Echoes of the Coliseum: Towards 3D Live streaming of Sports Events

JUNKAI HUANG*, CMU, USA
SASWAT SUBHAJYOTI MALLICK*, CMU, USA
ALEJANDRO AMAT, CMU, USA
MARC RUIZ OLLE†, CMU, USA
ALBERT MOSELLA-MONTORO, CMU, USA
BERNHARD KERBL, CMU, USA
FRANCISCO VICENTE CARRASCO, CMU, USA
FERNANDO DE LA TORRE, CMU, USA

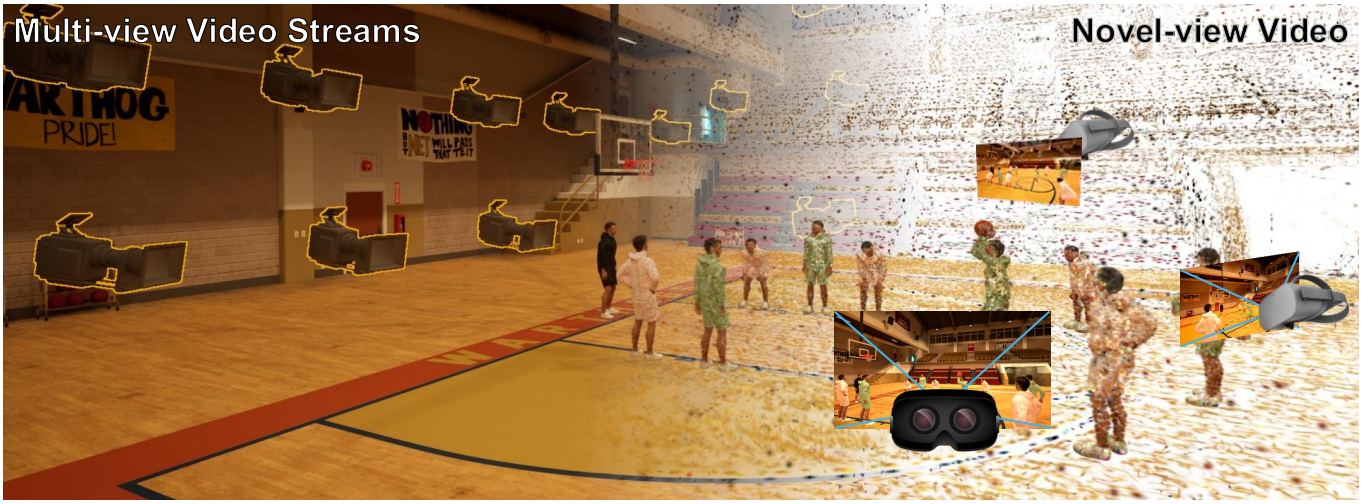

Fig. 1. **LiveSplats** takes multi-view video streams of sports events as input and reconstructs the dynamic 3D scene to allow viewers to watch from any novel view with interactive frame rates.

Human-centered live events have always played a pivotal role in shaping culture and fostering social connections. Traditional 2D live transmissions fail to replicate the immersive quality of physical attendance. Addressing this gap, this paper proposes **LiveSplats**, a framework towards real-time, photo-realistic 3D reconstructions of live events using high-performance 3D Gaussian Splatting.

Our solution capitalizes on strong geometric priors to optimize through distributed processing and load balancing, enabling interactive, freely explorable 3D experiences. By dividing scene reconstruction into actor-centric and environment-specific tasks, we employ hierarchical coarse-to-fine optimization to rapidly and accurately reconstruct human actors based on pose data, refining their geometry and appearance with photometric loss. For static environments, we focus on view-dependent appearance changes, streamlining rendering efficiency and maximizing GPU performance. To facilitate evaluation, we introduce (and distribute) a synthetic benchmark dataset of basketball games, offering high visual fidelity as ground truth. In both our synthetic benchmark and publicly available benchmarks, LiveSplats consistently outperforms existing approaches. The dataset is available at https://humansensinglab.github.io/basket-multiview.

CCS Concepts: • **Computing methodologies** → **Rendering**; **Tracking**; **Reconstruction**; **Rasterization**.

Additional Key Words and Phrases: Sports Events, Human-Centric Events, Gaussian Splatting, Real-time 3D Reconstruction

ACM Reference Format:
Junkai Huang, Saswat Subhajyoti Mallick, Alejandro Amat, Marc Ruiz Olle, Albert Mosella-Montoro, Bernhard Kerbl, Francisco Vicente Carrasco, and Fernando de la Torre. 2025. Echoes of the Coliseum: Towards 3D Live streaming of Sports Events . *ACM Trans. Graph.* 44, 4, Article 46 (August 2025), 17 pages. https://doi.org/10.1145/3731214

*Both authors contributed equally to this research.
†Contributed to dataset development

Authors' Contact Information: Junkai Huang, junkaih@andrew.cmu.edu, CMU, USA; Saswat Subhajyoti Mallick, smallick@andrew.cmu.edu, CMU, USA; Alejandro Amat, aamat@andrew.cmu.edu, CMU, USA; Marc Ruiz Olle, mruizoll@andrew.cmu.edu, CMU, USA; Albert Mosella-Montoro, CMU, USA; Bernhard Kerbl, bkerbl@andrew.cmu.edu, CMU, USA; Francisco Vicente Carrasco, fvicente@andrew.cmu.edu, CMU, USA; Fernando de la Torre, ftorre@andrew.cmu.edu, CMU, USA.

ACM 1557-7368/2025/8-ART46
https://doi.org/10.1145/3731214

1 Introduction

"Panem et circenses" coined by Roman poet Juvenal to critize the self-serving nature of Romans who, deprived of political influence, are placated soley by "bread and circuses" provided by the government. Although technology has advanced significantly, we remain the same humans, governed by similar systems. Entertainment continues to be at the heart of societal engagement, illustrating that some aspects of human nature endure across millennia.

To allow for widespread participation, live transmissions of such sports events are ubiquitously available for remote, visual consumption in a 2D format. Unfortunately, this does not allow viewers to immerse themselves in the same way that personal attendance would: at best, television services may allow viewers to cycle through cameras at will to enable a basic level of engagement.

In 2001, Takeo Kanade and collaborators [University 2001] developed the "EyeVision" technology for use during Super Bowl 2001. It involved an array of cameras mounted at the Raymond James Stadium, which provided a dynamic, panoramic view of the play. This system synthesized the inputs from multiple cameras into a single, flowing image, allowing viewers to see the action from virtually any angle. This innovation significantly enhanced the viewer's experience by offering detailed, 360-degree views of live action, which traditional single-point-of-view cameras could not capture. However, EyeVision did not allow real-time interaction, needed a lot of computing and was extremely costly to set up.

Building on this, Intel's TrueView (formerly freeD) technology uses an array of high-resolution cameras around stadiums to create immersive 360-degree replays, allowing viewers to experience key moments from various angles. Spiideo's Multi-Angle Autocasting utilizes AI-powered cameras to capture multiple perspectives of sports events automatically, enabling seamless switching between angles without manual intervention.

Anticipating the next frontier of visual content consumption, this paper lays the foundation to provide significant leaps forward for the remote experience of live sports events: Building on recent advances for high-performance radiance field representations, we describe a wholistic solution toward providing a freely explorable, photo-realistic 3D format for such events. Our approach, **LiveSplats**, is based on fast-to-train radiance fields from 3D Gaussian Splatting [Kerbl et al. 2023] (3DGS) . By focusing on human-centered content, we can exploit strong geometric priors to cut reconstruction time, facilitate distributed and parallel processing, as well as load balancing to produce high-quality 3D reconstructions for live content at interactive rates. Specifically, we separate the job of reconstructing the full scene at a given timestamp into *per-subject* and *environment* optimization tasks.

For each subject, we can seed their high-quality reconstruction at each timestamp from easy-to-obtain pose information, and further refine geometry and appearance from photometric loss via differentiable 3DGS rendering. This work proposes a hierarchical *coarse-to-fine* approach that progressively resolves increasingly fine aspects of subjects' pose and appearance, which forms a lightning-fast solution that's robust to noise, for reconstructing 3D humanoid actors from multi-view images.

In addition to subjects in the event, a complete visual experience also requires reacting to changes in the environment. To limit the problem space and allow highly effective run-time optimization, this research focuses on geometrically static environments. In other words, we assume that given an initial reconstruction of an environment, only its appearance (i.e., view-dependent color) may change as players perform actions this includes any changes in global illumination, shadows, or reflections as players move about the scene. Enforcing this restriction allows for a highly streamlined render pipeline design that alleviates several of the complex—and time-consuming—aspects of 3DGS rendering. We show how this simple assumption enables us to eschew all dynamic resource management challenges and use high-level (e.g., ideal load balancing) and low-level (e.g., CUDA graphs) optimizations to maximize the efficacy of available hardware for training.

A significant challenge in this research is the lack of available data and standardization for live event capture with ground-truth data. To enable an in-depth exploration of our target setting despite this issue, we have designed a comprehensive, synthetic benchmark dataset, focusing on sports activities with multiple actors. Designed with professional 3D authoring and rendering tools, our dataset provides high visual fidelity and multimodal reference outputs for optimization, as well as options for simulating real-live error sources (e.g., in pose detection).

In summary, we provide the following contributions:

(1) A scalable system design toward real-time reconstruction of sports events via distributed and parallel processing.
(2) A high-performance, coarse-to-fine solution for reconstructing subjects in the event.
(3) A streamlined pipeline for appearance optimization with optimal load balancing.
(4) A comprehensive benchmark dataset with multiple actors in human-centric settings.

2 Related works

Radiance Fields: Radiance fields are pivotal in computer graphics and vision, representing 3D scenes by modeling light distribution within a volume. Kajiya's rendering equation [Kajiya 1986] established the foundation for simulating radiance transfer in scenes. Precomputed Radiance Transfer (PRT) [Sloan et al. 2002] advanced real-time rendering by precomputing light interactions for dynamic lighting. Neural Radiance Fields (NeRF) [Mildenhall et al. 2020] revolutionized the field, using neural networks to synthesize high-fidelity novel views of complex scenes. Extensions like D-NeRF [Pumarola et al. 2021] capture non-rigid deformations over time, while Time-of-Flight Radiance Fields (TöRF) [Imaging Group 2021] incorporate depth sensing for improved dynamic scene reconstruction.

Despite their dominance in scene reconstruction and novel view synthesis, NeRF and its variants are computationally intensive, requiring significant optimization for each new scene. Recently, 3D Gaussian Splatting (3DGS) [Kerbl et al. 2023] has emerged as a more efficient alternative, modeling radiance fields with spatially distributed Gaussian primitives.

