Object-Wise Decomposition of 3D Gaussian Splatting Scenes for Physics-Enabled VR Interaction

by

Skylar Huang Knight

Department of Computer Science
Duke University

2026

Abstract

We present an end-to-end pipeline for converting multi-view RGB captures into object-centric, physics-enabled virtual reality scenes rendered with 3D Gaussian Splatting. Existing 3DGS reconstructions provide high visual fidelity but are typically scene-level representations without explicit object structure or physical affordances. Our method bridges this gap by combining structure-from-motion calibration, 3DGS reconstruction, prompt-driven instance segmentation, geometry-guided mask propagation, and per-Gaussian object labeling. The resulting representation separates the reconstructed scene into independently manipulable Gaussian assets and a static background. To support physically plausible interaction, the pipeline conservatively completes occlusion-induced missing regions and derives simplified mesh proxies for collision and rigid-body simulation. At runtime, each invisible proxy is synchronized with its corresponding visible Gaussian asset, preserving the appearance quality of neural rendering while enabling object grabbing, movement, and collision response in VR. This framework provides a practical approach for transforming image-based scene captures into interactive immersive environments, narrowing the gap between photorealistic neural reconstruction and real-time physics-based manipulation.

Contents

Acknowledgements

Thanks to my principal advisor, Dr. Maria Gorlatova, postdoctoral advisor Hanting Ye, committee members Michael Reed, Andrew Hurley, Josef Spjut, and the Duke Intelligent Interactive Internet of Things (I3T) Lab.

1. Background

1.1 3D Gaussian Splatting

3D Gaussian Splatting (3DGS) represents a scene as a collection of explicit 3D Gaussian primitives and enables high-quality real-time novel view synthesis [1]. Given a set of posed images, the scene is modeled by Gaussians $\{G_i\}_{i=1}^{N}$, where each Gaussian is parameterized by its center $m_i \in \mathbb{R}^3$, rotation $q_i \in \mathbb{R}^4$, scale $s_i \in \mathbb{R}^3$, opacity o_i, and view-dependent color coefficients c_i. The covariance of each Gaussian is constructed from its rotation and scale as

$$\Sigma_i = R(q_i)\, S(s_i) S(s_i)^\top R(q_i)^\top, \tag{1.1}$$

where $R(q_i)$ is the rotation matrix induced by q_i and $S(s_i)$ is the diagonal scaling matrix. During rendering, each 3D Gaussian is projected onto the image plane as a 2D Gaussian, and the projected primitives are rasterized using a tile-based splatting pipeline.

For a pixel u, the rendered color is computed by front-to-back alpha compositing:

$$C(u) = \sum_i c_i \alpha_i(u) \prod_{j=1}^{i-1} \big(1 - \alpha_j(u)\big), \tag{1.2}$$

where $\alpha_i(u)$ denotes the opacity contribution of the i-th projected Gaussian at pixel u. Because rendering is performed through explicit splatting rather than dense ray marching, 3DGS achieves substantially higher rendering efficiency while preserving high visual fidelity [1].

The explicit structure of 3DGS also makes it suitable for downstream scene understanding. Recent 3DGS-based methods augment Gaussian primitives with identity, affinity, or semantic features derived from 2D masks, prompts, or language supervision, enabling object-level grouping and querying [2]–[4]. In this setting, segmentation associates reconstructed Gaussian primitives with semantic or instance labels, so that subsets

of Gaussians can subsequently be selected, removed, or edited as object-level scene elements.

1.2 Interactive VR Systems

Interactive virtual reality systems combine low-latency tracking, real-time rendering, input handling, and simulation in a continuous update loop [5]. Unlike passive novel-view synthesis, VR requires scene content to be exposed as persistent entities whose state can be updated in response to user actions. Contemporary VR interaction is therefore built around a small set of object-centric primitives, including pointing, selection, grabbing, and transform manipulation. These operations are typically mediated through controllers or hand tracking, while physics engines provide collision handling and plausible rigid-body response for interactive objects [6].

This interaction model aligns naturally with object-centric Gaussian scenes. Once a subset of Gaussians is associated with an object or semantic region, it can be treated as an interactive entity with its own transform, collider, and interaction state. As a result, 3DGS offers not only an efficient representation for neural rendering, but also a practical interface between reconstructed scenes and immersive VR interaction.

2. Related Work

Neural scene representations have revolutionized photorealistic rendering by enabling high fidelity novel-view synthesis from multi-view imagery. However, bridging these representations to dynamic, interactive applications remains incomplete.

2.1 Neural Scene Representations for Real-Time Rendering

Early works in visual computing primarily modeled complex scenes through explicit spatial partitions, such as meshes [7] and octrees [8] or continuous implicit surface boundaries [9], optimizing these foundational data structures for tasks including synthesis, estimation, rigorous spatial analysis, and high-fidelity physical manipulation. However, these explicit and discrete representations often struggle with fixed topology constraints, heavy memory footprints, and the inability to natively model continuous, high-frequency physical phenomena efficiently with high fidelity.

The reconstruction of photorealistic 3D scenes from 2D image collections has undergone a dramatic transformation over the past several years. Early novel view synthesis (NVS) techniques such as Neural Radiance Fields (NeRF) [10] revolutionized the standard for visual fidelity through the introduction of scene representations via implicit radiance fields. NeRF pioneered this shift by proposing encoding scene representations as a continuous volumetric function optimized through multi-layer perceptrons (MLPs) to parameterize a scene from posed images and synthesizing novel views via differentiable volume integration. The versatility of NVS has driven progress across numerous domains within computer vision and graphics, proving critical for not just photorealistic rendering [11], but also interactive animation [12], [13], accurate 3D reconstruction [14], physics-based simulation [15].

While NeRF achieves remarkable photorealism, it suffers from significant time and

memory inefficiencies that limit interactive application. NeRF's reliance on dense ray-marching involving querying the MLP for hundreds of points per ray inherently bottlenecks real-time rendering and training speed by making the rendering cost scale with rays $\times$ samples per ray and makes direct geometric manipulation challenging.

This has motivated extensive work on accelerating training and inference while preserving view-dependent effects. In order to overcome the computational bottleneck of large MLPs, subsequent works have explored various acceleration techniques. These include leveraging sparse voxel grids [16], factorizing the radiance field via low-rank tensors [17], and fast-querying architectural changes like multi-resolution hash encodings [18] or optimized explicit structures [19]. Hybrid approaches [20]–[23] accelerate NeRF-like models via sparse discrete voxel grids or octrees, achieving interactive performance on commodity hardware. Anti-aliasing and multiscale sampling improvements [24], [25] primarily target quality and robustness across scales and unbounded scenes, but continue to rely on expensive per-ray evaluation at inference and thus inherit workflow friction for downstream tasks such as semantics, editing, and physics.

The introduction of 3D Gaussian Splatting (3DGS) [1] has redefined the NVS landscape by enabling faster, high-quality synthesis from point based primitives by representing scenes as explicit collections of anisotropic 3D Gaussians rendered using a differentiable tile-based rasterizer and optimized by interleaving densification and parameter updates. 3DGS possesses the advantages of volumetric rendering approaches, offering high-fidelity view synthesis for complex scenes, while also benefiting from the merits of rasterization approaches. By leveraging a tile-based differentiable rasterizer, 3DGS sidesteps the computational overhead of ray-marching, thus achieving state-of-the-art rendering quality with real-time (($\geqslant$ 30 FPS)) framerates for large-scale scenes.

