LUDVIG: Learning-Free Uplifting of 2D Visual Features to Gaussian Splatting Scenes

Juliette Marrie[1,2] Romain Menegaux[1] Michael Arbel[1] Diane Larlus[2] Julien Mairal[1]
[1] Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP, LJK [2] NAVER LABS Europe

Abstract

We address the problem of extending the capabilities of vision foundation models such as DINO, SAM, and CLIP, to 3D tasks. Specifically, we introduce a novel method to uplift 2D image features into Gaussian Splatting representations of 3D scenes. Unlike traditional approaches that rely on minimizing a reconstruction loss, our method employs a simpler and more efficient feature aggregation technique, augmented by a graph diffusion mechanism. Graph diffusion refines 3D features, such as coarse segmentation masks, by leveraging 3D geometry and pairwise similarities induced by DINOv2. Our approach achieves performance comparable to the state of the art on multiple downstream tasks while delivering significant speed-ups. Notably, we obtain competitive segmentation results using only generic DINOv2 features, despite DINOv2 not being trained on millions of annotated segmentation masks like SAM. When applied to CLIP features, our method demonstrates strong performance in open-vocabulary object segmentation tasks, highlighting the versatility of our approach.[1]

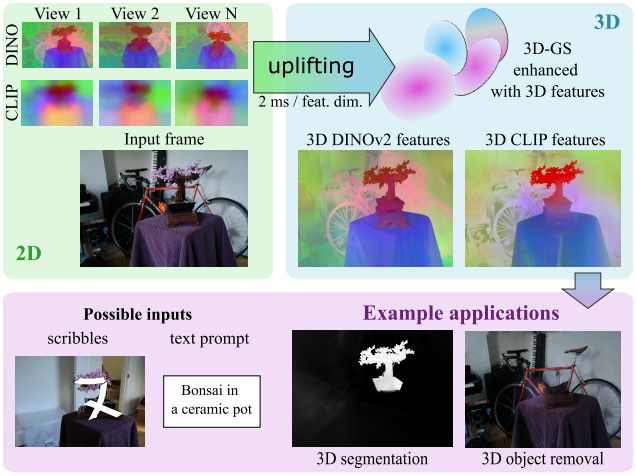


Figure 1. In this paper, we propose a simple, parameter-free method to uplift any 2D visual features (*e.g.*, CLIP or DINO) into a 3D Gaussian Splatting (3D-GS) representation. Uplifting is directly implemented into the rendering process, and takes about 2ms per image and feature dimension. The uplifted features can be leveraged for various 3D tasks, such as segmentation or object removal, based on various inputs, such as scribbles or text prompts.

1. Introduction

Image understanding has made remarkable progress, driven by large pretrained models such as CLIP [36], DINO [2, 33], or SAM [21], also called foundation models. A key factor behind their exceptional generalization capabilities lies in the size of their training datasets, often composed of millions or even billions of samples.

Meanwhile, 3D scene representation has also advanced through machine learning approaches like NeRF [31] and model fitting techniques such as Gaussian Splatting [18]. These methods typically reconstruct scenes from a few dozen views captured from different angles, effectively preserving both appearance and geometric details. However, they are not suited for semantic tasks.

The complementarity of these two families of approaches has been exploited to integrate geometry and image-level semantics extracted by large 2D pretrained models into NeRF or Gaussian Splatting 3D representations. This has led to a surge in methods for language-guided object retrieval [19, 28, 46, 56], scene editing [6, 10, 23], or semantic segmentation [4, 50, 52] in 3D scenes.

The main limitation of these approaches lies in their dependence on optimization, which requires an iterative process to learn a scene-specific 3D representation by minimizing a reprojection error across all training views. While this loss function is intuitive, a faster and more straightforward method for transferring 2D generic visual features to *already trained* Gaussian Splatting 3D models would be preferable, which is one of the purposes of this work.

In this paper, we demonstrate that a simple, learning-free process is highly effective for uplifting 2D features or semantic masks into 3D Gaussian Splatting scenes. This process, which can be viewed as an 'inverse rendering' operation, is both computationally efficient and adaptable to any feature type. We showcase its effectiveness by uplift-

[1] Project page: https://juliettemarrie.github.io/ludvig

ing visual features from DINOv2 [9, 33], semantic masks from SAM [21] and SAM2 [37], and visual features from CLIP [15], enabling language-driven applications. Then, we show that a graph diffusion mechanism [24, 42] is helpful for feature refinement in 3D scenes. This mechanism is rooted in spectral graph theory and spectral clustering [1, 29, 40]. In the context of our work, it transforms coarse segmentation inputs, such as scribbles or alignment scores between visual features and a text query, into accurate 3D segmentation masks without the need for segmentation models such as SAM. When evaluated on segmentation and open-vocabulary object localization tasks, our method matches state-of-the-art performance while being significantly faster than previous optimization-based approaches.

To summarize, our contributions are threefold: (i) We introduce a simple, learning-free uplifting approach that can be directly integrated into the rendering process (Sec. 3.2), achieving state-of-the-art segmentation results when applied to SAM-generated semantic masks (Sec. 4.1); (ii) We introduce a graph diffusion process for 3D segmentation based on uplifted features (Sec. 3.3) which yields competitive results in both foreground/background segmentation with DINOv2 alone and open-vocabulary segmentation when combined with CLIP (Secs. 4.1, 4.2); (iii) We show that our simple uplifting approach can replace more complex and time-consuming feature learning methods and yield strong accuracy gains (Sec. 4.3).

2. Related work

Learning 3D semantic scene representations with NeRF. NeRF [31] uses a multilayer perceptron to predict the volume density and radiance for any given 3D position and viewing direction. Such a representation can naturally be extended to semantic features. The early works N3F [43] and DFF [23] distill DINO 2D (*i.e.*, image-level) features [2] in scene-specific NeRF representations. Kobayashi et al. [23] also distill LSeg [26], a language-driven model for semantic segmentation. Building on this, LERF [19] and 3D-OVS [28] learn 3D CLIP [36] and DINO [2] features jointly for open-vocabulary segmentation. These works were extended to other pretrained models such as latent diffusion models [49] or SAM [21] for segmentation [4, 52].

Learning 3D semantic scene representations with GS. Subsequent work has relied on the Gaussian Splatting method [18], enabling rendering speeds orders of magnitude faster than NeRF-based models. Several tasks have been addressed such as semantic segmentation using SAM [3, 20, 46, 50], language-driven retrieval or editing using CLIP combined with DINO [56] or SAM [46, 49], scene editing using diffusion models [6, 44], and 3D-aware fine-tuning [53]. These works learn 3D semantic representations by minimizing a reprojection loss. As a single scene can be represented by over a million Gaussians, such optimization-based techniques have strong memory and computational limitations. To handle these, FMGS [56] employs a multi-resolution hash embedding (MHE) of the scene for uplifting DINO and CLIP representations, Feature 3DGS [55] learns a 1×1 convolutional upsampler of Gaussians' features distilled from LSeg and SAM's encoder, and LangSplat [35] learns an autoencoder to reduce CLIP feature dimension from 512 to 3. In contrast, our approach requires no learning, which significantly speeds up the uplifting process and reduces the memory requirements.

Direct uplifting of 2D features into 3D. Direct uplifting from 2D to 3D has been explored in prior works. GaussianEditor [6] uplifts 2D SAM masks into 3D to selectively optimize Gaussians for editing tasks, see details in the supplementary (Sec. D.4). Dr. Splat [17] also lifts 2D SAM masks, to create an assignment from object CLIP features to Gaussians. Both approaches are specific to segmentation and are ill-suited for uplifting generic representations. Semantic Gaussians [13] pairs 2D pixels with 3D Gaussians along each pixel's ray based on depth information but relies on a learned 3D convolutional network. OpenGaussian [46] also pairs pixels with Gaussians, but relies on pseudo-features learned with a contrastive loss on SAM masks. In contrast, our approach is simple to implement, applicable to any 2D representation and entirely parameter-free.

Leveraging 3D information to better segment in 2D. Most prior works focusing on semantic segmentation leverage 2D models specialized for this task. The early work of [51] uplifts learned 2D image inpainters by optimizing view consistency over depth and appearance. Subsequent works have mostly relied on uplifting either features from SAM's encoder [55], binary SAM masks [3, 4], or SAM masks automatically generated for all objects in the image [20, 50, 52]. The latter approach is computationally expensive, as it requires querying SAM on a grid of points over the image. It also requires matching inconsistent mask predictions across views, with *e.g.* a temporal propagation model [50] or a hierarchical learning approach [20], which introduces additional computational overhead. In this work, we focus on single instance segmentation and show that our uplifted features are on par with the state of the art [3, 4, 52]. Standing out from prior work uplifting DINO features [12, 19, 23, 28, 43, 49, 56], we show that DINOv2 features can be used on their own for semantic segmentation and rival SAM-based models through a simple graph diffusion process that leverages 3D geometry.

