

Splat, Click, Asset: An Artist-Centric Segmentation Pipeline for Gaussian Splats

1st Danial Barazandeh
University of Bremen (CGVR)
Bremen, Germany
danial.barazandeh@uni-bamberg.de

2nd Alireza Dehghanpour
University of Bremen (CGVR)
Bremen, Germany
alireza.dehghanpour@uni-bremen.de

3rd Gabriel Zachmann
University of Bremen (CGVR)
Bremen, Germany
zach@cs.uni-bremen.de



Fig. 1. The Splat, Click, Asset results: successfully isolating the object from the background and decomposing it according to the semantic hierarchy.

Abstract—3D Gaussian Splatting (3DGS) has revolutionized real-time rendering, but converting these Gaussians into interactive game assets remains a significant challenge. Recent methods, such as Feature 3DGS and Gaussian Grouping, attempt to address this by automatically distilling features during the expensive training phase. However, these “black-box” approaches are pipeline-dependent, render the pre-existing 3DGS models incompatible without costly retraining, and effectively remove the artist from the loop. Furthermore, purely automated algorithms can misinterpret artistic intent, treating asset definition as a classification problem rather than a creative decision. In this paper, we propose a model-agnostic, artist-centric segmentation tool that restores control to artists and game developers. Unlike prior work, our approach operates directly on standard *.ply* (Polygon File Format) files, leveraging fast GPU-accelerated geometric clustering as a “co-pilot” for the artist. We demonstrate that this strong clustering initialization enables users to interactively isolate, clean, and extract complex objects in minutes (~ 3 minutes in our study). Furthermore, we validate the system’s efficiency and user-friendliness through a formal user study, including a System Usability Scale (SUS) score of 70, demonstrating above-average usability for an early-stage academic prototype. Ultimately, we present an interactive workflow that prioritizes rapid iteration, universal compatibility with the Gaussian Splatting pipeline, and the validation of human agency as the ultimate arbiter of asset definition. Our source code is publicly available at: <https://danial-barazandeh.github.io/Splat-Click-Asset/>

I. INTRODUCTION

Creating 3D assets for video games and the cinematic pipeline is a labor-intensive bottleneck. To make a photorealistic 3D scene, the traditional workflow requires artists to meticulously model geometry, unwrap UV maps, and author proper textures. This process can often take hours or even days for a single asset. Consequently, there is growing interest in leveraging automated approaches like 3D Gaussian Splatting (3DGS) [1] to create photorealistic 3D representations directly from real-world footage.

However, while 3DGS addresses the challenge of visual fidelity, it introduces inherent structural limitations. Fundamentally, a 3DGS scene consists of millions of unstructured Gaussian primitives that lack semantic definition. This unstructured nature renders raw Gaussian splats unsuitable for game development pipelines, as objects cannot be selected as separate instances, physics colliders cannot be assigned, and editing individual elements becomes impractical.

To address this, recent works such as Feature 3DGS [2] and Gaussian Grouping [3] have proposed embedding semantic features directly into the Gaussians. While effective, these methods rely on distilling knowledge from 2D foundation models (like SAM or CLIP) [4] [5] during the computationally expensive training phase. This creates a rigid, pipeline-dependent workflow: if an artist downloads a pre-trained *.ply* file, they cannot segment it without retraining the model from scratch. Moreover, these results inherit the probabilistic biases

of the foundation model rather than being driven by the project’s artistic vision.

In this paper, we present Splat, Click, Asset, an interactive system designed to restore the user’s full agency in the asset creation process. In this context, we use the term ‘artist-centric’ to describe a workflow that prioritizes human decision-making, empowering developers, technical artists, and creators to manually define semantic boundaries rather than relying on black-box machine learning models. Our tool provides an intuitive graphical interface that supports a hierarchical, artist-driven refinement process. Users can iteratively isolate regions of interest, apply targeted outlier removal, and export clean assets in standard *.ply* or SPLAT formats. To validate our approach, we conducted a formal user study evaluating the system’s workflow velocity and overall usability.

The core contributions of this work are threefold:

- **An Interactive, Model-Agnostic Pipeline:** A post-hoc segmentation workflow operating directly on compiled *.ply* files, featuring GPU-accelerated clustering and an interactive 3D bounding box for explicit spatial control.
- **A Formal Usability Evaluation:** Empirical validation of the system’s artist-centric design using the industry-standard System Usability Scale (SUS) and a custom questionnaire evaluating individual pipeline components.
- **Quantitative Workflow Benchmarking:** A time-to-completion study providing hard metrics on the pipeline’s velocity in isolating and cleaning complex segmented assets.

II. RELATED WORK

A. 3D Reconstruction

While early novel view synthesis relied on implicit Neural Radiance Fields (NeRFs) [6], which suffered from prohibitive training and rendering costs, the field recently shifted back toward explicit geometry.

To address these computational limitations, Kerbl et al. [1] introduced 3D Gaussian Splatting (3DGS), which combines the speed of explicit geometry with the visual quality of volumetric rendering. Instead of a neural network, 3DGS represents the scene as a collection of millions of explicit 3D Gaussians, each defined by covariance, opacity, and spherical harmonics for view-dependent color. This allows for real-time rasterization via a tile-based sorter, significantly outperforming NeRFs in training and rendering speeds. However, strictly returning to explicit primitives reintroduces the challenge found in early SfM point clouds: the resulting scene is a massive, unstructured list of Gaussians with no inherent semantic understanding or object separability.