Subsequent literature has rapidly sought to improve upon the 3DGS pipeline, addressing limitations in memory footprint, aliasing artifacts, and scene scalability [26]–[28]. Hierarchical and scalable variants further extend 3DGS to large-scale scenes with level-of-detail control [29]. Concurrently, researchers have expanded the representational flexibil-

ity of the framework by integrating secondary paradigms, such as ray tracing [30], and generalizing the forward model to accommodate arbitrary camera geometries [31].

Beyond static reconstructions, 3DGS has emerged as a highly effective backbone for both generative pipelines and 4D view synthesis. By coupling the rapid rasterization of Gaussians with 2D diffusion priors, recent frameworks have drastically accelerated text-to-3D [32], [33] and image-to-3D generation tasks [34], [35]. In the temporal domain, dynamic formulations have been achieved either by explicitly tracking the kinematic trajectories of individual point primitives [36] or by continuously warping the canonical representation via neural deformation fields [37], [38].

Despite these substantial advancements in visual fidelity, scalability, and generation, the fundamental representation remains strictly photometric and decoupled from underlying physical laws. Existing pipelines optimize scenes exclusively as visual artifacts rather than reactive, physical environments. Consequently, current 3DGS formulations natively lack the capacity for real-time physical interaction, semantic reasoning, and immersive manipulation.

2.2 Open-Vocabulary 3D Scene Understanding

Open-vocabulary 3D scene understanding has emerged as a dynamic, rapidly evolving area of research. Foundational works in semantic based vision-language models were fundamentally constrained by closed-set label spaces and heavily reliant on domain-specific annotated datasets [39]. The emergence of large-scale vision-language models (VLMs), particularly CLIP [40], catalyzed a shift toward open-vocabulary semantic reasoning by aligning visual features with natural language supervision. Concurrently, prompt-driven 2D segmentation has been revolutionized by foundation models such as the Segment Anything Model (SAM) [41], with recent iterations improving boundary fidelity, temporal coherence, and zero-shot generalization [42], [43]. These developments enable the projection of semantically rich, viewpoint-consistent 2D features into 3D representa-

tions, facilitating open-vocabulary querying without scene-specific retraining.

Recent work has explored explicit scene representations, particularly 3D Gaussian Splatting (3DGS), as a more efficient substrate for semantic integration. Approaches such as Feature 3DGS [44] and Foundation Model Gaussian Splatting (FMGS) [45] associate compact semantic descriptors with individual Gaussians, enabling real-time rendering and querying. To further mitigate memory constraints, LangSplat [4] employs scene-specific feature compression, while Gaussian Grouping [3] leverages 2D segmentation priors to aggregate Gaussians into semantically meaningful object clusters. These methods benefit from advances in segmentation models, where high-quality mask generation provides fine-grained supervision for semantic lifting into 3D.

2.3 3D Editing and Inpainting

A critical challenge within 3D scene manipulation is editing and inpainting of neural 3D representations. Modifying underlying geometry, appearance, or semantic properties without the degradation of photorealism of the synthesized novel views is essential for supporting controllable scene manipulation. Early approaches within the domain relied on 2D-guided editing, where image-space modifications are projected into 3D via multi-view consistency. NeRF-based methods have introduced means of altering scenes via latent space manipulation or mask-based editing techniques [46]. Some methods [47] proved effective for completing missing scene content, particularly unobserved or poorly reconstructed regions, through the use of off-the-shelf 2D generative painters inside an iterative 3D distillation loop. Parallel work on object removal combined mask propagation, 2D inpainting, and NeRF re-training to delete designated objects while maintaining multi-view plausibility [48]. These approaches established the basic edit-reconstruct paradigm. However, the reliance of NeRF-based methods on volumetric optimization makes local control and rapid iteration computationally expensive [49].

In order to address the shortcomings of implicit representations, recent advancements

have increasingly leveraged explicit structures, primarily 3D Gaussian Splatting. Methods such as GaussianEditor [50] and InFusion [51] utilize these explicit representations in order to enable highly localized operations within a scene. In facilitating the direct manipulation of individual 3D Gaussian primitives, these frameworks achieve significant improvements in update rates and finer spatial control compared to their NeRF-based predecessors [1].

A critical and highly complex sub-task within 3D scene modification is inpainting, which aims to synthesize both geometry and textures to complete missing or occluded regions. Recent works enable semantic and instance-level manipulation by associating features with discrete primitives [3], [44], [45]. These methods support real-time editing and segmentation-driven operations, but primarily focus on appearance modification or object removal.

2.4 Neural Representations for Physical Simulation

Initial efforts to bring neural rendering into real-time, interactive applications exposed critical computational bottlenecks. While dynamic formations [36], [37] excel at kinematic replay or interpolated deformation [12], they fundamentally lack an understanding of physical laws, treating motion as a purely visual phenomenon. Recent frameworks leverage 3DGS for physical simulation by embedding kinematic properties directly into discrete kernels [52]–[54]. Coupled with particle-based simulators like the Material Point Method (MPM), 3DGS natively excels at modeling continuous materials, elastoplastics, and fluid dynamics. Yet, these point-based approaches introduce a severe computational bottleneck for rigid-body mechanics. Integrating consistent non-overlap constraints across millions of Gaussians inherently becomes extremely computationally intensive and as a result often yields numerical instability and "spongy" collision artifacts that break the perceptual illusion of solidity [55].

2.5 Interactive Neural Scenes in Virtual Reality

Transitioning novel-view synthesis from desktop environments to fully immersive six-degree-of-freedom (6-DOF) Head-Mounted Displays (HMDs) imposts stringent systems-level constraints. Immersive VR demands sustained high framerates with a recommended minimum framerate of $\geqslant$ 90 FPS for stereoscopic rendering and ultra-low motion-to-photon latency in order to mitigate cybersickness [56]. The explicit rasterization pipeline of 3DGS natively satisfies these rendering constraints, thereby unlocking the potential for photorealistic, room-scale VR [1].

Consequently, recent works within the field have aimed to construct end-to-end interactive VR environments using neural representations. Frameworks such as VR-GS [57] and LIVE-GS [58] have successfully deployed 3DGS to HMDs, enabling users to navigate reconstructed spaces and apply localized, soft-body deformations. However, these systems treat the scene as a continuous, deformable medium rather than a collection of discrete, actionable assets.

This architectural limitation precipitates significant user-experience failures during standard VR manipulation tasks. When a user physically grasps, extracts, or translates an object within existing neural VR systems, several critical breakdowns occur. Structurally, the manipulated object lacks a rigid boundary representation, rendering classical VR mechanics (e.g. rigid resting, stacking, or complex multi-body collisions) impossible [59]. Visually, the extraction of the object invariably exposes untreated background regions, leaving severe photometric artifacts and unstructured noise that fracture immersive presence. In this work, we propose a practical framework for deploying neural rendering-quality representations into fully interactive, rigid-body VR simulations, ensuring both visual integrity and physical plausibility during complex 6-DOF manipulation.

3. Methodology

Figure 3.1 summarizes the end-to-end pipeline, while Figure 3.2 shows the implementation-level data flow. The method consists of five main stages: scene-level 3D Gaussian Splatting reconstruction, geometry-guided multiview mask generation, Gaussian partitioning into object-centric assets, conservative object completion, and proxy mesh generation for VR interaction. The global 3DGS reconstruction provides the photorealistic scene representation, while the later stages impose object structure and physical affordances.

