential in immersive sports replay, telepresence, live performance capture, and robotics, where users need to view a dynamic scene from controllable viewpoints as it evolves. These applications require more than photorealistic rendering: the underlying 3D representation must also be updated efficiently enough to support streaming use. 3D Gaussian Splatting is a promising foundation for this setting because it represents scenes with explicit primitives that render quickly and can be locally optimized. However, many existing dynamic Gaussian pipelines still depend heavily on iterative photometric refinement to infer frame-to-frame motion. When objects undergo large displacement, much of the optimization budget can be spent recovering motion rather than improving appearance, making high-quality streaming updates expensive.

This thesis proposes *OpticalSplat*, a streaming Gaussian update framework that uses optical flow as a strong motion prior before refinement. The method estimates visible 2D displacement, lifts this evidence into group-level rigid motion proposals, and uses persistent ownership labels to move related Gaussians together. This motion-guided initialization places the Gaussian state closer to the next frame before optimization begins, allowing a small photometric refinement budget to focus on residual alignment, color, opacity, and local shape corrections.

OpticalSplat is evaluated against representative streaming and dynamic Gaussian baselines using reconstruction quality and per-frame update time. The experiments show a more favorable speed-quality trade-off: OpticalSplat reduces update time while preserving competitive reconstruction quality, narrowing the gap to more expensive tracking-based refinement without requiring the same optimization budget.

Acknowledgments

I would like to thank my advisor, Fernando De la Torre, for his guidance, support, and thoughtful feedback throughout this thesis. I am also grateful to Louis-Philippe Morency for serving on my thesis committee and for his valuable time and feedback.

I thank all of my lab mates for the many discussions and shared work along the way, especially Saswat. I am also deeply grateful to my friends and family for their support and encouragement throughout this journey.

Contents

1	Introduction	1
1.1	Background and Motivation	1
1.2	Thesis Goal and Scope	2
1.3	Overview	2
2	Background and Related Work	3
2.1	3D Gaussian Splatting	3
2.2	Dynamic 3D Scene Representations	4
2.3	Streaming Reconstruction	5
2.4	Evaluation of Dynamic View Synthesis	5
3	Baseline Methods	7
3.1	Overview	7
3.2	3DGStream	7
3.3	HiCoM	7
3.4	QUEEN	8
3.5	IGS	8
3.6	ReCon-GS	8
3.7	TrackerSplat	9
3.8	Baseline Comparison Summary	9
4	Proposed Method	11
4.1	Streaming Problem Setup	11
4.2	OpticalSplat Workflow	11
4.3	Canonical Semantic Ownership	11
4.4	Per-View Optical Flow	12
4.5	Lifting 2D Motion to 3D Gaussian Motion	13
4.6	Rigid Motion Solver	13
4.7	Short Photometric Refinement	14
5	Experiments	17
5.1	Experimental Setup	17
5.2	Metrics	17
5.3	Quantitative Results	17
5.4	Qualitative Short-Budget Comparison	17
5.5	Discussion	18

6	Discussion	21
6.1	Practical Lessons from Baseline Bring-up	21
6.2	Trade-offs	21
6.3	Limitations	22
7	Conclusion	23
A	Baseline Implementation Details	25
A.1	HiCoM Compact Initialization	25

List of Figures

4.1	Streaming reconstruction setup. A high-quality canonical Gaussian state and scene priors are built once from multi-view observations. Each following frame is updated causally from the previous Gaussian state and incoming image sequence, so the per-frame stage must spend most of its budget on fast motion alignment rather than full reconstruction.	12
4.2	OpticalSplat workflow. Canonical semantic segmentation provides persistent object and body-part ownership. Per-frame optical flow gives visible 2D motion cues, which are lifted to 3D Gaussian motion, segmented into rigid motion groups, and converted into SE(3) transformations before short photometric refinement.	13
4.3	Canonical-frame semantic ownership. SAM2 segments foreground objects such as the ball, while SAPIENs separates human body parts. These labels are computed once on the canonical frame and then reused throughout the stream to define which Gaussians should move together.	14
4.4	Per-view optical flow. OpticalSplat computes 2D motion vectors in each selected camera view. The vectors indicate visible displacement of the ball, arm, and hand regions, and form the evidence that is later lifted to 3D Gaussian motion.	15
4.5	Lifting 2D camera-space flow to 3D Gaussian motion. For a moving pixel, the Gaussian with the largest rendering contribution is treated as the 3D support of that observation. The image-space displacement therefore becomes a constraint on the motion of the corresponding 3D Gaussian.	16
4.6	Rigid motion solver. Each motion group is represented by its Gaussian centers and multi-view 2D flow observations. OpticalSplat solves for a group-level SE(3) transform by minimizing robust reprojection residuals, then applies the optimized transform to the group Gaussians.	16
5.1	PSNR versus per-frame update speed. OpticalSplat Short improves quality at a runtime close to the fastest streaming baselines, while OpticalSplat Long reaches the strongest PSNR among the reported methods with substantially lower update time than TrackerSplat.	19
5.2	Qualitative comparison on Panoptic seq_5 under limited per-frame refinement. OpticalSplat uses optical-flow motion cues to reduce the visible ball duplication and motion ghosts that remain in the TrackerSplat render.	19

List of Tables

3.1	Taxonomy of streaming 3D Gaussian update strategies. Each method reuses a previous scene state, but differs in the prior used to predict the next state and the unit on which that prior acts.	9
5.1	Panoptic results. QUEEN runtime is estimated from the logged per-frame update intervals after the initial frame. Lower update time is better; higher PSNR is better.	18

Chapter 1

Introduction

1.1 Background and Motivation

Free-viewpoint video aims to reconstruct a dynamic 3D scene so that a viewer can render it from arbitrary viewpoints over time. This capability is central to immersive media, telepresence, sports capture, virtual production, robotics, and other interactive applications where ordinary 2D video is too restrictive. In these settings, visual quality alone is not enough. A practical system must also update quickly, render at interactive rates, and keep the transmitted or stored representation compact.

3D Gaussian Splatting has changed the design space for novel view synthesis by combining explicit 3D primitives with fast differentiable rasterization [6]. Compared with many neural radiance field pipelines, Gaussian representations can train and render much faster while preserving high visual quality. This efficiency has made them a natural foundation for dynamic scene reconstruction and free-viewpoint video. Early dynamic Gaussian methods showed that Gaussians can be moved, rotated, and regularized over time to model non-rigid scenes and provide dense tracking [11]. However, many dynamic reconstruction methods still assume offline access to a complete video sequence and can spend substantial per-scene optimization time before the result is available.

Streaming reconstruction imposes a stricter requirement. Frames arrive over time, so the representation must be updated online instead of being optimized after the full sequence is known. Recent methods address this problem from several directions. 3DGStream uses a compact Neural Transformation Cache to model per-frame Gaussian motion efficiently [15]. HiCoM uses hierarchical coherent motion and compact refinement to reduce training time and storage [4]. QUEEN models and compresses Gaussian attribute residuals with quantization and sparsity for online free-viewpoint video [5]. ReCon-GS further studies adaptive hierarchical anchors and storage-aware optimization [3]. Other recent systems explore generalizable motion prediction, feature-stream compression, level-of-detail structures, feed-forward online reconstruction, and tracking assisted pre-alignment [19, 7, 10, 18, 20].

Together, these papers show that streaming 3D Gaussian reconstruction is not a single-metric problem. A method can improve PSNR by increasing optimization time, but that may make it less useful for online deployment. A method can compress aggressively, but that may reduce fidelity or temporal stability. A method can update quickly on short sequences, but may accumulate error over longer streams. The central question of this thesis is therefore how to improve the quality-speed trade-off for dynamic Gaussian streaming. The ideal method should shift the quality-time curve upward: for a fixed runtime budget it should produce better reconstruction quality, and for a target quality it should require less optimization time.

