GaussianCut: Interactive Segmentation via Graph Cut for 3D Gaussian Splatting

Umangi Jain, Ashkan Mirzaei, and Igor Gilitschenski
University of Toronto
`{umangi, ashkan, gilitschenski}@cs.toronto.edu`

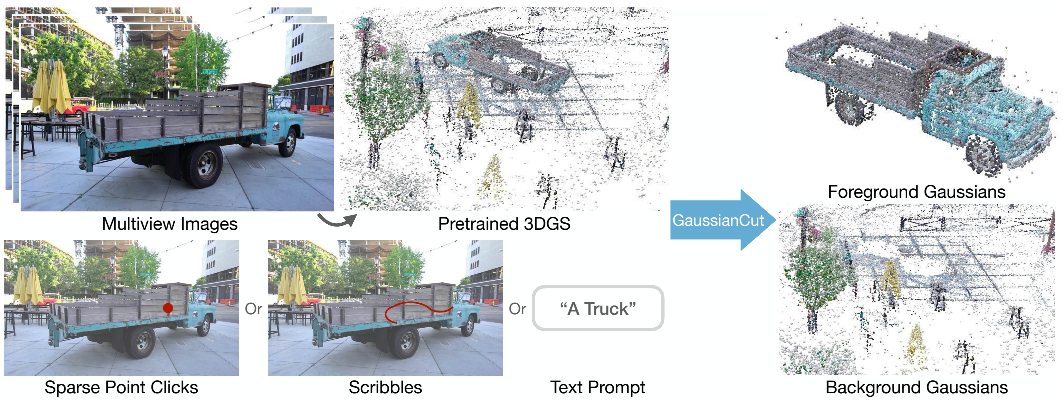

Figure 1: Our method, GaussianCut, enables interactive object(s) selection. Given an optimized 3D Gaussian Splatting model for a scene with user inputs (clicks, scribbles, or text) on any viewpoint, GaussianCut partitions the set of Gaussians as foreground and background.

Abstract

We introduce GaussianCut, a new method for interactive multiview segmentation of scenes represented as 3D Gaussians. Our approach allows for selecting the objects to be segmented by interacting with a single view. It accepts high-level user input, such as point clicks, coarse scribbles, or text. Using 3D Gaussian Splatting (3DGS) as the underlying scene representation simplifies the extraction of objects of interest which are considered to be a subset of the scene's Gaussians. Our key idea is to represent the scene as a graph and use the graph-cut algorithm to minimize an energy function to effectively partition the Gaussians into foreground and background. To achieve this, we construct a graph based on scene Gaussians and devise a segmentation-aligned energy function on the graph to combine user inputs with scene properties. To obtain an initial coarse segmentation, we leverage 2D image/video segmentation models and then use graph cut to further refine these coarse estimates. Our empirical evaluations show the adaptability of GaussianCut across a diverse set of scenes. GaussianCut achieves competitive performance with state-of-the-art approaches for 3D segmentation without requiring any changes to the underlying Gaussian-based scene representation. Project page: `https://umangi-jain.github.io/gaussiancut/`.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

1 Introduction

Recent advances in 3D scene representation have enabled unprecedented quality in 3D view synthesis without requiring specialized equipment or an excessively high computational budget. Fully leveraging these advances requires tools for scene understanding and manipulation specifically designed to operate on such representations. Object selection and segmentation often serve as a crucial first step in both scene understanding and editing tasks. While 2D image segmentation has been widely studied, developing analogous techniques for 3D remains challenging. One key challenge is accounting for the choice of underlying 3D scene representation in the segmentation method.

3D Gaussian Splatting (3DGS) [23] offers an explicit representation of a scene using a set of Gaussians. The nature of this representation motivates the idea that each Gaussian corresponds to either the segmented object or the background. Prior works in 3DGS segmentation involve augmenting each Gaussian with a low-dimensional feature, that is jointly optimized with the parameters of the Gaussians [6, 44, 55]. This is supervised by 2D features, which provide semantic information that can be used for segmentation. While this enables a 3D consistent segmentation, it significantly increases the fitting time and the already high memory footprint of the method. Thus, enabling 3DGS segmentation without modifying the optimization process is an important research challenge.

We address this challenge by proposing GaussianCut, a novel method for selecting and segmenting objects of interest in 3D Gaussian scenes. Our work taps directly into the representation created by 3DGS and maps each Gaussian to either the foreground or background. The proposed process mirrors the interactive nature of 2D segmentation tools, where users can engage through clicks, prompts, or scribbles. We require such user input on a single image and perform the object selection process in two steps. First, we obtain dense multiview segmentation masks from the user inputs using a video segmentation model. Subsequently, we construct a weighted graph, where each node represents a Gaussian. Graph cut then partitions the graph into two disjoint subsets by minimizing an energy function, which quantifies the cost of cutting the edges connecting the subsets. This approach effectively segments the selected foreground object from the background by using the energy function as a measure of dissimilarity between the nodes. An overview of the process is provided in Figure 1.

Our main contribution is a novel approach for segmentation in scenes obtained from 3DGS. Its main technical novelties are twofold: 1) we propose a method for graph construction from a 3DGS model that utilizes the properties of the corresponding Gaussians to obtain edge weights, and 2) based on this graph, we propose and minimize an energy function (Equation 3) that combines the user inputs with the inherent representation of the scene. Our experimental evaluations show that GaussianCut obtains high-fidelity segmentation outperforming previous segmentation baselines.

2 Related Work

2D image segmentation is a long studied problem in computer vision [17, 41, 57]. Recently, models like Segment Anything [25] and SEEM [64], have revolutionized 2D segmentation by employing interactive segmentation. A range of methods have also been developed for 3D segmentation, each tailored to different forms of representation, including voxels [10, 36], point clouds [42, 43], meshes [49, 62], and neural representations [7, 32, 48, 52]. The impressive capabilities of Neural Radiance Fields (NeRFs) [35] in capturing scene information implicitly have inspired numerous studies to explore 3D segmentation for NeRFs. Recent works have also explored segmentation with Gaussians as the choice for scene representation [6, 22, 44, 55, 63].

Training 3D segmentation with 2D masks/features: In addition to the wide adaptation of foundational models for 2D images [58], they are also used extensively by 3D editing and segmentation models. SAM has been used as an initial mask to facilitate 3D segmentation [6, 7, 33] and also for distillation into NeRF [59] and 3DGS models [63]. Semantic-NeRF [61] proposed 2D label propagation to incorporate semantics within NeRF so it can produce 3D consistent masks. MVSeg [37] propagates a 2D mask to different views using video segmentation techniques. ISRF [18] distills semantic features into the 3D scene of voxelized radiance fields. Nearest neighbor feature matching then identifies high-confidence seed regions. 2D features have also been used for facilitating the grouping of Gaussians [55] and for hierarchical semantics using language in 3DGS [44]. Distilled Feature Fields (DFF) [27] and Neural Feature Fusion Fields [50] distill 2D image embeddings from LSeg [30] and DINO [5] to enable segmentation and editing. SA3D [7] uses SAM iteratively to get 2D segments and then uses depth information to project these segments into 3D mesh grids. SANeRF-HQ [33] aggregates 2D masks in 3D space to enable segmentation with NeRFs.

Segmentation in 3D Gaussian Splatting: Gaussian Grouping [55], SAGA [6], LangSplat [44], CoSSegGaussians [11] and Feature 3DGS [63] require optimizing a 3DGS model with an additional identity or feature per Gaussian, which is usually supervised by 2D image features. These semantic features allow segmentation through user interaction. Gaussian Grouping and LangSplat also allow for textual prompts to segment objects supported through multimodal models like CLIP or grounding-DINO [31]. Feature-based methods alter the fitting process of 3DGS by adding additional attributes for each Gaussian and it facilitates learning features for everything in the scene. While useful, this limits the flexibility of interactivity with a single object. Our method is more flexible in choosing specific object(s) as we generate the 2D masks after the user interaction. Adding additional parameters also increases the fitting time for 3DGS. Moreover, such methods often rely on video segmentation models as they require 2D features from all training viewpoints. In contrast, we can operate on an arbitrary number of 2D masks, including just a single mask.

Graph cut for 3D segmentation: Boykov and Jolly [4] introduced a novel global energy function for interactive image segmentation using graph cut [15, 19]. Several follow-up works improved image segmentation using graph cut by designing a better energy function [13], improving optimization [3, 20], and reducing user input requirements [51]. Adapting energy minimization methods for 3D volumes has been difficult, requiring several modifications [20] to manage the higher memory demands. NVOS [45] trains a special multi-layer perceptron (MLP) to predict voxels in the foreground and background and applies graph cut on voxels as post-processing. However, this requires additional training and increases memory consumption. Guo *et al.* [21] propose 3D instance segmentation of 3D point clouds using graph cut. It involves constructing a superpoint graph and training a separate graph neural network for predicting the edge weights. Unlike their work, our method is a post hoc application and does not require any additional training. Our graph construction and edge weights have also been tailored specifically for 3D Gaussian Splatting.

Concurrent with our work, Segment Anything in 3D Gaussians (SAGD) [22], also performs interactive segmentation using 3D Gaussian Splatting without requiring any segmentation-aware training. However, their primary focus is on refining object boundaries by decomposing boundary Gaussians.

3 Method

3.1 Preliminaries

3D Gaussian Splatting (3DGS) [23] is a technique for creating a 3D representation of scenes based on a set of Gaussian ellipsoids $\mathcal{G}$. 3DGS facilitates real-time rendering and provides high-quality reconstruction. In this representation, each 3D Gaussian is characterized by a set of optimizable parameters. These include 3D position $\boldsymbol{\mu} \in \mathbb{R}^3$, spherical harmonics (SH) coefficients (for color) $\boldsymbol{\beta} \in \mathbb{R}^{3(d^2+1)}$ (where d is the degree of spherical harmonics), scale $\mathbf{s} \in \mathbb{R}^3$, rotation $\mathbf{r} \in \mathbb{R}^4$, and opacity $\sigma \in \mathbb{R}$. The optimization process involves iteratively rendering scenes and comparing the rendered images with the training views, interleaved with adaptive density control that handles the creation and deletion of the number of Gaussians. The differentiable rendering pipeline in 3DGS uses tile-based rasterization following [28] to ensure real-time rendering. 3DGS performs splatting by depth sorting the Gaussians and α-blending [14] them to project in 2D. The set of differentiable parameters for $\mathcal{G}$, $\mathcal{D} := \{\boldsymbol{\mu}_i, \boldsymbol{\beta}_i, \mathbf{s}_i, \mathbf{r}_i\, \sigma_i\}_{i=1}^{|\mathcal{G}|}$, are optimized from a set of posed images.