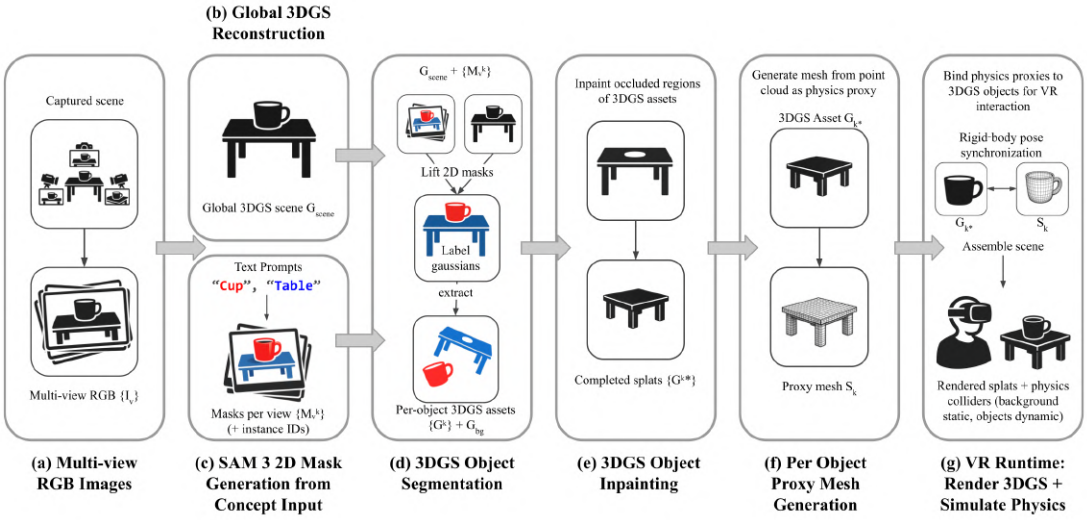

FIGURE 3.1: **Conceptual end-to-end system pipeline.** Multiview RGB images are reconstructed as a global 3D Gaussian scene. Concept-driven 2D masks are lifted into 3D to extract object-centric Gaussian assets. Missing object geometry is conservatively completed, converted into mesh proxies, and assembled in VR, where Gaussians provide photorealistic rendering and meshes provide collision and manipulation behavior.

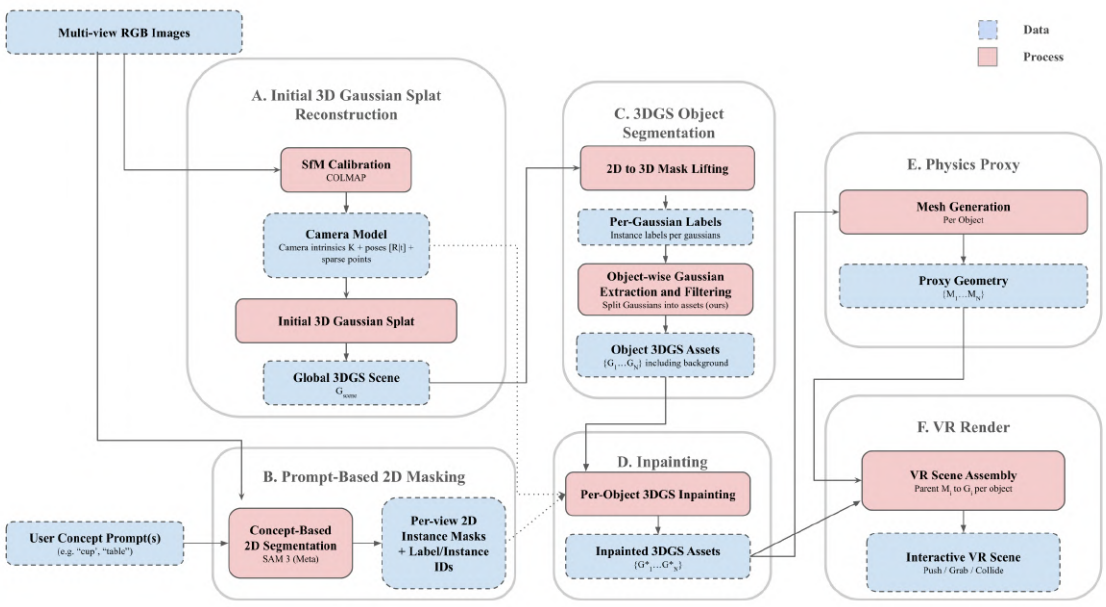

FIGURE 3.2**: Implementation data flow.** Blue nodes denote data products and red nodes denote processing stages. The pipeline estimates camera geometry, reconstructs a global 3DGS scene, generates concept-driven masks, partitions Gaussians into object assets, completes missing geometry, creates mesh proxies, and assembles the final VR scene.

3.1 Data Flow and Scene Representation

The input is a set of calibrated multiview RGB images. Camera geometry and sparse 3D structure are estimated using COLMAP, after which the scene is reconstructed as a global 3D Gaussian Splatting representation. Each Gaussian stores its 3D position, anisotropic shape, opacity, and view-dependent appearance parameters.

The user provides foreground object specifications. Each specification contains a text prompt and an anchor image in which the target object is visible. Label 0 is reserved for the background, and foreground objects are assigned labels from 1 to $K-1$. The pipeline outputs object-wise Gaussian assets, completed Gaussian assets, and foreground mesh proxies used for collision and rigid-body simulation.

3.2 Geometry-Guided Multiview Mask Generation

The first object-centric stage generates per-view object masks from user prompts. For each foreground object, we query a promptable segmentation model, SAM3 in our implementation [43], using the object text prompt and its anchor image. Because text prompting can return multiple plausible masks, we use sparse COLMAP geometry to select the candidate that best aligns with reconstructed 3D evidence.

When an object has sufficient COLMAP support, the selected anchor mask is the semantically plausible candidate with the strongest projected 3D point support relative to its size. This discourages selecting overly large regions while favoring masks that explain actual scene structure. For reflective, transparent, textureless, or otherwise weakly reconstructed objects, we fall back to the highest-confidence semantic mask and defer geometric support updates until later views provide more reliable evidence.

Masks are then propagated across views. Candidate masks in each view are evaluated using spatial continuity with the previously accepted mask, area consistency, segmentor confidence, and geometric support from visible COLMAP or persistent object points. A candidate is accepted when it satisfies the geometric support requirement. If no candidate meets this requirement, a fallback mask may be accepted only when it remains semantically plausible and spatially continuous. Such fallback masks do not update the persistent support set, preventing unsupported semantic drift.

Objects are propagated independently and may overlap in image space. Overlapping pixels are assigned to the object with stronger normalized geometric support, with segmentor confidence used as a tie-breaker. This produces a zero-based instance label map for each view.

