

ClipGStream: Clip-Stream Gaussian Splatting for Any Length and Any Motion Multi-View Dynamic Scene Reconstruction

Supplementary Material

A. Overview

Rendering videos on the [website](#) and Sec. B demonstrate the advantages of our method over FrameStream and Clip-based approaches: our model achieves significantly higher rendering quality on scenes with large motion and long temporal sequences. Furthermore, Sec. C details the design motivation, implementation specifics, and ablation studies for each component of our ClipGStream. Sec. D describes the hyperparameter settings of the model. The training pipeline is detailed in the *Training Pipeline* section at [website](#).

B. Video Demos

We provide rendering videos on the project [webpage](#), featuring qualitative comparisons between our ClipGStream and existing approaches (e.g., LocalDyGS, 3DGStream, SpaceTimeGS, and 4DGaussian).

Long 360. Our method achieves high-quality rendering even for scenes with large-motion objects (e.g., athletes), thanks to our Intra-Clip training strategy. In contrast, when 4DGaussian and LocalDyGS model the entire 1400-frame sequence using a single clip, the basketball player suffers from severe disappearance artifacts. On the other hand, modeling the sequence with multiple independent clips (LocalDyGS 140 Clips) leads to noticeable flickering on static regions, such as the floor, due to temporal inconsistency across clips. Thanks to our Inter-clip Inheritance strategy, ClipGStream maintains temporal coherence between clips and eliminates such flickering artifacts.

VRU GZ. Our method accurately models fast-moving objects—such as athletes and basketballs. In contrast, the streaming method 3DGStream suffers from error accumulation over time, resulting in progressively degraded rendering quality. Clip-based approaches, including SpaceTimeGS and 4DGaussian, also fail to effectively capture these high-velocity motions.

N3DV. Our method also excels at modeling scenes with fine-grained motion, achieving higher fidelity in capturing detailed deformations—such as facial expressions of dogs and hand movements of humans.

C. Implementation&Ablation

C.1. Intra-clip Training Strategy.

Residual Anchors Compensation Module. In Sec.3.2.1 (Fig. 4), we propose a geometry-based residual deduplication module to construct the input anchor set A_n for the Source

Table 1. We compare the voxel-based deduplication approach with our geometry-aware strategy. When the voxel size is too small (e.g., $v = 0.005$), excessive redundancy leads to out-of-memory errors. Conversely, increasing the voxel size removes too many points, degrading reconstruction performance. In contrast, our method achieves effective deduplication while maintaining higher rendering quality and stability.

	Point Nums ↓	PSNR ↑	DSSIM ₁ ↓	DSSIM ₂ ↓
$v = 0.005$	470848	-	-	-
$v = 0.01$	77372	23.88	0.071	0.034
$v = 0.05$	1165	23.55	0.074	0.037
ours	17252	24.08	0.069	0.032

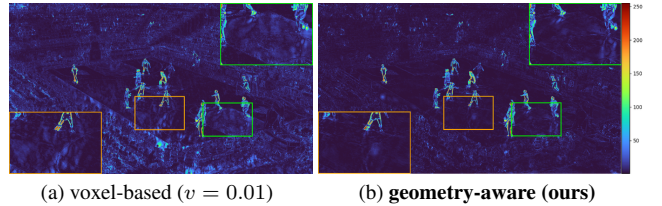


Figure 1. **Comparison of residual heatmaps.** (a) Voxel-based deduplication ($v = 0.01$) ignores the spatial extent already covered by A_0 , leading to ineffective redundancy removal and noticeable flickering artifacts. (b) Our geometry-aware deduplication effectively eliminates redundant anchors, thereby suppressing temporal flickering.

Clip $Clip_n$. Specifically, we first build a sphere coverage field from the anchors A_0 of the Reference Clip $Clip_0$ to represent regions that have already been well modeled. We then apply SDF to prune redundant anchors in the Source Clip that fall within this covered region, yielding a compact anchor set A_n .

This module effectively removes redundant anchors. On one hand, this reduces storage overhead; on the other hand, it prevents the introduction of new anchors in static regions. Such new anchors in static areas would otherwise be optimized during Source Clip training, leading to temporal flickering. We also compare against a simpler deduplication baseline based on voxel grids. Specifically, we construct a voxel grid from A_0 using a voxel size v . Any candidate anchor from the Source Clip that falls within an occupied voxel is treated as redundant and removed. However, we find that this voxel-based strategy fails to effectively eliminate redundancy.

This limitation stems from the fact that voxel-based deduplication assigns a uniform occupancy volume to all anchors. When the voxel size v is too small, anchors from A_0 occupy only tiny regions, failing to cover nearby redundant candidates—resulting in excessive residual anchors, increased

Table 2. We evaluate two feature fusion strategies: addition (add) and concatenation (concat). Both yield nearly identical performance in terms of PSNR, DSSIM₁, and DSSIM₂. However, since concatenation allows us to reduce the feature dimensionality from 128 to 64 per anchor, we adopt concat in our final implementation for improved memory efficiency.

	dims ↓	PSNR ↑	DSSIM ₁ ↓	DSSIM ₂ ↓
add	128	25.03	0.067	0.031
concat	64	24.92	0.069	0.032



(a) voxel-based ($v = 0.05$) (b) **geometry-aware (Ours)**

Figure 2. **Qualitative comparison between voxel-based and our geometry-aware deduplication.** (a) Using a large voxel size ($v = 0.05$) for deduplication results in too few residual anchors, which prevents the model from effectively capturing high-speed motion. (b) In contrast, our method adaptively preserves residual anchors in dynamic regions, leading to significantly improved rendering quality.

storage overhead (Tab. 1, $v = 0.005$), temporal flickering as shown in Fig. 1 (a). Conversely, when v is too large, the coarse voxel grid over-aggressively prunes anchors in dynamic regions, impairing the representation of fine motion and degrading reconstruction quality (Tab. 1, $v = 0.05$ and Fig. 2).