3D CLIP features for open-vocabulary localization. For learning 3D CLIP features, prior works also leverage vision models such as DINO or SAM. DINO is used to regularize and refine CLIP features [19, 28, 41, 56], while SAM is employed for generating instance-level CLIP represen-

tations [17, 35, 46]. These approaches suffer from high computational costs, resorting to dimensionality reduction or efficient multi-resolution embedding representations, and usually run for a total of one to two hours for feature map generation and 3D feature optimization. In contrast, our approach bypasses the high computational cost of gradient-based optimization and, combined with graph diffusion, is an order of magnitude faster than these prior works.

3. Uplifting 2D visual representations into 3D

We now present a simple yet effective method for lifting 2D visual features into 3D using Gaussian Splatting, and discuss its relation with optimization-based techniques.

3.1. Background on Gaussian Splatting

Scene representation. The Gaussian Splatting method consists in modeling a 3D scene as a set of n Gaussians densities $\mathcal{N}_i$, each defined by a mean μ_i in $\mathbb{R}^3$, a covariance Σ_i in $\mathbb{R}^{3\times3}$, an opacity σ_i in $(0,1)$, and a color function $c_i(d)$ that depends on the viewing direction d. d refers to the full camera pose, *i.e.* its extrinsics and intrinsics.

A 2D frame at a given view is an image $\hat{I}_d$ rendered by projecting the 3D Gaussians onto a 2D plane, parametrized by the viewing direction d. This projection accounts for the opacity of the Gaussians and the order in which rays associated with each pixel pass through the densities. More precisely, a pixel p for a view d is associated to an ordered set $\mathcal{S}_{d,p}$ of Gaussians and its value is obtained by summing their contributions:

$$\hat{I}_d(p) = \sum_{i\in\mathcal{S}_{d,p}} c_i(d) w_i(d,p). \tag{1}$$

The above weights are obtained by α-blending, *i.e.* $w_i(d,p) = \alpha_i(d,p)\prod_{j\in\mathcal{S}_{d,p},j<i}(1-\alpha_j(d,p))$, where the Gaussian contributions $\alpha_i(d,p)$ are given by the product of the opacity σ_i and the Gaussian density $\mathcal{N}_i$ projected onto the 2D plane at pixel position p.

Scene optimization. Let $I_1,\dots,I_m$ be a set of 2D frames from a 3D scene and $d_1,\dots,d_m$ the corresponding viewing directions. Gaussian Splatting optimizes the parameters involved in the scene rendering function described in the previous section. This includes the means and covariances of the Gaussian densities, their opacities, and the color function parametrized by spherical harmonics. Denoting these parameters by θ, the following reconstruction loss is used

$$\min_\theta \frac{1}{m}\sum_{k=1}^{m} \mathcal{L}(I_k, \hat{I}_{d_k,\theta}), \tag{2}$$

where $\hat{I}_{d_k,\theta}$ is the frame of the scene rendered in the direction d_k as in Eq. (1), using the parameters θ, and $\mathcal{L}$ is a combination of ℓ_1 and SSIM loss functions [18].

3.2. Uplifting 2D feature maps into 3D

Given a 3D Gaussian Splatting scene representation, we propose a method to uplift any set of 2D features maps $\mathbf{F} = \{\mathbf{F_1},...,\mathbf{F_m}\}$ associated to m views into 3D. These features may be obtained from pretrained models [33, 36] or segmentation masks. The uplifted features $\mathbf{f}$ should allow for downstream 3D tasks, such as 3D segmentation, while being 'faithful' when reprojected back in 2D.

Uplifting with simple aggregation. We construct uplifted features for each 3D Gaussian of the Gaussian Splatting scene as a weighted average of 2D features from all frames. Each 2D feature $F_{d,p}$ from a frame at a given viewing direction d and pixel p contributes to the feature f_i by a factor proportional to the rendering weight $w_i(d,p)$, if the Gaussian i belongs to the ordered set $\mathcal{S}_{d,p}$ associated to the view/pixel pair (d,p). Denoting $\mathcal{S}_i = \{(d,p), i\in\mathcal{S}_{d,p}\}$ as the set of view/pixel pairs contributing to the feature f_i, the resulting features are defined as follows:

$$f_i = \sum_{(d,p)\in\mathcal{S}_i} \bar{w}_i(d,p) F_{d,p}, \quad \bar{w}_i(d,p) = \frac{w_i(d,p)}{\sum_{(d,p)\in\mathcal{S}_i} w_i(d,p)}. \tag{3}$$

We can interpret this equation as a normalized version of the transposed rendering operation over the m viewing directions. More precisely, the rendering of any view-independent collection of features $\mathbf{f} = (f_i)$ attached to the n Gaussians into the m training frames can be represented as a linear operator W acting on the collection $\mathbf{f}$ and returning a collection of 2D feature maps $\hat{\mathbf{F}} = (\hat{F}_{d,p})$, see Eq. (4) below. Here, non-zero entries of the matrix W consists of all rendering weights $w_i(d,p)$ when $(d,p)\in\mathcal{S}_i$ is placed at row (d,p) and column i, and $\hat{\mathbf{F}}$ is a 2D matrix containing all (flattened) 2D feature maps generated for all cameras poses, with $\hat{\mathbf{F}}_{d,p}$ the feature of pixel p from view at direction d. Similarly, the uplifting expression introduced in Eq. (3) can be expressed in terms of the transpose of W and a diagonal matrix D of size m representing the normalization factor and whose diagonal elements are obtained by summing over the rows of W as in Eq. (5) below:

Rendering to m frames	Uplifting from m frames
$\hat{\mathbf{F}} = W\mathbf{f}$, (4)	$\mathbf{f} = D^{-1}W^\top\mathbf{F}$. (5)

Note that W and D are never explicitly constructed. Instead, they are computed by calling the forward rendering function for Gaussian Splatting and replacing the color vectors by the feature vectors (see Fig. 2). All these operations are performed within the CUDA rendering process.

Connection with optimization-based inverse rendering. An alternative approach to uplifting 2D features $\mathbf{F}$ is to minimize a reconstruction objective $\mathcal{L}(\mathbf{f})$, where the goal is to find uplifted features $\mathbf{f}$ whose rendering closely matches the

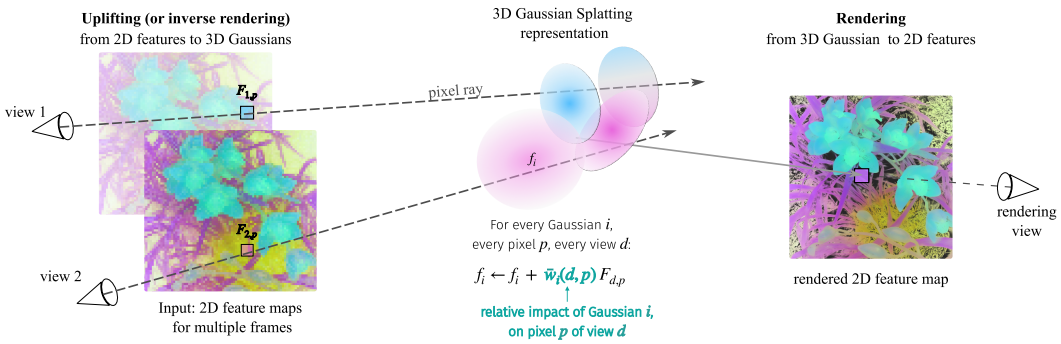


Figure 2. **Illustration of uplifting and rendering.** In the uplifting phase, features $\mathbf{f}$ are created for each 3D Gaussian by aggregating coarse 2D features $\mathbf{F}$ over all viewing directions. For rendering, the 3D features $\mathbf{f}$ are projected on any given viewing direction as in regular Gaussian Splatting. The rendering weight $\bar{w}_i(d,p)$ represents the relative influence of the Gaussian i on pixel p (Eq. (3)).

original 2D features $\mathbf{F}$ [19, 43, 56]. A natural choice is to minimize the mean squared error between the 2D features $\mathbf{F}$ and the rendered ones $\hat{\mathbf{F}}$ as defined by Eq. (4):

$$\min_{\mathbf{f}} \mathcal{L}(\mathbf{f}) := \frac{1}{2}\|\mathbf{F} - W\mathbf{f}\|^2. \tag{6}$$

Such an approach requires an optimization procedure which is costly compared to our proposed uplifting method. Nevertheless, it is possible to interpret the proposed uplifting scheme in Eq. (5) as a single pre-conditioned gradient descent step on the reconstruction objective, starting from a $\mathbf{0}$ feature, *i.e.*, $\mathbf{f} = -D^{-1}\nabla\mathcal{L}(\mathbf{0})$. In practice, we found that performing more iterations on the objective $\mathcal{L}(\mathbf{f})$ did not improve the quality of the features, thus suggesting that the computationally cheaper scheme in Eq. (5) is already an effective approach to uplifting.