1.2 Thesis Goal and Scope

This thesis studies streaming dynamic reconstruction through the lens of 3D Gaussian Splatting. The work has two connected parts. First, it brings together representative dynamic and streaming Gaussian baselines in a common local framework. This part is practical as much as it is experimental: each method comes with its own assumptions about data layout, initialization, camera calibration, optimization schedule, and output format. A shared framework makes it possible to reason about these systems in a consistent way rather than treating each result as an isolated repository-specific artifact.

Second, the thesis develops OpticalSplat, a streaming update framework that uses optical flow and persistent Gaussian ownership as strong priors before per-frame optimization. The working hypothesis is that a streaming method should not wait for photometric gradients alone to discover motion. If the method can first transport the Gaussian state close to the next frame, optimization can spend its limited iterations on refinement rather than recovering large inter-frame displacement.

The scope of the thesis is intentionally focused. The goal is not to solve all forms of online 4D reconstruction, nor to build a complete networked streaming system. Instead, the thesis focuses on the reconstruction side of the problem: how quickly and reliably a Gaussian representation can be updated for incoming dynamic frames. Transmission size, rendering speed, and memory footprint remain important supporting metrics, but the primary emphasis is the relationship between reconstruction quality and update cost.

1.3 Overview

Chapter 2 reviews 3D Gaussian Splatting, dynamic scene representations, and streaming reconstruction. Chapter 3 summarizes the baseline methods and their main advantages and limitations. Chapter 4 presents OpticalSplat, including canonical ownership, optical-flow lifting, rigid motion estimation, and short refinement. Chapter 5 evaluates the method on Panoptic sequences using PSNR and update time. Chapter 6 discusses trade-offs and limitations, and Chapter 7 concludes the thesis.

Chapter 2

Background and Related Work

2.1 3D Gaussian Splatting

Neural Radiance Fields represent a scene as a continuous function that maps a 3D position and viewing direction to density and color, and render images by sampling points along camera rays [12]. This formulation established a powerful approach for novel view synthesis, but the volumetric sampling process can make both training and rendering expensive. For streaming dynamic scenes, this cost is especially important: if each new time step requires substantial per-scene optimization, the representation cannot keep up with incoming video frames.

Before 3DGS, several works tried to reduce this cost by replacing a purely implicit MLP with more explicit or factorized structures. K-Planes represents static and dynamic radiance fields with collections of learned 2D planes, offering a compact and efficient alternative to dense high-dimensional grids [2]. HyperReel targets high-fidelity 6-DoF video with a compact dynamic volume and ray-conditioned sampling [1]. These methods are important because they show a broader trend: dynamic view synthesis needs representations that are not only expressive, but also efficient enough to train, render, store, and update.

3D Gaussian Splatting replaces implicit volumetric sampling with an explicit set of anisotropic 3D Gaussian primitives [6]. Each Gaussian stores a 3D mean, opacity, anisotropic covariance, and view-dependent color parameters, commonly represented with spherical harmonics. During rendering, Gaussians are projected into the image plane and alpha-composited in visibility order. This explicit representation preserves differentiability, so the Gaussian parameters can be optimized from posed multi-view images, while the rasterization-based renderer is much faster than dense ray marching.

The static 3DGS training pipeline usually starts from camera poses and a sparse Structure-from-Motion point cloud. The points initialize Gaussian centers, and optimization updates position, opacity, covariance, and appearance to minimize image reconstruction losses. Density control is interleaved with optimization: Gaussians can be split, cloned, pruned, or otherwise adjusted so that the representation allocates capacity where the scene requires detail. This combination of explicit primitives, differentiable rendering, and adaptive density control is a major reason 3DGS has become a common foundation for real-time novel view synthesis.

These same properties make 3DGS attractive for dynamic and streaming reconstruction. Because the scene is represented by explicit primitives, one can attempt to update the representation by moving existing Gaussians, changing their attributes, adding new Gaussians for emerging content, or pruning irrelevant ones. Dynamic 3D Gaussians, for example, treats Gaussians as persistent scene elements whose positions and rotations evolve over time [11]. Streaming methods build on the

same basic idea, but must update the representation without waiting for the full sequence [15, 4, 5].

At the same time, the flexibility of 3DGS creates challenges. The Gaussian set is unstructured, so correspondences across time are not automatically known. Fast motion can move scene content outside the region where gradient-based updates provide useful signals, leading to slow convergence or temporal artifacts [20]. Density control can improve quality, but it also changes the representation over time and can increase memory or storage cost. For this thesis, 3DGS is therefore both the enabling representation and the source of the central problem: how to update a high-quality explicit Gaussian scene fast enough for streaming dynamic reconstruction.

2.2 Dynamic 3D Scene Representations

Dynamic reconstruction extends novel view synthesis from a static radiance field to a time-varying scene. A common approach is to represent the scene with a canonical state and a deformation model that maps this state to each time step. This design separates appearance and geometry that persist over time from motion that changes across frames. It is useful because it can share parameters across a sequence, but it also requires the deformation model to explain all visible motion. When the motion is large, non-rigid, or partially occluded, the deformation field can become difficult to optimize and may blur together distinct physical elements.

This canonical-and-deformation view appears in early dynamic NeRF methods. D-NeRF extends NeRF to dynamic scenes by learning a deformation from observation space to a canonical space [14], while Nerfies models non-rigid scenes using deformable neural radiance fields [13]. Multi-view free-viewpoint video methods such as Neural 3D Video Synthesis further show how dynamic radiance fields can produce high-quality results from dense camera rigs, but they are generally designed for offline reconstruction rather than online updates [9]. These works establish the core modeling idea that later Gaussian methods inherit: preserve reusable scene structure when possible, and model time through motion, deformation, or residual change.

Gaussian-based dynamic methods inherit this tension. Dynamic 3D Gaussians keeps a persistent set of Gaussians and allows their positions and rotations to change over time while regularizing local neighborhoods to move coherently [11]. This gives the representation a Lagrangian interpretation: Gaussians can be viewed as scene particles that follow physical space over time. The same design also makes tracking and editing possible, because the identity of a Gaussian is preserved across frames. However, this kind of persistent dynamic modeling still uses sequence-level optimization and does not directly solve the online streaming problem.

Other dynamic Gaussian approaches place more emphasis on compact temporal encoding. Instead of optimizing a fully independent Gaussian set at each frame, they encode changes relative to a previous or canonical representation. This idea appears in residual-based streaming methods such as QUEEN, which models Gaussian attribute residuals and then compresses them with quantization and sparsity [5], as well as in feature-stream methods such as GIFStream, which adds time-dependent feature streams to improve motion modeling while preserving temporal correspondence for compression [7]. These methods show that the dynamic representation is not only a matter of reconstruction accuracy; it also determines how much data must be updated, stored, or transmitted over time.

A key challenge across dynamic representations is correspondence. If a method knows which scene content at time t corresponds to content at time $t + 1$, it can update motion and appearance more efficiently. Without such information, the method must discover motion through image reconstruction gradients alone. For slow motion, gradients may be sufficient. For fast motion, the relevant object may move far from where a Gaussian currently projects, so the gradient signal can

be weak or misleading. TrackerSplat makes this issue explicit by using point tracking to pre-align Gaussians before optimization, improving robustness under large inter-frame motion [20]. This observation motivates the motion-prior direction of this thesis: optical flow or tracking information may provide a useful bridge between frame-to-frame correspondence and efficient Gaussian updates.