Graph cut refers to the problem of partitioning the vertices $\mathcal{V}$ of a graph $\mathbf{G}$ with edges $\mathcal{E}$ weighted by $\{w_e\}_{e\in\mathcal{E}}$ into two disjoint, non-empty sets such that the sum of the weights of the edges between the two sets is minimized. This minimum-cost partitioning is known as the *minimum cut*. Segmentation can be reformulated as graph cut. This is achieved by forming a graph that includes the segmented entities (e.g. pixels, voxels, Gaussians) and introducing weighted edges. This allows defining an energy function, which includes unary terms representing the cost of assigning a node to a set based on individual properties, and pairwise terms that incorporate the cost of assigning neighboring nodes to different sets. The objective of the minimization is to find a cut that optimizes the overall energy, balancing individual preferences and neighborhood interactions. An efficient way for computing this minimum cut in a graph is the Boykov-Kolmogorov algorithm [2].

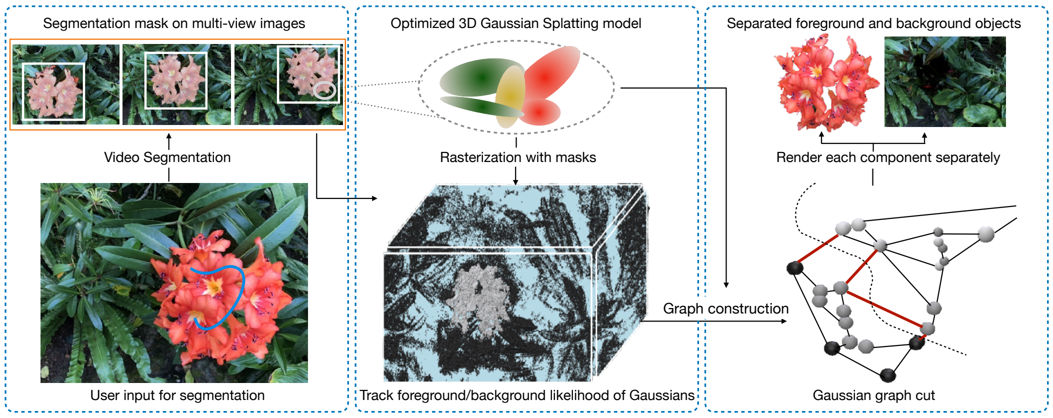
Segmentation mask on multi-view images
Video Segmentation
User input for segmentation
Optimized 3D Gaussian Splatting model
Rasterization with masks
Track foreground/background likelihood of Gaussians
Graph construction
Separated foreground and background objects
Render each component separately
Gaussian graph cut

Figure 2: Overall pipeline of GaussianCut. User input from any viewpoint is passed to a video segmentation model to produce multi-view masks. We rasterize every view and track the contribution of each Gaussian to masked and unmasked pixels. Then, Gaussians are formulated as nodes in an undirected graph and we adapt graph cut to partition the graph. The red edges in the graph highlight the set of edges graph cut removes for partitioning the graph.

3.2 Overview

Given a set of posed RGB images $\mathcal{I} = \{\mathbf{I}_i\}_{i=1}^{k}$ and an optimized reconstruction of the scene using a set of Gaussians $\mathcal{G}$, we define the task of interactive 3D segmentation as follows: given a user input (point clicks, scribbles, or text) on any image $\mathbf{I}_0 \in \mathcal{I}$, the objective is to partition the set of Gaussians in two non-empty disjoint sets $\mathcal{S}$ and $\mathcal{T}$ such that $\mathcal{S}$ represents the set of Gaussians representing the object(s) of interest and $\mathcal{T}$ represents the given 3D scene without these object(s). The extracted subset of Gaussians $\mathcal{S}$ can be rendered from any viewpoint to effectively cutout the 3D representation of the foreground without retraining. Innately, the other set of Gaussians $\mathcal{T}$ can be rendered to remove the foreground object(s). Figure 2 shows an overview of our pipeline.

In order to densify the object selection information provided by the user, we use an off-the-shelf video segmentation model to obtain dense segmentation masks from multiple views (discussed in section 4). For transferring the segmentation masks to the 3DGS representation, we trace the 3D Gaussians that map to the selected region in the masks (discussed in section 3.3). However, the masks used for propagation are not 3D consistent (as the underlying image segmentation model is 3D-unware). Moreover, the errors from 2D masks can propagate in the traced Gaussians and thereby provide a noisy 3D segmentation. To obtain a precise set of foreground Gaussians, we formulate the set of Gaussians $\mathcal{G}$ as nodes in a graph (discussed in section 3.4) and leverage graph cut to split the nodes into two sets: the foreground $\mathcal{S}$ and the background $\mathcal{T}$.

3.3 Mapping user inputs to Gaussians

We first feed the sparse single-view annotations by the user (e.g., point clicks) to a multiview/video segmentation model to obtain coarse segmentation masks across multiple training views. We then propagate the information from the 2D masks onto the set of Gaussians. For an already optimized 3DGS model of a scene, $\mathcal{G}$, we obtain n masks $\mathcal{M} := \{\mathbf{M}^j\}_{j=1}^{n}$ from a video segmentation model corresponding to any n viewpoints $\mathcal{I} := \{\mathbf{I}^j\}_{j=1}^{n}$. Here, $\mathbf{M}^j$ indicates the set of foreground pixels in the viewpoint $\mathbf{I}^j$. For each Gaussian $g \in \mathcal{G}$, we maintain a weight, w_g, that indicates the likelihood of the Gaussian belonging to the foreground. To obtain the likelihood term w_g^j pertaining to mask j for Gaussian g, we unproject the posed image $\mathbf{I}^j$ back to the Gaussians using inverse rendering and utilizing the mask information,

$$w_g^j = \frac{\sum_{\mathbf{p} \in \mathbf{M}^j} \sigma_g(\mathbf{p}) T_g^j(\mathbf{p})}{\sum_{\mathbf{p} \in \mathbf{I}^j} \sigma_g(\mathbf{p}) T_g^j(\mathbf{p})}, \tag{1}$$

where $\sigma_g(\mathbf{p})$ and $T_g^j(\mathbf{p})$ denote the opacity and transmittance from pixel $\mathbf{p}$ for Gaussian g. Combining over all the n masks,

$$w_g = \frac{\sum_j \sum_{\mathbf{p} \in \mathbf{M}^j} \sigma_g(\mathbf{p}) T_g^j(\mathbf{p})}{\sum_j \sum_{\mathbf{p} \in \mathbf{I}^j} \sigma_g(\mathbf{p}) T_g^j(\mathbf{p})}. \tag{2}$$

This likelihood, w_g, captures the weighted ratio of the contribution of Gaussian g to the masked pixels relative to the total number of pixels influenced by it. The complementary value of w_g, $1 - w_g$ provides the likelihood of Gaussian g contributing to the background. The value of w_g is updated using n 2D segmentation masks during rasterization. Since the rasterization of 3DGS is remarkably fast, each pass typically takes less than a second to update. GaussianEditor [8] also learns an additional tracing parameter but unlike our approach, it maps Gaussians to different semantic classes.

Having the likelihoods w_g, a naive approach to extract the 3D representation of the foreground is to threshold the Gaussians and prune those with values below a certain threshold τ. We denote this approach as "coarse splatting". Figure 4 demonstrates coarse splatting renders for a *plant* in the 360-garden scene [1]. Note that the renderings produced by coarse splatting are not accurate, particularly around the edges. This is due to two main reasons: 1) the 2D segmentation models are 3D inconsistent and can be imperfect, leading to artifacts in the final Gaussian cutout, and 2) lifting 2D masks to 3D can introduce its own artifacts.

3.4 Gaussian graph construction

After rasterization with the masks, each Gaussian $g \in \mathcal{G}$ in the 3DGS representation is characterized with parameters $\mathcal{D}_g := \{\boldsymbol{\mu}_g, \boldsymbol{\beta}_g, \mathbf{s}_g, \mathbf{r}_g\, \sigma_g, w_g\}$, where w_g captures the user requirement and the other parameters encapsulate the inherent properties of the scene. To fuse the two sources of information and obtain a precise set of foreground Gaussians, we formulate the optimized 3DGS model as an undirected weighted graph $\mathbf{G} := (\mathcal{G}, \mathcal{E})$, where each Gaussian in $\mathcal{G}$ is a node and $\mathcal{E}$ represents the set of edges connecting spatially adjacent Gaussians. We define the neighborhood $\mathcal{N} \subseteq \mathcal{G} \times \mathcal{G}$ of a node (Gaussian) as its k-nearest Gaussians in terms of their 3D position. The intuition behind constructing the edges is that Gaussians that map to the same object would be close spatially.

Gaussian graph cut partitions the Gaussians $\mathcal{G}$ into two disjoint and non-empty sets $\mathcal{S} \subset \mathcal{G}$ and $\mathcal{T} \subset \mathcal{G}$, that represent the foreground and background Gaussians, respectively. Our objective is to infer the foreground/background label $y_g \in \{0, 1\}$ of each Gaussian g. Let the unary term $\phi_g(\cdot, \cdot)$ represent the likelihood of node g being part of foreground or background and the pairwise term $\psi_{g,g'}(\cdot, \cdot)$ reflect the edge connection between node g and g'. To obtain the label for each Gaussian g, graph cut minimizes the aggregate of both unary and pairwise terms given by:

$$E = \sum_{g \in \mathcal{G}} \phi_g(\mathcal{D}_g, y_g) + \lambda \sum_{g, g' \in \mathcal{N}} \psi_{g,g'}(\mathcal{D}_g, \mathcal{D}_{g'}), \tag{3}$$

where λ provides a trade-off between the two terms.

Neighboring pairwise weights *(n-links)*: The pairwise term models the correlation between neighboring nodes. The neighbors for a node are based on its spatial proximity to other nodes. The edge weight between each pair of neighbors is a combination of its spatial distance and color similarity. While segments of an object can have multiple colors, and they often do, neighboring nodes with dissimilar colors can still be identified and grouped based on their spatial proximity. This ensures that parts of an object, despite varying in color, can be linked if they are close in space. For color similarity, we only use the zero-degree spherical harmonic to capture the ambient colors without any angular dependence. The correlation between the neighboring nodes is formulated as

$$\psi_{g,g'}(\mathcal{D}_g, \mathcal{D}_{g'}) = \mathbf{f}(\boldsymbol{\mu}_g, \boldsymbol{\mu}_{g'}) + \lambda_n \mathbf{f}(\boldsymbol{\beta}_g, \boldsymbol{\beta}'_g), \tag{4}$$

where λ_n is a hyperparameter balancing the contribution of position and color similarity, and the function $\mathbf{f}$ estimates similarity as $\mathbf{f}(\mathbf{x}, \mathbf{y}) = \exp(-\gamma \|\mathbf{x} - \mathbf{y}\|_2^2)$ (γ is a positive scalar).