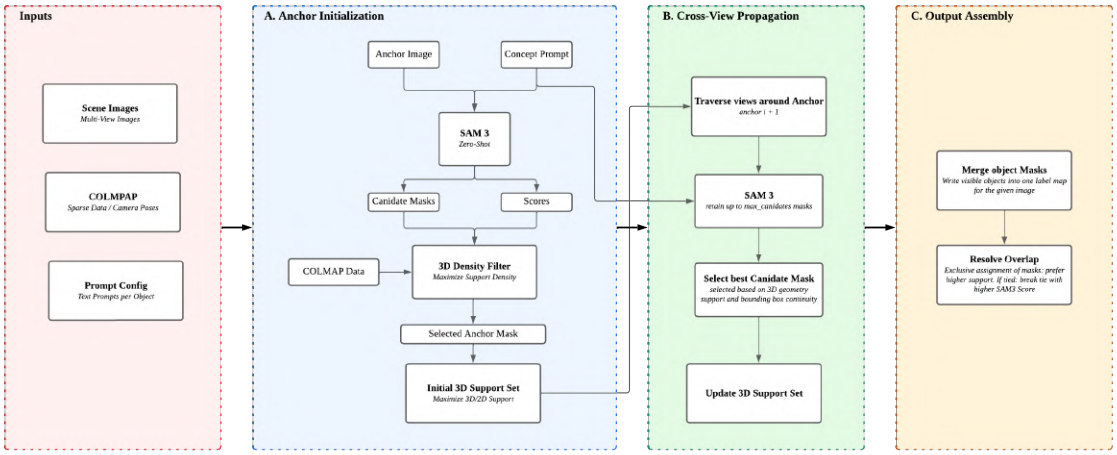

FIGURE 3.3**: Geometry-guided multiview mask generation.** This stage converts sparse user supervision into consistent per-view object labels. Instead of relying on semantic segmentation confidence alone, candidate masks are anchored and propagated using reconstructed 3D evidence, reducing instance ambiguity and cross-view drift before the masks are merged into label maps for downstream Gaussian partitioning.

3.3 Gaussian Partitioning by Masked Evidence

Given the global 3DGS scene and the multiview label maps, we assign Gaussians to object instances using masked rendering evidence. Each Gaussian is rendered into the training views, and its contribution to pixels labeled as each object is accumulated. This produces a per-Gaussian evidence score for each label.

A Gaussian is assigned to a foreground object only when its strongest foreground evidence exceeds a threshold. Ambiguous or weakly supported Gaussians are assigned to the background, which absorbs floaters and uncertain regions. The scene is then partitioned into object-wise Gaussian subsets. This produces separable object assets but does not yet synthesize missing geometry.

3.4 Conservative Multiview Object Completion

Object-wise Gaussian assets may be incomplete because occluded regions, backsides, and thin structures are poorly observed in the original capture. We address this using

a conservative render-and-verify completion procedure. The goal is not to hallucinate arbitrary geometry, but to add missing structure only where it improves object coverage without damaging already observed regions.

For each object, we render the isolated Gaussian asset from the training cameras and compare the rendered disparity support with the corresponding 2D object masks. Pixels labeled as the object but not explained by the current render are treated as holes. Completion is restricted to these masked holes, so the inpainting model cannot modify unrelated image regions.

Raw hole masks are refined using simple morphological operations. The resulting editable region is passed to the inpainting model, while confidently rendered pixels outside this region are protected and later used to verify that the update does not degrade existing content.

Views are selected for completion based on hole size, visible object area, boundary support, and overall visibility. Views with insufficient support or unreliable hole regions are discarded. For the remaining views, we adapt the depth-guided inpainting strategy of InFusion [60]. The model is conditioned on the editable mask, rendered disparity, camera parameters, and RGB context. Foreground objects use the original image as context, while background completion uses a hole-filled render to avoid leaking foreground appearance into the background asset.

Each selected view produces supplemental 3D points. These points are filtered to remove invalid samples, outliers, duplicates near existing geometry, and weakly supported single-view artifacts. Retained points are converted into additional Gaussians using local color, spacing, opacity, and orientation estimates.

After each candidate update, the object is re-rendered from all views and the hole statistics are recomputed. The update is accepted only if it reduces the mean hole ratio and satisfies conservative limits on added geometry, single-view support, and drift in protected RGB, depth, and coverage regions. This verification step favors stable partial completion over aggressive hallucination.

3.5 Proxy Mesh Generation

The completed Gaussian asset is converted into a mesh proxy for physics simulation. This proxy is not used for photorealistic rendering; it provides a compact collision representation that can be manipulated by the VR physics engine.

For each foreground object, we construct a backend input from the completed Gaussian point cloud and camera metadata. We then use the Gaussian-to-mesh extraction approach implemented in GauStudio [61] to obtain a reference mesh. The mesh is repaired using standard cleanup operations such as duplicate-vertex removal, optional hole filling, and connected-component filtering.

Because the reference mesh may be too detailed or topologically unstable for rigid-body simulation, we also generate simplified proxy candidates. These are created by voxelizing the reference mesh at multiple pitch scales, converting the voxel grids back into meshes, and applying the same repair operations. The candidate set therefore includes both the repaired reference mesh and several regularized voxel proxies.

Candidates are evaluated in held-out views where the object is visible. Each mesh is rasterized, and its silhouette is compared against the corresponding object mask. We prioritize candidates with high silhouette recall, since the proxy must cover the physical extent of the object. Among candidates with similar recall, we prefer higher IoU, watertightness, fewer boundary edges, a dominant connected component, and lower triangle count. The selected mesh is used as the object's invisible rigid-body proxy.

3.6 VR Scene Assembly and Runtime Interaction

The final VR scene pairs each completed Gaussian asset with its corresponding mesh proxy. The Gaussian asset remains visible and provides photorealistic appearance, while the mesh proxy remains invisible and provides collision, grabbing, and rigid-body behavior in the game engine. This separation allows the scene to preserve the visual fidelity of 3D Gaussian Splatting while supporting interactive object manipulation in VR.

4. Results

This chapter evaluates the asset-generation components of the proposed object-centric 3D Gaussian Splatting (3DGS) pipeline: prompt-conditioned segmentation, object-wise Gaussian decomposition, conservative completion, and collision-oriented proxy generation. The evaluation is stage-wise because no single benchmark provides ground truth for open-vocabulary 3D segmentation, object completion, and collision-proxy geometry.

4.1 Datasets and Evaluation Protocol

For open-vocabulary segmentation and object-centric Gaussian extraction, we use the LERF-Mask benchmark [46], evaluating on the *ramen*, *teatime*, and *figurines* scenes. These scenes contain cluttered tabletop layouts, small objects, partial occlusions, repeated structures, and concept-level prompts. For collision-oriented mesh evaluation, we use a 15-scan subset of DTU [74], which provides reference geometry for measuring compact proxy meshes.

For segmentation, we report scene-level mean Intersection over Union (mIoU), following the Gaussian Grouping protocol [3]. For decomposition, we report foreground and background Gaussian counts, number of exported foreground assets, and export time. For completion, we report target-region hole reduction where available. For proxy geometry, we report Chamfer Distance (CD), F-score (F1), and Hausdorff Distance (HD), with lower better for CD and HD and higher better for F1.

4.2 Open-Vocabulary Segmentation and Object-Centric Gaussian Extraction

Table 4.1 reports LERF-Mask mIoU for the three evaluated scenes. Our method obtains an average mIoU of 63.0. The strongest result occurs on *figurines*, where the method

Table 4.1: Open-vocabulary segmentation on LERF-Mask. Values are scene-level mIoU.