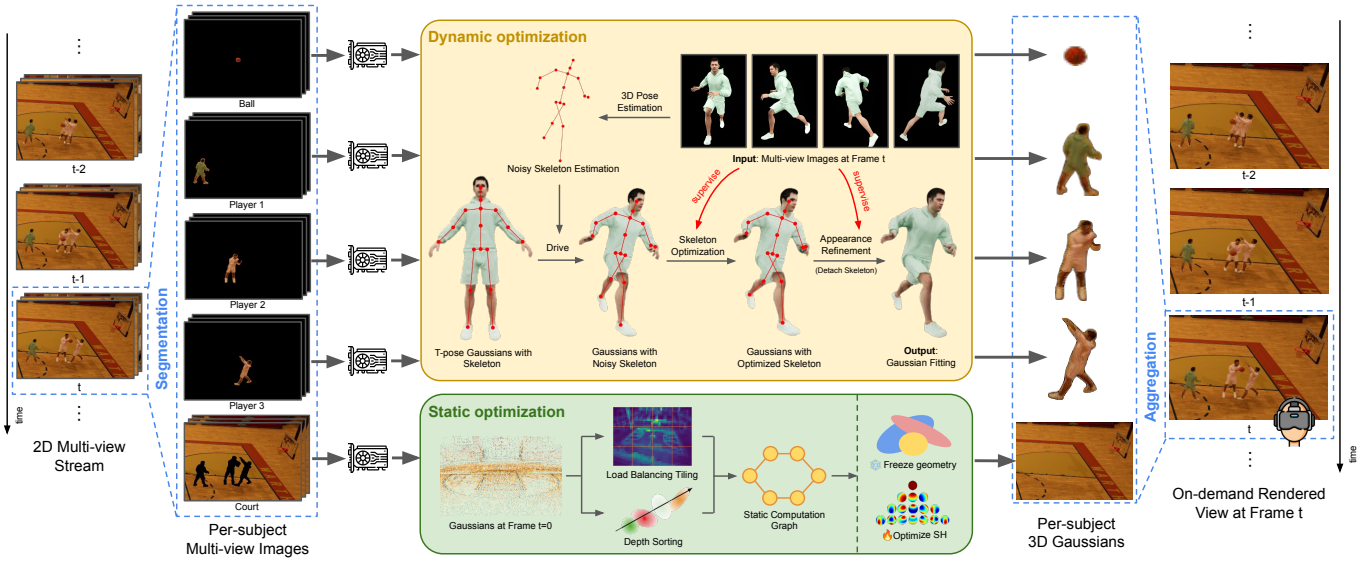

Fig. 2. System architecture of LiveSplats. We first segment multi-view images at time t, to obtain per-subject segmentation masks for each subject, and pass the masked RGB images of each subject to the nodes. Each node will then optimize 3D Gaussians at t for that subject, using either dynamic or static optimization logic. The optimized 3D Gaussians are collected from each node and aggregated into a single model, and then served for on-demand renderings from arbitrary viewpoints to clients.

3DGS Methods: Spacetime Gaussians [Li et al. 2024a] and Deformable 3D Gaussians [Yang et al. 2024] enhance 3DGS by incorporating temporal opacity, parametric motion, and annealing mechanisms to model dynamic scenes while addressing pose estimation inaccuracies. VideoRF [Wang et al. 2024a] and 4D Gaussian Splatting [Wu et al. 2024] utilize space-time mappings and multi-resolution Hex-Plane modules, respectively, to compress temporal redundancies and capture motion and shape changes with deformation fields. SurMo [Hu et al. 2024a] and Gaussian-Flow [Lin et al. 2024] model temporal dynamics using surface-based triplanes and dual-domain deformation models, to capture complex motions and deformations through frequency and polynomial fitting.
SWinGS [Shaw et al. 2024], Katsumata et al. [Katsumata et al. 2024], and DynMF [Kratimenos et al. 2024] represent scene dynamics with sliding windows, Fourier approximations, and basis trajectories, enabling efficient and controllable motion synthesis. Street Gaussians [Yan et al. 2024] and DualGS [Jiang et al. 2024] focus on human-centric dynamic scenes, leveraging tracked poses, and separate encodings for skeletal motion and surface appearance, with real-time rendering supported by entropy and codec-based compression.

3DGS for avatars: Several methods on driving human avatars such as GaussianAvatar [Hu et al. 2024b], GoMAvatar [Wen et al. 2024] and SplattingAvatar [Shao et al. 2024] achieve high-quality renderings from monocular videos by combining explicit mesh representations with Gaussian primitives. Concurrently, approaches such as HumanGaussian [Liu et al. 2024], SimAvatar [Li et al. 2024b], and PSHuman [Li et al. 2024c] focus on text-driven 3D human generation and photorealistic reconstruction, leveraging diffusion models and cross-scale techniques. V^3 [Wang et al. 2024b] and SqueezeMe [Saito et al. 2024] address the challenges of streaming volumetric videos on mobile devices and optimizing Gaussian avatars for VR applications, respectively.

NeRF streaming methods: StreamRF [Li et al. 2022] uses an explicit grid-based model with incremental learning to reconstruct streaming radiance fields, updating each frame as a difference from a base model, and optimizes only critical regions in each frame. ReRF [Wang et al. 2023] represents dynamic scenes by modeling inter-frame changes with a compact motion grid and residual feature grid, decoded by a lightweight MLP. NeRFPlayer [Song et al. 2023] enables efficient streamable representation of dynamic scenes by decomposing them into static, deform-able, and unseen areas, each modeled by separate neural fields, while employing a time-dependent sliding window for feature streaming.

3DGS streaming methods: Dynamic-3DGS [Luiten et al. 2024] (D-3DGS) models dynamic scenes with Gaussians with persistent geometry attributes that can orient freely, and enforcing local rigidity for spatial consistency. 3DGStream [Sun et al. 2024] uses a neural transformation cache (NTC) for Gaussian transformation and a refinement stage for detail reconstruction. HiCoM [Gao et al. 2024] adopts hierarchical voxel-based motion modeling, perturbation smoothing, and Gaussian merging for compact, streamable representations. DASS employs Gaussian inheritance, motion-aware alignment, and densification via optimization errors for iterative refinement. QUEEN [Girish et al. 2024] encodes Gaussian attributes with quantized residuals, sparsifies position data, and disentangles static and dynamic content using viewspace gradients.

3 Method

Our aim is to achieve real-time 3D reconstruction of sports-centered events from multi-view streaming videos, to enable unrestricted

6-DOF exploration, providing a uniquely interactive experience for (remote) audiences. However, delivering such up-to-date visuals poses a massive engineering challenge, requiring reconstructions to happen thousands of times faster than traditional methods.

To tackle this challenge, we model the reconstruction process using a two-level MapReduce [Dean and Ghemawat 2008] approach and parallelize it across GPU-powered nodes . Nodes in a node-pool process individual frames independently, while GPUs within a node handle the reconstruction of dynamic elements-such as players, the court, or the ball-in parallel. This design allows interactive frame rates, which improve linearly with increase in computational resources.

3.1 System overview

Figure 2 provides an overview of LiveSplats. At time t, we map the last N unprocessed frames (from t to $t-N$) to each of the N nodes. For each frame, we generate multi-view masks for all subjects using off-the-shelf models (SAM2 [Ravi et al. 2024]) and assign each subject to a dedicated GPU for reconstruction.

Each node optimizes 3D Gaussians at frame t, with a subject assigned to each GPU, employing either dynamic or static optimization, which we detail in Sections 3.2 and 3.3, respectively. After completing the 3D reconstruction for all subjects in the frame, the optimized 3D Gaussians from each node are aggregated into a unified model. We prioritize reconstruction in a manner that novel-view synthesis at high-fidelity is possible with just one optimization cycle, thereby amortizing on-demand rendering from arbitrary viewpoints for connected viewers.

A key design feature of LiveSplats that enables nearly unbounded scalability is that **scene reconstruction at frame t operates completely independently of other frames**. This added flexibility in distributed processing allows us to *multiplex* the reconstruction of multiple adjacent frames, fully leveraging the available distributed computing resources for real-time 3D reconstruction.

We elucidate the strength of this design through an example. The total time T_{total} for a frame to be generated by a node is comprised of image transfer (T_t), reconstruction (T_r), 3DGS transfer back (T_b), and merging (T_m). Since reconstruction is independent of previous frames and the remaining stages can be handled asynchronously, after an initial latency period of $T_t + T_b + T_m$, the per-frame time with N nodes is $\frac{T_r}{N}$, i.e., linear in the number of nodes. Theoretical performance gains from raising N saturate when $\frac{T_r}{N}$ matches the sampling rate of the input (e.g., 33 ms per frame), at which point real-time reconstruction is achieved. Recall that the time taken by segmentation and 3-D pose estimation are intentionally omitted as they run per frame, outside the critical path; This allows our timing measurements to focus solely on reconstruction and enables a fair comparison with prior work.

3.2 Dynamic optimization

Incremental frame optimization based on a robust prior is fundamental to several recent methods [Gao et al. 2024; Luiten et al. 2024; Shaw et al. 2024; Sun et al. 2024] for dynamic scene reconstruction using Gaussians. However, methods often lose structural detail over time because relying on rigid/no priors accumulates errors, which degrade fine details during motion. To overcome this limitation, we introduce a novel dynamic coarse-to-fine optimization for humans, capable of moving 3D Gaussians over long distances while maintaining fine-grained details.

Modeling human body motion relies on high-quality priors, such as 3D human skeletons, to guide the spatial transformations of the 3D Gaussians. We define two key concepts: *skeleton* and *skinning weights*. A *skeleton* is the kinematic tree structure of human body where each node represents a joint, characterized by properties such as global position $\mathbf{x} \in \mathbb{R}^3$, global rotation quaternion $\mathbf{q} \in \mathbb{H}$ in world coordinates, and a pointer p to its parent node. *Skinning weights* are values assigned to mesh vertices, specifying the influence of each joint on the vertex, allowing for realistic deformations as the joints rotate.

LiveSplats processes multi-view RGB images per frame to produce detailed, temporally consistent 3D Gaussian-based surface reconstructions. Initialization involves optimizing vanilla 3DGS [Kerbl et al. 2023] using multi-view images of a human in a T-pose (binding pose), to generate a detailed 3DGS reconstruction. Following [Bhatnagar et al. 2020a,b], we fit an SMPL [Loper et al. 2015] model to the converged Gaussian model of the T-pose, by minimizing the chamfer loss between Gaussian centers and SMPL mesh vertices, yielding the final mesh, skeleton, and linear blend skinning (LBS) weights. Each Gaussian is assigned to its nearest mesh vertex and inherits its skinning weights, enabling skeleton-driven manipulation. While related work ([Hu et al. 2024b; Li et al. 2024c,b; Liu et al. 2024; Shao et al. 2024; Wen et al. 2024]) learns this binding for improved visual quality and smoother transitions, we stick to principal, learning-free refinement approaches. We remain open to exploring such promising alternative Gaussian binding techniques in future work.

For dynamic frames, OpenPose [Cao et al. 2019] is used to extract multi-view 2D skeletons for frame t, which are then triangulated to compute the corresponding 3D skeleton. Gaussians are transformed from the T-pose to the pose at frame t using this derived 3D skeleton. If noise in the posed skeleton leads to misplacement of Gaussians, skeleton optimization is employed to refine their positions. Finally, the Gaussians are decoupled from the skeleton, allowing their parameters to be freely optimized to enhance visual quality.

Skeleton-driven Gaussians. Given the Gaussians $\mathcal{G}_0$ and skeleton $\mathcal{S}_0$ at T-Pose, and posed skeleton $\mathcal{S}_t$ at frame t, one can transform the Gaussian positions and quaternions from the T-pose to the pose t by:

$$\begin{aligned} \mathbf{x}_{i,t} &= \sum_{j=1}^{|\mathcal{S}_0|} w_{i,j} \left[\mathbf{q}_{j,t} \cdot \mathbf{q}_{j,0}^* \cdot (\mathbf{x}_{i,0} - \mathbf{x}_{j,0}) \cdot \mathbf{q}_{j,0} \cdot \mathbf{q}_{j,t}^* + \mathbf{x}_{j,t} \right] \\ \mathbf{q}_{i,t} &= \sum_{j=1}^{|\mathcal{S}_0|} w_{i,j}\ \mathbf{q}_{j,t} \cdot \mathbf{q}_{j,0}^* \cdot \mathbf{q}_{i,0} \end{aligned} \tag{1}$$

where $\mathbf{x}_{i,t}$, $\mathbf{q}_{i,t}$ are the position and quaternion of i^{th} Gaussian at frame t. $\mathbf{x}_{j,t}$, $\mathbf{q}_{j,t}$ are the position and quaternion of j^{th} joint at frame t. It is worth noting that subscript i is the index of Gaussians, and subscript j is that of joints. $w_{i,j}$ is the skinning weight of the i^{th} Gaussian w.r.t. the j^{th} joint. We obtain the joint quaternions by computing the rotation relative to a T-pose reference skeleton

Skeleton optimization. Skeleton-driven Gaussians assume accurate skeletons, but triangulated skeletons from OpenPose often suffer from noise due to occlusions and resolution limits, leading to suboptimal Gaussian initialization. To mitigate this, we introduce a skeleton optimization algorithm (Algorithm 1), that refines the 3D skeleton by rasterizing transformed Gaussians and minimizing 2D photometric loss against the ground truth image.

