Proceedings of CONF-CDS 2026 Symposium: Computer Vision-Based Multimodal Cognitive Load Estimation for Adaptive Media Communication
DOI: 10.54254/2755-2721/2026.GL35713

A Cost-Efficient Pipeline for Converting 3D Gaussian Splatting Representations into Simulation-Ready Meshes for NVIDIA Omniverse

Taolun Geng

College of Engineering, California Polytechnic State University, San Luis Obispo, USA
taolungeng@gmail.com

Abstract. Neural scene reconstruction has become an important tool for building digital environments used in robotics, autonomous systems, and physical AI training. However, NeRF-based reconstruction often requires high computational cost and does not directly produce simulation-ready mesh assets. 3D Gaussian Splatting offers a faster alternative by representing a captured scene as an explicit cloud of Gaussian primitives that can be rendered in real time. This paper proposes a practical pipeline that converts 3D Gaussian Splatting outputs into mesh-based assets and exports them into NVIDIA Omniverse and Isaac Sim workflows via OpenUSD. The proposed method extracts geometry from the Gaussian primitive cloud, reconstructs a watertight or simulation-usable mesh, bakes appearance information into textures and materials, and exports the result as an OpenUSD-compatible asset with physics and semantic metadata. The goal is to lower the cost of building realistic training environments while preserving visual and geometric fidelity sufficient for simulation, synthetic data generation, and robot learning. The paper also outlines a future extension toward live conversion, where streamed captures are incrementally transformed into simulation-ready scene updates.

Keywords: 3D Gaussian Splatting, mesh reconstruction, NVIDIA Omniverse, OpenUSD, Isaac Sim

1. Introduction

Digital simulation is becoming a central infrastructure for robotics, autonomous systems, and physical AI. Instead of collecting all training data in the real world, developers can create virtual environments, randomize lighting and object placement, generate synthetic annotations, and evaluate perception or control systems before real deployment. NVIDIA Isaac Sim is designed for robotics simulation, testing, and synthetic data generation in physically based virtual environments, and it is built on Omniverse libraries and OpenUSD workflows [1]. A major practical challenge is the cost of producing realistic simulation assets. Traditional manual modeling is slow, while NeRF methods can be computationally expensive and are not naturally compatible with mesh-based physics simulation [2]. A NeRF represents a scene as a neural radiance field and synthesizes images by querying a neural network along camera rays. This is powerful for view synthesis, but it does not

© 2026 The Authors. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

DOI: 10.54254/2755-2721/2026.GL35713

directly provide the triangle mesh, collision geometry, material structure, or semantic metadata needed for many simulation platforms.

3D Gaussian Splatting provides a promising alternative. It represents a scene with explicit Gaussian primitives and achieves high-quality real-time rendering with much faster optimization than that of many NeRF pipelines [3]. The output can be understood as a dense Gaussian primitive cloud containing position, covariance, opacity, and appearance parameters. This makes it attractive for low-cost capture of real environments. However, Gaussian splats are still not the same as simulation-ready assets. Omniverse, Isaac Sim, and many robotics workflows require mesh surfaces, collision structures, OpenUSD organization, and physically meaningful scene components. This paper therefore proposes a research direction: convert 3D Gaussian Splatting reconstructions into mesh-based OpenUSD assets that can be used in NVIDIA Omniverse and Isaac Sim for training. Our core contribution lies in a pipeline framework connecting visual scene reconstruction and simulation compatibility. Rather than treating 3DGS only as a rendering format, the proposed approach treats it as an intermediate reconstruction source from which geometry, texture, materials, and metadata can be extracted.

2. Background and motivation

2.1. Nerf-based reconstruction

NeRF introduced a major shift in novel view synthesis by representing a scene as a continuous function that maps spatial coordinates and view directions to density and color [2]. The method can produce photorealistic results, but training and rendering costs are high because the network must be sampled repeatedly along camera rays. When generating simulation assets, a critical drawback is that learned radiance fields lack native editability and physical simulation compatibility. A robot simulator usually needs surfaces, colliders, object hierarchy, and units rather than only an image-generating function.

2.2. 3D Gaussian Splatting as a low-cost alternative

3D Gaussian Splatting reduces this cost by optimizing an explicit set of anisotropic Gaussian primitives initialized from camera calibration and sparse points [3]. Because empty space is avoided and the scene is rendered through fast splatting, the approach supports real-time viewing and efficient reconstruction. NVIDIA Omniverse libraries also now include tools related to Gaussian splat conversion and OpenUSD workflows, supporting the connection between splat-based capture and simulation pipelines [4].