To address the geometric artifacts of 3DGS, 2D Gaussian Splatting (2DGS) [7] replaces volumetric ellipsoids with 2D oriented disks to produce cleaner surfaces and normals. While this improved geometric fidelity makes 2DGS highly attractive for game development and collision meshes, it remains an unstructured representation. Despite the cleaner geometry, the semantic definition of objects remains absent, necessitating the exact segmentation pipeline our tool provides.

B. Gaussian Splatting as Game Asset

Beyond visual fidelity, recent research has incorporated physical properties essential for interactive game environments into 3D Gaussian Splatting. PhysGaussian [8] represents a significant leap in this direction by integrating continuum mechanics via the Material Point Method (MPM) directly into the Gaussian kernels, enabling generative dynamics, such as elastic deformation and fracture, using the same unified representation for both rendering and simulation. Similarly, GASP [9] proposes a framework for physics-based simulation that operates directly on Gaussian primitives, facilitating real-time interactions without the need for a proxy mesh. While these methods successfully model the physical behavior of matter, they fundamentally lack a mechanism for semantic definition; they typically operate on the assumption that the target object has already been isolated from the larger scene. Consequently, deploying these advanced physics engines in a production pipeline remains contingent on an upstream step. Our work addresses this critical gap by enabling the rapid extraction of the distinct, semantic objects to which these physical properties can be applied.

Integrating character animation poses another significant challenge. While methods like GSAC [10] and SkinnedGaussian [11] enable rigging by binding Gaussians to standard armatures, they fundamentally rely on pre-defined templates or tightly coupled reconstruction pipelines. They cannot process an arbitrary, un-rigged Gaussian Splat scan and segment it into riggable components. This limitation highlights a critical workflow gap that our interactive clustering tool is designed to address.

Finally, the practical deployment of 3DGS in game environments is heavily constrained by rendering performance and memory budgets. LODGE [12] addresses the runtime bottleneck for large-scale scenes by introducing a comprehensive Level of Detail (LOD) system, dynamically adjusting the number of rendered Gaussians based on camera distance to maintain real-time frame rates. However, while this optimization is critical for performance, it operates on the global point cloud as a single entity. It does not inherently structure the data into the distinct, instanced assets required for a modular game level design.

C. Segmenting Gaussian Splatting

Recent works have proposed integrating feature learning directly into the 3DGS optimization pipeline. Feature 3DGS [2] introduces a general framework that augments each Gaussian primitive with a high-dimensional semantic feature vector, utilizing a custom parallel N-dimensional rasterizer to distill knowledge from 2D foundation models (such as SAM [4] and CLIP [5]) during training. Similarly, Gaussian Grouping [3] proposes a joint reconstruction-segmentation approach by assigning a compact, learnable ‘Identity Encoding’ to each Gaussian, which is supervised by 2D SAM masks and regularized via 3D spatial consistency to ensure coherent object grouping. However, while these methods achieve impressive segmentation results, they are fundamentally training-time

solutions. They rely on the availability of the original multi-view image dataset and require computationally intensive optimization to ‘bake’ semantics into the model. This creates a rigid pipeline dependency, rendering them unsuitable for processing pre-existing, third-party, or archived Gaussian Splat assets. This is the critical workflow gap that our model-agnostic approach addresses explicitly.

Addressing the challenge of inconsistent 2D segmentation masks across views, Contrastive Gaussian Clustering (CGC) [13] proposes a weakly supervised approach that leverages contrastive learning to enforce multi-view consistency in the 3D feature field. Instead of relying on pre-processed, consistent video tracking data (like Gaussian Grouping), CGC optimizes a segmentation feature vector for each Gaussian by maximizing the similarity of features within the same 2D SAM mask while minimizing it for different segments. To further refine the 3D structure, it introduces a spatial-similarity regularization term that encourages neighboring Gaussians to share semantic features. However, despite removing the need for consistent ground-truth masks, CGC remains a training-time intervention. It requires the original multi-view image set to compute its contrastive losses during the optimization phase, making it inapplicable to pre-existing, compiled 3DGS assets where the source training data is unavailable, which is the limitation that our post-processing workflow effectively addresses.

To address the multi-granularity ambiguity inherent in 3D segmentation, Segment Any 3D Gaussians (SAGA) [14] integrates the 2D promptable segmentation capabilities of the Segment Anything Model (SAM) directly into the 3DGS framework. SAGA achieves this by attaching a learnable ‘affinity feature’ to each Gaussian, optimized via a scale-aware contrastive training strategy that employs a soft scale-gate mechanism to dynamically adjust feature channels based on physical 3D scale. This allows for real-time, multi-granularity segmentation inference. However, similar to other distillation-based methods, SAGA’s reliance on 2D visual prompts and contrastive learning necessitates access to the original multi-view training data to supervise the affinity features. Consequently, it functions as a training-time augmentation rather than a post-processing tool, limiting its utility for pre-compiled or legacy 3DGS assets where the original source imagery is unavailable.

Recent interactive methods introduce user guidance to 3DGS, but face operational limitations: Click-Gaussian [15] requires a secondary training phase for feature distillation, while GaussianCut [16] uses a training-free approach but still relies on 2D user clicks projected into 3D space. Although highly effective for binary foreground extraction, this method depends on specific camera view interactions. In contrast, our pipeline operates entirely within explicit 3D space. By combining hierarchical clustering with direct 3D bounding box manipulation and outlier filtering, our system allows artists to functionally decompose and clean assets independent of original camera trajectories or retraining.