In Algorithm 1, drive$(\cdot)$ is the skeleton-driven Gaussian transformation explained by Equation 1; rasterize$(\cdot)$ is the Gaussian rasterizing function (implementation adopted from [Mallick et al. 2024]). During the skeleton optimization, all the Gaussians are bound to the skeleton, to ensure their movement aligns with joint motions. Only the joint rotations are updated at each step, while Gaussian opacity, scale, and spherical harmonics (SH) are held constant.

Algorithm 1 Skeleton Optimization

Input: A list of ground truth multi-view images $\{I_i{}^*\}$ of the player at frame t along with their corresponding camera poses $\{\text{cam}_i\}$, T-pose Gaussians $\mathcal{G}_0$, T-pose skeleton $\mathcal{S}_0$, frame t skeleton $\mathcal{S}_t$
Output: Skeleton-optimized 3D Gaussians $\mathcal{G}_t$ of the player
function OptimizeSkeleton($\{I_i{}^*\}$, $\{\text{cam}_i\}$, $\mathcal{G}_0$, $\mathcal{S}_0$, $\mathcal{S}_t$)
    for iter = 1 to maxIters **do**
        $\mathcal{G}_t \leftarrow \text{drive}(\mathcal{G}_0, \mathcal{S}_0, \mathcal{S}_t)$
        $i \leftarrow$ iter % numCams
        $I_i \leftarrow \text{rasterize}(\mathcal{G}_t, \text{cam}_i)$
        $\mathcal{L} \leftarrow \mathcal{L}_{\text{photo}}(I_i, I_i{}^*)$
        $\nabla\mathcal{S}_t \leftarrow \nabla_{\mathcal{S}_t}\mathcal{L}$
        $\mathcal{S}_t \leftarrow \mathcal{S}_t - \eta\nabla\mathcal{S}_t$
    end for
    $\mathcal{G}_t \leftarrow \text{drive}(\mathcal{G}_0, \mathcal{S}_0, \mathcal{S}_t)$
    return $\mathcal{G}_t$
end function

Appearance refinement. Once skeleton optimization converges, the Gaussians are detached from the skeleton, and their parameters are optimized using the photometric loss $\mathcal{L}_{\text{photo}}$, to enhance local structural details and adapt to changes in illumination.

To enhance speed and robustness, skeleton optimization optimizes global joint rotations in the world coordinate system instead of local rotations relative to parent joints, simplifying the computational graph. During appearance refinement, Gaussians, now independent of the skeleton, move freely. To prevent them from drifting into the background due to imperfect player masks, random backgrounds are applied to both the ground truth and rasterizer, penalizing Gaussians that move outside the player's body.

Figure 3 demonstrates the qualitative gains by skeleton optimization, and detailed ablation study of our skeleton optimization are provided in the supplemental.

3.3 Static optimization

In most sports, the only significant source of change in the scene arises from the dynamic movements of human participants (players)

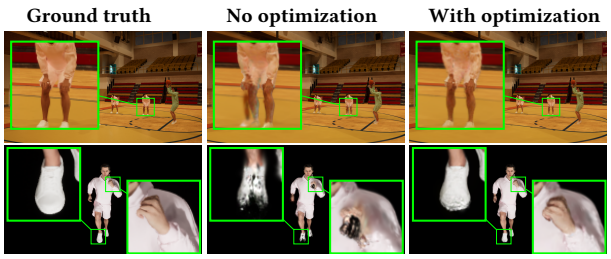

Fig. 3. Qualitative gains by skeleton optimization

and the ball. The rest of the environment remains static, with variations limited to global lighting and shadow dynamics influenced by these moving elements.

SH-only optimization. Following [Luiten et al. 2024], each Gaussian provides a soft representation of physical space and can be frozen once the geometry of the scene is accurately captured. We train vanilla 3DGS for 30,000 iterations on all the training views of an empty court/room. During this step, we apply scale regularization to prevent the formation of thin, elongated slivers. Additionally, to enhance visual fidelity around the basket—typically a region of high viewer interest, we strategically increase Gaussian density within its enclosing 3D bounding box. Subsequent optimization focuses exclusively on appearance (SH parameters); 200 iterations for static background and 500 iterations for the dynamic foreground, per frame, to account for lighting changes introduced by dynamic elements. This reduces computational overhead by excluding non-appearance parameters from gradient calculations, minimizing atomic operations during the backward pass, and significantly decreasing training time. Furthermore, decoupling geometry from appearance helps prevent artifacts such as flickering, which can occur when Gaussians shift, resize, or rotate.

Precomputation. Our empirical analysis reveals that depth-sorting of Gaussians is the primary bottleneck in rasterization. For Gaussians associated with static regions, their geometrical parameters (position, scale, rotation) are frozen, making their depth values invariant and cacheable. To address this, we implement a precomputed method that stores the sorted Gaussians for each rasterizer tile. These precomputed values are reused during the optimization of appearance changes in subsequent frames. This precomputation is performed after the optimization of the first frame has converged.

CUDA graphs. The precomputed structures reside in the VRAM which eliminate the expensive synchronization procedures between GPU and CPU. This gives way to implementing a purely CUDA graph-based [Ansel et al. 2024] solution devoid of any CPU routines or system calls. A CUDA graph is a software rendition that records GPU operations into a Directed Acyclic Graph (DAG) that can be used for asynchronous replays, minus the CPU overheads. Each replay reuses prerecorded execution paths and memory addresses, which removes any final unattended system-level constructs from limiting the speed of LiveSplats.

Load Balancing. Due to the non-deterministic nature of 3DGS, the number of Gaussians rendered per thread can vary significantly. To balance the workload across threads, we introduce a load-balancing

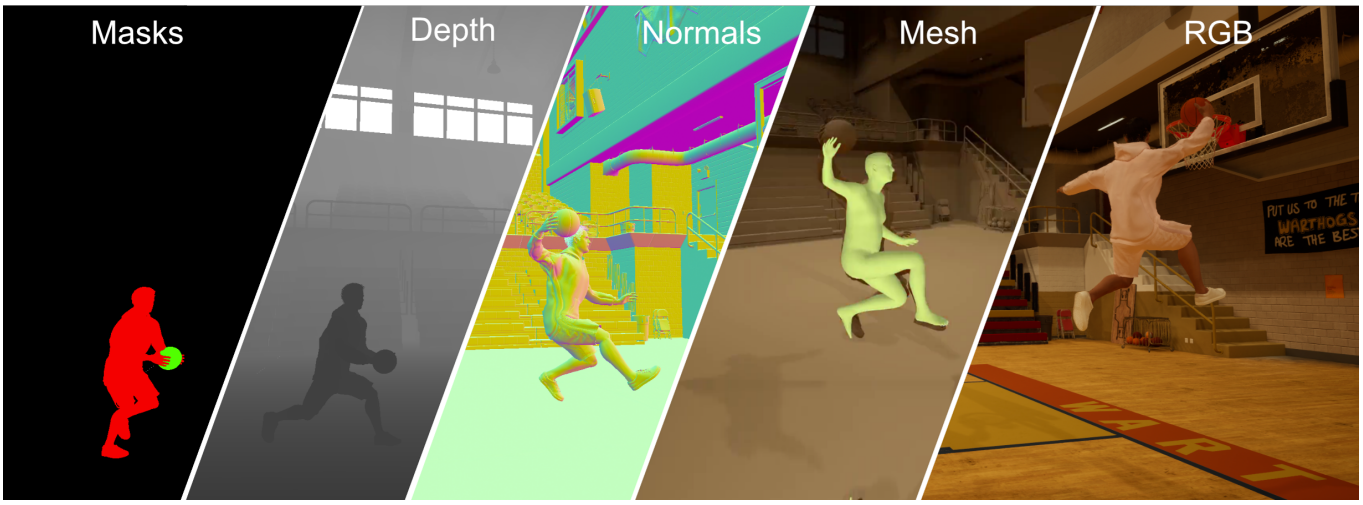

Fig. 4. Multi-modal data example from the BASKET-Multiview Dataset.

step after precomputation. For tile i, the ordered set of Gaussians $\mathcal{G}^i = \{g_1, g_2, \ldots, g_L\}$ is partitioned by order into M subtiles, each containing a maximum of N Gaussians ($M = \lceil L/N \rceil$). We have $\mathcal{G}^i = \bigcup_{j=1}^{M} \mathcal{B}_j^i$, where $\mathcal{B}_j^i$ is the set of Gaussians in subtile j with $|\mathcal{B}_j^i| \leq N$. Each $\mathcal{B}_j^i$ is processed by a separate thread block to maximize GPU occupancy.

Since only SH coefficients are optimized, transmittance values can be precomputed during the first frame. This allows independent blending within each subtile without requiring synchronization between thread blocks. Consequently, subtile-distributed alpha blending for pixel p can be expressed as:

$$\mathbf{I}(p) = \sum_{j=1}^{M} T_j^i(p) \cdot \sum_{g_k \in \mathcal{B}_j^i} \prod_{m=1}^{k-1} (1 - \alpha_m(p)) \cdot \alpha_k(p) \cdot \mathbf{C}_k(p) \quad (2)$$

where $T_j^i(p)$ is the transmittance of subtile j at pixel p; $\alpha_k(p)$, $\mathbf{C}_k(p)$ are the contributions of opacity and color of Gaussian k at pixel p. A representative flow for the static optimization and an ablation for each of the aforementioned optimizations is attached in the supplementary.

3.4 Loss

We adopt the photometric loss function in 3DGS [Kerbl et al. 2023]:

$$\mathcal{L}_{\text{photo}}(I, I^*) = (1 - \lambda)\mathcal{L}_1(I, I^*) + \lambda\mathcal{L}_{\text{D-SSIM}}(I, I^*) + R \quad (3)$$

for skeleton optimization, where I and I^* are the predicted and ground truth images, $\lambda = 0.2$. For the static optimization and appearance refinement in the dynamic optimization, only the subject-masked area is used to compute loss to avoid problems from occlusion among subjects: $\mathcal{L}_{\text{masked}} = \mathcal{L}_{\text{photo}}(MI, MI^*)$, where M is the subject mask. R is used to regularize Gaussian movements and scales, and is expressed as:

$$R = \lambda_x ||\mathbf{x} - \mathbf{x}^*|| + \lambda_s ||\mathbf{s} - \mathbf{s}^*||, \quad (4)$$

where $\mathbf{x}, \mathbf{x}^*, \mathbf{s}, \mathbf{s}^*$ are the optimized and initial Gaussian positions and scales. This regularization limits the geometrical changes to a Gaussian, thereby arresting any chances of flicker.

4 BASKET-Multiview Dataset

We introduce the BASKET (BAsketball Synthetic benchmarK for Enhanced Telepresence)-Multiview Dataset, a synthetic collection of scenarios representing common basketball plays generated using Unreal Engine 5 [Epic Games 2025] and EasySynth [YDRIVE 2023], leveraging the basketball court assets provided by [Studios 2025] and player models from [Pictures 2024]. For each scene, we provide comprehensive annotations that include calibrated cameras parameters, animations, RGB images, segmentation masks, depth maps, surface normal images and animations, as illustrated in Figure 4. All scenes are rendered at 1080p and 30 fps, with the exception of sequence *Attack 4*, which has been rendered in 4K resolution.

The dataset is divided into two partitions: *Core* and *Development*. The *Core* partition contains 7 scenes designed for evaluating reconstruction methods for sports events. Each *Core* scene is created within a basketball court environment, consisting of an 89-camera setup, optimized for capturing game-play during matches, as illustrated in Figure 5. More details on lighting and camera configurations are provided in the supplementary.

The *Development* partition contains 9 simpler sequences containing varying lighting conditions and backgrounds. *Development* sequences are designed to isolate features such as lighting dynamics, complex movements, close/far camera settings, and more, in order to test the capabilities of the method during development. The full list of scenes and their detailed specifications is provided in the supplementary.