2.3. Mesh extraction and OpenUSD motivation

Mesh extraction from 3DGS is an active research area. SuGaR aligns Gaussians with surfaces and uses Poisson reconstruction to extract meshes efficiently [5]. Later work improves mesh quality by adding normal supervision, confidence estimation, or mesh-in-the-loop optimization [6-8]. These studies show that Gaussian splats can be used not only for rendering but also as a source for surface reconstruction. This paper addresses how to connect mesh extraction to simulation-specific requirements.

OpenUSD provides a structured format for interoperable 3D scene composition, materials, geometry, and simulation workflows. NVIDIA describes Omniverse libraries as tools for physical AI

DOI: 10.54254/2755-2721/2026.GL35713

simulation and OpenUSD-based asset workflows [4]. Isaac Sim can ingest data from CAD, URDF, and real-world captures and convert or organize them into USD scenes for simulation, synthetic data generation, and robot learning [1]. This makes OpenUSD a natural target format for a 3DGS-to-simulation pipeline.

The motivation is therefore economic as well as technical. If a real warehouse corner, laboratory bench, road segment, or indoor room can be captured with ordinary cameras and converted into a reusable USD asset, the cost of building simulation environments decreases. This does not remove the need for expert validation, but it can reduce the manual modeling burden and make small-scale training environments available to teams that cannot build every mesh by hand.

3. Research problem and objective

3.1. Problem definition

The research problem is not simply rendering a Gaussian splat scene, but transforming a visually reconstructed Gaussian primitive cloud into a simulation-ready mesh asset. A useful asset for Omniverse or Isaac Sim should have at least four properties: geometric surfaces that approximate the real scene, appearance data preserving visual realism, physical information for collision and interaction, and a scene structure storable and reusable in OpenUSD.

3.2. Technical challenges

The first difficulty is geometric ambiguity. 3DGS is optimized mainly for photometric rendering, so a set of Gaussians may produce good images while still containing noisy surfaces, floating primitives, thin transparent regions, or incomplete geometry. Converting Gaussian centers to a point cloud and applying surface reconstruction may produce a plausible mesh that fails under physics simulation. For robot training, incorrect collision surfaces can be more damaging than small image-space artifacts [9, 10].

The second difficulty is asset compatibility. A mesh imported into Omniverse should have consistent scale, coordinate orientation, materials, texture maps, and a clean scene hierarchy. For Isaac Sim, the asset may also need collision approximations, semantic labels, and object-level segmentation so that synthetic data and training tasks can be generated. A raw mesh is therefore not enough; the pipeline must produce a SimReady-style asset rather than only a visual surface.

3.3. Research objective

This research aims to design and evaluate a low-cost conversion pipeline. The pipeline should use 3DGS as the front-end capture and reconstruction method, extract a simulation-usable mesh, convert materials and textures, export the result to OpenUSD, and validate the asset inside Omniverse or Isaac Sim. A future objective is to extend the pipeline toward live conversion, where new video or sensor frames update the Gaussian representation and produce incremental mesh or USD scene updates.

4. Proposed pipeline

Figure 1 illustrates the proposed five-stage 3DGS-to-Omniverse conversion pipeline. The workflow begins with real-world RGB images or video frames and camera poses, reconstructs a Gaussian primitive cloud, filters noisy Gaussians and reconstructs stable surfaces, converts the result into a

DOI: 10.54254/2755-2721/2026.GL35713

repaired mesh, bakes appearance information into textures, and finally exports a simulation-ready OpenUSD scene for Omniverse or Isaac Sim.

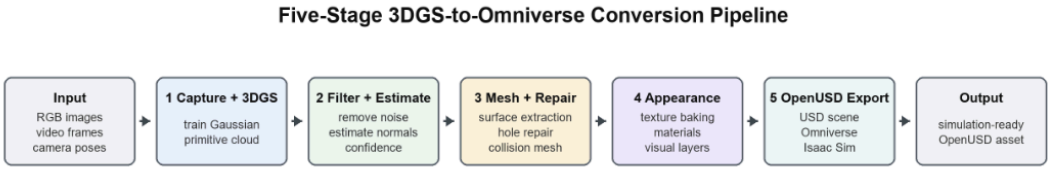


Figure 1. Five-stage workflow from real-world capture to simulation-ready OpenUSD output

4.1. Real-world capture and 3DGS reconstruction

The pipeline's first stage is real-world capture and Gaussian reconstruction. A set of RGB images or video frames is collected from a target environment, camera poses are estimated, and a 3DGS model is trained. This produces a Gaussian primitive cloud with centers, covariance matrices, opacity values, and appearance coefficients. Compared with NeRF, this front end is expected to reduce training and rendering costs while maintaining high visual quality.