D. Traditional Point Cloud Segmentation

Finally, before the advent of Gaussian Splatting, the segmentation of unstructured 3D data (such as LiDAR scans or photogrammetric point clouds) relied heavily on explicit clustering algorithms. Density-based methods like DBSCAN [17] and HDBSCAN [18], as well as partitioning methods like K-Means [19], have long been the standard for organizing raw geometric data based on spatial proximity and color similarity. While the field has recently pivoted toward neural feature distillation to handle semantic ambiguity, 3DGS’s return to an explicit representation invites a re-evaluation of these classical, highly optimized algorithms. Our work demonstrates that when paired with a human-in-the-loop workflow, these traditional clustering techniques offer computational efficiency and deterministic control that ‘black-box’ neural methods cannot match.

III. METHODOLOGY

A. System Architecture

Our system is a high-performance, interactive post-processing tool designed to bridge the gap between raw Gaussian Splatting reconstructions and game engine pipelines. Unlike training-time solutions that require the full source dataset and hours of optimization, our system operates directly on the compiled *.ply* asset, extracting the explicit attributes—Position $P \in \mathbb{R}^3$, Rotation $R \in \mathbb{R}^4$, Scale $S \in \mathbb{R}^3$, Opacity $\alpha \in \mathbb{R}$, and Spherical Harmonics $SH \in \mathbb{R}^k$ —into structured NumPy arrays. To achieve interactive frame rates with millions of primitives, we employ a client-server architecture where the backend leverages *PyTorch* for tensor management and *cuML* (*RAPIDS*) [20] and *Faiss* [21] for GPU-accelerated clustering operations. The frontend is rendered via *Viser* [22], a WebGL-based visualization library that supports real-time manipulation of massive point clouds. Crucially, we implement a *zero-copy* data workflow where possible, maintaining Gaussian attributes in GPU memory and selectively transferring only the cluster labels and subsampled visualization data to the frontend, ensuring that the ‘Splat-Click-Asset’ loop remains responsive even for scenes exceeding 5 million points.

B. Interface and Workflow

The core of our artist-centric approach is an interactive graphical interface that prioritizes responsiveness and granular control. To maintain real-time feedback loops even with scenes containing millions of Gaussians, we implement a dynamic sub-sampling mechanism. This allows the artist to perform clustering operations on a representative subset of the data (e.g., $N_{sub} \approx 10^5$) for instant visualization.

However, a critical challenge lies in translating these low-resolution decisions to the full-fidelity asset ($N_{total} \approx 10^7$) without losing high-frequency details. Rather than simple interpolation, we implement a *GPU-accelerated supervised classification step*. Upon finalization, the system treats the user-validated subset as a sparse, labeled training set $\mathcal{T} = \{(p_i, l_i) \mid p_i \in P_{sub}\}$. We then instantiate a k -Nearest Neighbor (k -NN) classifier (typically $k = 7$) to predict the

cluster identity for every unlabeled point in the full scene based on spatial proximity:

$$\hat{l}(x) = \arg \max_{c \in \mathcal{C}} \sum_{x_i \in \mathcal{N}_k(x)} \mathbb{I}(l_i = c) \quad (1)$$

where $\mathcal{N}_k(x)$ represents the set of k nearest neighbors in the subsample for a point x in the full cloud. By leveraging optimized GPU search kernels (via cuML), this inference step processes millions of points in seconds, ensuring that fine geometric structures, such as the edges of leaves or thin mechanical struts, are accurately captured in the final export, even if they were aliased out of the interactive viewport.

The interface exposes critical clustering hyperparameters, such as the minimum cluster size for density-based methods or the target cluster count for spatial partitioning, via intuitive sliders. This enables a non-destructive, exploratory workflow where artists can “dial in” the segmentation logic to match the geometric realities of the scene. Finally, the pipeline includes a dedicated *export module* that serializes the segmented clusters into standard formats. Users can export individual objects or the entire segmented scene as *.ply* files (preserving full spherical harmonics) or optimized *.splat* files, ensuring immediate compatibility with web viewers and game engine plugins.

C. Clustering Algorithms

To accommodate the diverse topological and semantic structures found in Gaussian Splatting scenes, our system integrates a suite of clustering strategies ranging from density-based to geometric partitioning methods.

1) *Density-Based Clustering (HDBSCAN)*: For decomposing complex objects, we employ Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN) [18]. Unlike partitioning methods, HDBSCAN does not require a pre-defined number of clusters. It constructs a hierarchy based on the mutual reachability distance $d_{mreach}(p, q) = \max\{core_k(p), core_k(q), d(p, q)\}$, where $core_k$ is the distance to the k -th nearest neighbor. This makes it particularly effective for decomposing objects into their functional parts.

2) *Geometric Partitioning (K-Means)*: For decomposing rigid, spatially distinct objects (e.g., a row of separate buildings), we utilize K-Means clustering accelerated via FAISS [21]. We minimize the within-cluster sum of squares inertia $\sum_{i=0}^n \min_{\mu_j \in \mathcal{C}} (\|x_i - \mu_j\|^2)$. While K-Means assumes spherical clusters, we find that this geometric bias is often advantageous for creating game assets, which are frequently modeled as compact, convex entities.