Fig. 5. Lighting and camera configuration in the court. White light: point light. Red light: spotlight. Some lights are not visible in this view due to occlusion.

Table 1. Comparison on available multi-view sports datasets

Dataset Name	Nature	Cameras	Annotations			
			Semantic Mask	Depth Map	Normal Map	Animation
TeamTrack	Real	3	✗	✗	✗	✗
SoccerNet	Real	1	✗	✗	✗	✗
SportsMOT	Real	1	✗	✗	✗	✗
KTH Multiview Football II	Real	3	✗	✗	✗	✗
FineSports	Real	1	✗	✗	✗	✗
APIDIS	Real	7	✗	✗	✗	✗
EPFL Basketball	Real	4	✗	✗	✗	✗
SoccerNet-Depth	Synthetic	1	✗	✓	✗	✗
Soccer on your tabletop	Synthetic	1	✗	✓	✗	✗
BASKET-Multiview	Synthetic	89	✓	✓	✓	✓

5 Experiments

This section describes the experiments we carry out for benchmarking the performance of LiveSplats. We post the results of our ablation studies on skeleton optimization and static optimization speed-up techniques in the accompanying supplemental.

5.1 Datasets

We performed experiments on the BASKET-Multiview (described in Section 4) and the CMU-Panoptic dataset [Joo et al. 2017]. For the CMU-Panoptic dataset, we selected three subsequences from *171204_pose1* sequence and conduct experiments on these. The BASKET-Multiview dataset provided in this paper includes enhancements incorporating diversity in gender, skin tone, height and clothing styles. However, the experimental results reported in this section were obtained using an earlier version of the dataset.

To evaluate the robustness of LiveSplats against imperfect priors, we generate skeletons, meshes, and skinning weights using OpenPose [Cao et al. 2019] and SMPL [Loper et al. 2015], and further introduce random perturbations to the skeletons (referred to as *noisy skl*). We mimic different noise levels in the skeletons by randomly perturbing the limb joints (shoulders, elbows, hips, and knees) by 10° to 20°. These scenarios provide a controlled testbed to validate the ability of our skeleton optimization method to correct errors introduced by traditional off-the-shelf models.

5.2 Baselines

We compared LiveSplats against all known state-of-the-art approaches that perform online training and have publicly available implementations, to the best of our knowledge. We consider 3DGS-based 3DGStream [Sun et al. 2024](CVPR'24 Highlight), HiCoM [Gao et al. 2024](Neurips'24), and NeRF-based StreamRF [Li et al. 2022] (Neurips'22). Additionally, we included Dynamic-3DGS (D-3DGS) [Luiten et al. 2024](3DV'24) as a baseline, prioritizing quality despite its lack of real-time training capabilities. In our experiments, we adhered to the recommended settings for each method wherever possible, adjusting hyperparameters only when necessary to achieve optimal results.

All experiments are conducted on an Nvidia RTX A4500 GPU, with training and evaluation performed at a resolution of 960×540. We optimize each scene component for 500 iterations, with the initial 60 iterations dedicated to skeleton optimization, if applicable. Since the preprocessing requirements vary across methods, we follow the protocol in [Luiten et al. 2024; Sun et al. 2024] to ensure a fair comparison by reporting only the training time.

Our distributed design allows the workload to be scaled across multiple, even moderately powerful, GPUs, enabling a flexible trade-off between performance and resources, making it practical and portable across typical platforms.

5.3 Metrics

We evaluated the per-frame reconstruction quality using established image-based metrics: PSNR, SSIM, and LPIPS. Additionally, we introduced masked PSNR (M-PSNR), which calculates the PSNR exclusively in dynamic regions to more accurately assess image quality in motion-intensive areas. The image metrics were averaged across all frames and views of each sequence.

For video evaluation, we adopted VMAF [Li et al. 2016], a widely used metric that integrates spatial and temporal quality measures, to account for multi-resolution quality and temporal coherence. We reported the VMAF averaged across all views of each sequence. The averaged training time (in seconds) per frame (SPF) was also reported as a measure of computational cost. Since LiveSplats processed each scene component independently in parallel, we reported the largest SPF across all components. Furthermore, we introduced VMAF efficiency (VE) and PSNR efficiency (PE), defined as $\text{VE} = \frac{\text{VMAF}}{\text{SPF}}$ and $\text{PE} = \frac{\text{PSNR}}{\text{SPF}}$, respectively, to highlight the quality achieved per unit time for each method.

5.4 Evaluations

Table 2 shows the evaluation results on BASKET-Multiview and the CMU-Panoptic dataset. Figure 8 houses all qualitative results and Figure 6 illustrates results on consecutive frames. From Table 2, we can see that LiveSplats outperforms other methods in terms of speed and quality on BASKET-Multiview. On CMU-Panoptic, LiveSplats aces in both video and image quality as well as reconstruction speed. This is particularly noteworthy given the imperfect segmentation masks and noisy priors in the dataset—conditions under which most methods degrade significantly—highlighting robustness and minimal reliance on perfect priors.

Table 2. Full evaluation on BASKET-Multiview core scenes and CMU-Panoptic dataset. For CMU-Panoptic, the detected noisy 3D skeletons are used and the skeleton optimization is applied. For BASKET-Multiview, skeleton optimization is not always applicable because the cameras are too far in some scenes, so the perfect skeletons are used.

Dataset	Method	VMAF↑	PSNR↑	M-PSNR ↑	SSIM↑	LPIPS↓	SPF↓	VE↑	PE↑
BASKET-Multiview	StreamRF [Li et al. 2022]	36.30 ± 2.47	26.82 ± 0.42	13.45 ± 1.20	0.756 ± 0.040	0.198 ± 0.028	48.6 ± 0.5	0.75 ± 0.06	0.55 ± 0.01
	3DGStream [Sun et al. 2024]	51.43 ± 6.05	28.36 ± 0.90	20.79 ± 1.52	0.875 ± 0.019	0.174 ± 0.024	12.6 ± 0.7	4.10 ± 0.43	2.26 ± 0.16
	D-3DGS [Luiten et al. 2024]	65.41 ± 0.87	28.75 ± 0.85	23.20 ± 1.64	0.901 ± 0.003	0.149 ± 0.009	74.3 ± 3.5	0.88 ± 0.05	0.39 ± 0.01
	HiCoM [Gao et al. 2024]	54.69 ± 10.61	28.08 ± 2.71	19.03 ± 1.26	0.888 ± 0.060	0.157 ± 0.077	6.1 ± 0.1	8.89 ± 1.65	4.57 ± 0.40
	LiveSplats	67.30 ± 2.41	29.75 ± 0.21	25.24 ± 0.86	0.912 ± 0.007	0.119 ± 0.006	5.2 ± 0.1	13.02 ± 0.67	5.75 ± 0.12
CMU-Panoptic	StreamRF [Li et al. 2022]	34.49 ± 2.93	23.46 ± 0.34	18.20 ± 1.09	0.852 ± 0.057	0.404 ± 0.042	9.0 ± 0.0	3.83 ± 0.33	2.61 ± 0.04
	3DGStream [Sun et al. 2024]	46.00 ± 6.03	25.68 ± 1.28	21.37 ± 1.79	0.902 ± 0.011	0.338 ± 0.008	5.0 ± 0.0	9.20 ± 1.21	5.13 ± 0.26
	D-3DGS [Luiten et al. 2024]	50.66 ± 4.30	26.41 ± 0.24	24.60 ± 0.94	0.908 ± 0.008	0.286 ± 0.006	88.3 ± 3.5	0.57 ± 0.04	0.30 ± 0.01
	HiCoM [Gao et al. 2024]	38.37 ± 4.10	25.35 ± 0.93	21.85 ± 1.83	0.904 ± 0.009	0.301 ± 0.009	4.0 ± 0.0	9.60 ± 1.03	6.34 ± 0.23
	LiveSplats	57.33 ± 3.31	26.53 ± 1.27	26.55 ± 1.22	0.879 ± 0.009	0.274 ± 0.008	4.7 ± 0.0	12.20 ± 0.70	5.64 ± 0.27

Table 3. Evaluation with different camera distances. Attack 4 noisy skeletons are perturbed from perfect skeletons. Far views in Defense 2 result in no meaningful priors and hence are skipped from the evaluation.

Sequence	Dribbling Player - close view					Attack 4 - mid view					Defense 2 - far view				
Method | Metric	VMAF↑	PSNR↑	M-PSNR↑	SPF↓	VE↑	VMAF↑	PSNR↑	M-PSNR↑	SPF↓	VE↑	VMAF↑	PSNR↑	M-PSNR↑	SPF↓	VE↑
3DGStream [Sun et al. 2024]	36.08	30.54	19.73	3	12.03	57.01	26.75	23.63	14	4.07	56.57	29.19	19.81	12.3	4.60
D-3DGS [Luiten et al. 2024]	55.84	31.03	19.41	54	1.03	65.37	26.82	24.33	67	0.98	65.25	29.18	23.06	76.0	0.86
HiCoM [Gao et al. 2024]	33.24	28.64	18.24	3	11.08	31.47	22.03	18.65	6	5.25	55.08	28.59	17.81	6.1	9.03
LiveSplats (perfect skl)	76.16	35.85	25.74	3.8	20.04	72.68	30.17	25.60	5	14.54	66.31	29.75	26.15	5.2	12.75
LiveSplats (noisy skl)	62.13	35.72	25.60	5.8	10.71	72.67	30.17	25.60	5.7	12.75	-	-	-	-	-

In the BASKET-Multiview dataset, three camera views—close, mid, and far—simulate court-side and spectator seat placements by varying focal lengths and camera-to-player distances. From Table 3 and Figure 8 (a), (b), and (e), we observe that LiveSplats achieves the best quality and convergence speed in mid views, excelling even with noisy skeletons. In close views, it maintains high reconstruction quality with minor speed trade-offs. For far views, we significantly outperforms others in both quality and speed, though skeleton detection and optimization become unreliable at extreme distances. The variance in speed with varying camera distance is because each player has a fixed number of Gaussians and closer views can distribute the update process over more tiles, thereby speeding up computation.

In real-world applications, lighting changes can come from difference in illumination between the animated player and the T-pose reference, or due to the dynamic, unpredictable movements of players. We evaluate our method's ability to handle these lighting variations using the Running Player sequence with and without light changes. Results are reported in Table 4 and illustrated in Figure 8 (e) (f). The results show that LiveSplats outperforms all baselines in quality, even with noisy priors while maintaining comparable training speeds, which demonstrates the effectiveness of our skeleton optimization.

Table 4. Evaluation under sudden lighting changes (LC).

LC	Method	VMAF↑	PSNR↑	SSIM↑	LPIPS↓	SPF↓	VE↑
✗	3DGStream [Sun et al. 2024]	32.61	29.89	0.973	0.037	2	16.31
	D-3DGS [Luiten et al. 2024]	56.12	31.60	0.976	0.024	57	0.98
	HiCoM [Gao et al. 2024]	31.03	27.22	0.965	0.038	3	10.34
	LiveSplats (perfect skl)	82.52	39.43	0.990	0.008	3.8	21.72
	LiveSplats (noisy skl)	70.80	35.22	0.987	0.013	5.8	12.21
✓	3DGStream [Sun et al. 2024]	25.81	30.33	0.971	0.041	2	12.91
	D-3DGS [Luiten et al. 2024]	39.25	29.31	0.972	0.034	57	0.69
	HiCoM [Gao et al. 2024]	16.31	27.44	0.963	0.044	3	5.44
	LiveSplats (perfect skl)	77.59	38.95	0.990	0.012	3.8	20.42
	LiveSplats (noisy skl)	63.61	35.23	0.984	0.018	5.8	10.97

In Section 3.3, we propose several quality-preserving, performance optimizations for static regions under lighting changes, which we test using three such scenes. As shown in Table 5, we achieve significant reduction in SPF with comparable VMAF and PSNR against other methods. Since different methods require different numbers of iterations, in addition to the quality and performance metrics, we report the milliseconds taken per iteration (ms/iter) to better enunciate the speedup of our optimizations.