4.2. Gaussian filtering and surface estimation

The second stage is Gaussian filtering and surface estimation. Gaussians with very low opacity, unstable covariance, or weak multi-view support are removed or down-weighted. For the remaining primitives, local normals and confidence values are estimated from covariance orientation, rendered depth consistency, and neighborhood structure. The goal is to identify which primitives are likely to belong to stable scene surfaces and which are view-dependent or noisy rendering artifacts.

4.3. Mesh reconstruction and repair

The third stage is mesh reconstruction. The stable Gaussian samples are converted into oriented points or local surface patches. Poisson surface reconstruction, Delaunay-based triangulation, or a SuGaR-like surface-aligned method can then generate a mesh [5]. The mesh is simplified, repaired, and separated into object or region components where possible. Repair includes removing isolated fragments, closing small holes, improving normal consistency, and creating simplified collision meshes for physical simulation.

4.4. Appearance transfer

The fourth stage is appearance transfer. Because 3DGS stores rich view-dependent appearance, directly baking it into ordinary textures may fail to preserve specular or transparent effects. The proposed method uses multi-view texture baking for diffuse appearance and stores more complex view-dependent effects as approximate material parameters or separate visual layers when needed. For robotics training, the system does not need perfect film-quality rendering in every case; it needs a balance between visual realism, physical usability, and runtime efficiency.

4.5. OpenUSD and Omniverse integration

The fifth stage is OpenUSD and Omniverse integration. The processed mesh, materials, texture maps, scale information, coordinate frame, semantic labels, and collision approximations are

DOI: 10.54254/2755-2721/2026.GL35713

exported into an OpenUSD scene. In Omniverse or Isaac Sim, the asset can then be placed inside a simulated environment, assigned physics properties, used for synthetic data generation, or connected to robot learning pipelines. This stage marks the transition from reconstruction quality to simulation readiness.

4.6. Future live conversion

For future live conversion, the same stages can be made incremental. A streaming camera system would update the Gaussian primitive cloud as new frames arrive. Only changed regions would be filtered, meshed, and re-exported to USD. A low-resolution collision mesh could update quickly for interaction, while a higher-quality visual mesh and texture set could update asynchronously. This design would support near-real-time digital twin creation, although it would require careful handling of latency, topology changes, and temporal stability.

The live setting should not attempt to rebuild the entire environment after every frame. A more realistic design is a two-layer update system. The first layer updates a fast visual representation for monitoring and navigation. The second layer updates simulation meshes only when a region becomes stable enough to support collision or training. This separation keeps the system responsive while avoiding unsafe physics updates caused by temporary reconstruction noise.

5. Evaluation design

5.1. Baseline comparison

The evaluation should compare the proposed 3DGS-to-mesh pipeline with NeRF-based reconstruction and conventional photogrammetry-style mesh generation. The comparison should include reconstruction time, GPU memory use, mesh quality, visual fidelity, and simulation usability. The main hypothesis is that 3DGS can reduce the cost of generating usable simulation assets while preserving enough realism for training and synthetic data generation (see Table 1).

Table 1. Summary of expected result

Evaluation Target	Example Object	Quantitative Indicators	Expected Output
Geometry quality	Recovered mesh	Chamfer distance; normal consistency; hole count;triangle count	Clean mesh usable for simulation
Simulation readine	OpenUSD / Isaac Sim asset	Collision-report count; object-scale error; Omniverse load time; Isaac Sim frame rate	Stable physics and efficient loading
Rendering fidelity	Held-out camera views	PSNR; SSIM; LPIPS	Visual quality comparable to source views
Live conversion	Incremental scene update	Frame-to-Gaussian latency; Gaussian-to-mesh latency; USD sync time; flicker count	Near-real-time digital twin update

5.2. Reconstruction and rendering metrics

Mesh quality can be measured with Chamfer distance, normal consistency, surface completeness, hole count, and triangle count. Rendering quality can be measured with PSNR, SSIM, and LPIPS against held-out views. Simulation readiness should be evaluated separately because a visually accurate mesh may still be unusable for physics simulation. Metrics include collision stability, object

DOI: 10.54254/2755-2721/2026.GL35713

scale accuracy, load time in Omniverse, frame rate in Isaac Sim, and success rate of synthetic annotation export.