D. Post-Process Outlier Removal

Gaussian Splatting reconstructions often exhibit “floaters” and high-frequency noise that can disrupt segmentation boundaries. To address this, our tool includes a hierarchical outlier-removal stage. We leverage GPU acceleration via FAISS [21] to process millions of Gaussians at interactive rates. The system presents three distinct strategies to the user, with parameters dynamically adjustable via the GUI.

1) *Statistical Outlier Removal (SOR)*: This method removes points that are statistically detached from the local distribution. For a set of points P , we utilize GPU-accelerated k -Nearest Neighbor (k -NN) search to compute the mean distance $\bar{d}_i$ for every point p_i to its k neighbors. We then compute the global mean μ_{dist} and standard deviation σ_{dist} of these distances. A point is rejected if:

$$\bar{d}_i > \mu_{dist} + \alpha \cdot \sigma_{dist} \quad (2)$$

where α is a user-controlled sensitivity parameter (*std_ratio*). This method is particularly effective for removing sparse noise surrounding dense objects.

2) *GPU-Optimized Density Filtering*: Density-based filtering removes points that lack a sufficient number of neighbors k_{min} within a radius r . However, standard GPU implementations of *Range Search* often suffer from thread divergence and uncoalesced memory access due to the ragged nature of dynamic neighbor counts (i.e., different points returning vastly different numbers of neighbors).

To ensure interactive performance on the GPU, we reformulated the density query into a fixed-memory k -NN problem. We observe that the condition of having at least k_{min} neighbors within radius r is mathematically equivalent to the distance to the k -th neighbor (d_k) being less than r . Our algorithm proceeds by first performing a GPU k -NN search where $k = k_{min} + 1$ (to account for the point itself). We then extract the distance to the furthest neighbor in this set, denoted as d_k . Finally, a point is classified as an inlier if and only if:

$$d_k^2 \leq r^2 \quad (3)$$

This approach allows us to utilize highly optimized fixed-matrix operations on the GPU, bypassing the limitations of dynamic range search while producing identical geometric results.

3) *Geometric Centroid Clipping*: For rapid elimination of gross artifacts (e.g., background points far from the object of interest), we implement a global distance filter. This method computes the geometric centroid $\mathbf{c} = \frac{1}{N} \sum p_i$ and removes points where $\|p_i - \mathbf{c}\| > \mu_{global} + \beta \cdot \sigma_{global}$. This serves as a computationally inexpensive first pass for volume reduction.

E. Interactive 3D Spatial Bounding Box

While unsupervised clustering effectively isolates distinct geometric and colorimetric manifolds, artists often require explicit spatial control to dictate semantic boundaries, particularly when objects overlap physically. To accommodate this, we introduce an interactive 3D bounding box into the viewport. Users can manually move, rotate, and scale a 3D widget to encapsulate specific regions of interest.

To process these selections at interactive frame rates across millions of primitives, we apply a rigid body transformation to the Gaussian centroids. For each point $P_i \in \mathbb{R}^3$, we compute its local position relative to the box’s center C and rotation matrix R :