Table 5. Quality and speed evaluation of static scene optimization with lighting changes only.

Scene	Method	VMAF↑	PSNR↑	SPF↓	VE↑	PE↑	ms/iter
Day loop	3DGStream [Sun et al. 2024]	30.42	23.84	10	3.04	2.38	41
	D-3DGS [Luiten et al. 2024]	62.19	27.16	29	2.14	0.94	29
	HiCoM [Gao et al. 2024]	40.74	24.85	7	5.82	3.55	35
	LiveSplats	53.31	26.69	2	26.66	13.35	18
Opera	3DGStream [Sun et al. 2024]	32.71	21.14	11	2.97	1.92	47
	D-3DGS [Luiten et al. 2024]	50.97	24.47	29	1.76	0.84	29
	HiCoM [Gao et al. 2024]	42.28	22.89	7	6.04	3.27	35
	LiveSplats	51.79	25.16	2	25.90	12.58	18
Factory	3DGStream [Sun et al. 2024]	11.61	27.91	9	1.29	3.10	36
	D-3DGS [Luiten et al. 2024]	45.94	28.62	31	1.48	0.92	32
	HiCoM [Gao et al. 2024]	35.49	28.61	6	5.92	4.77	28
	LiveSplats	41.61	28.82	2	20.81	14.41	13

5.5 Limitations

LiveSplats achieves state-of-the-art performance in human-centered scene reconstruction, demonstrating robust results on both the synthetic BASKET-Multiview dataset which exemplifies popular stadium sports like soccer, handball, or American football and the real-world CMU-Panoptic dataset, which represents casual indoor

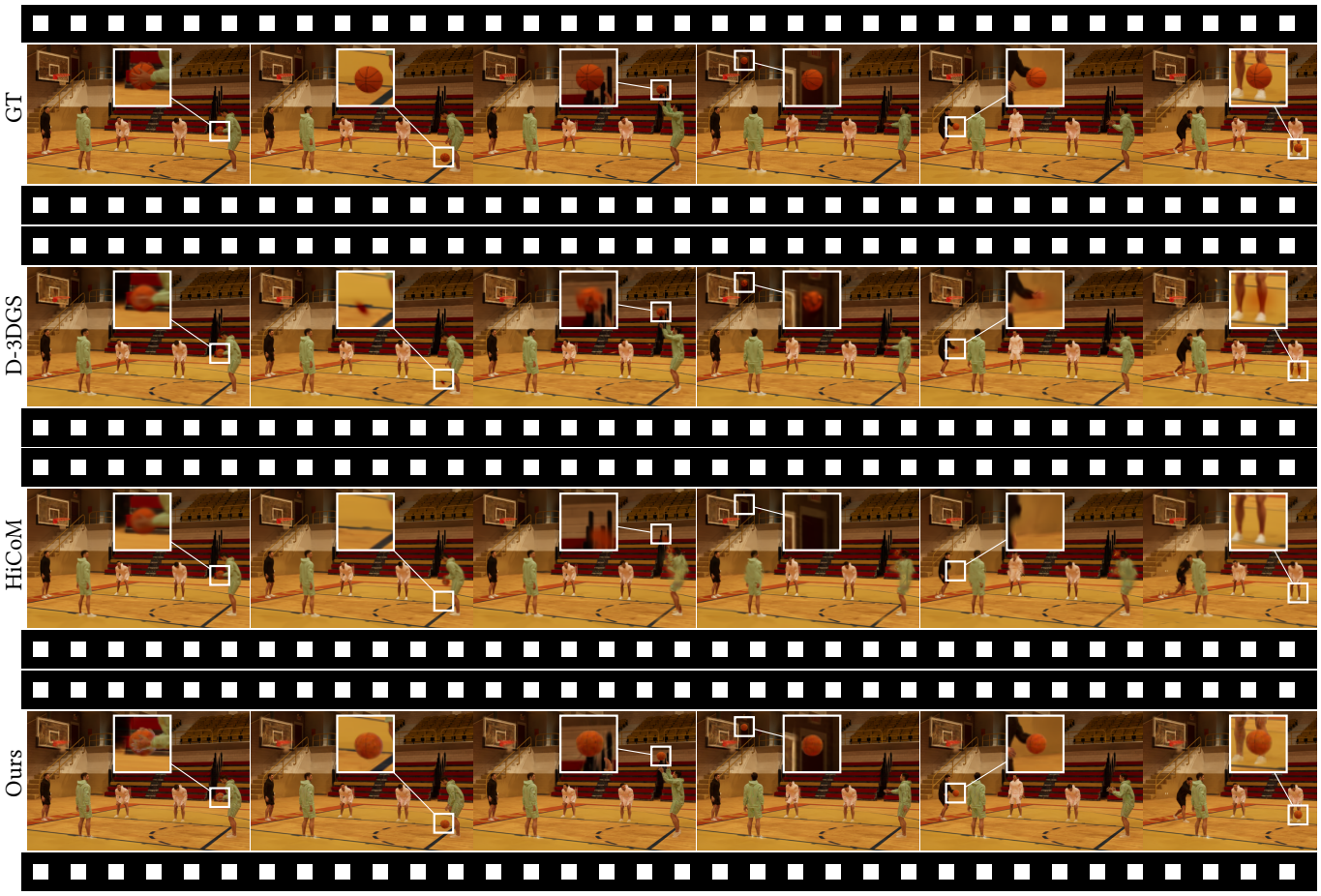

Fig. 6. Video qualitative results

settings. Nevertheless, several limitations and opportunities for future exploration remain.

First, despite robust performance demonstrated on the evaluated datasets, the range of human motions captured is relatively limited, primarily reflecting structured sports environments and casual indoor scenarios. Future research should address more complex interactions involving physical contact, such as wrestling or boxing, to rigorously assess and further enhance the robustness of our approach.

Second, we rely on human-specific priors and skeletal constraints, limiting validation thus far to human-centric datasets. Extending the underlying concept of binding Gaussians to generalized object skeletons represents a compelling and promising direction for future investigations, potentially expanding applicability beyond human motion capture.

Additionally, we currently assume a geometrically static environment, restricting its suitability in scenarios involving significant environmental dynamics, and investigating extensions to accommodate dynamic, evolving scenes is essential.

Finally, while the synthetic nature of the BASKET-Multiview dataset enables highly accurate ground-truth annotations, it inherently lacks some complexities encountered in real-world conditions. Therefore, comprehensive validation on more diverse, real-world datasets is imperative to fully establish the generalizability and robustness of LiveSplats in practical scenarios.

6 Conclusion

We propose LiveSplats, a framework towards real-time, photo-realistic 3D reconstruction of sports events, using 3DGS and distributed optimization. By computationally factorizing subjects and environment reconstruction, and employing an hierarchical optimization strategy with performance enhancements to the rasterization pipeline, we achieve scalable, high fidelity results that with the right computation could be run at interactive frame rates. Our BASKET-Multiview dataset, establishes a benchmark for evaluating methods in complex dynamic scenarios. Experiments demonstrate our approach surpasses SOTA methods in quality, robustness, and efficiency, setting the stage for immersive, real-time live event streaming that bridges physical and digital experiences.

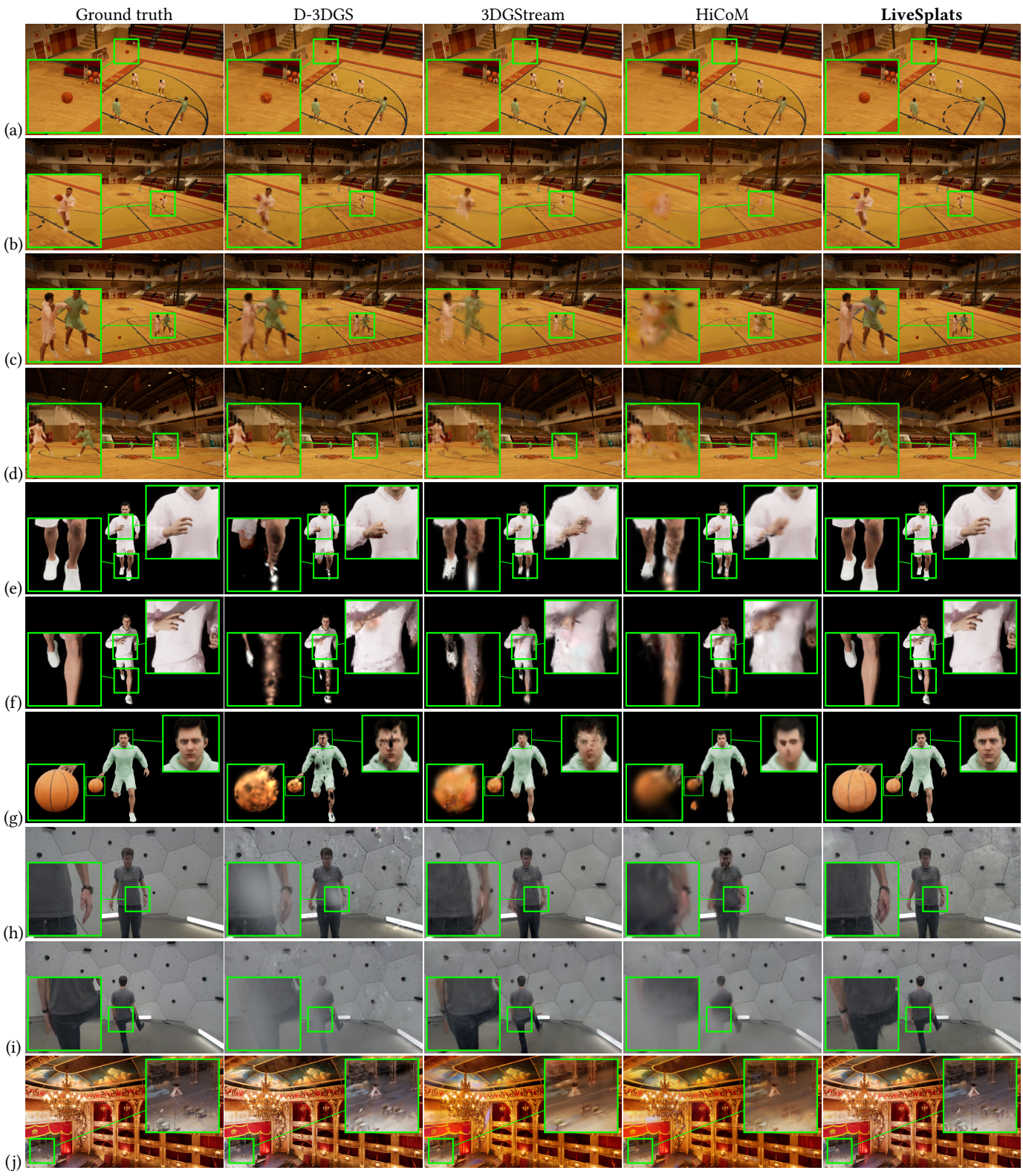

Fig. 7. Qualitative Results. Rows (a)-(d) are from BASKET-Multiview *Core* scenes. Rows (e)-(g) are from BASKET-Multiview *Development* scenes. Rows (h) and (i) are from CMU-Panoptic dataset. Row (j) is a static scene that we created to evaluate the static reconstruction quality.