2.3 Streaming Reconstruction

Streaming reconstruction differs from offline dynamic reconstruction in when information becomes available. In an offline setting, the method can observe the full sequence, optimize temporal structure globally, and revise earlier frames after seeing later ones. In a streaming setting, frames arrive incrementally. The system must update the 3D representation using only the current or recent observations, and the output should become useful before the stream has moved far ahead.

Streaming radiance field work studied this problem before the recent wave of Gaussian methods. StreamRF formulates dynamic radiance field reconstruction as incremental learning, training a base explicit-grid representation and then learning compact frame-to-frame model differences [8]. VideoRF takes a transmission-oriented view by representing dynamic radiance fields as 2D feature video streams that can benefit from standard video codecs [16]. These methods are not Gaussian-based, but they clarify two central streaming concerns that remain relevant here: how quickly the representation can be updated, and how compactly the update can be stored or transmitted.

3DGStream is an early Gaussian-based method designed around this online constraint. Rather than optimizing all Gaussian attributes independently for each frame, it uses a compact Neural Transformation Cache to model Gaussian translation and rotation, and adds new Gaussians when emerging content cannot be explained by the existing representation [15]. This reduces per-frame training time and storage compared with naive per-frame optimization. HiCoM further improves streamable reconstruction by exploiting hierarchical coherent motion, compact initialization, and ongoing refinement with pruning to control representation size [4]. ReCon-GS uses adaptive hierarchical anchor Gaussians and storage-aware optimization to preserve motion expressiveness while reducing memory requirements [3].

Recent work also explores different definitions of streaming efficiency. QUEEN focuses on online free-viewpoint video transmission, where each time step must be represented by a compact update packet [5]. StreamLoD-GS uses a level-of-detail Gaussian structure, motion partitioning, and quantized residual refinement to support sparse-view streaming video reconstruction [10]. StreamSplat takes a different direction by using a feed-forward model for online reconstruction from uncalibrated video streams, emphasizing latency and scalability rather than per-scene iterative optimization [18]. These methods share the same broad goal, but they optimize different bottlenecks: update time, rendering speed, storage, bandwidth, sparse-view robustness, or long-sequence stability.

This thesis focuses on the reconstruction update bottleneck. A method may render at hundreds of frames per second after a representation has been built, but still require seconds to update the representation for the next time step. For live or near-live reconstruction, this update cost is central. The streaming question is therefore not only whether a method can eventually render a high-quality dynamic scene, but whether it can produce a useful update under a short per-frame budget.

2.4 Evaluation of Dynamic View Synthesis

Dynamic view synthesis is commonly evaluated with image-quality metrics such as PSNR, SSIM, and LPIPS. PSNR measures pixel-level reconstruction fidelity, SSIM measures structural similarity,

and LPIPS measures perceptual similarity using deep image features [17, 21]. These metrics are useful because they allow direct comparison to ground-truth held-out camera views. For dynamic scenes, they are usually averaged over both camera views and time steps.

For streaming reconstruction, final image quality is only part of the story. A method that reaches high PSNR after many iterations may be less useful than a method that reaches slightly lower PSNR much faster. Therefore, update time per frame, reconstruction throughput, rendering FPS, and memory or storage per frame should be reported alongside image quality. Existing streaming Gaussian papers often emphasize different combinations of these quantities: 3DGStream reports per-frame training time, rendering speed, storage, and PSNR [15]; HiCoM emphasizes training time, rendering FPS, storage, and robustness [4]; QUEEN emphasizes quality, training speed, rendering speed, and compressed per-frame size [5].

In this thesis, the primary evaluation view is the quality-speed trade-off for reconstruction updates. The reported quality metrics are PSNR and LPIPS, and the reported speed metric is update seconds per frame. For selected longer runs, total wall-clock update time is also reported. These measurements support two direct comparisons: quality under the same short refinement budget, and quality achieved at different update-time operating points.

This framing is intentionally narrower than a full networked streaming evaluation. Bitrate, storage, rendering FPS, and temporal consistency remain important supporting measurements, but they are not the main evidence used in this thesis. The central experimental question is whether a motion-prior update can improve the reconstruction quality obtained per unit of optimization time.

Chapter 3

Baseline Methods

3.1 Overview

Each baseline is discussed in terms of four questions: what representation it updates, how it models motion, what it optimizes per frame, and what trade-off it makes among quality, update time, rendering speed, and storage.

The order of this chapter follows the approximate development of online Gaussian streaming methods: from 3DGStream, to compact regional motion in HiCoM, residual encoding in QUEEN, learned anchor-based motion in IGS, later storage-aware hierarchical anchors in ReCon-GS, and finally tracking- or flow-guided correspondence.

3.2 3DGStream

3DGStream introduces an online Gaussian pipeline for free-viewpoint video streaming [15]. Starting from an initial 3DGS model, it updates subsequent frames by learning a Neural Transformation Cache (NTC) that predicts Gaussian translations and rotations. This avoids directly optimizing a full independent Gaussian model for every frame.

The workflow is simple: initialize from the first frame, learn frame-to-frame Gaussian transformations with the NTC, and add new Gaussians when emerging scene content cannot be explained by the previous representation. The main advantage is that motion is represented compactly, reducing per-frame training time and storage. The main limitation is that the method still depends heavily on the initial 3DGS and on gradient-based learning of motion. Large motion or accumulated scene change can require additional Gaussians and may degrade the quality-speed trade-off.

3.3 HiCoM

HiCoM improves the 3DGStream-style pipeline by making the initial representation more compact and the motion model more structured [4]. It first trains a compact initial 3DGS model, then estimates frame-to-frame motion with a hierarchical regional model. Details of the compact initialization are included in Appendix A.

The regional model divides the scene into coarse-to-fine spatial regions. Instead of assigning an independent motion to every Gaussian, Gaussians inside the same region share coherent motion components. This reduces the number of motion parameters and encourages nearby Gaussians to move consistently. After motion estimation, HiCoM adds Gaussians for new or changed content and prunes low-opacity Gaussians to keep the representation compact.

HiCoM’s strength is efficient motion learning from spatial coherence. Its limitation for this thesis is that the prior is spatial rather than correspondence-based. If a fast-moving object shifts far between frames, explicit optical-flow or tracking cues may provide a better initial motion estimate than regional coherence alone.

3.4 QUEEN

QUEEN approaches streaming from a residual encoding perspective [5]. Rather than imposing a fixed motion structure, it learns Gaussian attribute residuals between consecutive frames. To make these residuals practical for transmission, it uses a quantization-sparsity framework: non-position attribute residuals are compressed through learned quantized latents, while position residuals are sparsified with a learned gating module.

The method also uses view-space Gaussian gradient differences to identify dynamic content. This helps focus computation and storage on the parts of the scene that actually change. QUEEN is therefore useful in this chapter because it shifts the question from only “how fast can the model update?” to also “how compact is the per-frame update?” Its limitation for our current focus is that compression and transmission are not the main bottleneck studied in this thesis; we focus first on improving reconstruction quality under short update budgets.

3.5 IGS

Instant Gaussian Stream (IGS) targets faster and more generalizable streaming updates [19]. Its main component is an anchor-driven Gaussian motion network. Instead of learning motion only through per-scene iterative optimization, IGS projects multi-view 2D motion features into 3D space and uses anchor points to drive Gaussian motion. This allows motion for a target frame to be produced by a network inference step.

IGS also introduces a key-frame-guided streaming strategy. Key frames are refined more accurately, and nearby frames are updated using the learned motion model, reducing per-frame reconstruction time while controlling error accumulation. For this thesis, IGS is important because it moves toward a learned motion prior rather than purely optimizing each frame from scratch. It also supports the broader hypothesis that better motion initialization can improve the quality-speed curve.