$$P_{local} = (P_i - C)R$$

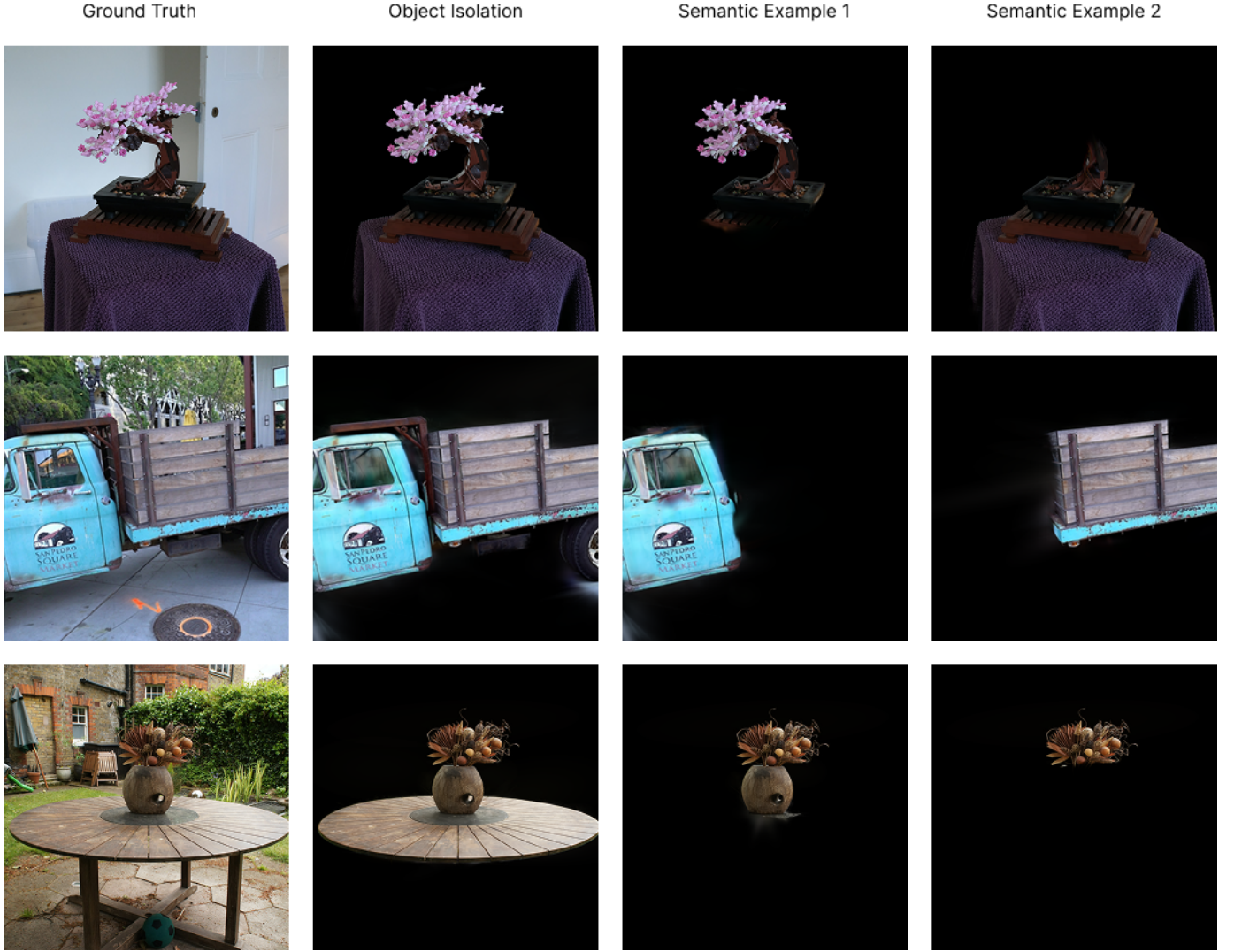


Fig. 2. This figure demonstrates the result of our work: a recursive coarse-to-fine segmentation workflow. The initial pass was performed using K-Means to separate the target asset (e.g., the Bonsai tree or Truck) from the background. Subsequently, HDBSCAN was used for further sub-clustering steps to deconstruct the isolated object into functional components, such as separating the organic foliage from the pot or the truck’s cargo bed from the cabin. This modularity enables the creation of independent, interactive game parts from a single Gaussian Splatting object. Geometric Centroid Clipping was also used to remove floaters.

We then perform a fast axis-aligned bounding check against the box’s extents E :

$$|P_{local}| \leq \frac{E}{2}$$

Based on this evaluation, the user can trigger Boolean operations, such as a set intersection (“Keep Inside”) or a set difference (“Keep Outside”), to instantly update the active scene mask. This allows for rapid manual pruning of complex scenes before or after algorithmic clustering.

F. User Study Design

To empirically validate the workflow velocity and usability of our proposed pipeline, we conducted a formal user study with 16 participants, predominantly Computer Science students with varying levels of expertise in computer graphics and game development. The study was designed to evaluate

how users interact with the system and to gather qualitative feedback. All experiments and time-to-completion metrics were recorded on a system equipped with an NVIDIA RTX 4090 GPU

Participants were first given a guided tutorial using a baseline scene (a truck) to familiarize themselves with the toolset. They were then asked to perform object isolation and cleanup on a complex target scene (a bonsai tree on a table). Participants executed two distinct workflows:

- 1) **Algorithmic Workflow:** Utilizing K-Means clustering for basic scene separation, followed by targeted outlier removal.
- 2) **Spatial and Compositional Workflow:** Utilizing the interactive 3D bounding box for initial scene separation, followed by HDBSCAN to semantically decouple the object (the tree) from adjacent surfaces (the table).

To quantify performance, we recorded the precise time-to-completion for each distinct step of the workflows. Following the practical tasks, participants completed a System Usability Scale (SUS) questionnaire to establish an industry-standard usability metric. Finally, participants completed a custom 5-point Likert scale questionnaire designed to evaluate the usefulness of individual algorithmic and spatial tools within the pipeline.

IV. RESULTS

A. Hierarchical Semantic Decomposition

Our system addresses the semantic complexity of 3D scenes by utilizing a hierarchical spatial strategy that mirrors the cognitive process of visual grouping. For general object isolation tasks or narrowing the region of interest, such as separating the table and bonsai tree, the Figure 2, we leverage K-Means clustering. However, to extract detailed hierarchical semantics, such as isolating the bonsai tree’s delicate leaves or distinguishing the individual wooden slats of the truck bed, we switch to HDBSCAN. With this, we can construct a cluster hierarchy that allows artists to narrow down clusters to reach the targeted object, effectively decomposing complex objects into their semantic parts. This hierarchical decomposition has profound implications for game physics by segmenting a monolithic scan into semantically meaningful sub-components (e.g., separating a vehicle’s door from its chassis), enabling semantically destructible environments. Unlike traditional Voronoi fracturing, which shatters geometry arbitrarily, our method prepares assets for logic-driven destruction systems where objects break apart into recognizable components, bridging the gap between visually impressive Gaussian points and game assets. We avoided using our spatial box method in Figure 2 to demonstrate the algorithm’s capability.

B. Post-Processing and Cleanup

While clustering algorithms successfully decompose the scene into semantic regions, raw Gaussian Splatting reconstructions often suffer from high-frequency artifacts(floaters). To address this, our system implements a hierarchical outlier removal pipeline that operates on the segmented clusters prior to export. As demonstrated in Figure 3, this stage is critical for refining the asset’s silhouette and ensuring clean geometry for collision mesh generation.