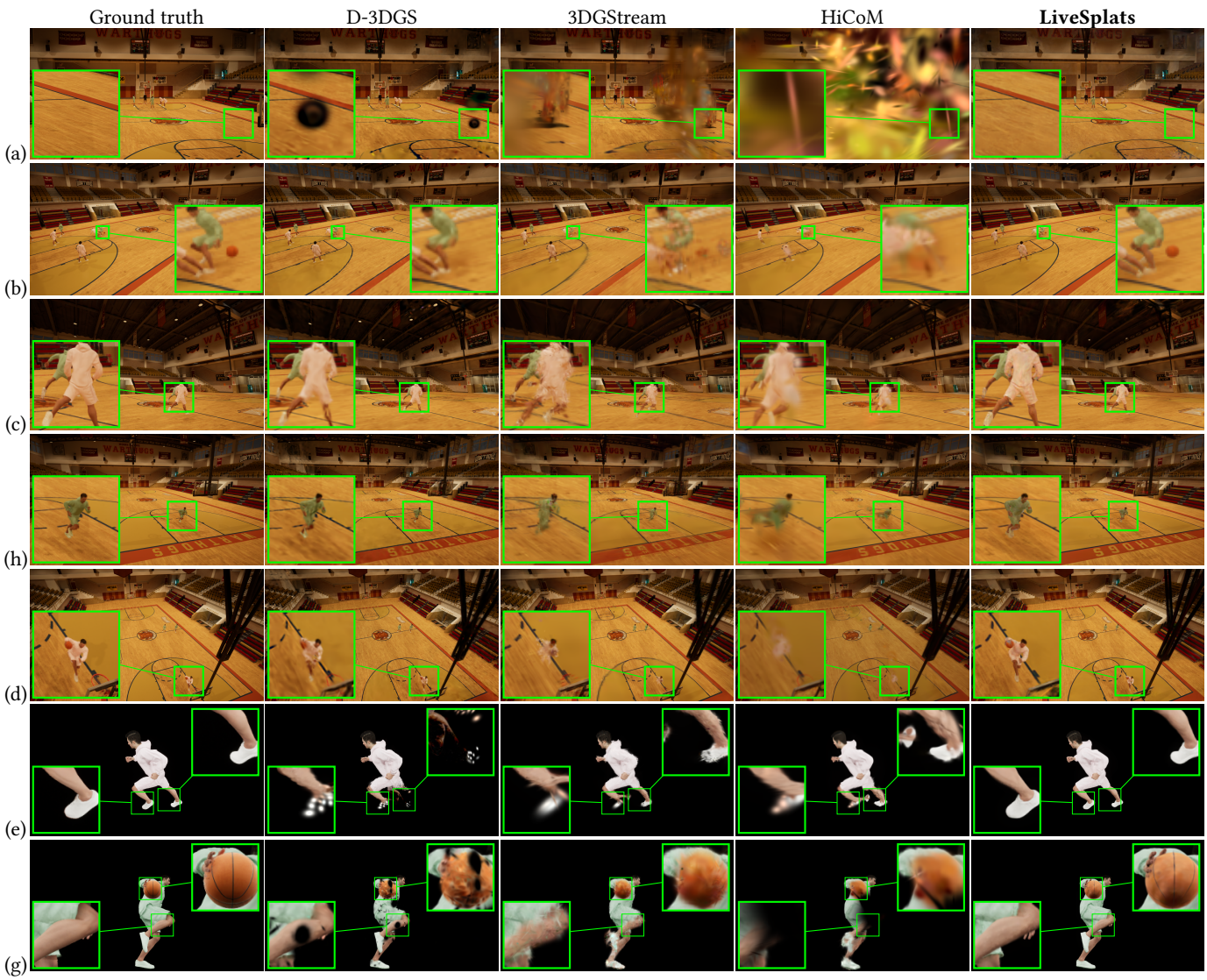

Fig. 8. Additional qualitative Results. Rows (a)-(d) are from BASKET-Multiview *Core* scenes. Rows (e)-(g) are from BASKET-Multiview *Development* scenes.

References

Jason Ansel, Edward Yang, Horace He, Natalia Gimelshein, Animesh Jain, Michael Voznesensky, Bin Bao, Peter Bell, David Berard, Evgeni Burovski, Geeta Chauhan, Anjali Chourdia, Will Constable, Alban Desmaison, Zachary DeVito, Elias Ellison, Will Feng, Jiong Gong, Michael Gschwind, Brian Hirsh, Sherlock Huang, Kshiteej Kalambarkar, Laurent Kirsch, Michael Lazos, Mario Lezcano, Yanbo Liang, Jason Liang, Yinghai Lu, CK Luk, Bert Maher, Yunjie Pan, Christian Puhrsch, Matthias Reso, Mark Saroufim, Marcos Yukio Siraichi, Helen Suk, Michael Suo, Phil Tillet, Eikan Wang, Xiaodong Wang, William Wen, Shunting Zhang, Xu Zhao, Keren Zhou, Richard Zou, Ajit Mathews, Gregory Chanan, Peng Wu, and Soumith Chintala. 2024. PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation. In *29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2 (ASPLOS '24)*. ACM. doi:10.1145/3620665.3640366

Bharat Lal Bhatnagar, Cristian Sminchisescu, Christian Theobalt, and Gerard Pons-Moll. 2020a. Combining Implicit Function Learning and Parametric Models for 3D Human Reconstruction. In *European Conference on Computer Vision (ECCV)*. Springer.

Bharat Lal Bhatnagar, Cristian Sminchisescu, Christian Theobalt, and Gerard Pons-Moll. 2020b. LoopReg: Self-supervised Learning of Implicit Surface Correspondences, Pose and Shape for 3D Human Mesh Registration. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Z. Cao, G. Hidalgo Martinez, T. Simon, S. Wei, and Y. A. Sheikh. 2019. OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2019).

Jeffrey Dean and Sanjay Ghemawat. 2008. MapReduce: simplified data processing on large clusters. *Commun. ACM* 51, 1 (Jan. 2008), 107–113. doi:10.1145/1327452.1327492

Epic Games. 2025. *Unreal Engine 5*. https://www.unrealengine.com

Qiankun Gao, Jiarui Meng, Chengxiang Wen, Jie Chen, and Jian Zhang. 2024. HiCoM: Hierarchical Coherent Motion for Dynamic Streamable Scenes with 3D Gaussian Splatting. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Sharath Girish, Tianye Li, Amrita Mazumdar, Abhinav Shrivastava, David Luebke, and Shalini De Mello. 2024. QUEEN: QUantized Efficient ENcoding for Streaming Free-viewpoint Videos. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*. https://openreview.net/forum?id=7xhwE7VH4S

Liangxiao Hu, Hongwen Zhang, Yuxiang Zhang, Boyao Zhou, Boning Liu, Shengping Zhang, and Liqiang Nie. 2024b. Gaussianavatar: Towards realistic human avatar modeling from a single video via animatable 3d gaussians. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 634–644.

Tao Hu, Fangzhou Hong, and Ziwei Liu. 2024a. Surmo: surface-based 4D motion modeling for dynamic human rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 6550–6560.

CMU Imaging Group. 2021. Time-of-Flight Radiance Fields (TöRF): A method for integrating depth sensing into dynamic radiance field modeling. *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2021).

Yuheng Jiang, Zhehao Shen, Yu Hong, Chengcheng Guo, Yize Wu, Yingliang Zhang, Jingyi Yu, and Lan Xu. 2024. Robust Dual Gaussian Splatting for Immersive Human-centric Volumetric Videos. *ACM Trans. Graph.* 43, 6, Article 265 (Nov. 2024), 15 pages. doi:10.1145/3687926

Hanbyul Joo, Tomas Simon, Xulong Li, Hao Liu, Lei Tan, Lin Gui, Sean Banerjee, Timothy Scott Godisart, Bart Nabbe, Iain Matthews, Takeo Kanade, Shohei Nobuhara, and Yaser Sheikh. 2017. Panoptic Studio: A Massively Multiview System for Social Interaction Capture. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2017).

James T Kajiya. 1986. The rendering equation. *ACM SIGGRAPH computer graphics* 20, 4 (1986), 143–150.

Kai Katsumata, Duc Minh Vo, and Hideki Nakayama. 2024. A Compact Dynamic 3D Gaussian Representation for Real-Time Dynamic View Synthesis. In *Computer Vision – ECCV 2024: 18th European Conference, Milan, Italy, September 29–October 4, 2024, Proceedings, Part LXXXVI* (Milan, Italy). Springer-Verlag, Berlin, Heidelberg, 394–412. doi:10.1007/978-3-031-73016-0_23

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 2023. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Transactions on Graphics* 42, 4 (July 2023). https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/

Agelos Kratimenos, Jiahui Lei, and Kostas Daniilidis. 2024. DynMF: Neural Motion Factorization for Real-time Dynamic View Synthesis with 3D Gaussian Splatting. *ECCV* (2024).

Lingzhi Li, Zhen Shen, Zhongshu Wang, Li Shen, and Ping Tan. 2022. Streaming radiance fields for 3d video synthesis. *Advances in Neural Information Processing Systems* 35 (2022), 13485–13498.

Peng Li, Wangguandong Zheng, Yuan Liu, Tao Yu, Yangguang Li, Xingqun Qi, Mengfei Li, Xiaowei Chi, Siyu Xia, Wei Xue, et al. 2024c. PSHuman: Photorealistic Single-view Human Reconstruction using Cross-Scale Diffusion. *arXiv preprint arXiv:2409.10141* (2024).

Xueting Li, Ye Yuan, Shalini De Mello, Gilles Daviet, Jonathan Leaf, Miles Macklin, Jan Kautz, and Umar Iqbal. 2024b. SimAvatar: Simulation-Ready Avatars with Layered Hair and Clothing. *arXiv preprint arXiv:2412.09545* (2024).

Zhi Li, Christos G Bampis, Ioannis Katsavounidis, Anne Aaron, and Alan C Bovik. 2016. Toward a practical perceptual video quality metric. *Netflix Technology Blog* (June 2016). https://medium.com/netflix-techblog/toward-a-practical-perceptual-video-quality-metric-653f208b9652

Zhan Li, Zhang Chen, Zhong Li, and Yi Xu. 2024a. Spacetime Gaussian Feature Splatting for Real-Time Dynamic View Synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 8508–8520.

Youtian Lin, Zuozhuo Dai, Siyu Zhu, and Yao Yao. 2024. Gaussian-flow: 4d reconstruction with dynamic 3d gaussian particle. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 21136–21145.

Xian Liu, Xiaohang Zhan, Jiaxiang Tang, Ying Shan, Gang Zeng, Dahua Lin, Xihui Liu, and Ziwei Liu. 2024. Humangaussian: Text-driven 3d human generation with gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 6646–6657.

Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. 2015. SMPL: A Skinned Multi-Person Linear Model. *ACM Trans. Graphics (Proc. SIGGRAPH Asia)* 34, 6 (Oct. 2015), 248:1–248:16.

Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. 2024. Dynamic 3D Gaussians: Tracking by Persistent Dynamic View Synthesis. In *3DV*.

Saswat Subhajyoti Mallick, Rahul Goel, Bernhard Kerbl, Markus Steinberger, Francisco Vicente Carrasco, and Fernando De La Torre. 2024. Taming 3dgs: High-quality radiance fields with limited resources. In *SIGGRAPH Asia 2024 Conference Papers*. 1–11.

Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. 2020. NeRF: Representing scenes as neural radiance fields for view synthesis. *Commun. ACM* 65, 1 (2020), 99–106.

Nice Pictures. 2024. Men's and Women's Sport Outfits 3 - Modular - Rigged. https://www.fab.com/listings/2eebf211-df72-4daf-a8c4-24bfadba9e7a 3D asset compatible with Metahuman skeletons, available at Fab.com, accessed on January 20, 2025.

Albert Pumarola, Enrique Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. 2021. D-NeRF: Neural Radiance Fields for Dynamic Scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 10318–10327.

Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollár, and Christoph Feichtenhofer. 2024. SAM 2: Segment Anything in Images and Videos. *arXiv preprint arXiv:2408.00714* (2024). https://arxiv.org/abs/2408.00714

Shunsuke Saito, Stanislav Pidhorskyi, Igor Santesteban, Forrest Iandola, Divam Gupta, Anuj Pahuja, Nemanja Bartolovic, Frank Yu, Emanuel Garbin, and Tomas Simon. 2024. SqueezeMe: Efficient Gaussian Avatars for VR. *arXiv preprint arXiv:2412.15171* (2024).