Gaussian filtering. The normalization term $\beta_i = \sum_{d,p\in\mathcal{S}_i} w_i(d,p)$ serves as an estimator of the relative importance of each Gaussian in the scene. Therefore, it can be used as a criterion to prune the set of Gaussians for memory efficiency. In our experiments, we filter out half of the Gaussians based on β_i and observe no qualitative nor quantitative degradation of the results. This approach is inspired by prior work on efficient Gaussian Splatting representation such as proposed by [10] that also prunes Gaussians based on their contribution to each pixel in the training frames.

3.3. Enriching features by graph diffusion

DINOv2 features have shown remarkable performance on semantic segmentation with simple linear probing [33], making them a good candidate to enrich features that lack such a property, such as those from CLIP [28, 47, 56]. Inspired by spectral clustering techniques [1, 24, 40] and manifold denoising [14], we propose to *diffuse* features that have been uplifted to 3D. This process aims to align semantic features with the scene layout and object boundaries implied by DINOv2. In contrast to prior work, we perform this landscaping with DINOv2 directly in the 3D scene, thereby taking 3D geometry into account as well.

Graph construction. We construct a graph whose nodes are the n 3D Gaussians and edges, represented by a matrix A of size $n \times n$, are based on 3D Euclidean geometry between the nodes and the similarity between their DINOv2 features. More precisely, we extract the k nearest neighbors $\mathcal{N}(i)$ for each node i, as measured by the Euclidean distance between the centers of the 3D Gaussians. Two nodes i and j in the graph are linked by an edge if $i \in \mathcal{N}(j)$ or $j \in \mathcal{N}(i)$, and the edge is assigned the following weight:

$$A_{ij} = S_f(f_i, f_j)\, P(f_i)^{\frac{1}{2}} P(f_j)^{\frac{1}{2}}, \tag{7}$$

with $S_f(f_i, f_j)$ a local similarity between features f_i and f_j, typically defined as a RBF kernel. For tasks requiring diffusion to be confined to a specific object instance, we prevent leakage into the background by introducing a node-wise unary regularization term $P(f_i)$ which quantifies the similarity between the node feature f_i and the features of the object of interest. Details on S_f and P are provided in the supplementary (Sec. A.3).

Diffusion on the graph. Given initial 3D features g_0 in $\mathbb{R}^n$, which we aim to improve by using information encoded in A (3D geometry and DINOv2 similarities), we perform T diffusion steps to construct a sequence of diffused features $(g_t)_{1\le t\le T}$ defined as follows:

$$g_{t+1} = A\tilde{g}_t, \quad \tilde{g}_t = g_t/\|g_t\|_2, \tag{8}$$

This can be seen as performing a few steps of the power method, projecting g_0 into the dominant eigenspace of A. Depending on the task, g_0 may represent generic features or task-specific features such as 3D segmentation masks.

4. From 3D uplifting to downstream tasks

In this section, we describe our pipeline around the uplifting process from Sec. 3, covering preceding and following stages for three tasks: multi-view segmentation, open-vocabulary object segmentation, and open-vocabulary semantic segmentation. As in Sec. 3, we are given a set of 2D frames $I_1, \ldots, I_m$, with viewing directions $d_1, \ldots, d_m$, and a 3D Gaussian Splatting representation of the scene.

4.1. Multi-view segmentation

We assume that a foreground mask of the object to be segmented is provided on the *reference frame* I_1. This mask can either be a full segmentation or a sparse set of *scribbles*, both defining a set of foreground pixels $\mathcal{P}$. The task is to generate a 2D segmentation mask on one or more *target frames* based on the foreground from the *reference frame*. It is evaluated with the intersection over union (IoU).

Segmentation with SAM. SAM [21, 37] is a supervised pretrained model that, given an input image and point prompts, generates segmentation masks. We uplift SAM masks from multiple views using Eq. (3); this aggregation improves cross-view consistency and leads to better single-view segmentation. On each view, SAM is run multiple times with different prompt sets obtained by uplifting and reprojecting the reference mask, and the resulting predictions are averaged (see Sec. A.1). The final prediction is obtained by rendering the 3D mask into the target frame and thresholding. Results obtained with this strategy are reported for SAM [21] and SAM2 [37] in Sec. 5.2.

Segmentation with DINOv2. We construct 2D feature maps using DINOv2 with registers [9], applying a sliding window over the image followed by dimensionality reduction of the DINOv2 features. To favor the first principal components, known to focus on the foreground objects [33], the features are re-weighted by the eigenvalues of the PCA decomposition. The 2D feature maps from the m training views are uplifted using Eq. (3) and the resulting 3D features are re-projected into any direction using Eq. (4). 2D segmentation is obtained by thresholding the similarity between projected features and those from the foreground pixels $\mathcal{P}$ in the reference frame (see Sec. A.2). This approach corresponds to an ablation discussed in Sec. 5.2.

Improving DINOv2 segmentation with graph diffusion. Segmentation based on DINOv2 alone may struggle to distinguish objects with similar features, as it lacks 3D spatial information. To address this, we leverage the graph diffusion process introduced in Sec. 3.3 and illustrated in Fig. 4. The foreground mask from the reference frame is first uplifted into 3D, defining a set of anchor Gaussians $\mathcal{M}$. This set is used to initialize the weight vector g_0 and to define the regularization term P, based on the similarity between each 3D feature and those of Gaussians in $\mathcal{M}$ (see supplementary Sec. A.3). For this task, we binarize A with a fixed threshold (set to 10^{-5}). After T diffusion steps, we recover the nodes $\mathcal{S}$ in g_T with strictly positive values (*i.e.*, those reachable after T steps). The final weight is defined as $h_i = P(f_i)$ if $i \in \mathcal{S}$ and 0 otherwise. Segmentation is then performed by projecting $\mathbf{h} = (h_i)$ into 2D and thresholding. This approach achieves results comparable to SAM, as reported and discussed in Sec. 5.2.

4.2. Open-vocabulary object segmentation

Following [19], we tackle the task of open-vocabulary object localization by uplifting CLIP features [15]. CLIP effectively aligns images and text in a shared representation space. As a measure of alignment, we use the relevancy score introduced by LERF [19], which measures the similarity between a CLIP visual feature and a text query.

Construction of CLIP feature maps. We follow common practice [19, 56] and construct multi-resolution CLIP 2D feature maps by querying CLIP on a grid of overlapping patches at different scales. As in [56], rather than keeping the resulting representations separate, we aggregate them via average pooling. These multi-resolution CLIP features are uplifted into 3D using Eq. (3).

Relevancy scores. After uplifting CLIP features, we compute relevancy scores for each Gaussian's feature to text queries embedded by CLIP. These relevancy scores can then be projected into 2D and used for both localization and segmentation. For localization, we choose the pixel with the highest relevancy score. For segmentation, we render the relevancy scores and either (i) directly threshold the rendered scores, or (ii) predict a SAM mask by selecting point prompts among pixels with the highest score. Specifics on the computation of relevancy scores and segmentation masks are provided in the supplementary (Sec. A.4).

Refining relevancy scores with DINOv2 graph diffusion. We refine 3D relevancy scores with the diffusion process described in Sec. 3.3. To this end, DINOv2 features are also uplifted, and the similarity matrix is built as in Eq. (7), with the unary term P constructed using a logistic regression over thresholded relevancies, see details in the supplementary (Sec. A.4). The diffusion process propagates CLIP relevancies to Gaussians with similar DINOv2 features. The resulting 3D relevancy scores span the object of interest without covering other objects with similar features and show a strong decay at the object's borders, as defined by DINOv2's feature landscape, resulting in improved segmentation results. This approach is evaluated and compared to direct segmentation without diffusion in Sec. 5.3.

4.3. Open-vocabulary semantic segmentation

In semantic segmentation, one needs to label each Gaussian with the appropriate text label from a predefined set. For this task, we directly replace OpenGaussian's 3D feature learning with our uplifting approach, keeping the rest of their protocol unchanged. This enables a fair comparison under the same evaluation setup. In particular, 2D feature map generation is performed using SAM in automatic mask generation mode and assigning a CLIP feature to each mask, as in [35, 41, 46]. After uplifting, each Gaussian is assigned the text label with the highest CLIP similarity. Results are reported in Sec. 5.4.

5. Experiments

In the following experiments, we train all scenes using the original Gaussian Splatting implementation [18] with default hyperparameters. To reduce memory usage, half of the Gaussians are filtered out based on their importance, as described in Sec. 3.2.