5.3. Simulation and robotics evaluation

A practical robotics experiment can further test whether the assets are useful for training. For example, a captured tabletop or small indoor environment could be converted into an OpenUSD scene. Isaac Sim could then generate RGB, depth, segmentation, and bounding-box data for object detection or navigation tasks. A model trained with the generated data could be evaluated on real images from the same environment to estimate the sim-to-real gap.

5.4. Live-conversion evaluation

The live-conversion extension should be evaluated with latency and stability metrics. Important measurements include time from new frame arrival to Gaussian update, time from Gaussian update to mesh patch update, time from mesh update to USD synchronization, and the amount of visual or physical flicker between frames. These metrics determine whether live conversion is only feasible offline or also useful for interactive digital twin workflows.

5.5. Ablation and cost analysis

Ablation studies should remove major pipeline components one at a time. The absence of Gaussian confidence filtering yields floating geometric artifacts on reconstructed meshes. Missing mesh repair operations will cause physical simulation failures of exported assets. Lack of texture baking results in physically valid but visually degraded simulation assets. Insufficient USD metadata allows scene loading but renders assets inapplicable for robotic training tasks. These ablations would show which parts of the pipeline are most important for the transition from reconstruction to simulation.

Cost should be reported in a way that is meaningful for users. The evaluation should include capture time, reconstruction time, manual cleanup time, GPU hours, and the number of usable simulation assets produced per scene. A pipeline that is slightly less photorealistic than a NeRF-based method may still be preferable if it produces editable meshes, colliders, and USD assets much faster.

6. Conclusion

This paper reinterprets 3D Gaussian Splatting as a cost-effective solution that bridges real-scene acquisition and simulation-compatible asset generation. Although 3DGS is often discussed as a rendering method, its explicit Gaussian primitive cloud can also serve as an intermediate structure for mesh extraction. By converting Gaussian reconstructions into OpenUSD-compatible meshes, the proposed pipeline aims to make captured real-world environments usable inside NVIDIA Omniverse and Isaac Sim.

The proposed approach mainly reduces cost. NeRF can produce high-quality novel views, but its computational cost and implicit representation make it less direct for robotics simulation. 3DGS offers faster reconstruction and real-time rendering, while mesh conversion provides compatibility with physics, collision, materials, and training workflows. The combination could help developers build digital twin environments more quickly and use them for synthetic data generation or robot learning. However, mesh extraction from Gaussian splats is still imperfect. Photorealistic rendering does not guarantee accurate geometry, and simulation requires stronger geometric consistency than

DOI: 10.54254/2755-2721/2026.GL35713

image synthesis alone. Future work should implement the pipeline, test multiple mesh extraction strategies, validate exported USD assets in Omniverse, and measure whether the generated environments improve training efficiency. Another promising direction involves live incremental conversion, which continuously updates simulation-compatible scene assets based on streaming scene captures.

References

[1] NVIDIA. (2026). NVIDIA Isaac Sim. *NVIDIA Developer*. https://developer.nvidia.com/isaac/sim
[2] Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., & Ng, R. (2020). NeRF: Representing scenes as neural radiance fields for view synthesis. In *Proceedings of the European Conference on Computer Vision (ECCV)*.
[3] Kerbl, B., Kopanas, G., Leimkuehler, T., & Drettakis, G. (2023). 3D Gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics, 42*(4).
[4] NVIDIA. (2026). NVIDIA Omniverse libraries. *NVIDIA Developer*. https://developer.nvidia.com/omniverse
[5] Guedon, A., & Lepetit, V. (2023). SuGaR: Surface-aligned Gaussian splatting for efficient 3D mesh reconstruction and high-quality mesh rendering. arXiv:2311.12775.
[6] Huang, B., Yu, Z., Chen, A., Geiger, A., & Gao, S. (2024). 2D Gaussian splatting for geometrically accurate radiance fields. arXiv:2403.17888.
[7] Krishnan, M., Fowl, L., & Duraiswami, R. (2025). 3D Gaussian splatting with normal information for mesh extraction and improved rendering. arXiv:2501.08370.
[8] Guedon, A., Gomez, D., Maruani, N., Gong, B., Drettakis, G., & Ovsjanikov, M. (2025). MILo: Mesh-in-the-loop Gaussian splatting for detailed and efficient surface reconstruction. arXiv:2506.24096.
[9] NVIDIA. (2026). OpenUSD overview. *NVIDIA Omniverse Documentation*. https://docs.omniverse.nvidia.com/usd/latest/index.html
[10] Schönberger, J. L., & Frahm, J.-M. (2016). Structure-from-motion revisited. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.