Zhijing Shao, Zhaolong Wang, Zhuang Li, Duotun Wang, Xiangru Lin, Yu Zhang, Mingming Fan, and Zeyu Wang. 2024. Splattingavatar: Realistic real-time human avatars with mesh-embedded gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 1606–1616.

Richard Shaw, Michal Nazarczuk, Jifei Song, Arthur Moreau, Sibi Catley-Chandar, Helisa Dhamo, and Eduardo Perez-Pellitero. 2024. SWinGS: Sliding Windows for Dynamic 3D Gaussian Splatting. arXiv:2312.13308 [cs.CV] https://arxiv.org/abs/2312.13308

Peter-Pike Sloan, Jan Kautz, and John Snyder. 2002. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. In *ACM SIGGRAPH 2002 Papers*. 527–536.

Liangchen Song, Anpei Chen, Zhong Li, Zhang Chen, Lele Chen, Junsong Yuan, Yi Xu, and Andreas Geiger. 2023. NeRFPlayer: A Streamable Dynamic Scene Representation with Decomposed Neural Radiance Fields. *IEEE Transactions on Visualization and Computer Graphics* 29, 5 (2023), 2732–2742. doi:10.1109/TVCG.2023.3247082

Decagon Studios. 2025. High School Basketball Gym - (Day/Night/Afternoon/Midnight Lighting). https://www.fab.com/listings/03e76034-abbf-4fc2-aa05-b025996eeb1d 3D asset available at Fab.com, accessed on January 20, 2025.

Jiakai Sun, Han Jiao, Guangyuan Li, Zhanjie Zhang, Lei Zhao, and Wei Xing. 2024. 3DGStream: 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 (CVPR)*. 20675–20685.

Carnegie Mellon University. 2001. Carnegie Mellon Professor's Unique New Vision Technology Will Be Used To Present Replays In Super Bowl XXXV. ScienceDaily. https://www.sciencedaily.com/releases/2001/01/010124075009.htm Accessed: 2025-01-19.

Liao Wang, Qiang Hu, Qihan He, Ziyu Wang, Jingyi Yu, Tinne Tuytelaars, Lan Xu, and Minye Wu. 2023. Neural Residual Radiance Fields for Streamably Free-Viewpoint Videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 76–87.

Liao Wang, Kaixin Yao, Chengcheng Guo, Zhirui Zhang, Qiang Hu, Jingyi Yu, Lan Xu, and Minye Wu. 2024a. Videorf: Rendering dynamic radiance fields as 2d feature video streams. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 470–481.

Penghao Wang, Zhirui Zhang, Liao Wang, Kaixin Yao, Siyuan Xie, Jingyi Yu, Minye Wu, and Lan Xu. 2024b. Vˆ 3: Viewing Volumetric Videos on Mobiles via Streamable 2D Dynamic Gaussians. *ACM Transactions on Graphics (TOG)* 43, 6 (2024), 1–13.

Jing Wen, Xiaoming Zhao, Zhongzheng Ren, Alexander G Schwing, and Shenlong Wang. 2024. Gomavatar: Efficient animatable human modeling from monocular video using gaussians-on-mesh. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2059–2069.

Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 2024. 4D Gaussian Splatting for Real-Time Dynamic Scene Rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 20310–20320.

Yunzhi Yan, Haotong Lin, Chenxu Zhou, Weijie Wang, Haiyang Sun, Kun Zhan, Xianpeng Lang, Xiaowei Zhou, and Sida Peng. 2024. Street gaussians: Modeling dynamic urban scenes with gaussian splatting. In *European Conference on Computer Vision*. Springer, 156–173.

Ziyi Yang, Xinyu Gao, Wen Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. 2024. Deformable 3d gaussians for high-fidelity monocular dynamic scene reconstruction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 20331–20341.

YDRIVE. 2023. *EasySynth: An Unreal Engine Plugin for Image Dataset Creation*. https://github.com/ydrive/EasySynth

A Ablations

To evaluate skeleton optimization with noisy priors, we mimic different noise levels in the skeletons by randomly perturbing the limb joints (shoulders, elbows, hips, and knees) by maximum 10° or 20°, and then compare the reconstruction quality with and without skeleton optimization. We choose two scenes: Running Player (with close views) and Attack 4 (with mid views) to experiment on. To quantify the skeleton quality, we report the skeleton error (SE), computed by $\text{SE} = \frac{1}{N}\sum_{i=1}^{N}||\mathbf{x}_i - \mathbf{x}_i^*||$, where $\mathbf{x}_i, \mathbf{x}_i^*$ are the optimized and ground truth joint positions respectively, and N is the number of joints in the skeleton. From the results in Table 6 and Figure 3, we can see that the skeleton optimization can effectively reduce the SE and improve the video quality under various noise levels and camera settings.

Table 6. Ablation study of skeleton optimization. Max ptb: max degrees of random perturbation.

Scene	Max ptb	Skl opt	SE↓	VMAF↑	PSNR↑	M-PSNR↑	SSIM↑	LPIPS↓
Running Player (LC)	10°	✗	2.356	58.83	33.06	22.17	0.981	0.023
		✓	**0.593**	**73.77**	**37.43**	**26.61**	**0.987**	**0.014**
	20°	✗	4.519	43.78	29.77	18.55	0.976	0.031
		✓	**1.258**	**65.11**	**34.84**	**24.27**	**0.989**	**0.019**
Attack 4	10°	✗	2.588	71.33	29.88	24.15	0.924	0.107
		✓	**1.334**	**72.59**	**30.17**	**26.22**	**0.927**	**0.106**
	20°	✗	5.134	69.88	29.59	22.90	0.922	0.110
		✓	**2.563**	**72.27**	**30.10**	**25.72**	**0.926**	**0.106**

We evaluate the speed-up contributions of techniques in our static optimization method: optimizing SH parameters (Only SH), precomputing Gaussian sortings (Precomp), CUDA computation graph (Graph), and load balancing (LB) through an ablation study on the empty stadium scene with SH1 and SH3. Table 7 shows that each technique provides substantial speed improvements for both.

Table 7. Ablation study of static optimization techniques.

SH Only	Precomp	Graph	LB	ms / iter SH1	ms / iter SH3
				24	34
✓				20	30
✓	✓			13	25
✓	✓	✓		9	21
✓	✓	✓	✓	**6**	**18**

B Per-scene evaluation results.

We report the quality and performance metrics per scene on CMU-Panoptic dataset in Table 8 and BASKET-Multiview in Table 9.

C BASKET-Multiview dataset

The full dataset specifications are shown in Table 10 and Table 11. Our dataset does not simulate any crowd dynamics among the audience.

Table 8. CMU-Panoptic dataset per-scene evaluation results.

Scene	Method	VMAF↑	PSNR↑	M-PSNR↑	SSIM↑	LPIPS↓	SPF↓	VE↑	PE↑
Scene 1	StreamRF	37.86	23.82	19.31	0.819	0.412	9.0	4.21	2.65
	3DGStream	50.69	26.25	22.58	0.909	0.331	5.0	10.14	5.25
	D-3DGS	53.84	26.69	25.51	0.914	0.281	88.0	0.61	0.30
	HiCoM	42.58	26.08	23.42	0.909	0.297	4.0	10.65	6.52
	Ours	53.80	25.58	25.89	0.876	0.274	4.7	11.45	5.44
Scene 2	StreamRF	33.10	23.39	17.14	0.819	0.359	9.0	3.68	2.60
	3DGStream	39.19	24.21	19.31	0.889	0.347	5.0	7.84	4.84
	D-3DGS	52.38	26.23	23.64	0.899	0.292	92.0	0.57	0.29
	HiCoM	34.38	24.31	19.84	0.894	0.311	4.0	8.60	6.08
	Ours	57.82	26.03	25.81	0.873	0.282	4.7	12.30	5.54
Scene 3	StreamRF	32.51	23.16	18.14	0.917	0.441	9.0	3.61	2.57
	3DGStream	48.11	26.57	22.22	0.908	0.335	5.0	9.62	5.31
	D-3DGS	45.77	26.32	24.64	0.912	0.284	85.0	0.54	0.31
	HiCoM	38.15	25.67	22.28	0.909	0.295	4.0	9.54	6.42
	Ours	60.36	27.97	27.96	0.889	0.267	4.7	12.84	5.95

Table 9. BASKET dataset per-scene evaluation results.

Scene	Method	VMAF↑	PSNR↑	M-PSNR↑	SSIM↑	LPIPS↓	SPF↓	VE↑	PE↑
Attack 1	StreamRF	33.79	26.49	12.15	0.816	0.192	49.0	0.69	0.54
	3DGStream	50.44	28.42	20.67	0.874	0.173	12.5	4.04	2.27
	D-3DGS	64.87	29.07	22.03	0.901	0.147	76.0	0.85	0.38
	HiCoM	61.21	29.77	21.49	0.917	0.125	6.1	10.03	4.88
	Ours	65.69	29.50	24.05	0.908	0.123	5.2	12.63	5.67
Attack 2	StreamRF	34.12	26.82	13.67	0.755	0.259	49.0	0.70	0.55
	3DGStream	41.34	27.73	19.36	0.844	0.213	12.3	3.36	2.25
	D-3DGS	66.03	29.08	21.48	0.901	0.146	78.0	0.85	0.37
	HiCoM	62.18	29.48	18.06	0.914	0.127	6.3	9.87	4.68
	Ours	66.44	29.65	24.22	0.910	0.121	5.2	12.78	5.70
Attack 3	StreamRF	40.95	27.14	11.86	0.721	0.193	48.0	0.85	0.57
	3DGStream	55.97	29.21	21.59	0.896	0.148	12.5	4.48	2.34
	D-3DGS	63.99	29.01	25.38	0.901	0.146	74.0	0.86	0.39
	HiCoM	56.37	28.64	19.81	0.906	0.134	6.2	9.09	4.62
	Ours	66.74	29.80	26.01	0.911	0.122	5.2	12.83	5.73
Attack 4	StreamRF	37.86	27.42	14.75	0.721	0.182	48.0	0.79	0.57
	3DGStream	57.01	26.75	23.63	0.869	0.179	14.0	4.07	1.91
	D-3DGS	65.37	26.82	24.33	0.894	0.169	67.0	0.98	0.40
	HiCoM	31.47	22.03	18.65	0.751	0.331	6.0	5.25	3.67
	Ours	72.68	30.17	25.60	0.927	0.105	5.0	14.54	6.03
Defense 1	StreamRF	35.81	26.41	13.84	0.787	0.189	49.0	0.73	0.54
	3DGStream	53.19	28.98	21.06	0.884	0.159	12.2	4.36	2.38
	D-3DGS	66.74	29.04	24.77	0.903	0.143	74.0	0.90	0.39
	HiCoM	56.14	28.53	18.45	0.907	0.131	6.1	9.20	4.68
	Ours	66.97	29.68	25.78	0.909	0.121	5.2	12.88	5.71
Defense 2	StreamRF	36.46	26.36	14.96	0.784	0.177	48.0	0.76	0.55
	3DGStream	56.57	29.19	19.81	0.895	0.151	12.3	4.60	2.37
	D-3DGS	65.25	29.18	23.06	0.903	0.145	76.0	0.86	0.38
	HiCoM	55.08	28.59	17.81	0.905	0.131	6.1	9.03	4.69
	Ours	66.31	29.75	26.15	0.910	0.121	5.2	12.75	5.72
Interval 1	StreamRF	35.14	27.11	12.89	0.711	0.195	49.0	0.72	0.55
	3DGStream	45.46	28.22	19.38	0.862	0.195	12.1	3.76	2.33
	D-3DGS	65.59	29.05	21.36	0.901	0.146	75.0	0.87	0.39
	HiCoM	60.39	29.49	18.94	0.914	0.121	6.2	9.74	4.76
	Ours	66.28	29.69	24.86	0.910	0.123	5.2	12.75	5.71

C.1 Config sets

We use several illumination setups (denoted by *Config set* in 11) to better simulate environments for diverse benchmarking of our method.