Scene	LERF [46]	SA3D [75]	LangSplat [4]	Gaussian Grouping [3]	Ours
ramen	28.2	7.8	50.8	**77.0**	62.7
teatime	47.4	42.5	48.8	**71.7**	46.3
figurines	36.1	24.9	67.3	69.7	**79.9**
Average	37.2	25.1	55.6	**72.8**	63.0

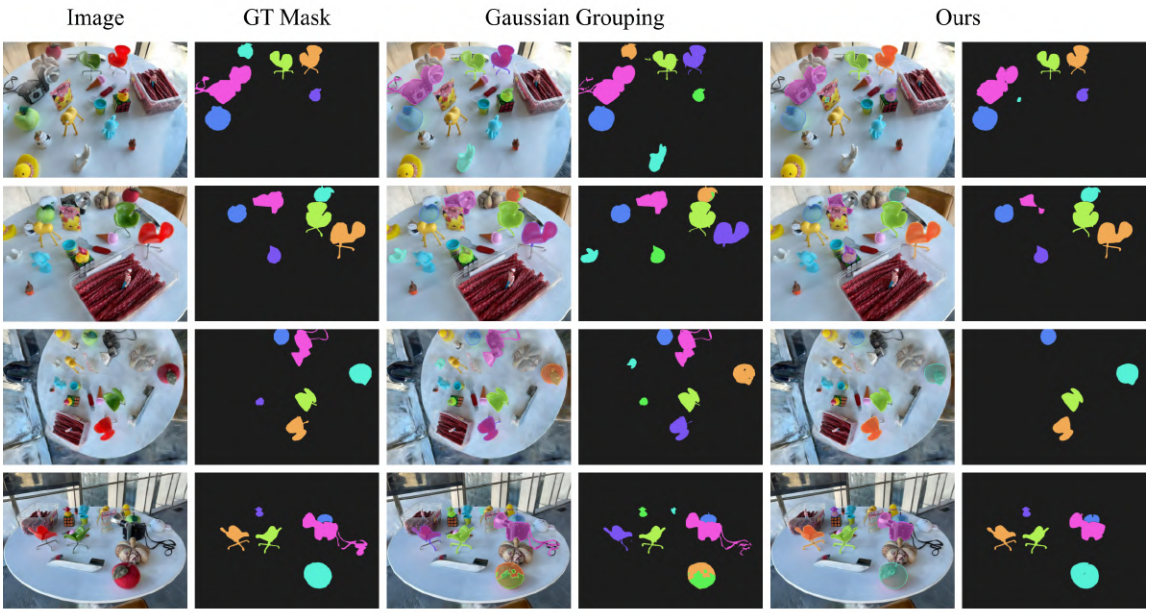

FIGURE 4.1: **Qualitative 2D segmentation results on the *Figurines* scene from the LERF-Mask dataset.** Each row shows a different evaluation view. Columns show the input image, ground-truth instance masks, Gaussian Grouping predictions, and our prompt-conditioned geometry-guided segmentation results.

reaches 79.9 mIoU, outperforming Gaussian Grouping by 10.2 points and LangSplat by 12.6 points. The method underperforms Gaussian Grouping on *ramen* and *teatime*, indicating that ambiguous prompts, color-dominated concepts, and visually entangled objects remain challenging.

Figure 4.1 provides qualitative context for the *figurines* result. The method produces concept-aligned masks for several small foreground objects, while visually similar or partially occluded instances remain difficult.

Table 4.2 summarizes the resulting Gaussian decomposition. Across all three scenes,

Table 4.2: Object-centric Gaussian decomposition on the primary LERF-Mask scenes. Foreground object count excludes the background asset.

Scene	Total Gaussians	Background	FG Objects	FG Gaussians	FG Fraction	Export
teatime	2,250,850	2,202,019	5	48,831	2.2%	1.33 s
ramen	758,947	641,161	6	117,786	15.5%	0.47 s
figurines	1,929,139	1,876,328	6	52,811	2.7%	1.17 s

the pipeline separates a large static background from a small set of foreground assets. Foreground objects account for 2.2% of Gaussians in *teatime*, 15.5% in *ramen*, and 2.7% in *figurines*; all selected foreground assets are exported in less than 1.5 seconds.

Figure 4.2 shows that the 2D masks can be lifted into persistent object-specific Gaussian assets. The extracted objects remain recognizable from multiple viewpoints, but some contain missing backsides, boundary artifacts, or residual ghosting, especially near occlusions and visually entangled regions.

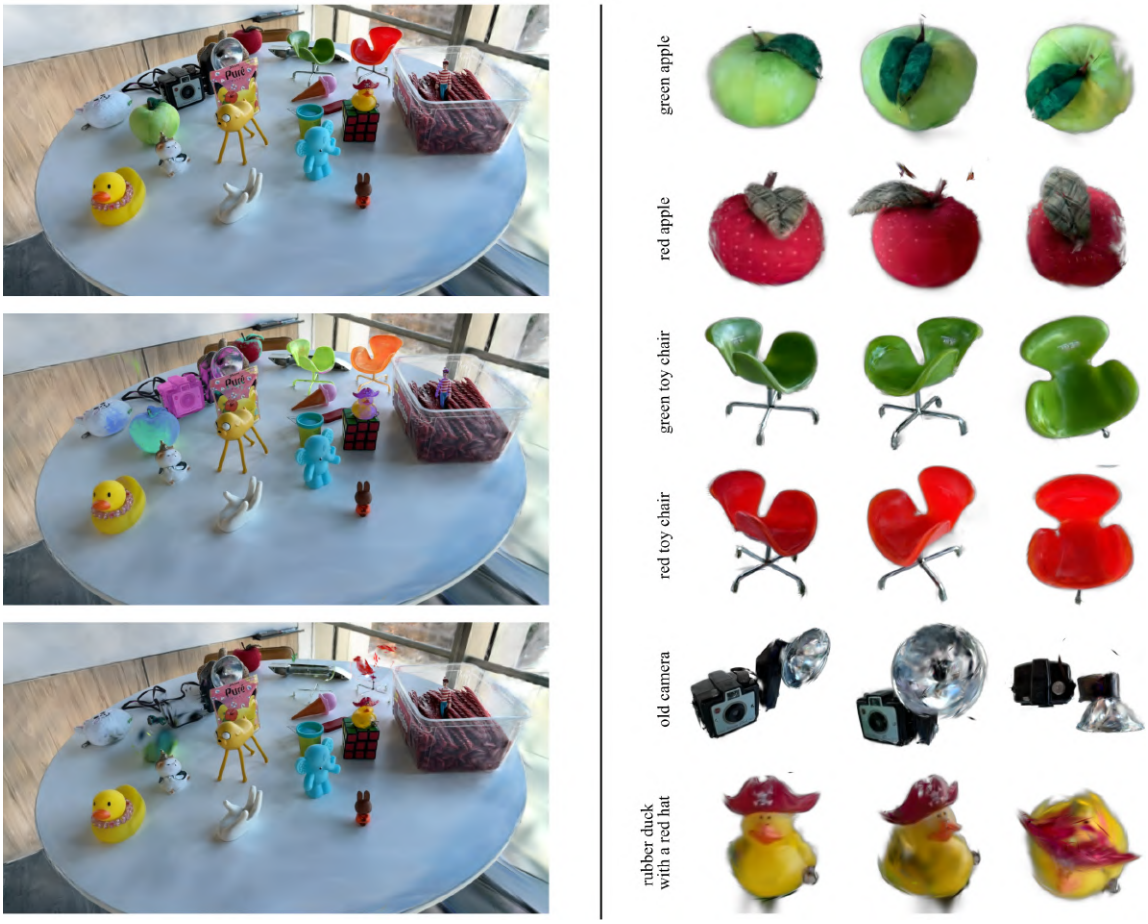

FIGURE 4.2: **Qualitative Stage 3 object-centric 3DGS decomposition on the *Figurines* scene from the LERF-Mask dataset.** Left: three renders from the same camera view showing, from top to bottom, the original full-scene 3DGS reconstruction; the corresponding 3D Gaussian object assignment visualization, where foreground object labels are shown as semi-transparent color overlays; and the residual background-only 3DGS after removing the segmented foreground objects. Right: independently extracted object-centric Gaussian assets rendered from multiple viewpoints and grouped by their original concept prompts.