We provide artists with three distinct filtering strategies to accommodate different noise profiles. For removing sparse, detached artifacts (floaters) that often appear in the background of object scans, we employ *Statistical Outlier Removal (SOR)*. By computing the mean k -nearest neighbor distance for every point in the cluster, SOR effectively prunes Gaussians that lie more than α standard deviations from the local density distribution (typically $\alpha = 2.0$). Conversely, for correcting the fuzzy boundaries of an object to sharpen its edges, as seen in the refinement of the rooftop of the toy house, Figure 3, we utilize *Density-Based Filtering*. This method rejects points that lack a sufficient number of neighbors (k_{min}) within a fixed radius r , acting as a morphological erosion operation that

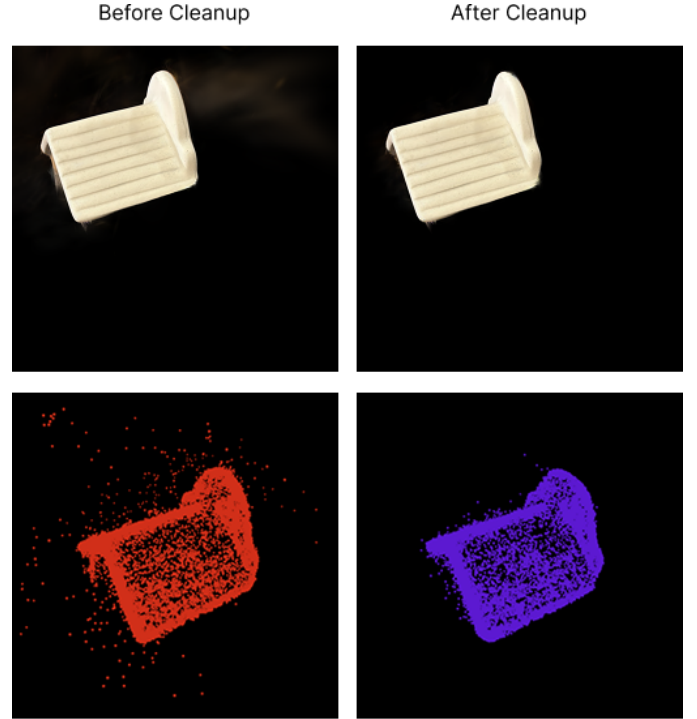


Fig. 3. Impact of our outlier removal pipeline. The left column shows the initial segmentation of a toy house, which retains significant “floaters” and background noise characteristic of raw Gaussian Splatting. The right column demonstrates the result after applying our density-based filtering and distance-based filtering.

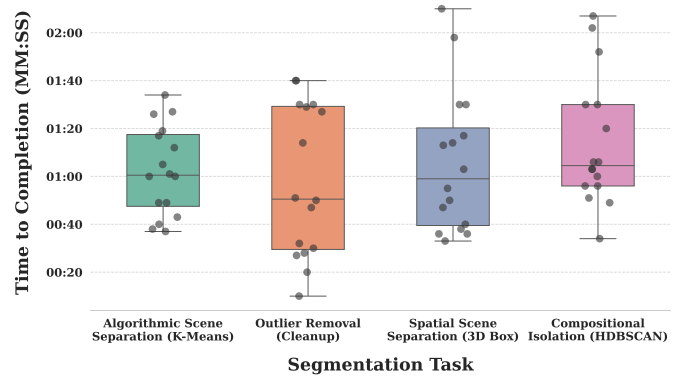


Fig. 4. Time-to-completion for N=16 users across core pipeline tasks. By operating post-hoc on pre-compiled assets, the system enables users to complete individual isolation and cleanup tasks in roughly one minute, bypassing the lengthy retraining phases required by learning-based segmentation methods.

tightens the object’s hull. Finally, a *Geometric Centroid Clip* allows for the rapid rejection of gross outliers that fall outside a global bounding volume. These cleanup options transform the noisy output of the initial segmentation into a polished asset.

C. Workflow Velocity and Empirical Performance

While recent segmentation methods (e.g., SAGA) offer rapid human interaction, they impose a subsequent 15 to 120-minute computational bottleneck to distill 2D foundation

model features into 3D space. Our system operates directly on the pre-compiled Gaussian *.ply* asset, completely bypassing this secondary training phase and enabling immediate, real-time asset extraction.

To quantify the impact of this post-hoc approach, we conducted a user study with 16 participants. Users were tasked with performing both algorithmic and spatial isolation, as well as cleanup workflows, on a complex target scene. As illustrated in Figure 4, the empirical velocity of our tool allows artists to execute complete extraction workflows in a matter of minutes.

Participants completed individual segmentation sub-tasks in approximately one minute per task, including both human interaction and GPU algorithmic execution. Even the most time-intensive operations, such as 3D bounding box manipulation, were completed by all users in under two and a half minutes. This validates the system’s utility for rapid, iterative asset development, proving that training-free spatial control can provide production-ready utility without the computational overhead of learning-based models.

D. Usability of Hybrid Interactive Tools

To empirically validate the system’s artist-centric design and understand user preferences across different interaction paradigms, we measured the perceived usefulness of individual pipeline features using a custom 5-point Likert scale (1 = Strongly Disagree, 5 = Strongly Agree).

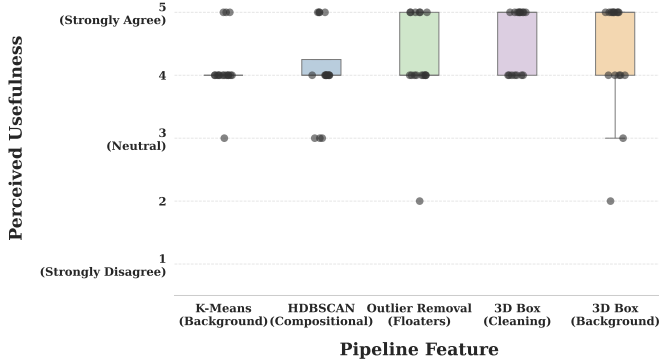


Fig. 5. User evaluation of specific pipeline features on a 5-point Likert scale (N=16). The data demonstrates high perceived utility for algorithmic tools (K-Means) during macro-level background separation, and equally high utility for explicit manual control (3D Box) during micro-level cleaning and refinement.