(1) Set 1 (Figure 9): The map consists of a basketball court with the following illumination setup.
Point lights: 46 point lights are strategically placed through the scene to simulate localized lightning. These lights are

Table 10. Core BASKET-Multiview dataset composition

Sequence Name	# Players	#Cameras	Shot	#Frames	Resolution
Attack 1	11	83	Far	321	1080p
Attack 2	6	83	Far	130	1080p
Attack 3	4	83	Far	319	1080p
Attack 4	5	83	Mid	321	4k
Defense 1	2	83	Far	160	1080p
Defense 2	3	83	Far	220	1080p
Interval 1	5	83	Far	185	1080p

placed near the assets that give illumination to the scene as fluorescent lights and the intensity of these point lights are adjusted for realism.

Spot Lights: 21 Spot Lights, with two types of intensity (15 lux and 10 lux) are placed on the top of the court and stands. The use of two types of illumination aims to enhance visibility on the court while creating a realistic ambiance by reducing light intensity in the stands, thereby directing spectators' focus toward the court.

Sky Light: The Sky Light setup uses the SLS Specified Cubemap source type with a cubemap of Tx_HDRI_07a at an angle of 175.0 and a distance threshold of 150, 000. The light intensity is set to 5.0, with a white color (FFFFFFFF). It affects the world and casts shadows, with indirect lighting intensity and volumetric scattering intensity both set to 1.0.

Post Process Volume: These settings include an Auto Exposure Histogram with an exposure compensation of −1.24, a minimum brightness of 0.5, and a maximum brightness of 8.0. The color temperature is set to 6036.0.

For global illumination, the *Lumen* method is used with a scene lighting quality of 5.0, scene detail of 4.0, and a scene view distance of 100.0. Reflections use a quality setting of 2.0, hit lighting for reflections, high-quality translucency reflections enabled, and a maximum of 3 reflection bounces.

(2) Set 2: This is simpler than Set 1 as it only consists of a Skylight. Here the scenario is completely empty as we want to place the player in a black environment with a sky light for global illumination.

Sky Light: The Sky Light setup uses the SLS Specified Cubemap source type with a cubemap GrayLightTextureCube. The light intensity is set to 3.0, with a white color (FFFFFFFF).

Post Process Volume: This sets up the lightning and reflection methods to use Lumen.

(3) Set 3 (Figure 10): This is similar to Set 2 with an additional point light moving around the player, to better understand the effect of lighting changes on our method's ability.

Point light: It is placed at 170.0 units from the player and rotates around it. It has an intensity of 10000.0 unitless, with a white color (FFFFFFFF) and an attenuation radius of 1000.0.

Sky light: This setup is same as Set 2 but with an intensity of 0.5.

(4) Set 4: The map is same as Set 1, a basketball court. The lighting configuration is the same as Set 1, with an additional spotlight on the middle of the path that the player is following. **Spot light**: It has an intensity of 2000.0 cd(candelas), it has a white color (FFFFFFFF), an attenuation radius of 1140.0 and an outer cone angle of 52.0, the rest of the values are set at 0.0.

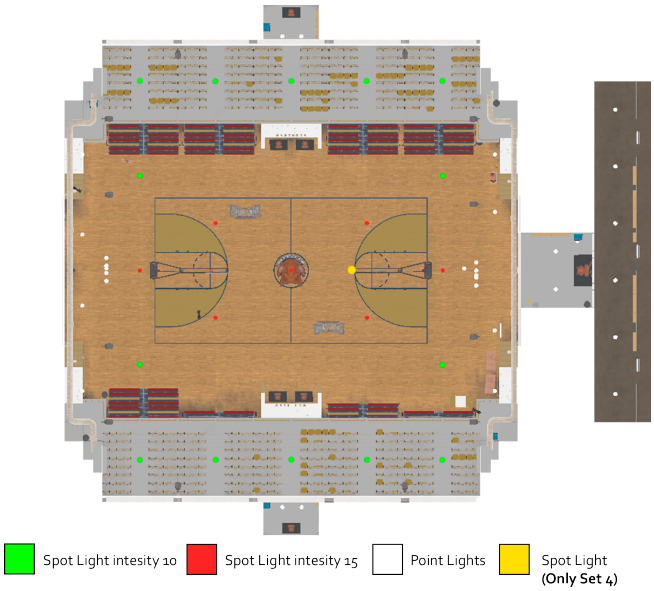

Fig. 9. Lighting layout in the basketball court.

Fig. 10. Lighting layout for the lighting change sequence.

Table 11. Development BASKET-Multiview dataset composition

Sequence Name	Animation name	Background	#Players	Ball	#Cameras	Shot	#Frames	Config set
Running Player w/o ball	Running Player	Black	1	✗	40	Close	57	Set 2
Running Player w/o ball with light dynamics		Black	1	✗	40	Close	57	Set 3
Running Player w/o ball		Court	1	✗	83	Far	105	Set 1
Running Player w/o ball long path		Court	1	✗	83	Far	600	Set 1
Running Player w/o ball with light dynamics		Court	1	✗	83	Far	105	Set 4
Dribbling Player w/o ball	Dribbling Player	Black	1	✗	40	Close	48	Set 2
Dribbling Player with ball		Black	1	✓	40	Close	48	Set 2
Dribbling Player w/o ball		Court	1	✗	83	Far	160	Set 1
Day Cycle		Court	0	✗	83	Far	76	Set 1

C.2 Camera setup

We use two different camera setups, one for all the scenes in the stadium (Config Set 1 & 4) and another for scenes with a black background (Config Set 2 & 3)

Fig. 11. 83-camera layout in the basketball court.

(1) **83-Camera Setup**: The camera rig consists of 83 cameras designed to capture all views of the stadium, as illustrated in Figure 11. Using the stadium's square shape, we place cameras across multiple levels: the highest focuses on the play, while the others capture the stadium's structure.
All cameras have the same configuration, with a sensor width of 23.76 mm and a sensor height of 13.366 mm. They recreate a 16:9 Digital film and capture with a lens of focal length of 12 mm, with a min FStop of 2.8 and a Max FStop of 22.0.
The rig for Attack 4 has a slight change in the orientation. We place an empty actor on the basket and activate the *look at* function on the camera settings. The camera configurations are similar with a focal length of 22.00 mm for simulating a zoomed-in view.
(2) **40-Camera Setup**: This setup is designed to capture multiple angles of the player, and consists of 40 cameras, as illustrated in Figure 13. The cameras are arranged in a sphere and the player is located at the center. To improve the view quality, we activate the *look at* function with an offset of 94.0 on the Z axis. The camera configurations are similar to the 83 Camera setup, but with a sensor width of 24 mm and a sensor height of 13.5 mm. We use this camera layout to capture the binding pose players (as illustrated in Figure 12) and black background sequences.

Fig. 12. Sample binding pose player.

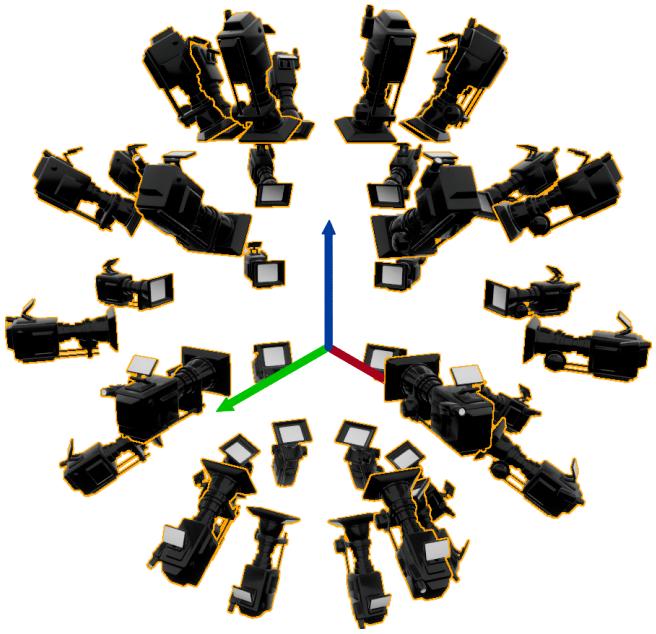

Fig. 13. 40-camera layout for capturing the binding pose pictures and black background sequences.

C.3 Render settings

(1) EasySynth [YDRIVE 2023]: This setup is configured to generate RGB (Color) images, depth maps, and normal maps. The color images are output in PNG format, depth maps in

EXR format with a 16-bit precision, and normal maps in PNG format. The resolution is set to 1920x1080 and depth range configured to 100 meters.

(2) Movie Render Queue: Anti-aliasing is turned on with spatial sample count set to 4 and temporal sample count set to 1. Anti-aliasing is overridden and the anti-aliasing method is set to *None* to ensure no additional smoothing artifacts are introduced. The render warm-up count is set to 32, and the engine warm-up count is set to 4. Both "Use Camera Cut for Warm-Up" and "Render Warm-Up Frames" are disabled.
In the console variables section, Temporal AA upsampling (`r.TemporalAA.Upsampling`) is set to 0.0 to disable any upscaling associated with temporal anti-aliasing. Motion blur quality (`r.MotionBlurQuality`) is also set to 0.0 to remove motion blur effects from the render. The screen percentage (`r.ScreenPercentage`) is set to 70.0, likely to balance performance and output quality. For the output settings, the image size is set to 1920x1080.

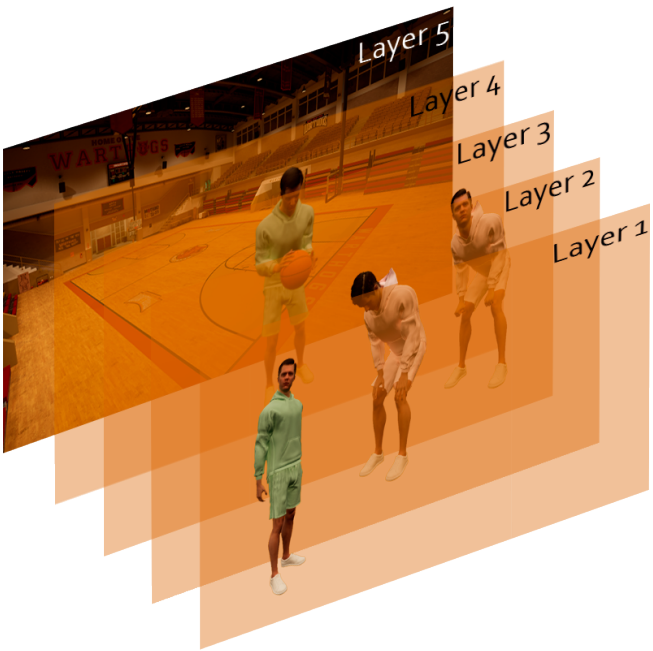

Fig. 14. Layer rendering illustration.

C.4 Data generating workflow

We outline a brief sketch of the steps taken to design a sequence in order.

(1) The character skeleton is exported from Unreal Engine 5 (UE5) to Maya. In Maya, a plugin is applied to generate control rigs, which facilitate precise control over the animation process and result in realistic, fluid movements.

(2) Using references from real sports plays, animations are carefully created to replicate the movements accurately. Once the animations are completed, they are exported from Maya and re-imported into UE5 as animated skeletons.

(3) The play sequences are created in UE5 by combining player animations with elements like the ball. These sequences are assembled and organized in the sequence editor.

(4) To generate detailed visual data, Easy Synth is configured. It generates multiple types of images, such as Base Color renders, Depth maps (up to 100 meters, exported in EXR format for improved precision), and Normal maps capturing surface geometry.

(5) Finally, semantic images are generated to enhance accuracy. Each player, the ball, and the background are separated into individual layers with transparency, as illustrated in Figure 14. This layered approach results in more precise semantic images.