3.6 ReCon-GS

ReCon-GS can be viewed as a later continuation of the hierarchical, compact-streaming direction explored by HiCoM [3]. Instead of using a uniform regional motion model, it allocates multi-level anchor Gaussians in a density-adaptive way. The anchors capture inter-frame geometric deformation from coarse to fine levels, while hierarchy reconfiguration updates the anchor structure when local motion requires more expressiveness.

The key workflow is anchor allocation, hierarchical deformation inheritance, and storage-aware optimization. Deformation inheritance keeps motion priors within each hierarchy level, helping temporal consistency. Storage-aware optimization adjusts anchor density to trade reconstruction fidelity against memory usage. Compared with HiCoM, ReCon-GS is more explicitly focused on preserving compactness while retaining local motion expressiveness.

3.7 TrackerSplat

TrackerSplat explicitly addresses a failure mode of gradient-based dynamic Gaussian updates: fast motion [20]. If an object moves far between frames, the Gaussian that represented it may no longer project near the correct image region. Image reconstruction gradients can then be too weak or misleading, causing fading, wrong recoloring, or unstable motion.

TrackerSplat uses off-the-shelf point tracking to estimate pixel trajectories and triangulates them onto 3D Gaussians. Before gradient-based optimization, it updates Gaussian position, rotation, and scale according to the tracking signal. This pre-alignment gives optimization a better starting point, especially for large inter-frame displacement.

This baseline is closest in spirit to the direction of this thesis. It suggests that explicit correspondence cues can complement 3DGS optimization. The 3DGStreamOptical repository follows a related idea with optical flow: precompute 2D flow, lift or aggregate it into per-Gaussian motion signals, and use those signals for initialization, masking, or regularization.

3.8 Baseline Comparison Summary

Table 3.1: Taxonomy of streaming 3D Gaussian update strategies. Each method reuses a previous scene state, but differs in the prior used to predict the next state and the unit on which that prior acts.

Method	Main prior	Update unit	Short-budget bottleneck
3DGStream	Photometric deformation learned online	Gaussian-level deformation field plus added Gaussians	Large motion must still be learned by local optimization
HiCoM	Spatial coherence	Coarse-to-fine regions	Compact regions do not provide object correspondence
QUEEN	Sparse attribute residuals	Quantized residual codes and gates	Compression improves rate, not large-motion initialization
IGS	Learned multi-view motion features	Anchor-driven Gaussian motion	Depends on learned priors and key-frame/cache quality
ReCon-GS	Adaptive anchor hierarchy	Density-adaptive anchors with inherited deformation	Anchor hierarchy may miss fine or fast correspondence
TrackerSplat	Point trajectories	Track-inferred per-Gaussian motion	Track failures under occlusion or thin structures
OpticalSplat	Optical flow and Gaussian ownership	Flow-component motion propagated to Gaussians	Flow/support errors and part-level motion limits

Chapter 4

Proposed Method

4.1 Streaming Problem Setup

OpticalSplat separates a dynamic sequence into a high-quality canonical initialization and fast per-frame updates. The canonical frame is processed once using all available posed views to build a clean Gaussian representation and scene priors. After this one-time step, the streaming stage receives the previous Gaussian state and the next image sequence, then updates the representation as quickly as possible before the next frame arrives.

At time t , the scene is represented by Gaussian primitives

$$\mathcal{G}_t = \{g_i^t\}_{i=1}^{N_t}, \quad g_i^t = (\mathbf{x}_i^t, \Sigma_i^t, \alpha_i^t, c_i^t, s_i), \quad (4.1)$$

where $\mathbf{x}_i^t$ is the 3D center, Σ_i^t is the covariance, α_i^t is opacity, c_i^t stores color parameters, and s_i is a persistent semantic or part ownership label. Given posed images $\mathcal{I}_{t+1}$, OpticalSplat predicts a transported Gaussian state before running short photometric refinement:

$$\mathcal{G}_{t+1} = \text{Refine}_K(\text{MotionProposal}(\mathcal{G}_t, \mathcal{I}_t, \mathcal{I}_{t+1})), \quad K \ll K_{\text{long}}. \quad (4.2)$$

The goal is to make K small by placing the Gaussians near their next-frame positions before iterative optimization begins.

4.2 OpticalSplat Workflow

Figure 4.2 summarizes the full update pipeline. The method first assigns persistent object and body-part labels on the canonical frame. For each new frame, it estimates 2D optical flow in selected views, lifts visible 2D motion to 3D Gaussian motion, groups compatible moving Gaussians into rigid bodies, and solves an SE(3) transformation for each group. The transformed Gaussian state is then passed to photometric refinement.

This design reflects the streaming constraint: expensive semantic reasoning is performed only on the canonical frame, while the per-frame stage uses faster geometric and optical-flow computations.

4.3 Canonical Semantic Ownership

The first-frame Gaussian model needs persistent ownership labels so that motion observed in one view can be propagated to the correct 3D support. We therefore run semantic segmentation only on the canonical frame. SAM2 is used to identify foreground objects, while SAPIENs provides

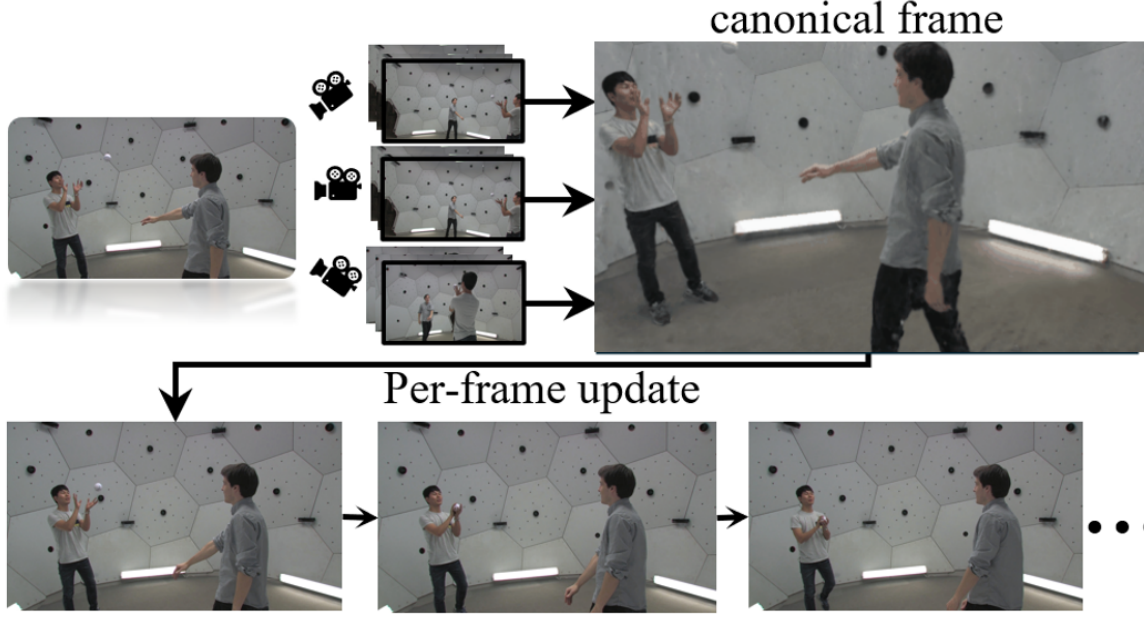


Figure 4.1: Streaming reconstruction setup. A high-quality canonical Gaussian state and scene priors are built once from multi-view observations. Each following frame is updated causally from the previous Gaussian state and incoming image sequence, so the per-frame stage must spend most of its budget on fast motion alignment rather than full reconstruction.

human body-part labels. The resulting masks are fused with the canonical Gaussian projection to assign each Gaussian a persistent label s_i .