Unary weights *(t-links)*: We designate two terminal nodes for the graph cut algorithm, the source and the sink node. These terminals represent the foreground (source) and the background (sink) in segmentation tasks. t-links connect all the nodes to both the terminal nodes and the edge weight for

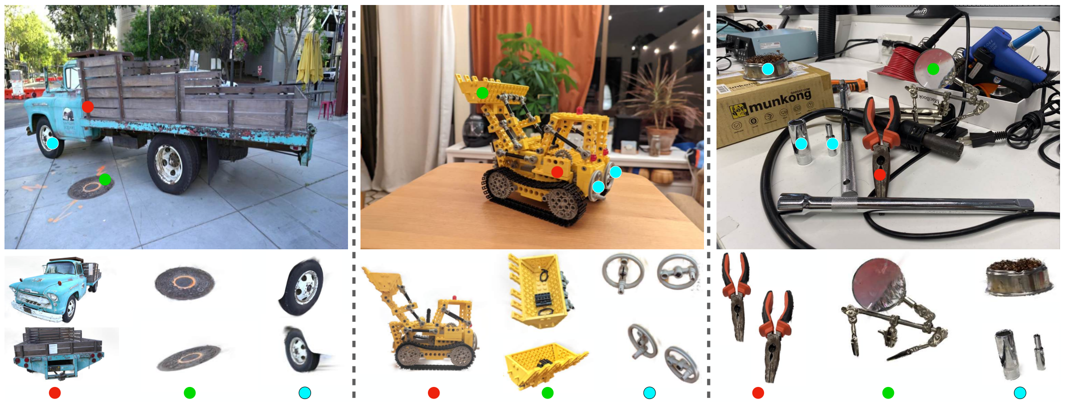
Figure 3: Visualization results of different objects in the following scenes: truck from Tanks and Temples [26], kitchen from Mip-NeRF 360 [1], tools from Shiny [54].

these links represents the pull of that node towards each terminal node. We assign the edge weights connecting each non-terminal node to the source node to reflect its likelihood of belonging to the foreground set $\mathcal{S}$ (and as belonging to the background set $\mathcal{T}$ for edges connecting to the sink node).

Gaussian tracking, from section 3.3, provides the connection of each Gaussian g to the source and sink terminal nodes, using w_g and $1 - w_g$, respectively. However, these weights can be noisy estimates. Therefore, we introduce an additional term to the edge weights that captures the similarity of node g to the other nodes that are well-connected to the terminal nodes. To do so, we identify high-confidence nodes for both the source and the sink terminals. A Gaussian g is considered as a high-confidence node for the source terminal if $w_g \approx 1$ and for a sink terminal if $w_g \approx 0$. Since computing the similarity of a node to all the high-confidence nodes is computationally expensive, we cluster all the high-confidence nodes (denoted as $\mathcal{F}$ and $\mathcal{B}$ for the source and sink, respectively) based on their position. For each node g, we then determine the closest cluster by finding $g_f = \text{argmin}_{g' \in \mathcal{F}} \mathbf{f}(\boldsymbol{\mu}_g, \boldsymbol{\mu}_{g'})$ for the source, and similarly g_b for the sink. Consequently, the unary term based on the user input is,

$$\phi_g(\mathcal{D}_g, y_g) = \begin{cases} w_g + \lambda_u \psi_{g,g_f}(\mathcal{D}_g, \mathcal{D}_{g_f}) & \text{if } y_g = 1, \\ 1 - w_g + \lambda_u \psi_{g,g_b}(\mathcal{D}_g, \mathcal{D}_{g_b}) & \text{if } y_g = 0. \end{cases} \tag{5}$$

We minimize the objective E in Equation 3 to partition the set of nodes $\mathcal{G}$ as $\mathcal{S}$ (foreground Gaussians) and $\mathcal{T}$ (background Gaussians). To render the foreground object from any viewpoint, we simply render the Gaussian collected in $\mathcal{S}$ with $\mathcal{T}$ as background.

4 Experimental Setup

Datasets: For quantitative evaluation, we test the scenes from LLFF [34], Shiny [54], SPIn-NeRF [37], and 3D-OVS [29]. All selected scenes from the LLFF and Shiny datasets are real-world front-facing scenes, with 20-62 images each. SPIn-NeRF provides a collection of scenes from some of the widely-used NeRF datasets [16, 26, 34, 35, 56]. It contains a combination of front-facing and 360° inward-facing real-world scenes. 3D-OVS contains scenes featuring long-tail objects.

Input types: Our model accepts all input processed by SAM-Track [9]. It uses grounding-DINO [31] to process text inputs. For the LLFF scenes used in NVOS [45], we follow their input scribbles to obtain the initial mask. For SPIn-NeRF and Shiny, we use clicks (each scene typically requires 1-4 clicks). For the 3D-OVS dataset evaluation, we use text query as input (results in Table 11).

Evaluation metrics: Different Image-Based Rendering (IBR) models represent 3D scenes in different ways. Thus, obtaining universal ground-truth 3D masks is difficult. To avoid this challenge, we evaluate the segmentation mask of the projected 2D rendering from the scene. The ground-truth 2D masks are typically obtained from professional image segmentation tools. NVOS provides one ground-truth mask for every scene in LLFF. SPIn-NeRF and 3D-OVS provide masks for multiple images in every scene. Shiny dataset does not contain any ground-truth masks so we create our own ground-truth mask. For evaluation, we generate 2D foreground masks by rendering the Gaussians from the desired viewpoint. We use pixel classification accuracy (Acc) and foreground intersection-over-union (IoU) for evaluating the segmentation masks.

Following NVOS, we also assess the photo-realistic appearance of the segmented object by rendering it against a black background. We trim both the rendered image and the ground-truth image to the foreground object by applying a bounding box that fits the ground-truth mask. This prevents the evaluation from being biased by the background, especially when the object of interest is relatively small. The metrics we report are PSNR, SSIM [53], LPIPS [60].

Implementation details: To obtain segmentation masks from the user inputs (used in Section 3.3), we leverage the advancements in video segmentation models. The user selects the foreground objects on $\mathbf{I}_0$, and we obtain dense masks for multiple views using SAM-Track [9]. Note that the use of the video segmentation model is done to enhance the performance further and our method can also work with a single image mask (Table 4). We use KD-Tree for efficiently finding the k nearest neighbors to construct the edges between the nodes.

For all the evaluations, we resize the longer image size to 1008, as commonly practiced in novel-view synthesis. We optimize 3DGS model for each of the considered scenes, without making any changes to the original 3DGS code. For coarse splatting, we keep the cut-off threshold $\tau = 0.9$ for front-facing views and $\tau = 0.3$ for the 360° inward-facing scenes. This disparity stems because parts of objects might not be observed from every viewpoint for the latter and also because of the relative ineffectiveness of video tracking for inward-facing scenes (Figure 8). For graph cut, we keep $\gamma = 0.1$ for neighboring pairwise position weights and $\gamma = 1$ for all other weights, the number of neighbors for every node as 10, and the number of clusters for high-confidence nodes as 4 for sink and 1 for source. λ, λ_n, λ_u can be adjusted depending on the scene and the quality of coarse splatting but generally, $\lambda_n = \lambda_u = 1$ and $\lambda = 0.5$ give decent results.

Baselines: Our comparison includes a selection of baseline models such as NVOS [45], MVSeg [37], Interactive segmentation of radiance fields (ISRF) [18], Segment Anything in 3D with NeRFs (SA3D) [7], Segment Any 3D Gaussians (SAGA) [6], Segment Anything in 3D Gaussians (SAGD) [22], Gaussian Grouping [55], and LangSplat [44]. Unlike our approach, SAGA, Gaussian Grouping, and LangSplat alter the Gaussian optimization process by learning additional features per Gaussian that increases the optimization time (Table 9). SAGD is a concurrent work also designed for 3DGS segmentation and has not yet been published. Thus, their results may be subject to change. SAGD, similar to our approach, does not require any segmentation-aware training and uses a cross-view label voting approach to segment selected objects. All the baselines allow for selecting objects using clicks, except LangSplat, for which we use text queries. Further details on baseline implementation are provided in Appendix A.1.

5 Results

5.1 Quantitative results

Dataset from NVOS: We take the seven scenes from LLFF dataset used in NVOS. NVOS contains a reference image with input scribbles and a target view with an annotated 2D segmentation mask. As shown in Table 1, GaussianCut outperforms other approaches. Unlike ISRF, SAGA, Gaussian Grouping, and LangSplat, GaussianCut works on pretrained representations and does not require any changes to the training process. Owing to the fast rasterization, 3DGS-based approaches can also

Table 1: Quantitative results for 2D mask segmentation on NVOS dataset [45].

Method	IoU (%)↑	Acc (%)↑
graph cut (3D) [45, 46]	39.4	73.6
NVOS [45]	70.1	92.0
ISRF [18]	83.8	96.4
SA3D [7]	90.3	98.2
SAGD [22]	72.1	91.7
SAGA [6]	90.9	98.3
Gaussian Grouping [55]	90.6	98.2
LangSplat [44]	74.0	94.0
GaussianCut (Ours)	**92.5**	**98.4**

Table 2: Quantitative results on the SPIn-NeRF dataset [37].

Method	IoU (%)	Acc (%)
MVSeg [37]	90.4	98.8
ISRF [18]	71.5	95.5
SA3D [7]	92.4	98.8
SAGD [22]	89.7	98.1
SAGA [6]	88.0	98.5
Gaussian Grouping [55]	88.4	99.0
LangSplat [44]	69.5	94.5
GaussianCut (Ours)	**92.9**	**99.2**

Table 3: Object rendering results on NVOS [45].

Metrics	SSIM↑	PSNR (dB)↑	LPIPS↓
graph cut (3D) [45, 46]	0.600	15.03	0.415
NVOS [45]	0.767	18.40	0.213
SA3D [7]	0.794	20.76	0.198
GaussianCut (Ours)	**0.840**	**22.45**	**0.132**

Table 4: Quantitative results on Shiny [54].

Scenes	IoU (%)↑	Acc (%)↑
SA3D [7]	93.3	98.5
SAGD [22]	83.3	84.7
Coarse Splatting	94.3	99.4
GaussianCut (Ours)	**95.0**	**99.5**

render foreground Gaussians in real-time. To compare the rendering quality of the segmented objects using 3DGS, we train a NeRF model at the same resolution and segment it using SA3D. Table 3 shows the photo-realistic quality of the foreground image against a black background. Gaussian Splatting provides significant gains over NVOS and SA3D for rendering quality, providing a boost of $+4.05$ dB PSNR and $+1.69$ dB PSNR, respectively.

Dataset from SPIn-NeRF: We compare our model on all scenes from the SPIn-NeRF dataset, which includes four 360° inward-facing scenes and six front-facing scenes. Our model gives an overall better performance compared to other baselines. Compared to MVSeg, on 360° scenes such as lego and truck, GaussianCut provides an absolute IoU gain of 14.3% and 10.5%, respectively. Our model also performs better compared to other 3DGS baselines as shown in Table 2. The 360° scenes for ISRF were run at one-fourth resolution due to memory constraints. We show the scene-wise results in Table 10. Feature-based 3DGS segmentation methods, such as Gaussian Grouping and LangSplat, outperform GaussianCut on certain scenes, but their interactivity can be limited if the optimized features do not delineate the object of interest. We show such cases in Figure 5. Moreover, we also show in Figure 11 that segmentation masks from GaussianCut contain finer details and a better segmentation quality than the ground-truth masks provided in SPIn-NeRF.