4.3 Completion for Mesh Readiness

Object-centric Gaussian assets are often incomplete after hard partitioning because occluded backsides, contact regions, and thin structures may be absent from the original monolithic 3DGS. We therefore evaluate completion as a conservative mesh-readiness step rather than as unconstrained generative inpainting.

Table 4.3 reports the available aggregate hole accounting for *teatime*. Completion re-

Table 4.3: Completion summary for the *teatime* scene. Holes are counted over target object regions across the evaluated views.

Scene	Holes Before	Holes After	Reduction	Added Gaussians
teatime	1,378,616	600,702	56.4%	234

duces target-region holes from 1,378,616 pixels to 600,702 pixels, a 56.4% reduction, while adding 234 Gaussians. The same acceptance policy adds 11 Gaussians in *ramen* and 43 in *figurines*, but comparable aggregate hole accounting is not available for those scenes.

These results support a narrow interpretation: completion can reduce locally constrained missing regions, but it does not recover semantic object parts that were never assigned during segmentation.

4.4 Collision-Oriented Mesh Generation

The final geometric stage converts object-centric Gaussian assets into compact mesh proxies for downstream collision use. Table 4.4 compares representative proxy choices on 15 DTU scans.

Bounding boxes and convex hulls are compact but too coarse for accurate object-level geometry. The selected voxel proxy achieves CD 5.767, F1 0.977, and HD 68.318 at 2554.3 faces on average. Relative to the convex hull, it reduces CD by 81.0%, reduces HD by 42.4%, and improves F1 by 160.5%.

The face-budget comparison shows a practical knee near 2.5k faces. Increasing to the Voxel-4200 setting improves CD from 5.767 to 5.719 and F1 from 0.977 to 0.978, but increases the average face count by more than 50%. The selected proxy therefore provides the stronger accuracy–complexity balance. The broader mesh sweep also shows that smoothing is harmful for collision-oriented fidelity: across 85 matched voxel configurations, two smoothing iterations increase CD by 38.9% and HD by 68.5%, while reducing F1 by 15.8%.

Table 4.4: Collision-proxy accuracy–complexity trade-off on 15 DTU scans. Values are means. Lower is better for CD and HD; higher is better for F1.

Proxy	Faces	CD ↓	F1 ↑	HD ↓
BBox	12.0	118.874	0.003	304.566
Convex hull	308.3	30.344	0.375	118.599
Voxel-2800 selected	2554.3	5.767	0.977	68.318
Voxel-4200	3949.8	**5.719**	**0.978**	**61.335**

4.5 Discussion

The results show that prompt-conditioned 2D masks can be lifted into 3D Gaussian object assignments, producing recognizable foreground assets and a complementary background. The strongest segmentation result occurs on *figurines*, where the objects are visually distinct and well suited to concept-level prompts. The weaker results on *ramen* and *teatime* show that ambiguous prompts, repeated instances, and color-based distinctions remain major sources of error.

The completion and proxy results show that downstream stages can improve asset readiness but cannot fully correct front-end mistakes. Completion reduces locally constrained holes, while voxel-derived proxies provide substantially better geometry than bounding boxes or convex hulls. However, if segmentation omits an object part, later stages generally cannot reconstruct it reliably.

The current evaluation is limited by its stage-wise structure. Existing datasets do not jointly provide ground truth for text-conditioned 3D segmentation, object completion, and collision-oriented proxy generation. The system also generates geometry but does not estimate physical material properties such as mass, friction, restitution, or compliance.

4.6 Conclusion

The results support three conclusions: prompted multiview segmentation can produce independent 3DGS object assets; conservative completion improves mesh readiness when missing regions are locally constrained; and compact voxel-derived proxies substantially

outperform bounding boxes and convex hulls while remaining near 2.5k faces on average. Overall, object-wise decomposition of 3DGS scenes provides a practical foundation for scene-native asset generation. The approach preserves the visual representation of Gaussian splatting while exposing selected foreground objects as persistent assets with associated proxy geometry.

4.7 Future Work

Future work should prioritize more robust segmentation and Gaussian assignment, since most downstream errors originate from ambiguous or inconsistent object masks. Stronger 3D-consistent language grounding, better instance tracking, and tighter coupling between 2D confidence and 3D evidence would improve asset reliability.

A second direction is more principled object completion. The current completion stage avoids aggressive hallucination but limits recovery under severe occlusion. Category-aware or uncertainty-aware completion could distinguish reliable geometry from speculative geometry.

Future systems should also estimate physical properties such as mass, friction, restitution, density, or material class, and should extend beyond rigid objects to articulated or deformable assets. Finally, unified benchmarks are needed to evaluate text-conditioned 3D segmentation, object-centric decomposition, completion, and proxy generation within a single end-to-end protocol.

Appendix A.

A.1 Hardware and Software Environment

All reported experiments were run on a workstation with an NVIDIA RTX 3090 GPU with 24 GB of VRAM, an AMD Ryzen 9 3950X CPU with 16 cores, 64 GB of system memory, and NVMe solid-state storage. The software stack used CUDA 12.4, PyTorch 2.5.1, a FlashSplat-based 3DGS training and rendering backend, and a SAM3 inference wrapper integrated with COLMAP camera geometry.

Table A.1: Hardware and software configuration used for the reported experiments.

Component	Configuration
GPU	NVIDIA RTX 3090, 24 GB VRAM
CPU	AMD Ryzen 9 3950X, 16 cores
System memory	64 GB RAM
Storage	NVMe SSD
CUDA	12.4
PyTorch	2.5.1
3DGS backend	FlashSplat-based training/rendering
2D Segmentation	SAM3 wrapper with COLMAP integration

Bibliography

[1] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, *3d gaussian splatting for real-time radiance field rendering*, 2023. arXiv: 2308.04079 [cs.GR]. [Online]. Available: https://arxiv.org/abs/2308.04079.

[2] J. Cen, J. Fang, C. Yang, L. Xie, X. Zhang, W. Shen, and Q. Tian, *Segment any 3d gaussians*, 2025. arXiv: 2312.00860 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2312.00860.

[3] M. Ye, M. Danelljan, F. Yu, and L. Ke, *Gaussian grouping: Segment and edit anything in 3d scenes*, 2024. arXiv: 2312.00732 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2312.00732.

[4] M. Qin, W. Li, J. Zhou, H. Wang, and H. Pfister, *Langsplat: 3d language gaussian splatting*, 2024. arXiv: 2312.16084 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2312.16084.

[5] D. A. Bowman and L. F. Hodges, "An evaluation of techniques for grabbing and manipulating remote objects in immersive virtual environments," in *Proceedings of the 1997 Symposium on Interactive 3D Graphics*, ser. I3D '97, Providence, Rhode Island, USA: Association for Computing Machinery, 1997, 35–ff. ISBN: 0897918843. DOI: 10.1145/253284.253301. [Online]. Available: https://doi.org/10.1145/253284.253301.

[6] A. Helgert, A. Groeneveld, and S. C. Eimler, "A qualitative analysis of interaction techniques in a virtual reality instruction environment: Experiences from a case study," in *2022 IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR)*, 2022, pp. 171–175. DOI: 10.1109/AIVR56993.2022.00034.

[7] H. Kato, Y. Ushiku, and T. Harada, *Neural 3d mesh renderer*, 2017. arXiv: 1711.07566 [cs.CV]. [Online]. Available: https://arxiv.org/abs/1711.07566.