Computing segmentation only once avoids repeatedly running large segmentation models during the streaming update. It also stabilizes ownership over time: when optical flow is sparse, noisy, or partially occluded, Gaussians with the same canonical ownership label can still inherit the same group motion.

4.4 Per-View Optical Flow

For each selected camera view v , OpticalSplat estimates the optical flow between frames t and $t+1$:

$$F_{v,t} : \Omega_v \rightarrow \mathbb{R}^2, \quad \mathbf{p}_v^1 = \mathbf{p}_v^0 + F_{v,t}(\mathbf{p}_v^0). \quad (4.3)$$

The flow is measured in 2D camera space and provides a dense image-space motion prior for visible regions.

Rather than treating every flow vector independently, the method aggregates flow within semantic support regions. For each label or connected moving component, robust statistics remove boundary outliers and unreliable vectors. This gives a set of observations

$$\mathcal{O}_k = \{(\mathbf{x}_i, \mathbf{p}_{i,v}^0, \mathbf{p}_{i,v}^1)\}, \quad (4.4)$$

where Gaussian i belongs to motion group k , projects to $\mathbf{p}_{i,v}^0$ before motion, and is expected to project near $\mathbf{p}_{i,v}^1$ after motion.

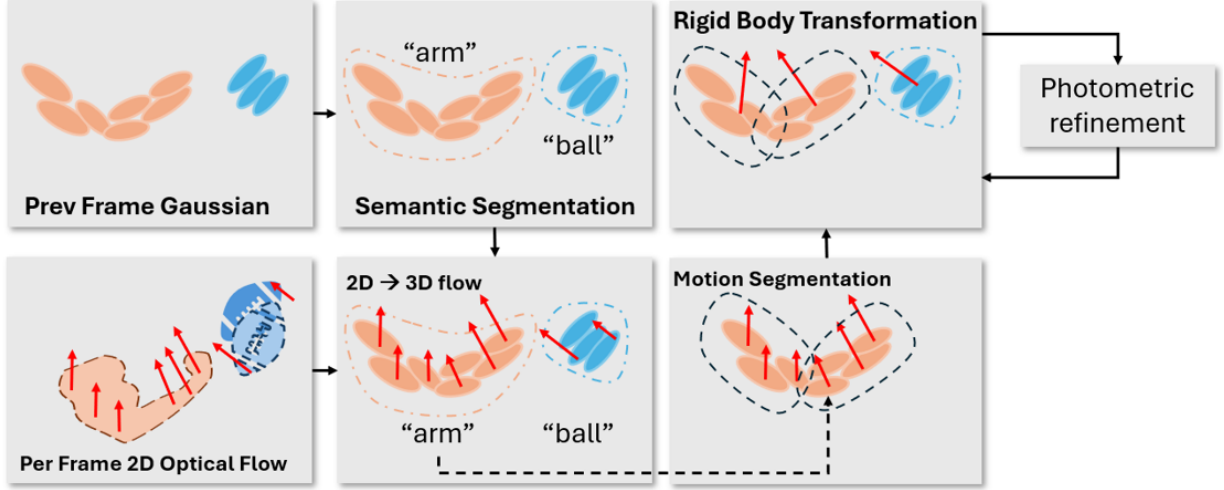


Figure 4.2: OpticalSplat workflow. Canonical semantic segmentation provides persistent object and body-part ownership. Per-frame optical flow gives visible 2D motion cues, which are lifted to 3D Gaussian motion, segmented into rigid motion groups, and converted into SE(3) transformations before short photometric refinement.

4.5 Lifting 2D Motion to 3D Gaussian Motion

The optical-flow vectors live in the camera image plane, but the representation must be updated in 3D. OpticalSplat bridges this gap by using the Gaussian that contributes most to a moving pixel as that pixel’s 3D support. For a pixel $\mathbf{p}$ in view v , differentiable Gaussian rasterization provides the front-to-back contribution weights of nearby Gaussians. We select the dominant Gaussian

$$i^*(\mathbf{p}, v) = \arg \max_i w_{i,v}(\mathbf{p}), \quad (4.5)$$

and associate the 2D flow vector at $\mathbf{p}$ with the 3D center $\mathbf{x}_{i^*}$. Aggregating these correspondences over all selected views produces a 3D motion proposal for each semantic or motion group.

This lifting step is intentionally lightweight. It does not solve a complete scene-flow problem for every Gaussian; it extracts enough reliable 3D motion evidence to initialize the next frame close to the target state.

4.6 Rigid Motion Solver

After lifting, OpticalSplat estimates a rigid SE(3) update for each motion group. Let

$$\mathcal{G}_k = \{\mathbf{x}_i\}_{i=1}^{N_k}, \quad \mathcal{O}_k = \{(\mathbf{x}_i, \mathbf{p}_{i,v}^0, \mathbf{p}_{i,v}^1)\} \quad (4.6)$$

denote the Gaussians and multi-view flow observations assigned to rigid body group k . A transformation with rotation vector $\boldsymbol{\omega}$ and translation $\mathbf{t}$ is parameterized as

$$T_\theta(\mathbf{x}_i) = R(\boldsymbol{\omega})(\mathbf{x}_i - \mathbf{c}_k) + \mathbf{c}_k + \mathbf{t}, \quad \theta = [\boldsymbol{\omega}, \mathbf{t}], \quad (4.7)$$

where $\mathbf{c}_k$ is the group center and $R(\boldsymbol{\omega})$ is the exponential-map rotation. The reprojection residual compares the observed 2D flow to the projected 3D motion:

$$\mathbf{r}_{i,v}(\theta) = (\mathbf{p}_{i,v}^1 - \mathbf{p}_{i,v}^0) - (\pi_v(T_\theta(\mathbf{x}_i)) - \mathbf{p}_{i,v}^0). \quad (4.8)$$



Figure 4.3: Canonical-frame semantic ownership. SAM2 segments foreground objects such as the ball, while SAPIENs separates human body parts. These labels are computed once on the canonical frame and then reused throughout the stream to define which Gaussians should move together.

The rigid update is obtained by minimizing a robust reprojection objective:

$$\theta^* = \arg \min_{\omega, \mathbf{t}} \sum_v \sum_{i \in \mathcal{G}_k} \rho(\|\mathbf{r}_{i,v}(\theta)\|_2^2). \quad (4.9)$$

In practice, Equation 4.9 is solved with a damped Gauss–Newton update:

$$(J^\top W J + \lambda D) \Delta \theta = J^\top W \mathbf{r}, \quad \theta \leftarrow \theta + \Delta \theta, \quad \mathbf{x}'_j = T_{\theta^*}(\mathbf{x}_j). \quad (4.10)$$

The damping term stabilizes updates when the flow observations are sparse or nearly degenerate, while the robust weights reduce the influence of unreliable flow near boundaries and occlusions.

4.7 Short Photometric Refinement

The transformed Gaussian state is finally refined with the standard rendering loss for a small fixed budget. For view v , the Gaussian renderer produces $\hat{I}_v(\mathcal{G})$, and the refinement minimizes

$$\mathcal{L}_{\text{photo}} = \sum_v \left[(1 - \lambda) \left\| I_v^{t+1} - \hat{I}_v(\mathcal{G}) \right\|_1 + \lambda \mathcal{L}_{\text{DSSIM}} \right]. \quad (4.11)$$

This step corrects residual misalignment, color, opacity, and local shape errors. It is not intended to rediscover all motion from scratch; the optical flow and rigid-body proposal should already move the Gaussian state close to the next frame.