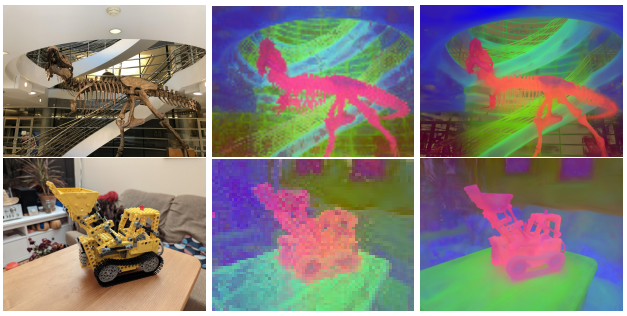
We uplift features from DINOv2 ViT-g with registers [9], SAM [21], SAM 2 [37], and OpenCLIP ViT-B/16 [15]. Key parameters (*e.g.* the segmentation threshold and the definitions of S_f and P in graph diffusion) are either fixed across all scenes of a task or automatically selected, as detailed in the supplementary material (Secs. A.1, A.3, A.4). Results are averaged over three independent runs.

5.1. Qualitative results

DINOv2 feature uplifting. First, we illustrate the effectiveness of our simple uplifting approach. Fig. 3 shows the first three PCA components (one channel per component) over DINOv2's patch embeddings. The coarse patch-level representations from every view (middle) are aggregated using Eq. (5) to form a highly detailed 3D semantic representation, and reprojected into 2D (right) using Eq. (4). The aggregation is very fast, being directly implemented in the Gaussian Splatting CUDA-based rendering process, and takes about 2ms per view and feature dimension. The first principal component (encoded in the red channel) mostly captures the foreground object, and the subsequent ones allow refining the foreground representations and delivering a detailed background. In the supplementary, we provide additional comparative visualizations of our learned 3D features and of 3D segmentation for object removal.

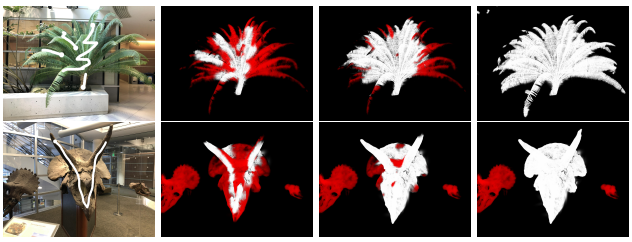
Graph diffusion. Fig. 4 illustrates the diffusion process. The graph nodes are initialized with the reference scribbles, and the diffusion spreads through the object of interest. As illustrated with the case of Horns, diffusion filters out unwanted objects that are similar to the object of interest (here, the two skulls on the side). In the Fern scene, diffusion progressively spreads through the branches to their extremities, with the regularization (red background) constraining it to the trunk and preventing leakage, even after a large number of iterations. Fig. C in the supplementary also illustrates this for the Flower and Trex scenes: diffusion rapidly spreads, achieving near-full coverage after only 5 steps before reaching all the much smaller Gaussians on the border, allowing for a refined segmentation.

Open-vocabulary object removal. Fig. 1 and 5 provide qualitative results for the object removal task, performed by discarding all Gaussians corresponding to a binary 3D segmentation mask. This mask is obtained by computing relevancy between 3D CLIP features and the scene-specific textual prompt "bonsai in a ceramic pot" or "teddy bear", followed by DINOv2-guided graph diffusion.



(a) RGB image (b) Single-view PCA (c) Multi-view PCA

Figure 3. **PCA visualizations.** The DINOv2 patch-level representations (middle) predicted from the RGB images (left) are aggregated into highly detailed 3D representations (right) using Eq. (3).



(a) RGB image (b) 3 steps (c) 7 steps (d) 100 steps

Figure 4. **Diffusion process.** 2D projection of the weight vector g_t (white) and unary regularization term (red) at diffusion steps t. The diffusion process filters out unwanted objects that have similar features to the object of interest (such as the two smaller skulls on *horns*, bottom-row), but are disconnected in space. The regularization term (red background) prevents leakage from the object to the rest of the scene (such as through the *fern*'s trunk, top-row).

Figure 5. **Open-vocabulary object removal.** Removing the teddy bear from the CO3D dataset [38], using DINOv2-guided graph diffusion based on 3D CLIP relevancy scores.

5.2. Multi-view segmentation results

Task. In this section, we consider two segmentation tasks: (i) Neural Volumetric Object Selection (NVOS) [39], which is derived from the LLFF dataset [30], and (ii) SPIn-NeRF, which contains a subsets of scenes from NeRF-related datasets [11, 22, 30, 31, 51]. NVOS consists of forward-facing sequences where the task is to predict the segmentation mask of a labeled frame based on reference scribbles from another frame. SPIn-NeRF contains both forward-facing and 360-degree scenes, in which all frames are la-

beled with segmentation masks, and the standard evaluation protocol uses the segmentation mask from the first frame as reference to label the subsequent frames.

Setup. We evaluate segmentation based on SAM and SAM2 mask uplifting and on DINOv2 feature uplifting combined with graph diffusion (see Sec. 4.1). We compare our segmentation results (IoU) to the current state of the art: SA3D [4], SA3D-GS [3], SAGA [5], OmniSeg3D [52]. All these methods are specifically designed for uplifting the 2D segmentation masks produced by SAM into 3D using gradient-based optimization of a projection loss. We also report results from NVOS [39] and MVSeg [51], who respectively introduced the NVOS and SPIn-NeRF datasets.

Results. Table 1 reports the average IoU across all scenes for NVOS and SPIn-NeRF. Per-scene results can be found in supplementary Tables C and D. Our results are comparable to the state of the art, while not relying on gradient-based optimization. Surprisingly, our segmentation with DINOv2 using graph diffusion also gives results on par with models leveraging SAM masks. Compared to SAM, DINOv2 better captures complex objects but sometimes also captures background noise. This can be seen in supplementary Fig. B, T-Rex example: while SAM misses the end of the tail and ribs, DINOv2 captures the whole T-Rex but also includes part of the stairs behind. Our lower segmentation results compared to OmniSeg's can partly be attributed to the poor Gaussian Splatting reconstruction of highly specular scenes, such as Fork. As also noted by [5], the reconstruction includes semi-transparent Gaussians floating over the object, attempting to represent reflections or surface effects, which are challenging to capture using standard rasterization techniques [16].

Ablation study. We compare our segmentation protocol using DINOv2 and SAM2 to multiple simpler variants. More precisely, we evaluate (i) a purely geometrical variant that reprojects the reference mask on the other views, without using SAM2 or DINOv2, (ii) single-view segmentation in 2D based on SAM2 or DINOv2 2D predictions, (iii) uplifting DINOv2 features or SAM2 masks into 3D then rendering them for segmentation, and (iv) segmenting using graph diffusion over DINOv2 3D feature similarities. Results are reported in Table 2, and per-scene IoU as well as a detailed analysis can be found in supplementary Table E and Sec. C.1. We observe that the purely geometrical approach works well on the forward-facing scenes and fails on 360-degree scenes. The single-view variant performs reasonably well on average, but the low resolution of patch-level representations (illustrated in Fig. 3) lead to a coarser segmentation. 3D uplifting considerably boosts results compared to single-view approaches, and introducing 3D spatial information through 3D graph diffusion further enhances results on the more challenging 360-degree scenes.

5.3. Open-vocabulary object localization

Task. We evaluate on the LERF dataset [19], which includes localization and segmentation tasks on complex in-the-wild scenes. We report our results on the extended evaluation task introduced by LangSplat [35] containing additional challenging localization samples.

Setup. For localization, we simply reproject the 3D CLIP relevancy scores and follow the evaluation protocol from LangSplat [35]. For segmentation, we consider two evaluation protocols: (i) *standard 2D segmentation*, as introduced by [19] (top part of the table), and (ii) *3D object selection*, which consists of binarizing the 3D masks before rendering (bottom part). The latter protocol was introduced in [17] as a way to better assess the quality of 3D semantic features, such as 3D CLIP features, compared to prior existing protocols which focus on the performance on 2D downstream tasks. For both approaches, we refine raw 3D CLIP relevancy scores using graph diffusion and use them for segmentation as follows. For (i) we perform 2D segmentation from rendered features using SAM, as described in Sec. 4.2, which we compare to 2D segmentation by LERF [19] and LangSplat [35] (top of table). For (ii) we directly segment in 3D by binarizing the diffused features before rendering them, then computing 2D segmentation masks from these rendered features. We use automatic thresholding for binarization before and after rendering.

Results. Table 3 reports object localization results, where our method outperforms LangSplat despite not relying on SAM for this task. Table 4 presents segmentation results along with average runtimes, which account for feature map generation and 3D feature training when applicable. Across both evaluation settings, our method consistently surpasses prior approaches while achieving a 5 to 10$\times$ speedup in feature generation and uplifting.