Dataset from Shiny: We test the segmentation performance of our model on four scenes from the Shiny dataset: tools, pasta, seasoning, and giants. We create ground-truth masks for 4 test-view images for each scene and compare our model against non-feature learning-based baselines: SA3D and SAGD. We also report the performance of Coarse Splatting (no graph cut) in Table 4. Figure 10 also shows the quality of segmented images for all the scenes.

5.2 Qualitative results

Similar to SA3D, we perform segmentation in three modes: object segmentation, part segmentation, and text-prompting based segmentation. Figure 3 shows object and part segmentation. GaussianCut can retrieve complex objects (such as truck, lego bulldozer, mirror) precisely. It can segment smaller part segmentation (manhole cover, lego wheels, and socket wrenches). Our method can also retrieve multiple objects together (socket wrenches and metallic bowl are extracted together).

Table 5: Ablation of the energy function averaged over the seven scenes from LLFF dataset.

Energy	NVOS
Single	86.6
Coarse	91.2
w/o *n-link* spatial similarity	92.4
w/o *n-link* color similarity	92.3
w/o *t-link* cluster similarity	91.5
GaussianCut (Ours)	92.5

Figure 4 demonstrates the performance of objection selection using text input. We do a qualitative comparison of GaussianCut with ISRF, SA3D, and SAGD. Feature-based 3DGS methods run into memory issues for this scene. Out of these, SA3D and SAGD also use a text-based prompt to segment the plant. ISRF uses stroke for segmentation. GaussianCut retrieves finer details in the plant with a higher perceptual quality. We also show the rendered image from different viewpoints. It can also be seen that coarse splatting (before the Gaussian graph cut) misses finer details, such as the decorations on the plant, which can be retrieved using GaussianCut.

5.3 Ablation and sensitivity study

Number of views: To obtain the 2D segmentation masks, we run the camera on a fixed trajectory and get rendering from different viewpoints (spiral trajectory for front-facing and circular for 360° inward scene). We limit the number of frames to 30 for front-facing and 40 for 360° scenes. Using more segmentation masks can boost performance, however it might not be always preferred, especially for scenes with a large number of training views. SAM-Track can also handle segmentation for unordered multi-frame images. Table 6 shows the effect of varying the number of masks on two scenes from the

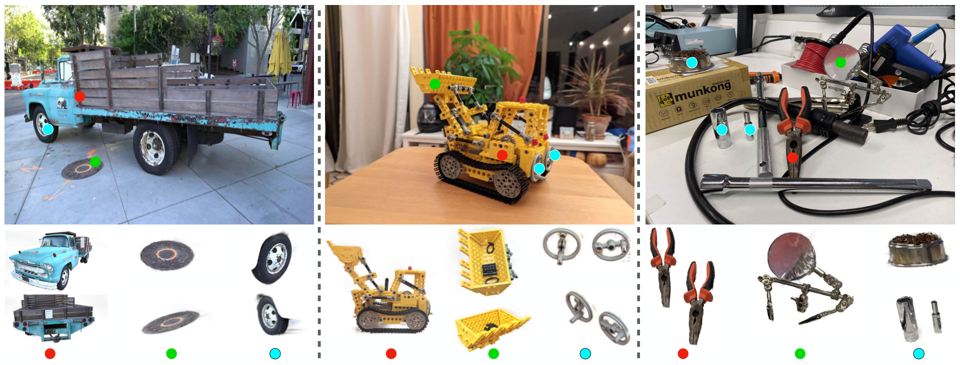

Figure 4: Qualitative comparison: 3D segmentation results of GaussianCut using text on 360-garden [1] scene. Compared to ISRF [18], SA3D [7], SAGD [22], GaussianCut segment contain finer details. The graph cut component of GaussianCut also retrieves fine details (like decorations on the plant) that are missed in coarse splatting.

SPIn-NeRF dataset (images used were unordered). Since 3DGS offers fast rasterization, the overall time cost for segmentation does not grow linearly with the number of masks as the time taken for segmentation dominates. We also show the qualitative performance with a single mask (no video segmentation model) and just scribbles (no image segmentation model) in Figure 13.

Sensitivity of each term in the graph cut energy function: In order to understand the contribution of each term in the energy function, Table 5 shows the average IoU on the NVOS dataset with each term removed from Equation 4 and 5. Each term contributes to the overall performance and the cluster similarity, in particular, gives a significant boost.

Sensitivity of graph cut hyper-parameters: We test the sensitivity of our Gaussian graph cut algorithm on the number of neighbors (number of edges for each node) and the number of high-confidence clusters. As the number of neighbors increases, the number of edges in the graph also increases (so does the time taken for graph cut). As seen in Table 7, adding more edges can help in modeling more long-distance correlations. However, after a limit, the effects of adding more edges diminish. Adding a large number of clusters for the high-confidence nodes, in Table 8, does not affect the performance drastically and the optimal number can vary depending on the scene. We show sensitivity to other hyper-parameters in Appendix E.

6 Discussion

Our results demonstrate that 3DGS allows for direct segmentation using a pretrained model. Developments in 2D segmentation and tracking have played a crucial role in 3D segmentation. We observe that GaussianCut not only generates 3D consistent masks but also improves the segmentation quality of 2D masks by capturing more details (Figure 12). This is more prominent for 360° scenes, where the tracker can partially or fully miss the object of interest (Figure 8).

Table 6: Performance of GaussianCut with varying the number of views passed to the video segmentation models. The number in parenthesis is the percentage of total views for the scene.

Number of views	5 (10%)	9(20%)	21 (50%)	43 (100 %)
Coarse Splatting on Fortress	96.1	96.3	96.5	96.8
GaussianCut on Fortress	97.7	97.8	97.8	97.9
Time Cost (s)	51	55	59	71
Number of views	11 (10%)	21(20%)	51 (50%)	102 (100 %)
Coarse Splatting on Lego	85.5	88.0	88.4	88.9
GaussianCut on Lego	87.3	89.1	89.2	89.2
Time Cost (s)	58	62	72	90

Table 7: Ablation on the number of neighbors.

#Neighbors	1	10	50	100
Horns	91.9	93.6	93.8	94.3
Time (s)	18	57	209	410
Truck	93.3	95.7	95.3	95.2
Time (s)	32	96	393	738

Table 8: Ablation on the number of clusters for high-confidence nodes.

#Clusters	1	5	10	20
Fortress	97.3	97.8	97.6	97.5
Horns	93.8	93.9	94.0	94.0
Truck	95.6	95.7	95.6	95.5

Time requirement: Since we use pretrained 3DGS models, the optimization time for the Gaussians remains the same as in [23] (it took under 15 minutes for every scene we used). For inference, masked rasterization of Gaussians is fast and the time taken for graph cut grows roughly linearly with the number of Gaussians. Table 9 shows a detailed breakdown of time taken in each step: preprocessing (obtaining the features from 2D image/video segmentation models), fitting time (3DGS optimization time), and segmentation time (time taken to obtain the segmented output). Compared to feature-based methods, like Gaussian Grouping, LangSplat, and SAGA, our method does not require any alteration to the fitting process and, therefore, has a shorter fitting time. While the segmentation time is higher for GaussianCut, it still has a much shorter overall time requirement. All reported times have been obtained using an NVIDIA RTX 4090 GPU and an Intel Core i9-13900KF CPU.

Table 9: Comparison of segmentation time (in seconds) on the NVOS benchmark.

Method	Preprocessing time	Fitting time	Segmentation time	Performance (IoU)
SAGA [6]	71.17 ± 22.74	1448.50 ± 205.07	0.35 ± 0.05	90.9
Gaussian Grouping [55]	13.72 ± 4.63	2096.07 ± 251.96	0.55 ± 0.09	90.6
LangSplat [44]	2000.34 ± 1222.19	1346.92 ± 247.00	0.82 ± 0.02	74.0
Coarse Splatting (Ours)	6.11 ± 0.38	510.97 ± 106.42	19.48 ± 4.31	91.2
GaussianCut (Ours)	6.11 ± 0.38	510.97 ± 106.42	88.77 ± 33.68	92.5

Memory requirement: While 3DGS has a higher footprint than NeRF-based models, several recent works reduce the memory footprint with limited loss of quality [12, 38–40]. Our method only stores one additional parameter w_g for every Gaussian and is less memory-intensive than methods requiring learning a feature field [6, 13].

Limitations: GaussianCut can address some inaccuracies in 2D video segmentation models, but it may still lead to partial recovery when the initial mask or tracking results are significantly off (Figure 7). By segmenting directly at the level of Gaussians, the resulting segmentation boundaries may inherit inaccuracies of the underlying representation that are not visible in the unsegmented scene due to alpha-blending. While GaussianCut does not require additional scene optimization, our method can still take up to a few minutes for the graph cut component, which makes the segmentation not real-time. The implementation could be improved by applying graph cut on a subset of Gaussians. We leave this to future work. Additionally, extending our energy function to include a feature similarity term (in equation 3) is another potential improvement. We also discuss some failure cases in section B.

7 Conclusion

In this paper, we introduce GaussianCut, a novel approach that taps into the underlying explicit representation of 3D Gaussian Splatting to accurately delineate 3D objects. Our approach takes in an optimized 3DGS model along with sparse user inputs on any viewpoint from the scene. We use video segmentation models to propagate the mask along different views and then track the Gaussians that splat to these masks. In order to enhance the precision of partitioning the Gaussians, we model them as nodes in an undirected graph and devise an energy function that can be minimized using graph cut. Our approach shows the utility of explicit representation provided by 3DGS and can also be extended for downstream use cases of 3D editing and scene understanding.

8 Acknowledgment

This work was supported by the Sony Research Award Program and NSERC Alliance.

References

[1] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *CVPR*, 2022.

[2] Yuri Boykov and Vladimir Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *TPAMI*, 2004.

[3] Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. *TPAMI*, 2001.

[4] Yuri Y Boykov and M-P Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in nd images. In *ICCV*, 2001.

[5] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *ICCV*, 2021.

[6] Jiazhong Cen, Jiemin Fang, Chen Yang, Lingxi Xie, Xiaopeng Zhang, Wei Shen, and Qi Tian. Segment any 3d gaussians. *arXiv*, 2023.

[7] Jiazhong Cen, Zanwei Zhou, Jiemin Fang, Wei Shen, Lingxi Xie, Dongsheng Jiang, Xiaopeng Zhang, Qi Tian, et al. Segment anything in 3d with nerfs. *NeurIPS*, 2023.

[8] Yiwen Chen, Zilong Chen, Chi Zhang, Feng Wang, Xiaofeng Yang, Yikai Wang, Zhongang Cai, Lei Yang, Huaping Liu, and Guosheng Lin. Gaussianeditor: Swift and controllable 3d editing with gaussian splatting. In *CVPR*, 2024.