The method has clear limitations. Incorrect flow can move Gaussians in the wrong direction, incomplete semantic ownership can miss support, and rigid groups cannot fully represent all non-rigid deformation. These limitations motivate the experiments in Chapter 5, which evaluate OpticalSplat as a quality-speed trade-off rather than as a universal replacement for long-refinement tracking-based reconstruction.

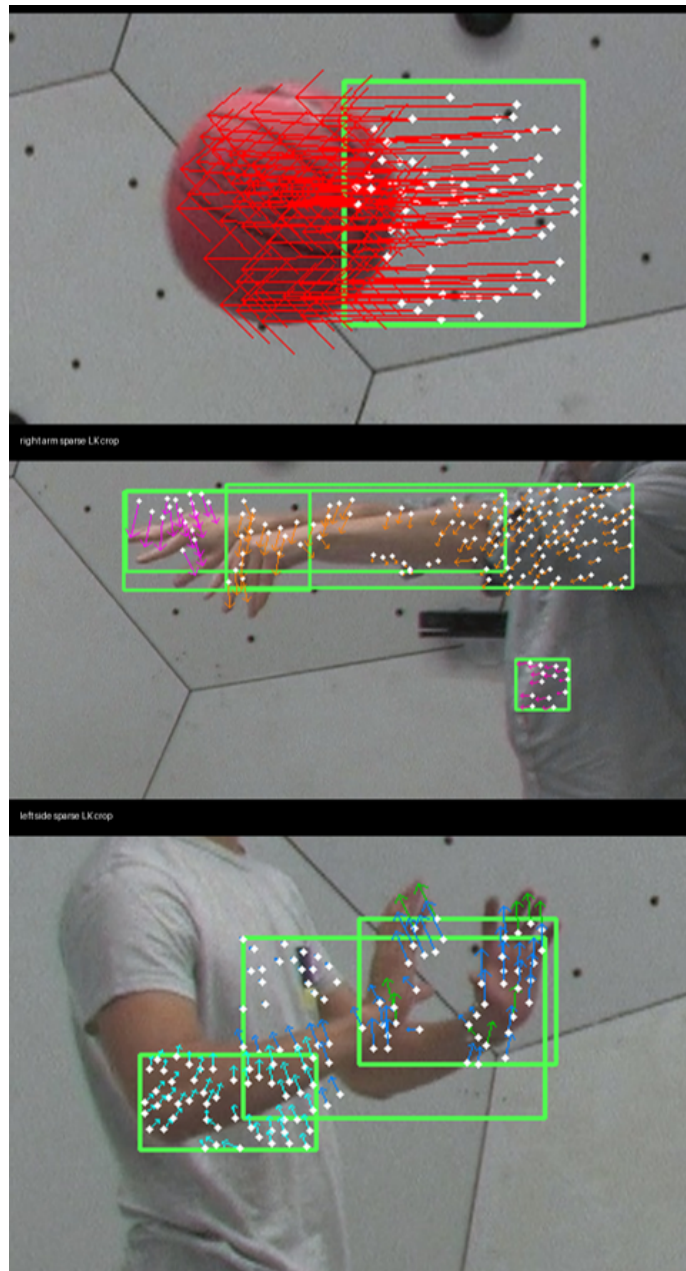


Figure 4.4: Per-view optical flow. OpticalSplat computes 2D motion vectors in each selected camera view. The vectors indicate visible displacement of the ball, arm, and hand regions, and form the evidence that is later lifted to 3D Gaussian motion.

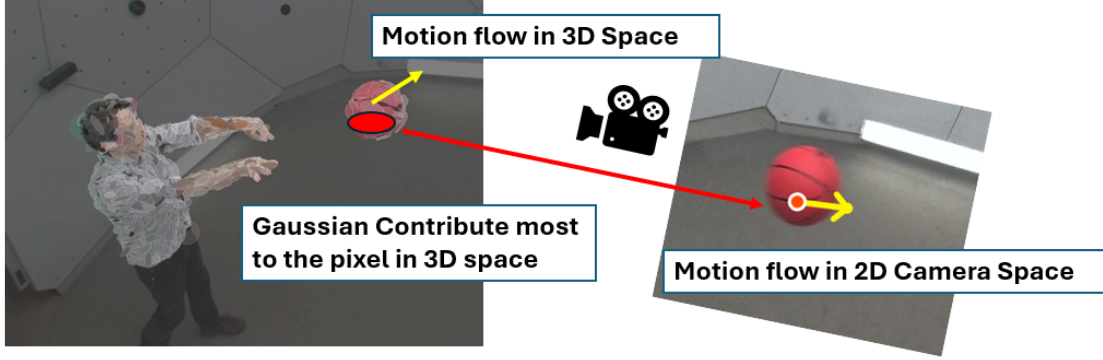


Figure 4.5: Lifting 2D camera-space flow to 3D Gaussian motion. For a moving pixel, the Gaussian with the largest rendering contribution is treated as the 3D support of that observation. The image-space displacement therefore becomes a constraint on the motion of the corresponding 3D Gaussian.

$$\mathcal{G}_k = \{\mathbf{x}_i\}_{i=1}^{N_k}, \quad \mathcal{O}_k = \{(\mathbf{x}_i, \mathbf{p}_{i,v}^0, \mathbf{p}_{i,v}^1)\}$$

$$T_\theta(\mathbf{x}_i) = R(\omega)(\mathbf{x}_i - \mathbf{c}_k) + \mathbf{c}_k + \mathbf{t}, \quad \theta = [\omega, \mathbf{t}]$$

$$\mathbf{r}_{i,v}(\theta) = (\mathbf{p}_{i,v}^1 - \mathbf{p}_{i,v}^0) - (\pi_v(T_\theta(\mathbf{x}_i)) - \mathbf{p}_{i,v}^0)$$

$$\theta^* = \arg \min_{\omega, \mathbf{t}} \sum_v \sum_{i \in \mathcal{G}_k} \rho(\|\mathbf{r}_{i,v}(\theta)\|_2^2)$$

$$(J^\top WJ + \lambda I) \Delta \theta = J^\top W \mathbf{r}, \quad \theta \leftarrow \theta + \Delta \theta, \quad \mathbf{x}'_j = T_{\theta^*}(\mathbf{x}_j)$$

Figure 4.6: Rigid motion solver. Each motion group is represented by its Gaussian centers and multi-view 2D flow observations. OpticalSplat solves for a group-level SE(3) transform by minimizing robust reprojection residuals, then applies the optimized transform to the group Gaussians.

Chapter 5

Experiments

5.1 Experimental Setup

The experiments use Panoptic multi-camera dynamic sequences and report the baseline results available in the local evaluation logs, together with the short and long operating points of OpticalSplat. All methods update a dynamic Gaussian representation over time under a shared evaluation format for PSNR and update time.

5.2 Metrics

Image quality is reported primarily with PSNR. LPIPS [21] is computed for the TrackerSplat and OpticalSplat runs where the local evaluation pipeline stored it, but it is omitted from the aggregate baseline table because the other wrapper baselines did not save comparable LPIPS summaries. Runtime is reported as update seconds per frame, including the motion proposal and photometric refinement needed to produce the next Gaussian state.

5.3 Quantitative Results

Table 5.1 summarizes the Panoptic results. The table is intentionally compact: each row is one method or method setting, and the columns report only PSNR and update time. OpticalSplat Short is the default low-latency setting, while OpticalSplat Long uses a larger refinement budget to show the quality-speed trade-off.