Runtime analysis. Our approach based on graph diffusion offers fast feature map generation compared to leveraging SAM's automatic mask generation in 2D as in [35, 46], but it comes with additional inference-time overhead. This tradeoff is further discussed in supplementary Sec. B.2. When inference time is a critical constraint, LUDVIG's uplifting can also be applied directly to full segmentation masks, following the same setup as LangSplat [35] and OpenGaussian [46], as shown in the next section.

Ablation study. Table 5 presents an ablation study on the effect of graph diffusion and SAM-based segmentation following the first evaluation protocol (top part) in Table 4.

5.4. Open-vocabulary semantic segmentation

Task. We evaluate on ScanNetv2 [8], which provides posed RGB images from video scans, reconstructed point clouds, and ground-truth 3D point-level semantic labels.

	NVOS [39]	MVSeg [32]	SA3D-TRF [4]	SA3D-GS [3]	SAGA [5]	OmniSeg3D [52]	**LUDVIG (Ours)**		
3D representation			TensoRF	3D-GS	3D-GS	NeRF	3D-GS		
Uplifting			SAM	SAM	SAM	SAM	DINOv2	SAM	SAM2
NVOS	70.1	-	90.3	92.2	**92.6**	91.7	92.4	91.3	91.3
SPIn-NeRF	-	90.9	93.7	93.2	93.4	**94.3**	93.8	93.8	93.8

Table 1. **Multi-view segmentation (IoU) on NVOS [39] and SPIn-NeRF [32].**

Geometry only	Single view		Uplifting		Uplifting + Dif.
Reference mask	DINOv2	SAM2	DINOv2	SAM2	DINOv2
73.1	88.5	88.6	91.6	93.8	93.8

Table 2. **Ablation on SPIn-NeRF segmentation**. We compare purely geometrical reference mask reprojection and single-view prediction with our feature/mask uplifting and graph diffusion.

	ramen	figurines	teatime	waldo	**overall**
LERF [19]	62.0	75.0	84.8	72.7	73.6
LangSplat [35]	73.2	80.4	88.1	**95.5**	84.3
LUDVIG (ours)	**78.9**	**80.4**	**94.9**	90.9	**86.3**

Table 3. **LERF object localization.** Comparison with the state of the art on the dataset introduced by LangSplat [35].

	ramen	figurines	teatime	waldo	**overall**	time (min)
Initial evaluation protocol from [19, 35]: 2D object segmentation						
LERF [19]	28.2	38.6	45.0	37.9	37.4	45
LangSplat [35]	51.2	44.7	65.1	44.5	51.4	105
LUDVIG (ours)	**58.1**	**63.3**	**77.1**	**58.5**	**64.3**	10
New evaluation protocol introduced in OpenGaussian [46]: 3D object selection						
OpenGaussian [46]	23.9	59.3	54.4	34.6	43.1	50
Dr. Splat [17]	24.7	53.4	57.2	39.1	43.6	-
LUDVIG (ours)	**42.3**	**58.0**	**58.6**	**42.8**	**50.4**	10

Table 4. **LERF object segmentation.** We evaluate on the LERF dataset introduced by LangSplat [35] with two different evaluation protocols: i) segmentation based on the 2D reprojected features, ii) 3D segmentation (or *object selection* and reprojection of the *binary 3D masks*, proposed by [19, 35] and [46] respectively.

SAM	Graph diffusion	ramen	figurines	teatime	waldo	overall
✗	✗	27.8	37.8	38.2	30.4	33.6
✗	✓	42.3	58.0	58.6	42.9	50.4
✓	✗	52.2	51.8	68.9	56.4	57.3
✓	✓	**58.1**	**63.3**	**77.1**	**58.5**	**64.3**

Table 5. **Ablation on LERF object segmentation.** Results (IoU) with and without using 3D graph diffusion and/or 2D SAM segmentation, evaluated on the dataset introduced by LangSplat [35].

Setup. We follow the evaluation protocol of OpenGaussian [46]. In particular, we train Gaussian Splatting by initializing from the ground-truth point cloud, keeping the 3D positions fixed and disabling densification. For this task, we replace OpenGaussian's quantized feature learning with our uplifting process, leaving the rest of the protocol unaltered. Specifically, we uplift 2D segmentation masks generated by SAM based on textual queries, as in the prior works we compare to [35, 41, 46].

Methods	19 classes		15 classes		10 classes	
	mIoU	mAcc	mIoU	mAcc	mIoU	mAcc
LangSplat*	3.8	9.1	5.4	13.2	8.4	22.1
LEGaussians*	3.8	10.9	9.0	22.2	12.8	28.6
OpenGaussian	24.7	41.5	30.1	48.3	38.3	55.2
LUDVIG	**33.9**	**51.4**	**37.4**	**57.2**	**46.4**	**66.2**

Table 6. **ScanNet semantic segmentation.** For LUDVIG, we replace OpenGaussian's quantization-based feature training stage ($\sim$ 40min) with our direct uplifting process ($\sim$ 3min), starting from the same 2D feature maps ($\sim$ 50min) and GS pre-training stage ($\sim$ 9min). *Results reported by OpenGaussian.

Results. As shown in Table 6, our approach yields significant accuracy gains over OpenGaussian [46], while being considerably simpler and faster than their quantization-based learning process.

6. Concluding remarks and limitations

In this work, we introduce a simple yet effective aggregation mechanism for transferring 2D visual representations into 3D, bypassing traditional optimization-based approaches. The aggregation builds upon already trained Gaussian Splatting representations and is implemented within the CUDA rendering process, making 2D-to-3D uplifting as fast as 3D-to-2D rendering. Our approach can serve as an off-the-shelf module for transferring 2D features into 3D across a wide range of applications, as demonstrated in our experiments (Sec. 5.4) and its recent integration into Panst3R [57] for novel-view panoptic segmentation.

We also propose a graph diffusion process that combines spatial structure with DINOv2 similarity to generate accurate 3D segmentation masks, starting from coarse inputs such as scribbles or CLIP relevancy maps. This leads to strong gains over SAM-based open-vocabulary segmentation baselines while remaining computationally efficient.

However, the quality of the resulting 3D features depends on the underlying 3D scene reconstruction, which remains challenging in cases of high specularity [16, 48] or motion blur [25, 54]. In such scenarios, learning 3D features *along with* 3D Gaussian Splatting reconstruction may serve as a regularization and improve scene geometry.

Acknowledgments

This project was supported by ANR 3IA MIAI@Grenoble Alpes (ANR-19-P3IA-0003) and by ERC grant number 101087696 (APHELEIA project). This work was granted access to the HPC resources of IDRIS under the allocation [AD011013343R2] made by GENCI.

References

[1] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. *Advances in neural information processing systems (NIPS)*, 2001. 2, 4

[2] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021. 1, 2

[3] Jiazhong Cen, Jiemin Fang, Zanwei Zhou, Chen Yang, Lingxi Xie, Xiaopeng Zhang, Wei Shen, and Qi Tian. Segment anything in 3d with radiance fields. *arXiv preprint arXiv:2304.12308*, 2023. 2, 7, 8, 15

[4] Jiazhong Cen, Zanwei Zhou, Jiemin Fang, Wei Shen, Lingxi Xie, Dongsheng Jiang, Xiaopeng Zhang, Qi Tian, et al. Segment anything in 3d with nerfs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. 1, 2, 7, 8

[5] Jiazhong Cen, Jiemin Fang, Chen Yang, Lingxi Xie, Xiaopeng Zhang, Wei Shen, and Qi Tian. Segment any 3d gaussians. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2025. 7, 8, 16

[6] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 1, 2, 18, 19, 23

[7] Jaeyoung Chung, Jeongtaek Oh, and Kyoung Mu Lee. Depth-regularized optimization for 3d gaussian splatting in few-shot images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 18

[8] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 7

[9] Timothée Darcet, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. Vision transformers need registers. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2024. 2, 5, 6, 13

[10] Zhiwen Fan, Kevin Wang, Kairun Wen, Zehao Zhu, Dejia Xu, and Zhangyang Wang. Lightgaussian: Unbounded 3d gaussian compression with 15x reduction and 200+ fps. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. 1, 4

[11] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 6

[12] Rahul Goel, Dhawal Sirikonda, Saurabh Saini, and P.J. Narayanan. Interactive segmentation of radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 2

[13] Jun Guo, Xiaojian Ma, Yue Fan, Huaping Liu, and Qing Li. Semantic gaussians: Open-vocabulary scene understanding with 3d gaussian splatting. *arXiv preprint arXiv:2403.15624*, 2024. 2

[14] Matthias Hein and Markus Maier. Manifold denoising. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2006. 4