Clip-Specific Spatio-Temporal Fields. Here, we provide a detailed description of the design of our spatio-temporal field (STF). As mentioned in the main text, STF comprises a hash grid h followed by MLPs ϕ . In practice, h is implemented as a multi-resolution structure consisting of L individual hash grids $\{s_l\}_{l=1}^L$.

Given an anchor and query time $(\mu, t) \in R^m$, the hash encoding at resolution level l is denoted as $s_l(\mu, t) \in R^m$. This encoding is obtained via trilinear interpolation of the feature vectors stored at the vertices of the 4D grid cell containing (μ, t) . The dynamic feature f_d is then aggregated across all L resolution levels as:

$$f_d = \phi(h(\mu, t)) = \phi([s_1(\mu, t), \dots, s_L(\mu, t)]), \quad (1)$$

In our default configuration, both the dynamic and static features are 64-dimensional, and they are concatenated to form the final feature representation. We also evaluate an

Table 3. **Ablation on hash table size of STF in the Long 360.** Due to the shorter length (M) of our Clips compared to the full sequence, even a reduced size suffices to model dynamic content effectively. Modeling 1,400 frames requires only 2.xG of storage.

size	storage	PSNR ↑	DSSIM ₁ ↓	DSSIM ₂ ↓	LPIPS ↓
2^{20}	~35GB	24.64	0.077	0.036	0.142
2^{18}	~9GB	24.50	0.078	0.036	0.144
2^{17}	~4.5GB	24.60	0.078	0.036	0.145
2^{16}	~2.24GB	24.54	0.079	0.036	0.146

alternative fusion strategy in which both dynamic and static features are 128-dimensional and combined via element-wise addition. We find that this variant achieves comparable reconstruction performance, yet our design significantly reduces storage consumption thanks to its lower feature dimensionality (Tab. 2)

C.2. Inter-clip Inheritance Strategy

Decoder. Our decoder takes concatenated dynamic features f_d and static features f_s as input and outputs the attributes of Temporal Gaussians G_t . Specifically, for each Gaussian attribute, we employ a dedicated two-layer MLP followed by an activation function as the decoding head.

Anchors Inheritance Module. During Source Clip training, Clip-Stream inherits the anchor set from the Reference Clip. These anchors comprise two distinct components:

- **Static background anchors** (e.g., floor regions) that are shared between the Reference and Source Clips. While their geometry remains unchanged, their appearance may vary over time. To model such appearance variations, ClipGStream learns a new spatio-temporal feature (STF) in the Source Clip. The resulting dynamic feature is concatenated with the static feature and fed into appearance-aware decoders (e.g., the color decoder) to reconstruct the updated appearance.
- **Dynamic anchors** corresponding to moving objects (e.g., athletes in the Reference Clip). Their geometry has significantly changed in the Source Clip. The model controls the visibility of these anchors through the opacity decoder, thereby enabling explicit modeling of geometric deformation or displacement.

D. Hyperparameters

The size of STF . We initially create the original STF for each Clip, which yields solid accuracy but incurs heavy storage. A single field occupies about 400 MB, and a 1400-frame sequence exceeds 30 GB. Profiling the field inputs shows many dynamic entries are zero, indicating substantial redundancy. We therefore reduce the hash grid capacity and allocate a compact spatio-temporal field to each Clip (in Tab. 3). In practice, the hash table size of each STF is set to 2^{16} to trade off performance against storage, with other settings consistent with INGP.

The size of Clip (M). A critical limitation of prior methods is their inherent requirement for $M = N$, meaning the entire

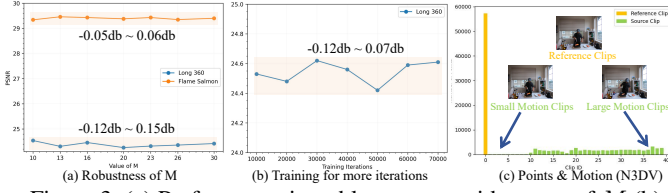


Figure 3. (a) Performance is stable across a wide range of M . (b) The Source Clip converges quickly. (c) Few points are added for small motions while sufficient points are added for large ones.

video sequence must be processed at once. Directly extending it to multiple clips by partitioning frames (i.e. $M < N$) results in noticeable temporal inconsistencies (LocalDyGS 140 Clips video). Our method addresses this by enforcing consistency across clips, enabling flicker-free handling when $M < N$. Furthermore, we find that using the same M on VRU Long and N3DV does not significantly affect the experimental results (Fig. 3 (a)). Future work could explore an adaptive M to further boost generalization.

Training iterations of Source Clip. We train the f_s of the few residual points, along with the STF , to efficiently model dynamic information in Source Clip. Since most regions have already been learned in the Reference Clip, the model can converge rapidly when training on the Source Clip. (Fig. 3 (b), (c)).