As shown in Figure 5, the user feedback strongly validates our hybrid approach to asset extraction (extracting the main object from the background). For macro-level tasks, such as initial scene separation, both K-Means clustering and the 3D Bounding Box received high usability scores. This indicates that users highly value having the flexibility to choose between algorithmic assistance and explicit spatial boundaries depending on the specific geometry of the target asset.

However, for micro-level refinement, explicit spatial control emerged as the undisputed preference. The use of the 3D Bounding Box for final cleanup and isolation was the highest-rated feature in the study, with nearly all users strongly

agreeing on its utility. This data supports our core hypothesis: while unsupervised clustering (K-Means, HDBSCAN) provides rapid and effective initialization, providing artists with explicit, manual spatial control is the critical requirement for making game assets.

V. DISCUSSION

Our results demonstrate that relocating semantic segmentation from the reconstruction phase to the post-processing phase offers distinct operational advantages for game development. By analyzing the user feedback and qualitative output of our pipeline, two key implications for production workflows emerge.

A. Decoupling Segmentation from Reconstruction

The primary contribution of our work is giving the artist control of Gaussians. Works like Gaussian Grouping [3] treat segmentation as an optimization problem. While mathematically elegant, this imposes significant constraints on the pipeline, effectively removing artist influence and creating incompatibility with emerging architectural approaches (such as 2D Gaussian Splatting) or third-party assets trained without specific semantic embeddings.

Our results suggest that, for interactive applications, the geometric features inherent in the compiled 3DGS format are often sufficient for separating objects from backgrounds and breaking objects into their functional components. This shift effectively democratizes the use of Gaussian Splatting, enabling artists to cluster and edit precompiled scenes and objects with full control.

B. Limitations and Future Work

It was observed that intuitive UI design can significantly improve system usability. For example, multiple users in our experiment noted that having the same color for the hierarchical cluster menu and the clusters in the viewport would significantly improve selection. Other feedback highlighted a strong interest in selecting the clusters by simply selecting them in the viewport.

Furthermore, observing users’ interactions showed that they actively avoided exploring the Statistical Outlier Removal (SOR) parameters because it was not intuitive to them, which increased cognitive load and led the majority to use Centroid Clipping because the radius parameter provided predictable feedback.

The system achieved an average SUS score of 70. While this is above the historical baseline of 68 (indicating general usability), individual variance in the scores strongly correlated with users’ comments and our observations, showing that improvement to the UI design is necessary.

Regarding algorithmic limitations, our clustering approach struggles with highly noisy reconstructions or deeply intertwined geometry where spatial boundaries are ambiguous. In these scenarios, algorithms like K-Means or HDBSCAN fail to separate distinct objects. While our 3D Box method mitigates this by enforcing explicit spatial bounds, dealing with these

scenarios remains a challenge we need to explore. We need to integrate fine-grained manual selection tools, such as 3D volumetric brushes, to fully address these failure cases.

Finally we acknowledge future work must include quantitative benchmarking (e.g., mIoU) against other models to formally evaluate comparative segmentation accuracy.

VI. CONCLUSION

In this work, we introduced an interactive segmentation pipeline to bridge the gap between Gaussian Splatting and game asset production. Our system proposes a fundamental paradigm shift: decoupling segmentation from reconstruction. By analyzing the explicit geometric features of the compiled asset, we reduce the segmentation loop from hours to minutes. This enables a true *Human-in-the-Loop* workflow, where the algorithm serves as an amplifier of artistic intent rather than a replacement for it. We demonstrated that standard geometric clustering algorithms (K-Means), when guided by artist inputs and specialized sub-clustering (HDBSCAN), can achieve results comparable to those of complex neural foundations in separating functional game parts. We validated the usability of our tool by demonstrating its ability to decompose complex assets into sub-components and robustly clean noisy real-world scans using our GPU-accelerated outlier removal pipeline. Ultimately, our approach transforms the 3DGS format from a rigid visualization medium into a flexible, editable asset. By providing a universal Gaussian splatting, model-agnostic tool that works on standard *.ply* files, whether generated by 3DGS, 2DGS, or future explicit representations, we aim to facilitate the use of Gaussian Splatting in game development.

ACKNOWLEDGMENT

The authors acknowledge the use of Gemini 3.1 Pro to assist with grammar correction and structural refinement during the drafting of this manuscript. All concepts, data, and conclusions remain solely the intellectual property and responsibility of the authors.