[15] Gabriel Ilharco, Mitchell Wortsman, Ross Wightman, Cade Gordon, Nicholas Carlini, Rohan Taori, Achal Dave, Vaishaal Shankar, Hongseok Namkoong, John Miller, Hannaneh Hajishirzi, Ali Farhadi, and Ludwig Schmidt. Openclip, 2021. 2, 5, 6

[16] Yingwenqi Jiang, Jiadong Tu, Yuan Liu, Xifeng Gao, Xiaoxiao Long, Wenping Wang, and Yuexin Ma. Gaussianshader: 3d gaussian splatting with shading functions for reflective surfaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 7, 8, 17

[17] Kim Jun-Seong, GeonU Kim, Kim Yu-Ji, Yu-Chiang Frank Wang, Jaesung Choe, and Tae-Hyun Oh. Dr. splat: Directly referring 3d gaussian splatting via direct language embedding registration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025. 2, 3, 7, 8

[18] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. In *Proceedings of the ACM SIGGRAPH Conference on Computer Graphics and Interactive Techniques*, 2023. 1, 2, 3, 6

[19] Justin Kerr, Chung Min Kim, Ken Goldberg, Angjoo Kanazawa, and Matthew Tancik. Lerf: Language embedded radiance fields. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2023. 1, 2, 4, 5, 7, 8, 14, 15, 16, 20

[20] Chung Min Kim, Mingxuan Wu, Justin Kerr, Ken Goldberg, Matthew Tancik, and Angjoo Kanazawa. Garfield: Group anything with radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 2

[21] 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. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2023. 1, 2, 5, 6, 12

[22] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics (ToG)*, 36 (4):1–13, 2017. 6

[23] Sosuke Kobayashi, Eiichi Matsumoto, and Vincent Sitzmann. Decomposing nerf for editing via feature field distillation. *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. 1, 2

[24] Risi Imre Kondor and John Lafferty. Diffusion kernels on graphs and other discrete structures. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2002. 2, 4

[25] Byeonghyeon Lee, Howoong Lee, Xiangyu Sun, Usman Ali, and Eunbyung Park. Deblurring 3d gaussian splatting. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2024. 8

[26] Boyi Li, Kilian Q Weinberger, Serge Belongie, Vladlen Koltun, and Rene Ranftl. Language-driven semantic segmentation. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2022. 2

[27] Chun Hung Li and CK Lee. Minimum cross entropy thresholding. *Pattern recognition*, 26(4):617–625, 1993. 13

[28] 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. *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. 1, 2, 4

[29] Marina Meila and Jianbo Shi. Learning segmentation by random walks. *Advances in neural information processing systems (NIPS)*, 13, 2000. 2

[30] 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. *ACM Transactions on Graphics (ToG)*, 38(4):1–14, 2019. 6

[31] 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. 1, 2, 6

[32] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20669–20679, 2023. 8, 17

[33] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel HAZIZA, Francisco Massa, Alaaeldin El-Nouby, Mido Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research (TMLR)*, 2024. 1, 2, 3, 4, 5, 13

[34] Nobuyuki Otsu et al. A threshold selection method from gray-level histograms. *Automatica*, 11(285-296):23–27, 1975. 14, 15

[35] Minghan Qin, Wanhua Li, Jiawei Zhou, Haoqian Wang, and Hanspeter Pfister. Langsplat: 3d language gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 2, 3, 5, 7, 8, 14, 15, 16, 18, 20

[36] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2021. 1, 2, 3

[37] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, et al. Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024. 2, 5, 6, 12

[38] Jeremy Reizenstein, Roman Shapovalov, Philipp Henzler, Luca Sbordone, Patrick Labatut, and David Novotny. Common objects in 3d: Large-scale learning and evaluation of real-life 3d category reconstruction. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021. 6

[39] Zhongzheng Ren, Aseem Agarwala, Bryan Russell, Alexander G Schwing, and Oliver Wang. Neural volumetric object selection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 6, 7, 8, 17, 18, 19

[40] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence (PAMI)*, 22(8):888–905, 2000. 2, 4

[41] Jin-Chuan Shi, Miao Wang, Hao-Bin Duan, and Shao-Hua Guan. Language embedded 3d gaussians for open-vocabulary scene understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 2, 5, 8

[42] Alexander J Smola and Risi Kondor. Kernels and regularization on graphs. In *Learning Theory and Kernel Machines: 16th Annual Conference on Learning Theory and 7th Kernel Workshop*, 2003. 2

[43] Vadim Tschernezki, Iro Laina, Diane Larlus, and Andrea Vedaldi. Neural feature fusion fields: 3d distillation of self-supervised 2d image representations. In *Proceedings of the International Conference on 3D Vision (3DV)*, 2022. 2, 4, 18, 22

[44] Junjie Wang, Jiemin Fang, Xiaopeng Zhang, Lingxi Xie, and Qi Tian. Gaussianeditor: Editing 3d gaussians delicately with text instructions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 2

[45] Mengzhao Wang, Xiaoliang Xu, Qiang Yue, and Yuxiang Wang. A comprehensive survey and experimental comparison of graph-based approximate nearest neighbor search. *arXiv preprint arXiv:2101.12631*, 2021. 15

[46] Yanmin Wu, Jiarui Meng, Haijie Li, Chenming Wu, Yahao Shi, Xinhua Cheng, Chen Zhao, Haocheng Feng, Errui Ding, Jingdong Wang, et al. Opengaussian: Towards point-level 3d gaussian-based open vocabulary understanding. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. 1, 2, 3, 5, 7, 8, 16, 18

[47] Monika Wysoczańska, Oriane Siméoni, Michaël Ramamonjisoa, Andrei Bursuc, Tomasz Trzciński, and Patrick Pérez.

Clip-dinoiser: Teaching clip a few dino tricks for open-vocabulary semantic segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2024. 4

[48] Ziyi Yang, Xinyu Gao, Yangtian Sun, Yihua Huang, Xiaoyang Lyu, Wen Zhou, Shaohui Jiao, Xiaojuan Qi, and Xiaogang Jin. Spec-gaussian: Anisotropic view-dependent appearance for 3d gaussian splatting. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. 8

[49] Jianglong Ye, Naiyan Wang, and Xiaolong Wang. Featurenerf: Learning generalizable nerfs by distilling foundation models. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2023. 2

[50] Mingqiao Ye, Martin Danelljan, Fisher Yu, and Lei Ke. Gaussian grouping: Segment and edit anything in 3d scenes. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2024. 1, 2

[51] 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 *Proceedings of the International Conference on Learning Representations (ICLR)*, 2022. 2, 6, 7

[52] Haiyang Ying, Yixuan Yin, Jinzhi Zhang, Fan Wang, Tao Yu, Ruqi Huang, and Lu Fang. Omniseg3d: Omniversal 3d segmentation via hierarchical contrastive learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 1, 2, 7, 8, 16

[53] Yuanwen Yue, Anurag Das, Francis Engelmann, Siyu Tang, and Jan Eric Lenssen. Improving 2d feature representations by 3d-aware fine-tuning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2024. 2

[54] Lingzhe Zhao, Peng Wang, and Peidong Liu. Bad-gaussians: Bundle adjusted deblur gaussian splatting. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2024. 8

[55] 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. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 2

[56] Xingxing Zuo, Pouya Samangouei, Yunwen Zhou, Yan Di, and Mingyang Li. Fmgs: Foundation model embedded 3d gaussian splatting for holistic 3d scene understanding. *International Journal of Computer Vision (IJCV)*, 133(2):611–627, 2025. 1, 2, 4, 5, 15

[57] Lojze Zust, Yohann Cabon, Juliette Marrie, Leonid Antsfeld, Boris Chidlovskii, Jerome Revaud, and Gabriela Csurka. Panst3r: Multi-view consistent panoptic segmentation. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2025. 8

Supplementary Material

Contents

A. Using LUDVIG for downstream tasks

In this section, we describe our approach for uplifting DINOv2, SAM and CLIP models and evaluating the 3D features on two downstream tasks: segmentation and open-vocabulary object detection. As in Sec. 3, we are given a set of 2D frames $I_1, \dots, I_m$, with viewing directions $d_1, \dots, d_m$ and a corresponding 3D scene obtained using the Gaussian Splatting method.

Multi-view segmentation. For this task, we assume that a *foreground mask* of the object to be segmented is provided on a *reference frame* taken to be the first frame I_1. We consider two types of foreground masks: either *scribbles* or a whole *reference mask* of the object, both of which define a set of *foreground pixels* $\mathcal{P}$. In the following subsections, we present the proposed approaches for segmentation using SAM and DINOv2 features, based on both types of foreground masks.

A.1. Multiple-view segmentation with SAM

SAM [21, 37] is a powerful image segmentation model, that can generate object segmentation masks from point prompts on a single 2D image. Aggregating SAM 2D segmentation masks in 3D allows for cross-view consistency and improves single-view segmentation results. In order to leverage SAM, we propose a simple mechanism for generating SAM 2D features for each frame from a *foreground mask* in the *reference frame*.