[8] D. Maturana and S. A. Scherer, "Voxnet: A 3d convolutional neural network for real-time object recognition," *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 922–928, 2015. [Online]. Available: https://api.semanticscholar.org/CorpusID:14620252.

[9] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger, *Occupancy networks: Learning 3d reconstruction in function space*, 2019. arXiv: 1812.03828 [cs.CV]. [Online]. Available: https://arxiv.org/abs/1812.03828.

[10] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, *Nerf: Representing scenes as neural radiance fields for view synthesis*, 2020. arXiv: 2003.08934 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2003.08934.

[11] R. Martin-Brualla, N. Radwan, M. S. M. Sajjadi, J. T. Barron, A. Dosovitskiy, and D. Duckworth, *Nerf in the wild: Neural radiance fields for unconstrained photo collections*, 2021. arXiv: 2008.02268 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2008.02268.

[12] A. Pumarola, E. Corona, G. Pons-Moll, and F. Moreno-Noguer, "D-nerf: Neural radiance fields for dynamic scenes," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. [Online]. Available: http://arxiv.org/abs/2011.13961v1.

[13] E. Tretschk, A. Tewari, V. Golyanik, M. Zollhöfer, C. Lassner, and C. Theobalt, *Non-rigid neural radiance fields: Reconstruction and novel view synthesis of a dynamic scene from monocular video*, 2021. arXiv: 2012.12247 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2012.12247.

[14] P. Wang, L. Liu, Y. Liu, C. Theobalt, T. Komura, and W. Wang, *Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction*, 2023. arXiv: 2106.10689 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2106.10689.

[15] X. Li, Y.-L. Qiao, P. Y. Chen, K. M. Jatavallabhula, M. Lin, C. Jiang, and C. Gan, *Pac-nerf: Physics augmented continuum neural radiance fields for geometry-agnostic system identification*, 2023. arXiv: 2303.05512 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2303.05512.

[16] L. Liu, J. Gu, K. Z. Lin, T.-S. Chua, and C. Theobalt, *Neural sparse voxel fields*, 2021. arXiv: 2007.11571 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2007.11571.

[17] A. Chen, Z. Xu, A. Geiger, J. Yu, and H. Su, *Tensorf: Tensorial radiance fields*, 2022. arXiv: 2203.09517 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2203.09517.

[18] T. Müller, A. Evans, C. Schied, and A. Keller, "Instant neural graphics primitives with a multiresolution hash encoding," *ACM Trans. Graph.*, vol. 41, no. 4, Jul. 2022, ISSN: 0730-0301. DOI: 10.1145/3528223.3530127. [Online]. Available: https://doi.org/10.1145/3528223.3530127.

[19] C. Sun, M. Sun, and H.-T. Chen, *Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction*, 2022. arXiv: 2111.11215 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2111.11215.

[20] T. Neff, P. Stadlbauer, M. Parger, A. Kurz, J. H. Mueller, C. R. A. Chaitanya, A. Kaplanyan, and M. Steinberger, "Donerf: Towards real-time rendering of compact neural radiance fields using depth oracle networks," *Computer Graphics Forum*, vol. 40,

no. 4, pp. 45–59, Jul. 2021, ISSN: 1467-8659. DOI: 10.1111/cgf.14340. [Online]. Available: http://dx.doi.org/10.1111/cgf.14340.

[21] A. Yu, R. Li, M. Tancik, H. Li, R. Ng, and A. Kanazawa, *Plenoctrees for real-time rendering of neural radiance fields*, 2021. arXiv: 2103.14024 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2103.14024.

[22] P. Hedman, P. P. Srinivasan, B. Mildenhall, J. T. Barron, and P. Debevec, *Baking neural radiance fields for real-time view synthesis*, 2021. arXiv: 2103.14645 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2103.14645.

[23] A. Yu, S. Fridovich-Keil, M. Tancik, Q. Chen, B. Recht, and A. Kanazawa, *Plenoxels: Radiance fields without neural networks*, 2021. arXiv: 2112.05131 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2112.05131.

[24] J. T. Barron, B. Mildenhall, M. Tancik, P. Hedman, R. Martin-Brualla, and P. P. Srinivasan, *Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields*, 2021. arXiv: 2103.13415 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2103.13415.

[25] J. T. Barron, B. Mildenhall, D. Verbin, P. P. Srinivasan, and P. Hedman, *Mip-nerf 360: Unbounded anti-aliased neural radiance fields*, 2022. arXiv: 2111.12077 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2111.12077.

[26] Z. Yu, A. Chen, B. Huang, T. Sattler, and A. Geiger, *Mip-splatting: Alias-free 3d gaussian splatting*, 2023. arXiv: 2311.16493 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2311.16493.

[27] S. S. Mallick, R. Goel, B. Kerbl, M. Steinberger, F. V. Carrasco, and F. De La Torre, "Taming 3dgs: High-quality radiance fields with limited resources," in *SIGGRAPH Asia 2024 Conference Papers*, ser. SA '24, Tokyo, Japan: Association for Computing Machinery, 2024, ISBN: 9798400711312. DOI: 10.1145/3680528.3687694. [Online]. Available: https://doi.org/10.1145/3680528.3687694.

[28] Y. Seo, Y. S. Choi, H. Son, and Y. Uh, "Flod: Integrating flexible level of detail into 3d gaussian splatting for customizable rendering," *ACM Trans. Graph.*, vol. 44, no. 4, Jul. 2025, ISSN: 0730-0301. DOI: 10.1145/3731430. [Online]. Available: https://doi.org/10.1145/3731430.

[29] B. Kerbl, A. Meuleman, G. Kopanas, M. Wimmer, A. Lanvin, and G. Drettakis, *A hierarchical 3d gaussian representation for real-time rendering of very large datasets*, 2024. arXiv: 2406.12080 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2406.12080.

[30] N. Moenne-Loccoz, A. Mirzaei, O. Perel, R. de Lutio, J. M. Esturo, G. State, S. Fidler, N. Sharp, and Z. Gojcic, *3d gaussian ray tracing: Fast tracing of particle scenes*, 2024. arXiv: 2407.07090 [cs.GR]. [Online]. Available: https://arxiv.org/abs/2407.07090.

[31] Q. Wu, J. M. Esturo, A. Mirzaei, N. Moenne-Loccoz, and Z. Gojcic, *3dgut: Enabling distorted cameras and secondary rays in gaussian splatting*, 2025. arXiv: 2412.12507 [cs.GR]. [Online]. Available: https://arxiv.org/abs/2412.12507.

[32] Z. Chen, F. Wang, Y. Wang, and H. Liu, "Text-to-3d using gaussian splatting," in *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024, pp. 21 401–21 412. DOI: 10.1109/CVPR52733.2024.02022.

[33] B. Xiong, J. Liu, J. Hu, C. Wu, J. Wu, X. Liu, C. Zhao, E. Ding, and Z. Lian, "Texgaussian: Generating high-quality pbr material via octree-based 3d gaussian splatting," in *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025, pp. 551–561. DOI: 10.1109/CVPR52734.2025.00060.

[34] J. Tang, J. Ren, H. Zhou, Z. Liu, and G. Zeng, *Dreamgaussian: Generative gaussian splatting for efficient 3d content creation*, 2024. arXiv: 2309.16653 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2309.16653.

[35] J. Chung, S. Lee, H. Nam, J. Lee, and K. M. Lee, "Luciddreamer: Domain-free generation of 3d gaussian splatting scenes," *IEEE Transactions on Visualization and Computer Graphics*, vol. 31, no. 12, pp. 10 640–10 651, 2025. DOI: 10.1109/TVCG.2025.3611489.