REFERENCES

- [1] B. Kerbl, G. Kopanas, T. Leimkuehler, and G. Drettakis, “3d gaussian splatting for real-time radiance field rendering,” *ACM Trans. Graph.*, vol. 42, no. 4, Jul. 2023. [Online]. Available: <https://doi.org/10.1145/3592433>
- [2] 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,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 21 676–21 685.
- [3] M. Ye, M. Danelljan, F. Yu, and L. Ke, “Gaussian grouping: Segment and edit anything in 3d scenes,” in *ECCV*, 2024.
- [4] 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,” *arXiv:2304.02643*, 2023.
- [5] 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,” in *ICML*, 2021.
- [6] 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,” *Commun. ACM*, vol. 65, no. 1, p. 99–106, Dec. 2021. [Online]. Available: <https://doi.org/10.1145/3503250>
- [7] B. Huang, Z. Yu, A. Chen, A. Geiger, and S. Gao, “2d gaussian splatting for geometrically accurate radiance fields,” in *SIGGRAPH 2024 Conference Papers*. Association for Computing Machinery, 2024.
- [8] T. Xie, Z. Zong, Y. Qiu, X. Li, Y. Feng, Y. Yang, and C. Jiang, “PhysGaussian: Physics-Integrated 3D Gaussians for Generative Dynamics,” in *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Los Alamitos, CA, USA: IEEE Computer Society, Jun. 2024, pp. 4389–4398. [Online]. Available: <https://doi.ieeeecomputersociety.org/10.1109/CVPR52733.2024.00420>
- [9] P. Borycki, W. Smolak, J. Wacziarg, M. Mazur, S. Tadeja, and P. Spurek, “Gasp: Gaussian splatting for physic-based simulations,” 2024. [Online]. Available: <https://arxiv.org/abs/2409.05819>
- [10] R. Zhang, A. Watkins, and N. Sarkar, “Gsac: Leveraging gaussian splatting for photorealistic avatar creation with unity integration,” 07 2025, pp. 168–178.
- [11] N. Kondo, Y. Asano, and Y. Ochiai, “Instant skinned gaussian avatars for web, mobile and vr applications,” in *Proceedings of the 2025 ACM Symposium on Spatial User Interaction*, ser. SUI ’25. New York, NY, USA: Association for Computing Machinery, 2025. [Online]. Available: <https://doi.org/10.1145/3694907.3765954>
- [12] J. Kulhanek, M.-J. Rakotosaona, F. Manhardt, C. Tsalicoglou, M. Niemeyer, T. Sattler, S. Peng, and F. Tombari, “LODGE: Level-of-detail large-scale Gaussian splatting with efficient rendering,” in *Proceedings of the 39th International Conference on Neural Information Processing Systems*, 2025.
- [13] N. Park, R. Rossi, E. Koh, I. A. Burhanuddin, S. Kim, F. Du, N. Ahmed, and C. Faloutsos, “Cgc: Contrastive graph clustering for community detection and tracking,” in *Proceedings of the ACM Web Conference 2022*, ser. WWW ’22. New York, NY, USA: Association for Computing Machinery, 2022, p. 1115–1126. [Online]. Available: <https://doi.org/10.1145/3485447.3512160>
- [14] J. Cen, J. Fang, C. Yang, L. Xie, X. Zhang, W. Shen, and Q. Tian, “Segment any 3d gaussians,” in *Proceedings of the Thirty-Ninth AAAI Conference on Artificial Intelligence and Thirty-Seventh Conference on Innovative Applications of Artificial Intelligence and Fifteenth Symposium on Educational Advances in Artificial Intelligence*, ser. AAAI’25/IAAI’25/EAAI’25. AAAI Press, 2025. [Online]. Available: <https://doi.org/10.1609/aaai.v39i2.32193>
- [15] S. Choi, H. Song, J. Kim, T. Kim, and H. Do, “Click-gaussian: Interactive segmentation to any 3d gaussians,” in *Computer Vision – ECCV 2024: 18th European Conference, Milan, Italy, September 29–October 4, 2024, Proceedings, Part III*. Berlin, Heidelberg: Springer-Verlag, 2024, p. 289–305. [Online]. Available: <https://doi.org/10.1007/978-3-031-72646-0-17>
- [16] U. Jain, A. Mirzaei, and I. Gilitschenski, “Gaussiancut: interactive segmentation via graph cut for 3d gaussian splatting,” in *Proceedings of the 38th International Conference on Neural Information Processing Systems*, ser. NIPS ’24. Red Hook, NY, USA: Curran Associates Inc., 2024.
- [17] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, “A density-based algorithm for discovering clusters in large spatial databases with noise,” in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, ser. KDD’96. AAAI Press, 1996, p. 226–231.
- [18] R. J. G. B. Campello, D. Moulavi, and J. Sander, “Density-based clustering based on hierarchical density estimates,” in *Advances in Knowledge Discovery and Data Mining*, J. Pei, V. S. Tseng, L. Cao, H. Motoda, and G. Xu, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 160–172.
- [19] J. A. Hartigan and M. A. Wong, “Algorithm as 136: A k-means clustering algorithm,” *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 28, no. 1, pp. 100–108, 1979. [Online]. Available: <http://www.jstor.org/stable/2346830>
- [20] S. Raschka, J. Patterson, and C. Nolet, “Machine learning in python: Main developments and technology trends in data science, machine learning, and artificial intelligence,” *arXiv preprint arXiv:2002.04803*, 2020.
- [21] J. Johnson, M. Douze, and H. Jégou, “Billion-scale similarity search with GPUs,” *IEEE Transactions on Big Data*, vol. 7, no. 3, pp. 535–547, 2019.
- [22] B. Yi, C. M. Kim, J. Kerr, G. Wu, R. Feng, A. Zhang, J. Kulhanek, H. Choi, Y. Ma, M. Tancik, and A. Kanazawa, “Viser: Imperative, web-based 3d visualization in python,” 2025. [Online]. Available: <https://arxiv.org/abs/2507.22885>