Generating 2D query points for SAM. The key idea is to generate point prompts on each frame from the *foreground mask* provided on the *reference frame*. To this end, we perform an uplifting of the *foreground mask* (Eq. (5) from the main paper) and re-project it on all frames (Eq. (4) from the main paper). From the reprojected mask for viewing direction d, further normalized by its average value, we retain a subset $\mathcal{P}_d$ of pixels with values higher than a threshold τ fixed for all scenes ($\tau = 0.4$ for SPIn-NeRF and $\tau = 1$ for NVOS). We then predict a SAM mask based on these point prompts as described below.

Predicting SAM masks from sets of query points. Given a set of pixels $\mathcal{P}_d$ pertaining to the foreground, we compute 2D segmentation masks using SAM by randomly selecting 3 points prompts from $\mathcal{P}_d$, repeating the operation 10 times and averaging the resulting masks for each view to obtain the final 2D SAM feature maps.

Segmentation with uplifted SAM masks. The 2D segmentation masks generated by SAM are uplifted using the aggregation scheme described in Sec. 3.2. Our final prediction is obtained by rendering the uplifted feature maps into the target frame and thresholding.

Evaluation and hyperparameter tuning. Segmentation with 3D SAM masks requires setting a threshold for foreground/background pixel assignment, and optionally choosing one of the three masks proposed by SAM (representing different possible segmentations of the object of interest). For SPIn-NeRF, the threshold and mask prediction are chosen based on the IoU for the available reference mask. For NVOS, only reference scribbles are provided; hence, a single mask is predicted, and the segmentation threshold is

fixed across all scenes for SAM, and automatically chosen using Li's iterative Minimum Cross Entropy method [27] for SAM 2.

A.2. Multiple-view segmentation with DINOv2

DINOv2 [33] is a self-supervised vision model recognized for its generalization capabilities. In this work, we aggregate the patch-level representations produced by DINOv2 with registers [9] into a high resolution and fine-grained 3D semantic representation.

Construction of 2D feature maps. We construct the 2D feature maps using a combination of a sliding windows mechanism and dimensionality reduction of the original DINOv2 features. Specifically, we i) extract DINOv2 patch-level representations across multiple overlapping crops of each image, ii) apply dimensionality reduction over the set of all patch embeddings, ii) upsample and aggregate the dimensionality-reduced patch embeddings to obtain pixel-level features for each image. The process is illustrated in Fig. A. This approach enhances the granularity of spatial representations by aggregating patch-level representations to form pixel-level features. To favor the first principal components, known to focus on the foreground objects [33], the features are re-weighted by the eigenvalues of the PCA decomposition.

Segmentation with uplifted DINOv2 features. The 2D feature maps from the m views are uplifted using Eq. (5) from the main paper, and the resulting 3D features are re-projected into any viewing direction d using Eq. (4) from the main paper to compute rendered 2D features $(\hat{F}_{d,p})$. To obtain segmentation masks, we construct a score $P(\hat{F}_{d,p})$ for a 2D pixel p to belong to the foreground, based on its corresponding rendered feature. More precisely, P relies on the rendered *foreground features* $\mathcal{F}_{\text{ref}} := (\hat{F}_{d_1,p})_{p\in\mathcal{P}}$ corresponding to the *foreground mask* computed on the *reference frame* I_1. We propose two approaches for constructing P. The first one is a simple approach that sets $P(\hat{F}_{d,p}) = \mathcal{S}_F(\hat{F}_{d,p}, \bar{F})$ where $\bar{F}$ is the average over foreground features $\mathcal{F}_{\text{ref}}$, and $\mathcal{S}_F$ is defined based on the cosine similarity. The second approach is more discriminative and first trains a logistic regression model P on all rendered 2D features of the reference frame, so that the foreground features $\mathcal{F}_{\text{ref}}$ are assigned a positive label. Then $P(\hat{F}_{d,p})$ gives the probability that a pixel p belongs to the foreground. The final mask is then obtained by thresholding.

Experimentally, the second approach is extremely efficient when the set of *foreground pixels* $\mathcal{P}$ covers the whole object to segment, so that P captures all relevant features. This is the case when a whole *reference mask* of the object is provided. When the *foreground pixels* $\mathcal{P}$ does not cover the whole object, as with scribbles, P can be discriminative to parts of the object that are not covered by $\mathcal{P}$. Therefore, we rely on the second approach for tasks where a reference mask is provided, and use the simpler first approach when only scribbles serve as reference.

A.3. Enhancing segmentation with DINOv2 using 3D graph diffusion

DINOv2 provides generic visual features that do not explicitly include information for segmentation, unlike models such as SAM that were specifically trained for such a task. Consequently, using the 2D projections of uplifted DINOv2 features, as proposed in Sec. A.2, might fail to separate different objects that happen to have similar features while still being distinct entities. This challenge can be mitigated by incorporating 3D spatial information in which the objects are more likely to be well-separated. To this end, we propose to leverage the graph diffusion process introduced in Sec. 3.3. Below, we describe the initialization of the weight vector g_0 and the construction of the adjacency matrix A.

Initialization of the weight vector. The initial vector of weights $g_0 \in \mathbb{R}^n$ representing a coarse estimation of the contribution of each Gaussian to the segmentation mask. It is computed by uplifting the 2D *foreground mask* (either scribbles or a reference mask) from the *reference frame* using Eq. (3) from the main paper, and retaining only the top 10% of Gaussians with positive mask values, setting the rest to zero. The nodes for which g_0 has a positive value define a set of anchor nodes $\mathcal{M}$ that are more likely to contribute to the foreground. The resulting weight vector is a coarse estimation of how much each Gaussian contributes to a rendered 2D segmentation mask.

Construction of the graph edges. We define the pairwise similarity function S_f as:

$$S_f(f_i, f_j) = \exp\left(-\frac{||f_i - f_j||_2^2}{b s_f^2}\right) \tag{9}$$

where f_i, f_j are the l_2-normalized DINOv2 features, s_f is the median of pairwise l_2 distances and b is a bandwidth parameter. We choose a global unary regularization term $P(f_i)$ on each node i to contain diffusion to nodes with features similar to those of the foreground. More precisely, P is defined using a similar approach as in Sec. A.2:

1. When scribblers are provided, $P(f_i) = \mathcal{S}_f(f_i, \bar{f})$ with the averaged feature $\bar{f}$ over the anchor nodes $\mathcal{M}$, and a different value for the bandwidth b.
2. When a full foreground mask is available, we train a logistic regression model on the uplifted features with positive labels for the anchor nodes' features. The unary term is then defined as $P(f_i) = \mathcal{L}(f_i)^{1/b}$, with b a bandwidth parameter and $\mathcal{L}(f_i)$ is the model's predicted probability for f_i.

The local term S_f allows diffusing to neighbors that have similar features while the unary term prevents leakage to

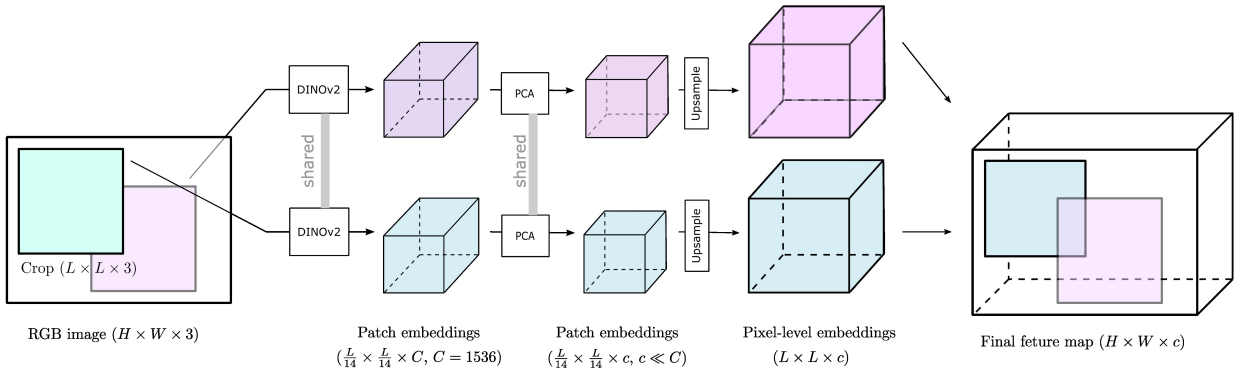


Figure A. Sliding windows for construction of DINOv2 feature maps.

background nodes and allows using an arbitrary number of diffusion steps.