[9] Yangming Cheng, Liulei Li, Yuanyou Xu, Xiaodi Li, Zongxin Yang, Wenguan Wang, and Yi Yang. Segment and track anything. *arXiv*, 2023.

[10] Özgün Çiçek, Ahmed Abdulkadir, Soeren S Lienkamp, Thomas Brox, and Olaf Ronneberger. 3d u-net: learning dense volumetric segmentation from sparse annotation. In *MICCAI*, 2016.

[11] Bin Dou, Tianyu Zhang, Yongjia Ma, Zhaohui Wang, and Zejian Yuan. Cosseggaussians: Compact and swift scene segmenting 3d gaussians. *arXiv*, 2024.

[12] Zhiwen Fan, Kevin Wang, Kairun Wen, Zehao Zhu, Dejia Xu, and Zhangyang Wang. Lightgaussian: Unbounded 3d gaussian compression with 15x reduction and 200+ fps. *arXiv*, 2023.

[13] Pedro F Felzenszwalb and Daniel P Huttenlocher. Efficient graph-based image segmentation. *IJCV*, 2004.

[14] James D Foley. *Computer graphics: principles and practice*, volume 12110. Addison-Wesley Professional, 1996.

[15] Lester Randolph Ford and Delbert Ray Fulkerson. *Flows in networks*. Princeton university press, 2015.

[16] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *CVPR*, 2022.

[17] Alberto Garcia-Garcia, Sergio Orts-Escolano, Sergiu Oprea, Victor Villena-Martinez, and Jose Garcia-Rodriguez. A review on deep learning techniques applied to semantic segmentation. *arXiv*, 2017.

[18] Rahul Goel, Dhawal Sirikonda, Saurabh Saini, and PJ Narayanan. Interactive segmentation of radiance fields. In *CVPR*, 2023.

[19] Andrew V Goldberg and Robert E Tarjan. A new approach to the maximum-flow problem. *JACM*, 1988.

[20] Leo Grady. Random walks for image segmentation. *TPAMI*, 2006.

[21] Haoyu Guo, He Zhu, Sida Peng, Yuang Wang, Yujun Shen, Ruizhen Hu, and Xiaowei Zhou. Sam-guided graph cut for 3d instance segmentation. *arXiv*, 2023.

https://doi.org/10.52202/079017-2830

[22] Xu Hu, Yuxi Wang, Lue Fan, Junsong Fan, Junran Peng, Zhen Lei, Qing Li, and Zhaoxiang Zhang. Semantic anything in 3d gaussians. *arXiv*, 2024.

[23] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), 2023.

[24] Justin Kerr, Chung Min Kim, Ken Goldberg, Angjoo Kanazawa, and Matthew Tancik. Lerf: Language embedded radiance fields. In *ICCV*, 2023.

[25] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. *arXiv*, 2023.

[26] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ToG*, 2017.

[27] Sosuke Kobayashi, Eiichi Matsumoto, and Vincent Sitzmann. Decomposing nerf for editing via feature field distillation. *NeurIPS*, 2022.

[28] Christoph Lassner and Michael Zollhofer. Pulsar: Efficient sphere-based neural rendering. In *CVPR*, 2021.

[29] Kunhao Liu, Fangneng Zhan, Jiahui Zhang, Muyu Xu, Yingchen Yu, Abdulmotaleb El Saddik, Christian Theobalt, Eric Xing, and Shijian Lu. Weakly supervised 3d open-vocabulary segmentation. *NeurIPS*, 2023.

[30] Quande Liu, Youpeng Wen, Jianhua Han, Chunjing Xu, Hang Xu, and Xiaodan Liang. Open-world semantic segmentation via contrasting and clustering vision-language embedding. In *ECCV*, 2022.

[31] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. *arXiv*, 2023.

[32] Yichen Liu, Benran Hu, Junkai Huang, Yu-Wing Tai, and Chi-Keung Tang. Instance neural radiance field. In *ICCV*, 2023.

[33] Yichen Liu, Benran Hu, Chi-Keung Tang, and Yu-Wing Tai. Sanerf-hq: Segment anything for nerf in high quality. *arXiv*, 2023.

[34] Ben Mildenhall, Pratul P Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *TOG*, 2019.

[35] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

[36] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *3DV*, 2016.

[37] Ashkan Mirzaei, Tristan Aumentado-Armstrong, Konstantinos G Derpanis, Jonathan Kelly, Marcus A Brubaker, Igor Gilitschenski, and Alex Levinshtein. Spin-nerf: Multiview segmentation and perceptual inpainting with neural radiance fields. In *CVPR*, pages 20669–20679, 2023.

[38] KL Navaneet, Kossar Pourahmadi Meibodi, Soroush Abbasi Koohpayegani, and Hamed Pirsiavash. Compact3d: Compressing gaussian splat radiance field models with vector quantization. *arXiv*, 2023.

[39] Simon Niedermayr, Josef Stumpfegger, and Rüdiger Westermann. Compressed 3d gaussian splatting for accelerated novel view synthesis. *arXiv*, 2023.

[40] Panagiotis Papantonakis, Georgios Kopanas, Bernhard Kerbl, Alexandre Lanvin, and George Drettakis. Reducing the memory footprint of 3d gaussian splatting. *PACMCGIT*, 2024.

[41] Bo Peng, Lei Zhang, and David Zhang. A survey of graph theoretical approaches to image segmentation. *Pattern Recognition*, 2013.

[42] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, 2017.

[43] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *NeurIPS*, 2017.

[44] Minghan Qin, Wanhua Li, Jiawei Zhou, Haoqian Wang, and Hanspeter Pfister. Langsplat: 3d language gaussian splatting. *arXiv*, 2023.

[45] Zhongzheng Ren, Aseem Agarwala, Bryan Russell, Alexander G Schwing, and Oliver Wang. Neural volumetric object selection. In *CVPR*, 2022.

[46] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. " grabcut" interactive foreground extraction using iterated graph cuts. *ACM transactions on graphics (TOG)*, 23(3):309–314, 2004.

[47] Myrna C Silva, Mahtab Dahaghin, Matteo Toso, and Alessio Del Bue. Contrastive gaussian clustering: Weakly supervised 3d scene segmentation. *arXiv*, 2024.

[48] Karl Stelzner, Kristian Kersting, and Adam R Kosiorek. Decomposing 3d scenes into objects via unsupervised volume segmentation. *arXiv*, 2021.

[49] Wenming Tang and Guoping Qiu. Dense graph convolutional neural networks on 3d meshes for 3d object segmentation and classification. *Image and Vision Computing*, 2021.

[50] Vadim Tschernezki, Iro Laina, Diane Larlus, and Andrea Vedaldi. Neural feature fusion fields: 3d distillation of self-supervised 2d image representations. In *3DV*, 2022.

[51] Zhuowen Tu. Auto-context and its application to high-level vision tasks. In *CVPR*, 2008.

[52] Suhani Vora, Noha Radwan, Klaus Greff, Henning Meyer, Kyle Genova, Mehdi SM Sajjadi, Etienne Pot, Andrea Tagliasacchi, and Daniel Duckworth. Nesf: Neural semantic fields for generalizable semantic segmentation of 3d scenes. *arXiv*, 2021.

[53] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *TIP*, 2004.

[54] Suttisak Wizadwongsa, Pakkapon Phongthawee, Jiraphon Yenphraphai, and Supasorn Suwajanakorn. Nex: Real-time view synthesis with neural basis expansion. In *CVPR*, 2021.

[55] Mingqiao Ye, Martin Danelljan, Fisher Yu, and Lei Ke. Gaussian grouping: Segment and edit anything in 3d scenes. *arXiv*, 2023.

[56] Lin Yen-Chen, Pete Florence, Jonathan T Barron, Tsung-Yi Lin, Alberto Rodriguez, and Phillip Isola. Nerf-supervision: Learning dense object descriptors from neural radiance fields. In *ICRA*, 2022.

[57] Nida M Zaitoun and Musbah J Aqel. Survey on image segmentation techniques. *Procedia Computer Science*, 65:797–806, 2015.

[58] Chaoning Zhang, Fachrina Dewi Puspitasari, Sheng Zheng, Chenghao Li, Yu Qiao, Taegoo Kang, Xinru Shan, Chenshuang Zhang, Caiyan Qin, Francois Rameau, et al. A survey on segment anything model (sam): Vision foundation model meets prompt engineering. *arXiv*, 2023.

[59] Hao Zhang, Fang Li, and Narendra Ahuja. Open-nerf: Towards open vocabulary nerf decomposition. In *WACV*, 2024.

[60] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.

[61] Shuaifeng Zhi, Tristan Laidlow, Stefan Leutenegger, and Andrew J Davison. In-place scene labelling and understanding with implicit scene representation. In *ICCV*, 2021.

[62] Kaichen Zhou, Lanqing Hong, Enze Xie, Yongxin Yang, Zhenguo Li, and Wei Zhang. Serf: Fine-grained interactive 3d segmentation and editing with radiance fields. *arXiv*, 2023.

[63] Shijie Zhou, Haoran Chang, Sicheng Jiang, Zhiwen Fan, Zehao Zhu, Dejia Xu, Pradyumna Chari, Suya You, Zhangyang Wang, and Achuta Kadambi. Feature 3dgs: Supercharging 3d gaussian splatting to enable distilled feature fields. *arXiv*, 2023.

[64] Xueyan Zou, Jianwei Yang, Hao Zhang, Feng Li, Linjie Li, Jianfeng Wang, Lijuan Wang, Jianfeng Gao, and Yong Jae Lee. Segment everything everywhere all at once. *NeurIPS*, 2024.

A Implementation details

We run our segmentation algorithm on 3D Gaussian Splatting representations, following the code provided by Kerbl *et al.* [23]. All scenes are optimized for 30,000 steps using the default parameters. We use SAM-Track [9] as the video segmentation model.

For the datatset used in NVOS [45], we use the provided reference image with the user scribbles for a fair comparison. For the SPIn-NeRF dataset [37], we use the first image in the directory as the reference image for the user input. The scenes reported throughout the paper are selected from the following datasets:

- NVOS (LLFF [34] subset): flower, fortress, fern, horns, orchids, trex, leaves
- SPIn-NeRF (collection from some widely used datasets [16, 26, 34, 35, 56]): orchids, leaves, fortress, horns, truck, lego bulldozer
- Shiny [54]: giants, tools, seasoning, pasta
- Mip-NeRF [1]: cycle, garden, bonsai
- LERF [24]: figurines
- 3D-OVS [29]: lawn, sofa, bed, bench, room

Mask evaluation: The Gaussians optimized for 3DGS can have ambiguous structures as they are not geometrically constrained. When partitioning the Gaussians as foreground or background, the boundary Gaussians can appear as having shard-like artifacts (or "spiky" Gaussians). Since the goal of this work is to effectively characterize a Gaussian as foreground or background, we render the foreground mask by overriding the colors of background Gaussians. To generate object assets, our algorithm can be combined with Gaussian decomposition based approached [22].