Figure 5.1 visualizes the same comparison as a quality-speed trade-off. The upper-left region corresponds to methods that update quickly while maintaining high PSNR. OpticalSplat occupies the most useful part of this frontier: the short setting is close to the fastest streaming baselines but provides substantially higher PSNR, while the long setting reaches the highest PSNR in this comparison at less than half of TrackerSplat’s update time. In this sense, OpticalSplat is not merely the fastest or merely the highest-quality method; it offers a better operating point among methods that care about both streaming efficiency and reconstruction fidelity.

5.4 Qualitative Short-Budget Comparison

Figure 5.2 compares TrackerSplat and OpticalSplat on the same Panoptic `seq_5` view. Each panel shows the ground-truth image on the left and the rendered result on the right. Under a short

Table 5.1: Panoptic results. QUEEN runtime is estimated from the logged per-frame update intervals after the initial frame. Lower update time is better; higher PSNR is better.

Method	PSNR $\uparrow$	Sec/frame $\downarrow$
3DGStream	26.107	13.014
HiCoM	24.581	13.313
Dynamic 3DGS	28.475	126.210
QUEEN	25.817	24.318
ReCon-GS	25.105	22.273
TrackerSplat	32.012	91.201
OpticalSplat Short	28.949	15.926
OpticalSplat Long	32.268	40.680

update budget, TrackerSplat can leave visible duplicated ball artifacts and motion ghosts, while OpticalSplat places the moving ball closer to the target frame before refinement.

5.5 Discussion

The results show the intended trade-off. OpticalSplat Short provides a low-latency operating point with competitive reconstruction quality, while OpticalSplat Long reaches the strongest PSNR among the reported rows at less than half of the TrackerSplat update time. These results support the main hypothesis of this thesis: a strong optical-flow motion prior can reduce the amount of per-frame refinement needed for streaming Gaussian reconstruction.

Broader dataset coverage remains important future work, but the observed operating points already show that motion-guided initialization can move the quality-speed curve in the desired direction.

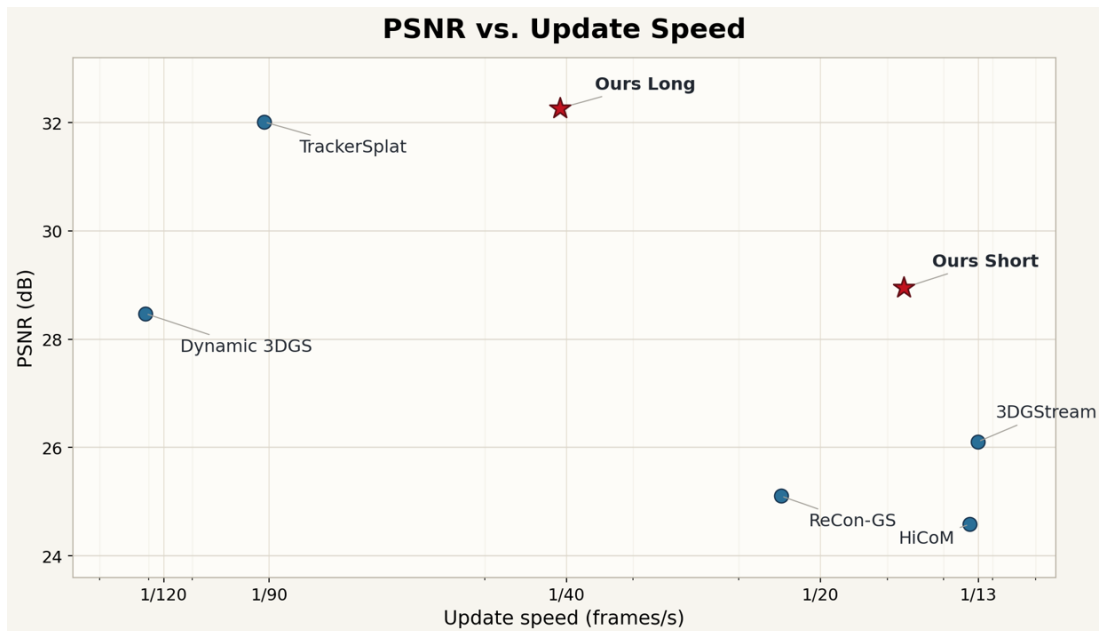


Figure 5.1: PSNR versus per-frame update speed. OpticalSplat Short improves quality at a runtime close to the fastest streaming baselines, while OpticalSplat Long reaches the strongest PSNR among the reported methods with substantially lower update time than TrackerSplat.

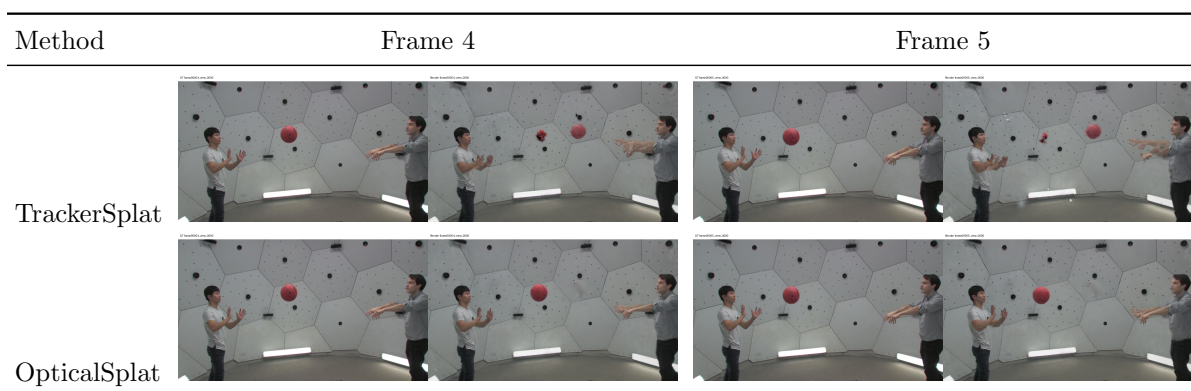


Figure 5.2: Qualitative comparison on Panoptic seq_5 under limited per-frame refinement. OpticalSplat uses optical-flow motion cues to reduce the visible ball duplication and motion ghosts that remain in the TrackerSplat render.

Chapter 6

Discussion

6.1 Practical Lessons from Baseline Bring-up

A major practical lesson is that streaming Gaussian baselines are difficult to compare without careful control of data conventions. Many repositories assume a specific camera format, image directory layout, initialization procedure, or coordinate scale. Small mismatches in camera translation scale or source-cache layout can create large quality differences that are unrelated to the method itself. For this reason, the experiments in this thesis use a shared Panoptic preparation path and refined canonical Gaussian sources whenever possible.

The second lesson is that smoke tests are necessary but not sufficient. A method can work well for the first few frames and still accumulate drift over longer sequences. Conversely, a method can look weak in a very short strict-streaming prototype but become useful after a better source convention, part bank, or refinement protocol is fixed. The thesis therefore reports diagnostic quality-speed results while treating broader long-horizon evaluation as an important next step.

The third lesson is that runtime must be reported together with iteration budget. A method may be faster than TrackerSplat1000 simply because it uses fewer refinement iterations. This is still meaningful for streaming, but the claim should be phrased as faster convergence or better use of a short update budget, not as a universally faster optimizer.

6.2 Trade-offs

The central trade-off is between update speed and reconstruction quality. TrackerSplat remains a strong quality baseline, but its update time is much higher in the reported comparison. OpticalSplat reduces update time by using optical-flow motion priors and short refinement. The short setting improves the low-latency operating point, while the long setting reaches the strongest reported PSNR with substantially lower update time than TrackerSplat. This means the method should be understood as a better quality-speed operating point, not as a complete replacement for high-iteration tracking-based optimization.