[36] J. Luiten, G. Kopanas, B. Leibe, and D. Ramanan, *Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis*, 2023. arXiv: 2308.09713 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2308.09713.

[37] G. Wu, T. Yi, J. Fang, L. Xie, X. Zhang, W. Wei, W. Liu, Q. Tian, and X. Wang, *4d gaussian splatting for real-time dynamic scene rendering*, 2024. arXiv: 2310.08528 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2310.08528.

[38] Z. Yang, X. Gao, W. Zhou, S. Jiao, Y. Zhang, and X. Jin, *Deformable 3d gaussians for high-fidelity monocular dynamic scene reconstruction*, 2023. arXiv: 2309.13101 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2309.13101.

[39] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, *Pointnet++: Deep hierarchical feature learning on point sets in a metric space*, 2017. arXiv: 1706.02413 [cs.CV]. [Online]. Available: https://arxiv.org/abs/1706.02413.

[40] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, *Learning transferable visual models from natural language supervision*, 2021. arXiv: 2103.00020 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2103.00020.

[41] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollár, and R. Girshick, *Segment anything*, 2023. arXiv: 2304.02643 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2304.02643.

[42] N. Ravi, V. Gabeur, Y.-T. Hu, R. Hu, C. Ryali, T. Ma, H. Khedr, R. Rädle, C. Rolland, L. Gustafson, E. Mintun, J. Pan, K. V. Alwala, N. Carion, C.-Y. Wu, R. Girshick, P. Dollár, and C. Feichtenhofer, *Sam 2: Segment anything in images and videos*, 2024. arXiv: 2408.00714 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2408.00714.

[43] N. Carion, L. Gustafson, Y.-T. Hu, S. Debnath, R. Hu, D. Suris, C. Ryali, K. V. Alwala, H. Khedr, A. Huang, J. Lei, T. Ma, B. Guo, A. Kalla, M. Marks, J. Greer, M. Wang, P. Sun, R. Rädle, T. Afouras, E. Mavroudi, K. Xu, T.-H. Wu, Y. Zhou, L. Momeni, R. Hazra, S. Ding, S. Vaze, F. Porcher, F. Li, S. Li, A. Kamath, H. K. Cheng, P. Dollár, N. Ravi, K. Saenko, P. Zhang, and C. Feichtenhofer, *Sam 3: Segment anything with concepts*, 2025. arXiv: 2511.16719 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2511.16719.

[44] S. Zhou, H. Chang, S. Jiang, Z. Fan, Z. Zhu, D. Xu, P. Chari, S. You, Z. Wang, and A. Kadambi, *Feature 3dgs: Supercharging 3d gaussian splatting to enable distilled feature fields*, 2024. arXiv: 2312.03203 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2312.03203.

[45] X. Zuo, P. Samangouei, Y. Zhou, Y. Di, and M. Li, *Fmgs: Foundation model embedded 3d gaussian splatting for holistic 3d scene understanding*, 2024. arXiv: 2401.01970 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2401.01970.

[46] A. Mirzaei, T. Aumentado-Armstrong, K. G. Derpanis, J. Kelly, M. A. Brubaker, I. Gilitschenski, and A. Levinshtein, *Spin-nerf: Multiview segmentation and perceptual inpainting with neural radiance fields*, 2023. arXiv: 2211.12254 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2211.12254.

[47] E. Weber, A. Hołyński, V. Jampani, S. Saxena, N. Snavely, A. Kar, and A. Kanazawa, *Nerfiller: Completing scenes via generative 3d inpainting*, 2023. arXiv: 2312.04560 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2312.04560.

[48] Y. Yin, Z. Fu, F. Yang, and G. Lin, *Or-nerf: Object removing from 3d scenes guided by multiview segmentation with neural radiance fields*, 2023. arXiv: 2305.10503 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2305.10503.

[49] G. Chen and W. Wang, *A survey on 3d gaussian splatting*, 2025. arXiv: 2401.03890 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2401.03890.

[50] Y. Chen, Z. Chen, C. Zhang, F. Wang, X. Yang, Y. Wang, Z. Cai, L. Yang, H. Liu, and G. Lin, *Gaussianeditor: Swift and controllable 3d editing with gaussian splatting*, 2023. arXiv: 2311.14521 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2311.14521.

[51] Z. Liu, H. Ouyang, Q. Wang, K. L. Cheng, J. Xiao, K. Zhu, N. Xue, Y. Liu, Y. Shen, and Y. Cao, *Infusion: Inpainting 3d gaussians via learning depth completion from diffusion prior*, 2024. arXiv: 2404.11613 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2404.11613.

[52] T. Xie, Z. Zong, Y. Qiu, X. Li, Y. Feng, Y. Yang, and C. Jiang, *Physgaussian: Physics-integrated 3d gaussians for generative dynamics*, 2024. arXiv: 2311.12198 [cs.GR]. [Online]. Available: https://arxiv.org/abs/2311.12198.

[53] P. Borycki, W. Smolak, J. Waczyńska, M. Mazur, S. Tadeja, and P. Spurek, *Gasp: Gaussian splatting for physic-based simulations*, 2025. arXiv: 2409.05819 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2409.05819.

[54] H. Zhao, H. Wang, X. Zhao, H. Fei, H. Wang, C. Long, and H. Zou, *Efficient physics simulation for 3d scenes via mllm-guided gaussian splatting*, 2025. arXiv: 2411.12789 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2411.12789.

[55] M. Müller, B. Heidelberger, M. Hennix, and J. Ratcliff, "Position based dynamics," *Journal of Visual Communication and Image Representation*, vol. 18, no. 2, pp. 109–118, 2007.

[56] J. Wang, R. Shi, W. Zheng, W. Xie, D. Kao, and H.-N. Liang, "Effect of frame rate on user experience, performance, and simulator sickness in virtual reality," *IEEE Transactions on Visualization and Computer Graphics*, vol. 29, no. 5, pp. 2478–2488, 2023. DOI: 10.1109/TVCG.2023.3247057.

[57] Y. Jiang, C. Yu, T. Xie, X. Li, Y. Feng, H. Wang, M. Li, H. Lau, F. Gao, Y. Yang, and C. Jiang, *Vr-gs: A physical dynamics-aware interactive gaussian splatting system in virtual reality*, 2024. arXiv: 2401.16663 [cs.HC]. [Online]. Available: https://arxiv.org/abs/2401.16663.

[58] J. Park, C. Park, M. Kim, M. Kim, and S. Kim, *Live-gs: Online lidar-inertial-visual state estimation and globally consistent mapping with 3d gaussian splatting*, 2026. arXiv: 2507.23273 [cs.RO]. [Online]. Available: https://arxiv.org/abs/2507.23273.

[59] D. A. Bowman, E. Kruijff, J. J. LaViola Jr, and I. Poupyrev, *3D User Interfaces: Theory and Practice*. Addison-Wesley Professional, 2004.

[60] Z. Liu, H. Ouyang, Q. Wang, K. L. Cheng, J. Xiao, K. Zhu, N. Xue, Y. Liu, Y. Shen, and Y. Cao, "Infusion: Inpainting 3d gaussians via learning depth completion from diffusion prior," *arXiv preprint arXiv:2404.11613*, 2024.

[61] C. Ye, Y. Nie, J. Chang, Y. Chen, Y. Zhi, and X. Han, *Gaustudio: A modular framework for 3d gaussian splatting and beyond*, 2024. arXiv: 2403.19632 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2403.19632.