A.1 Baseline implementation details

A.1.1 Gaussian Grouping

Gaussian Grouping [55] learns an additional feature per Gaussian that can be used to group Gaussians belonging to the same object. We use SAM-Track to get all the 2D segmentation masks. While the default implementation of Gaussian Grouping uses DEVA [9] masks, we chose SAM-Track for both GaussianCut and Gaussian Grouping to maintain consistency in mask quality across the methods. Similar to GaussianCut, we use clicks to segment objects in NVOS and SPIn-NeRF benchmarks. However, Gaussian Grouping uses a mask for segmenting everything and can, therefore, sometimes

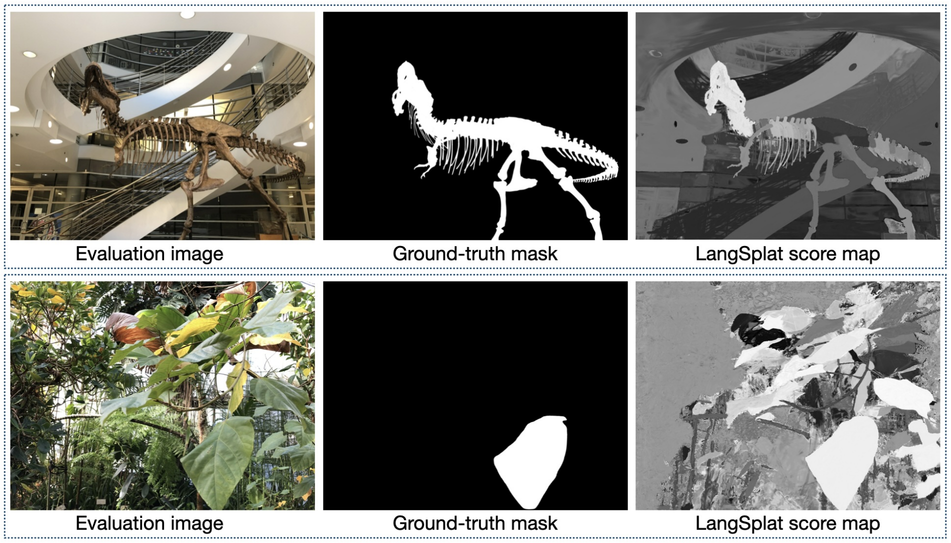

Figure 5: Limitation of LangSplat on Trex and Leaves scenes from NVOS benchmark. Parts of the trex can not be extracted in the top row. In the bottom row, background leaves are also selected along with front leaf.

produce over-segmented object segments. To handle such cases and obtain the final segmentation, we aggregate all the segments that constitute the object. We also use the same number of 2D segmented masks as the number of training views. In contrast, for GaussianCut, we limit the number of masks to 30 for front-facing scenes and 40 for 360° inward-facing scenes. To prevent memory issues in Gaussian Grouping, we restrict the total number of Gaussians across all scenes to 2M.

A.1.2 LangSplat

We obtained all 2D features at feature level "part" for LangSplat [44]. Since we could not use clicks and scribbles to obtain the segment, we have used text queries. We tried multiple text queries for each scene and reported the results on the best performing query. For certain scenes, text queries can constrain the selection of an object. For instance, in Figure 5, multiple instances of the leaves get a high relevancy score when segmenting the front leaf.

B Limitations

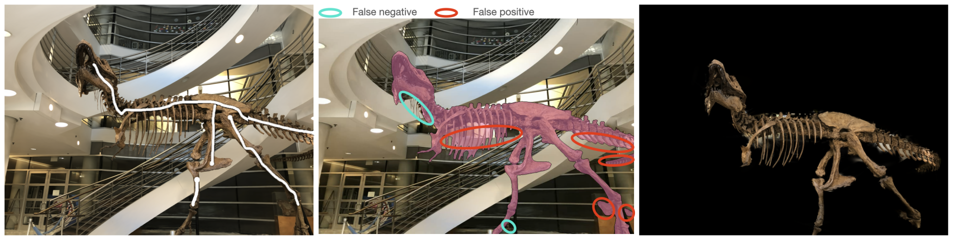

Figure 6: Trex scene from the LLFF [34] dataset. **Left:** reference image with scribbles provided by NVOS [45]. **Center:** segmentation mask provided by SAM [25]. The obtained mask misses finer details and also groups multiple intricate features together. **Right:** segmented using GaussianCut. While it adds finer details (like near the ribs), the tail still contains some background elements.

The performance of our method depends on the robustness of image and video segmentation models. For all the scenes tested, we do not tune SAM-Track and use the default settings. SAM-Track (built on SAM) can provide coarse segments, even on the reference image, especially for irregular scenes, as shown in Figure 6. GaussianCut improves the segmentation details of SAM but there still remains scope for improving the segmentation performance further for the more intricate patterns.

Similar to image-segmentation models, video-segmentation models can also have inaccurate segmentation masks. This issue is more pronounced in complex 360° scenes, where an object can entirely

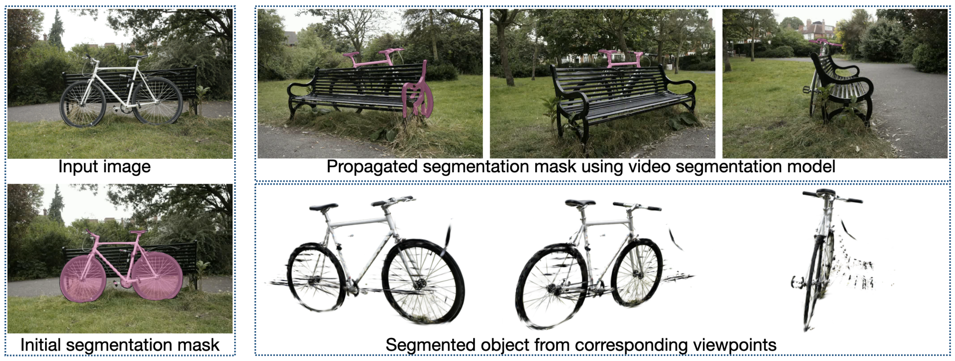
Input image
Propagated segmentation mask using video segmentation model
Initial segmentation mask
Segmented object from corresponding viewpoints

Figure 7: SAM-Track fails to capture major sections of the bicycle when its orientation significantly deviates from the initial position. Even in the reference image, the segmentation mask omits finer details such as the bicycle wheel rims, pedals, and bottle holder. GaussianCut improves segmentation by eliminating substantial portions of the bench to isolate the bicycle, and it partially restores the visibility of the wheel rims. Despite these improvements, the segmentation remains imprecise.

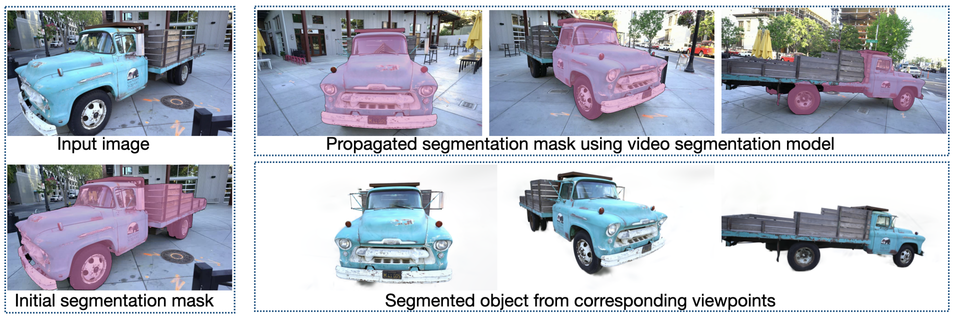
Input image
Initial segmentation mask
Propagated segmentation mask using video segmentation model
Segmented object from corresponding viewpoints

Figure 8: GaussianCut precisely retrieves fine details, such as the mirrors on the front of the truck, even in instances where video-segmentation model struggles to maintain consistency across different views in the scene.

change its orientation, which can lead the trackers to fail in segmenting all views effectively. We illustrate two instances in Figure 7 and 8, where GaussianCut corrects the inaccuracies of SAM-Track with varying levels of effectiveness.

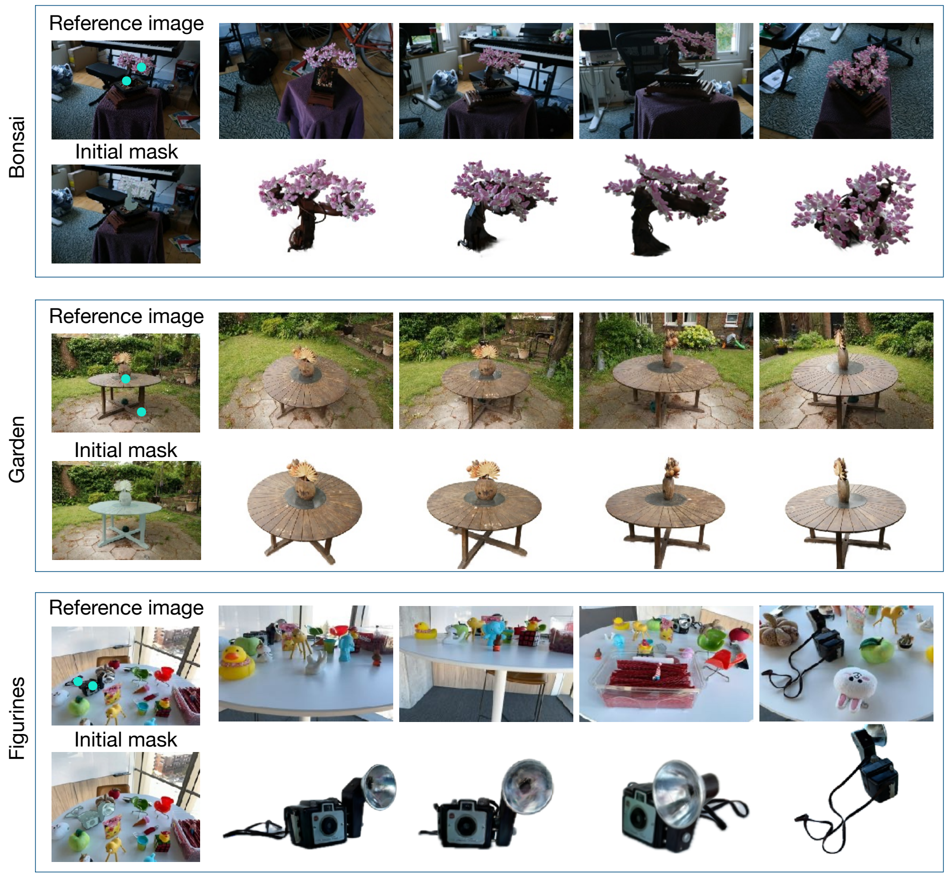

Figure 9: Visualization of selected objects on the Mip-NeRF and LERF dataset. Initial object selection, based on point clicks, and the reference image is shown on the left.

C More visualization results

We present additional segmentation visualizations for 360° inward scenes taken from Mip-NeRF [1] and LERF datasets [24] in Figure 9. GaussianCut segments complex and irregular features, including the leaves and wire in the bonsai lego, the detailed decorations of the plant and table in the garden scene, as well as the cord, viewfinder, and flashbulb of the vintage camera.