The method also trades flexibility for stability. Part-level motion is more stable and cheaper than estimating a separate unconstrained motion for every Gaussian. However, it is less expressive for articulated motion, fine hand motion, and interactions between small objects. Ownership labels help define which Gaussians should move together, but if the labels are coarse or misaligned, they can propagate motion to the wrong support.

LPIPS and PSNR can disagree. A method that improves perceptual artifacts or temporal stability may not always improve pixel-level alignment, and the reverse can also happen. For future

evaluation, qualitative inspection and temporal metrics should complement the numeric tables.

6.3 Limitations

The first limitation is that the current method is evaluated mainly on Panoptic-style sequences. N3DV-style scenes remain important for a broader thesis evaluation, especially because they include different camera layouts, motion patterns, and appearance changes.

The second limitation is that OpticalSplat depends on the quality of the first-frame ownership labels. If semantic labels are too coarse, moving and static regions can be merged into one part. If labels cover too little of the object, support propagation may miss part of the moving object. A stronger ownership or motion-part segmentation module could improve this component.

The third limitation is that the current motion model is rigid at the part level. This is efficient and more expressive than pure translation, but it still cannot represent all non-rigid motion. More expressive local transformations, confidence-weighted deformation, or learned residual motion could improve quality while preserving the short-refinement goal.

Finally, the experiments measure reconstruction update time, not a full networked streaming system. Bitrate, compression, rendering latency, and client delivery remain outside the main scope of this thesis. Those metrics are important for deployment, but the thesis first isolates the reconstruction question: how much quality can be obtained quickly after each new frame arrives?

Chapter 7

Conclusion

This thesis studies streaming 3D Gaussian Splatting through baseline analysis, method development, and controlled experiments. The main motivation is that dynamic Gaussian methods should not be judged only by final reconstruction quality. For streaming use, the representation must also update quickly as new frames arrive.

The proposed OpticalSplat method uses optical-flow motion priors for short-budget Gaussian updates. It combines persistent first-frame Gaussian ownership labels with online optical-flow motion components and group-level rigid motion estimation, then applies a short photometric refinement. Experiments show that this design improves the quality-speed operating point: OpticalSplat Short provides a low-latency setting with competitive quality, while OpticalSplat Long reaches the strongest reported PSNR at substantially lower update time than TrackerSplat.

The main conclusion is therefore measured but positive. Optical-flow guided Gaussian transport is a useful direction for improving the quality-speed trade-off in streaming Gaussian reconstruction, especially as a way to reduce dependence on long per-frame optimization. The next step is to make the motion prior more reliable: better persistent ownership labels, broader long-horizon evaluation, stronger handling of articulated motion, and adaptive refinement policies that spend iterations only where the initialization is uncertain.

Appendix A

Baseline Implementation Details

A.1 HiCoM Compact Initialization

HiCoM uses perturbation smoothing when training the initial 3DGS model from the first frame. During optimization, it applies a small random perturbation to each Gaussian center:

$$\mu_p = \mu + \lambda_{\text{noise}}\epsilon, \quad \epsilon \sim \mathcal{N}(0, 1), \quad (\text{A.1})$$

where μ is the learned Gaussian center, μ_p is the perturbed center used during training, and λ_{noise} controls the perturbation scale.

The purpose is to regularize sparse-view initial-frame training. With limited input views, standard 3DGS can overfit by growing many small Gaussians that fit the training cameras but form a less stable 3D representation. Perturbing the centers during training encourages Gaussians to remain useful under small position changes, which makes the initial representation smoother and more compact.

This compact initial model matters for streaming because later frames are updated from the existing Gaussian set. Fewer and more stable Gaussians reduce the cost of motion learning, rendering, storage, and subsequent refinement.

Bibliography

- [1] Benjamin Attal, Jia-Bin Huang, Christian Richardt, Michael Zollhoefer, Johannes Kopf, Matthew O’Toole, and Changil Kim. Hyperreel: High-fidelity 6-dof video with ray-conditioned sampling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.
- [2] Sara Fridovich-Keil, Giacomo Meanti, Frederik Warburg, Benjamin Recht, and Angjoo Kanazawa. K-planes: Explicit radiance fields in space, time, and appearance. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.
- [3] Jiaye Fu, Qiankun Gao, Chengxiang Wen, Yanmin Wu, Siwei Ma, Jiaqi Zhang, and Jian Zhang. Recon-gs: Continuum-preserved gaussian streaming for fast and compact reconstruction of dynamic scenes. In *Advances in Neural Information Processing Systems*, 2025.
- [4] Qiankun Gao, Jiarui Meng, Chengxiang Wen, Jie Chen, and Jian Zhang. Hicom: Hierarchical coherent motion for streamable dynamic scene with 3d gaussian splatting. In *Advances in Neural Information Processing Systems*, 2024.
- [5] Sharath Girish, Tianye Li, Amrita Mazumdar, Abhinav Shrivastava, David Luebke, and Shalini De Mello. QUEEN: QUantized efficient ENcoding of dynamic gaussians for streaming free-viewpoint videos. In *Advances in Neural Information Processing Systems*, 2024.
- [6] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), 2023.
- [7] Hao Li, Sicheng Li, Xiang Gao, Abudouaihati Batuer, Lu Yu, and Yiyi Liao. GIFStream: 4d gaussian-based immersive video with feature stream, 2025.
- [8] Lingzhi Li, Zhen Shen, Zhongshu Wang, Li Shen, and Ping Tan. Streaming radiance fields for 3d video synthesis. In *Advances in Neural Information Processing Systems*, 2022.
- [9] Tianye Li, Mira Slavcheva, Michael Zollhoefer, Simon Green, Christoph Lassner, Changil Kim, Tanner Schmidt, Steven Lovegrove, Michael Goesele, and Zhaoyang Lv. Neural 3d video synthesis from multi-view video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.
- [10] Xinhui Liu, Can Wang, Lei Liu, Zhenghao Chen, Wei Jiang, Wei Wang, and Dong Xu. LoD-structured 3d gaussian splatting for streaming video reconstruction, 2026.
- [11] Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. In *International Conference on 3D Vision*, 2024.

- [12] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. In *European Conference on Computer Vision*, 2020.
- [13] Keunhong Park, Utkarsh Sinha, Jonathan T. Barron, Sofien Bouaziz, Dan B. Goldman, Steven M. Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021.
- [14] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-NeRF: Neural radiance fields for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021.
- [15] Jiakai Sun, Han Jiao, Guangyuan Li, Zhanjie Zhang, Lei Zhao, and Wei Xing. 3dstream: On-the-fly training of 3d gaussians for efficient streaming of photo-realistic free-viewpoint videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.
- [16] Liao Wang, Kaixin Yao, Chengcheng Guo, Zhirui Zhang, Qiang Hu, Jingyi Yu, Lan Xu, and Minye Wu. VideoRF: Rendering dynamic radiance fields as 2d feature video streams, 2023.
- [17] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.
- [18] Zike Wu, Qi Yan, Xuanyu Yi, Lele Wang, and Renjie Liao. Streamsplat: Towards online dynamic 3d reconstruction from uncalibrated video streams. In *International Conference on Learning Representations*, 2026.
- [19] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025.
- [20] Daheng Yin, Isaac Ding, Yili Jin, Jianxin Shi, and Jiangchuan Liu. Trackersplat: Exploiting point tracking for fast and robust dynamic 3d gaussians reconstruction. In *SIGGRAPH Asia Conference Papers*, 2025.
- [21] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.