D Additional results

D.1 Shiny dataset

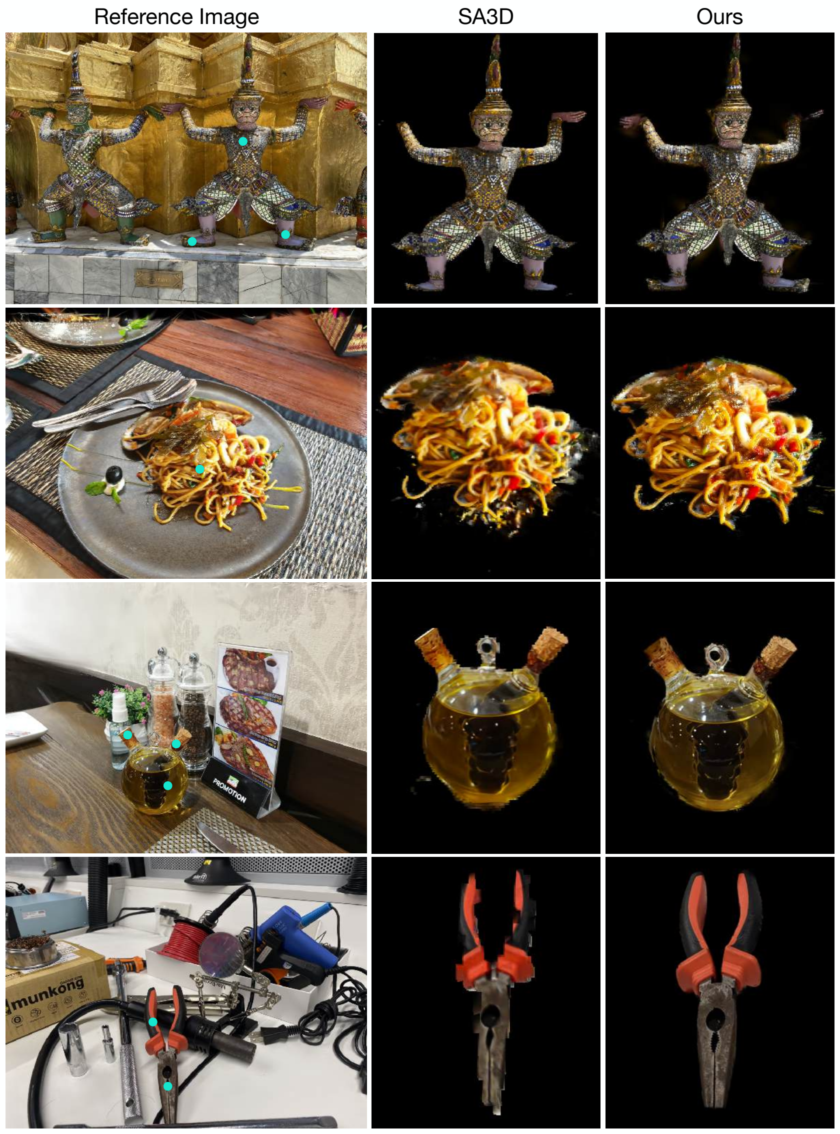

Figure 10: Qualitative results on the Shiny dataset, compared against SA3D [7]. The points used as user inputs are highlighted in the reference image.

Scenes in Shiny [54] dataset have complex shapes of objects (pasta), are placed in more cluttered environments (tools), or possess a subtler distinction between the foreground and background colors (giants). We test four scenes from the Shiny dataset and label four images from each scene as ground-truth. Table 4 shows the improvement of GaussianCut against coarse splitting and SA3D [7] with an overall +0.7% and +1.7% absolute gain in foreground mIoU, respectively. Qualitative results from the dataset are shown in Figure 10. GaussianCut can retrieve fine details (like the strands of the pasta) more accurately.

D.2 SPIn-NeRF

Table 10 shows the performance of GaussianCut on each scene of the SPIn-NeRF dataset. Furthermore, we show in Figure 11 that the quality of the mask produced by GaussianCut contains finer details than the ground-truth labels from SPIn-NeRF.

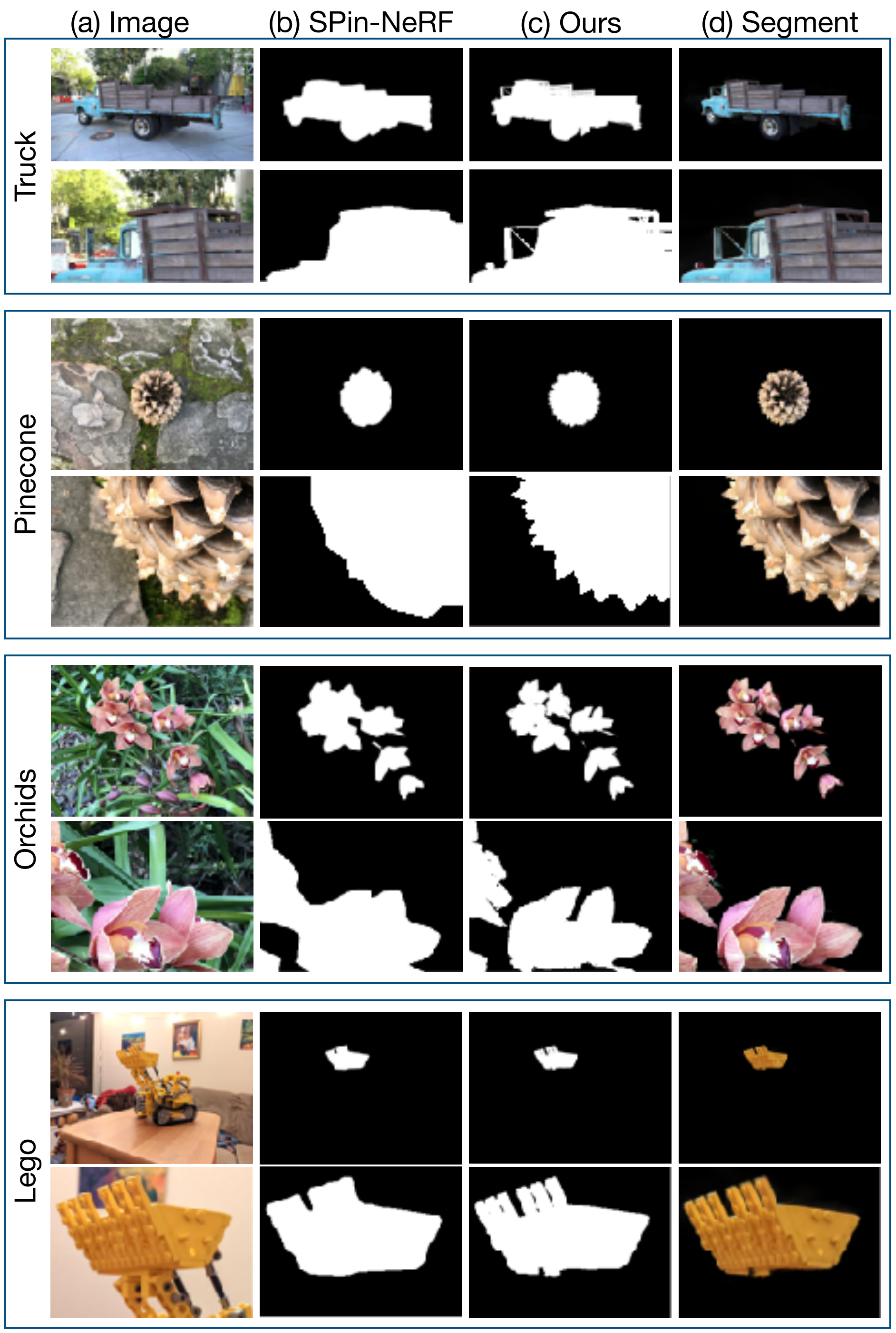

Figure 11: Qualitative comparison of segmentation masks obtained from GaussianCut and the ground-truth used in SPIn-NeRF dataset.

Table 10: Quantitative results on each scene in the SPIn-NeRF dataset.

Scenes	MVSeg [37]	ISRF [18]	SA3D [6]	SAGD [22]	Gaussian Grouping [55]	LangSplat [44]	GaussianCut)
Orchids	92.7	84.2	83.6	85.4	89.4	88.2	87.8
Leaves	94.9	87.2	97.2	87.9	96.7	46.5	96.3
Fern	94.3	83.1	97.1	92.0	97.3	97.3	96.1
Room	95.6	67.6	88.2	86.5	96.2	64.5	95.2
Horns	92.8	84.3	94.5	91.1	92.5	87.1	93.6
Fortress	97.7	92.6	98.3	96.6	97.8	95.8	97.7
Fork	87.9	19.9	89.4	83.4	90.0	70.1	85.4
Pinecone	93.4	60.0	92.9	92.0	92.2	54.4	91.9
Truck	85.2	79.2	90.8	93.0	95.9	53.8	95.7
Lego	74.9	57.3	92.2	88.4	36.0	36.3	89.2
mean	90.9	71.5	92.4	89.7	88.4	69.5	92.9

D.3 3D-OVS

To test the performance of our method using text queries, we test on the 3D-OVS [29] dataset and compare it against Gaussian Grouping [55], LangSplat [44], and Contrastive Gaussian Clustering [47] in Table 11. We use the grounding-DINO integration in SAM-Track to obtain the initial segments. The baseline numbers reported in Table 11 are taken from [47]. We use a different text query for some objects than [47]. This was done to ensure that we have a decent initial mask from SAM-Track as the goal of our work is not to improve language understanding in 3D models.

D.4 Qualitative comparison with 2D segmentation model

The objects segmented by GaussianCut exhibit fine details, as depicted in the masks presented in Figure 12. Although our method uses SAM predictions as an initial mask, the segregation of Gaussians provides information with greater precision compared to SAM alone.

E Additional sensitivity analysis and ablations

E.1 Binary weights for coarse splatting

As mentioned in Section 3.3, the likelihood term w_g for each Gaussian g is obtained by taking a weighted ratio of g's contribution on the masked pixels compared to the total number of pixels it affects. Instead of using weighted assignment, we can also have a hard binary assignment where a g either contributes to a foreground pixel or it doesn't. For the n viewpoints, $\mathcal{I} := \{\mathbf{I}^j\}_{j=1}^n$ that have corresponding masks $\mathcal{M} := \{\mathbf{M}^j\}_{j=1}^n$,

$$w_g = \frac{\sum_j \sum_{\mathbf{p}\in\mathbf{M}^j} \mathbb{I}(T_g^j > 0)}{\sum_j \sum_{\mathbf{p}\in\mathbf{I}^j} \mathbb{I}(T_g^j > 0)}, \tag{6}$$

which reflects the ratio of the number of pixels that g has contributed to in $\mathbf{M}^j$ and $\mathbf{I}^j$. As shown in Table 12, since soft assignment has marginally better performance, it is our default implementation.

E.2 Sensitivity of hyperparameters

We share a default setting in section 4 which performs reasonably well on all our datasets. The sensitivity of each parameter can be very scene-dependent. For instance, in a scene where parts of an object have different colors, a very high weight on the color similarity can affect adversely. We show the effect of λ (controls the pull of neighboring vs terminal edges) and γ (decay constant of the similarity function) on two scenes in Table 13 and Table 14, respectively. The reported metric is IoU.

E.3 Threshold of coarse splatting

For the four 360-degree inward scenes in the SPIn-NeRF benchmark, we show a sweep of the threshold τ (default is 0.3 in our implementation) used for Coarse Splatting. GaussianCut outperforms all the thresholds considered for coarse splatting as shown in Table 15.

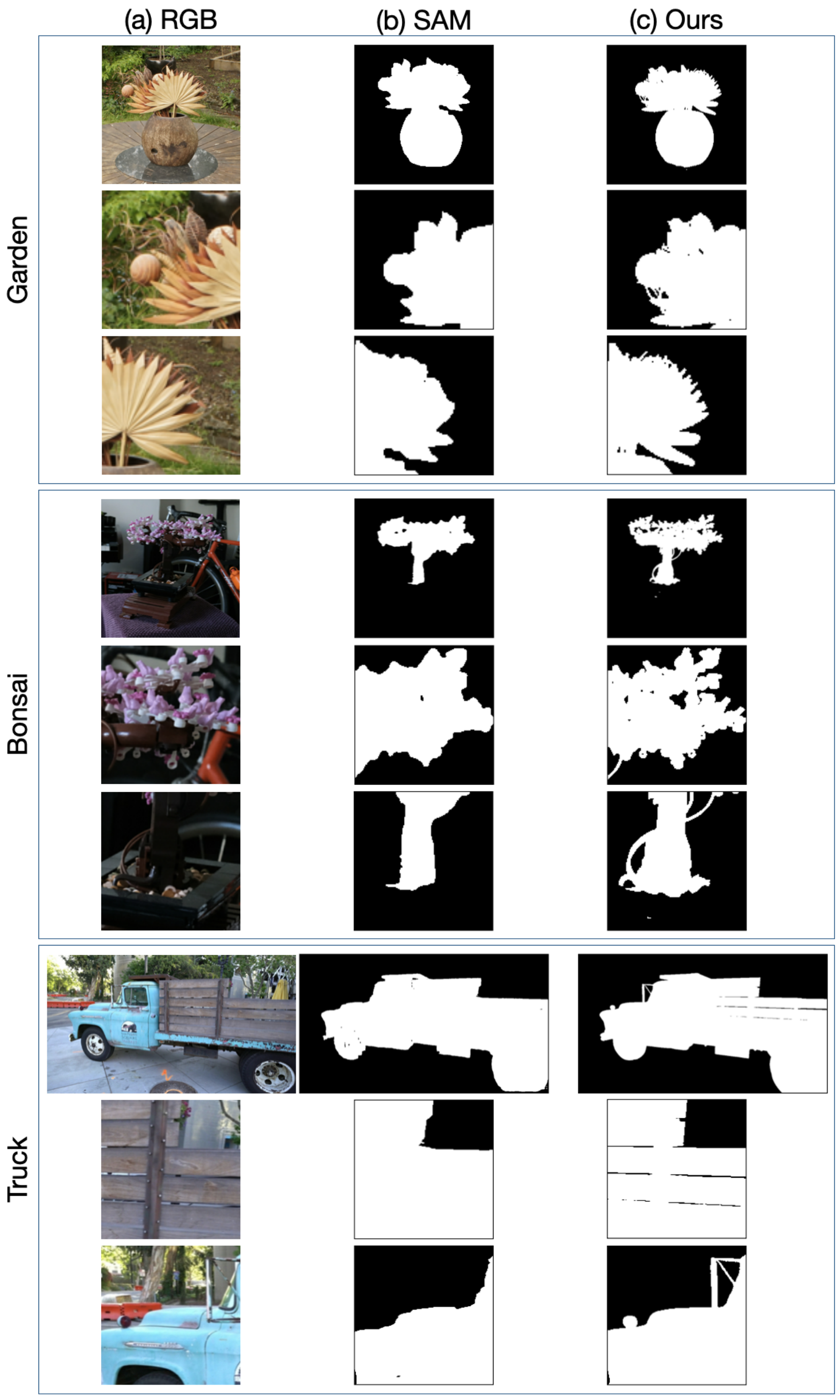

Figure 12: Visualization of segmentation masks from SAM and GaussianCut.

Table 11: Quantitative evaluation on 3D-OVS [29] dataset. CGC refers to Contrastive Gaussian Clustering method.

	Item	Gaussian Grouping [55]	LangSplat [44]	CGC [47]	GaussianCut (Ours)
Bed	Banana	96.9	17.8	95.9	**97.2**
	Leather Shoe	97.6	71.4	**97.9**	96.4
	Camera	**95.3**	3.3	85.0	**95.3**
	Hand	**96.8**	6.9	95.5	94.0
	Red Bag	98.5	81.2	98.1	**98.8**
	White Sheet	**99.0**	25.5	**99.0**	98.9
	Average	**97.3**	34.3	95.2	96.8
Bench	Wall	**98.7**	42.6	**98.7**	97.7
	Wood	97.8	88.0	**98.0**	96.2
	Egg Tart	95.4	87.4	**97.2**	93.2
	Orange Cat	94.5	96.4	97.3	**97.6**
	Green grape	0.0	93.7	95.5	94.6
	Offroad car	93.7	93.7	93.6	**95.2**
	Doll	36.0	92.1	92.2	**93.2**
	Average	73.7	84.8	**96.1**	95.4
Room	Wall	**99.4**	24.3	43.6	99.1
	Chicken	0.0	**94.7**	91.4	91.2
	Basket	87.9	**98.3**	97.4	96.3
	Rabbit	96.3	45.3	**97.0**	93.8
	Dinosaur	92.7	6.8	**94.9**	93.1
	Baseball	**97.5**	68.4	97.4	95.6
	Average	79.0	56.3	86.9	**94.9**
Sofa	Pikachu	48.8	89.6	0.0	**94.3**
	UNO cards	**95.8**	79.6	95.4	95.5
	Nintendo switch	90.9	89.8	92.6	93.7
	Gundam	16.1	79.5	76.9	**91.0**
	Xbox controller	59.9	67.8	96.1	**97.7**
	Sofa	97.4	0.0	44.0	**98.0**
	Average	68.1	67.7	67.5	**95.0**
Lawn	Apple	**96.3**	94.0	93.8	88.8
	Cap	**98.4**	**98.4**	97.9	92.0
	Stapler	94.7	**96.2**	95.6	88.1
	Headphone	**94.5**	91.7	70.2	84.2
	Hand soap	**96.0**	95.2	93.7	91.3
	Lawn	99.2	**99.5**	99.3	94.1
	Average	**96.5**	95.8	91.8	89.8

Table 12: Comparison of soft and hard weight assignment of w_g.

Scene	Soft assignment	Hard assignment
Fern (NVOS)	83.06	82.56
Fortress (NVOS)	97.97	98.12
Leaves (NVOS)	95.95	95.60
Lego (SPIn-NeRF)	89.18	88.95
Pinecone (SPIn-NeRF)	91.89	91.99

E.4 How important are the 2D segmentation masks?

In order to understand the extent to which the performance of our model depends on the initial 2D segmentation mask, we do the masked rasterization with just scribbles, a single mask, and multi-view masks. Figure 13 shows the segmentation result of Coarse Splatting and GaussianCut. The effectiveness of GaussianCut is heightened further when the initial segmentation mask is sparse. Table 16 also shows the performance improvement when running GaussianCut directly on user scribbles.

Table 13: Performance comparison for different λ values.

Scene	$\lambda = 0.5$	$\lambda = 1$	$\lambda = 2$	$\lambda = 4$
Fortress	97.67	97.99	97.95	97.80
Lego	89.15	89.18	89.18	88.49

Table 14: Performance comparison for different γ values.

Scene	$\gamma = 0.5$	$\gamma = 1$	$\gamma = 2$	$\gamma = 4$
Fortress	96.12	97.95	97.56	96.04
Lego	89.20	89.18	89.18	89.19

Table 15: Coarse splatting baseline with different thresholds.

Threshold	IoU	Acc
Coarse@0.1	88.47 ± 4.85	98.96 ± 0.53
Coarse@0.3	89.67 ± 3.18	98.94 ± 0.72
Coarse@0.5	87.76 ± 3.06	98.45 ± 1.50
Coarse@0.7	83.30 ± 6.04	97.58 ± 2.84
Coarse@0.9	72.13 ± 11.26	96.08 ± 4.69
GaussianCut	90.55 ± 3.76	99.18 ± 0.41

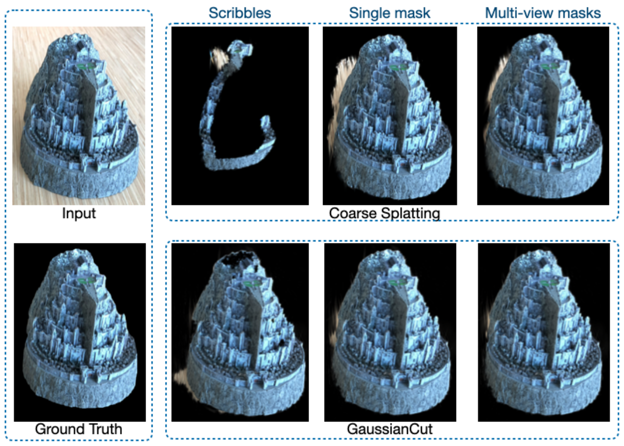

Figure 13: We compare coarse splatting (w/o graph cut) and GaussianCut. Scribbles refer to using direct input, single mask refers to taking the mask from one viewpoint, and multi-view masks refer to using video segmentation. The effectiveness of GaussianCut becomes more prominent when the inputs are sparse.

Table 16: Segmentation performance with just user scribbles for NVOS scenes.

Scene	Scribbles	Scribbles (with graphcut)	GaussianCut
Fern	8.17	47.97	83.06
Flower	7.48	85.30	95.37
Fortress	15.12	95.67	97.95
Trex	6.74	50.44	83.43
Orchids	6.17	85.25	95.80

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The introduction clearly lists the contribution of this work.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We have written about the limitations in the main paper. We have also included a few failure cases in the appendix.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

Justification: Our method does not have any theoretical result.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Yes, we mention all the hyperparameters and the off-the-shelf models we have used. When building on another codebase, we use their default settings (or mention explicitly if something is changed).

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Code is available at: https://github.com/umangi-jain/gaussiancut

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: Yes, we list it in the implementation details.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [NA]

Justification: Factors for stochasticity are less.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)

- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Provided in the discussion section.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification:

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification:

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification:

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification:

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification:

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification:

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification:

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.