The matrix A of graph edges is then defined based on S_f and P as in Eq. (7) from the main paper. For this task, we binarize A with a fixed threshold (set to 10^{-5}). After the T diffusion steps, we recover the nodes $\mathcal{S}$ in g_T with strictly positive values (*i.e.*, those reachable after T iterations). The final weight is defined as $h_i = P(f_i)$ if $i \in \mathcal{S}$ and 0 otherwise. Segmentation is then performed by projecting $\mathbf{h} = (h_i)$ into 2D and thresholding. The selection process of the threshold and bandwidth parameters is detailed below.

Evaluation and hyperparameter tuning. Segmentation relies on three hyperparameters: the bandwidths for S_f and P, and the threshold for foreground/background pixel assignment. For SPIn-NeRF, all hyperparameters are chosen based on the IoU for the available reference mask. For NVOS, only reference scribbles are provided, hence we predict a SAM mask based on the scribbles of the reference frame, and choose the hyperparameters maximizing the IoU with this SAM mask. This is consistent with a scenario where the user, here SAM, would choose hyperparameters based on visual inspection on one of the frames.

A.4. Open-vocabulary object localization

For the open-vocabulary object localization task, multi-resolution CLIP feature maps are constructed as described in Sec. 4.2 and uplifted along with DINOv2 features using Eq. (3) from the main paper. The refined 3D CLIP features are then evaluated on the LERF localization and segmentation tasks as described below.

A.4.1. Relevancy scores and object localization.

We consider uplifted 3D CLIP features f. We follow LERF [19] and LangSplat [35] to compute alignment scores between CLIP visual features and a text query, denoted as *relevancy score*, and for object localization based on these relevancy scores.

Relevancy scores. The relevancy $r_{i,q}$ between a feature f_i and text query ϕ_q is defined as follows:

$$r_{i,q} = \min_k \frac{\exp(T \cdot f_i \cdot \phi_q)}{\exp(T \cdot f_i \cdot \phi_q) + \exp(T \cdot f_i \cdot \phi^k_{cano})}, \quad (10)$$

where T is a temperature parameter set to 10 by [19] and ϕ^k_cano is the CLIP embedding of a predefined canonical phrase chosen from "object," "things," "stuff," and "texture." Note that [35] compute the relevancy scores for 2D pixels, while we directly compute them for 3D Gaussians, allowing their manipulation in 3D.

Localization. The 3D CLIP relevancies can be projected into 2D for a given camera pose, and used for localization and segmentation for each text query. For localization, we follow [35] and choose the pixel with the highest relevancy score, following a 2D smoothing using a mean filter with kernel size $K = 5$ in our work.

A.4.2. Object segmentation

Segmentation based on raw CLIP relevancies is challenging, as fully covering the object of interest without capturing other objects of a similar nature is challenging.

We first describe our process for segmenting directly based 3D relevancies. We then present two complementary approaches that allow for a more targeted segmentation: predicting 2D SAM masks by retrieving query points with high relevancies, and refining 3D relevancy scores using graph diffusion based on 3D DINOv2 features.

Segmentation from 3D relevancies. Given a camera pose and a text query, a segmentation mask is obtained by first applying a rough thresholding over projected relevancies rescaled by their maximum value, with a fixed threshold value $\tau = 0.8$, followed by automatic thresholding with Otsu's method [34].

Improving 2D segmentation with SAM. For segmentation with SAM, we use the pixels with the highest relevancy scores as query points for a given camera pose and text

query. More precisely, we first obtain a mask $\mathcal{M}$ by projecting and thresholding the relevancies as described above, using $\tau = 0.93$. We then use the approach described in Sec. A.1, paragraph *Predicting a SAM segmentation mask from a set of query points* and average 20 mask predictions. We choose the top-q percent of pixels as the subset of query points for SAM, where q is the proportion of positive pixels in $\mathcal{M}$, hence extracting a larger subset of point prompts for larger objects. The scalar map obtained by averaging the 20 predicted masks is then automatically thresholded again using Otsu's method [34].

Refining 3D CLIP relevancies with graph diffusion based on DINOv2 features. We refine 3D CLIP relevancy scores using graph diffusion based on 3D DINOv2 features (f), as in Sec. A.3. The diffusion process runs in parallel for all text queries. For initialization, we keep positive a very restricted set of nodes with the highest relevancy, whose weight propagate to neighboring nodes with similar DINOv2 features, progressively spanning the object of interest. The diffusion process results in a better segmentation both with and without leveraging SAM. When using SAM, the set of query pixels can have a larger coverage of the object of interest without extending to other objects.

Details on graph construction and node initialization for refining 3D CLIP relevancies. The pairwise similarity function S_f is defined as in Eq. 9 with a bandwidth value $b = 0.5$. For each text query ϕ_q, we define a unary regularization term P_q using a logistic regression model $\mathcal{L}_q$ that predicts the probability that a DINOv2 feature f_i belongs to the object corresponding to query ϕ_q. The set of nodes $\mathcal{P}$ with positive labels is defined based on 3D CLIP relevancy scores for prompt ϕ_q. More precisely, we rescale 3D relevancies to $[0, 1]$ and apply Otsu's method [34] over relevancies above 0.5. We use a regularization $C = 0.001$ and equal class weighting for training the model. The unary regularization term P_q is then defined as $P_q(f_i) = \mathcal{L}_q(f_i)^{1/b}$, with $b = 0.025$ for segmentation with SAM, and $b = 2$ otherwise. The initial weight vector g_0 is defined by applying two more iterations of Otsu's method among nodes in $\mathcal{P}$ and setting to zero all relevancies below the given threshold. Restricting the set of initial points ensures diffusion only happens within the object of interest. Segmentation based on the resulting 3D relevancy scores is then performed as described in the previous paragraphs, using $\tau = 0.01$ for segmentation with SAM and $\tau = 0$ otherwise.

B. Runtime analysis

B.1. Runtimes for LUDVIG

In this section, we detail our running times for feature uplifting and evaluation, conducted on a GPU RTX 6000 ADA. Table A shows a breakdown of running times between feature uplifting (Up.) and generation (Gen.), graph diffusion and 2D segmentation for evaluation on LERF segmentation. The total reported times can be divided between pre-uplifting, uplifting and post-uplifting. In our experiments, the pre-processing and uplifting steps are independent from the downstream tasks (except for our foreground/background segmentation with SAM), and part of the graph-diffusion process is task-dependent. Below we detail our runtimes for every step and compare them to the literature.

Pre-uplifting: feature map generation. The time this step takes (Gen. in Table A) depends on the backbone model, on the number of images and on the number of calls to the model per image. The total time ranges from a few seconds up to an hour for approaches such as LangSplat [35], that queries SAM over a grid of points on the image at various resolutions to generate full image segmentation masks.[3] In our experiments, the feature generation step takes from 1 to 5 minutes, except for Sec. 5.4 where we uplift semantic maps generated with LangSplat's approach.

Uplifting. For LUDVIG, uplifting time is linear in the number of images (given a 3D scene representation). Experimentally, it is also linear in the number of feature dimensions, taking 2ms per dimension for an image of size 724×986. As reported in Table A (Up.), uplifting 100 images of dimension 40 takes 9s on average. By contrast, gradient-based optimization requires approximately n_{steps} times this duration, where the number of gradient steps n_{steps} typically ranges from 3,000 to 30,000 for 3D feature distillation [19, 35, 56]. Gradient-based optimization can still be very fast for low-dimensional features such as SAM masks (can take as little as a few seconds, as reported by SA3D-GS [3]) or dimensionality-reduced features (LangSplat [35] trains an autoencoder to reduce the CLIP feature dimension from 512 to 3 and runs for 25 minutes). However, optimization becomes intractable for high-dimensional features such as CLIP and DINO; FMGS [56] relies on an efficient multi-resolution hash embedding of the scene; however, their total training time still amounts to 1.4 hours.

Post-uplifting: graph diffusion. After uplifting, LUDVIG performs graph diffusion using pairwise DINOv2 feature similarities for the segmentation tasks in Sec. 5.2 and 5.3. In Table A, we divide runtimes in two categories:

- **Scene**: operations performed once for the whole scene. This includes querying the Euclidean neighbors for each node, which is log-linear in the number of Gaussians. With 600,000 Gaussians as in our experiments, the step takes about 30s, and can be further optimized by using approximate nearest neighbor search algorithms [45]. Defining S_f based on DINOv2 features is also indepen-

[3] This process takes 24s/image on a GPU 6000 ADA and amounts to an average of 80 minutes for the evaluated scenes.

	Images (#)		Text queries (#)		DINOv2 (s)		CLIP (s)	Graph diffusion (s)		2D segmentation (s)		Total
Scene	Train	Test	Unique	Total	Gen.	Up.	Gen.+Up.	Scene	Prompt	w/ SAM	w/o SAM	(mins)
Teatime	177	6	14	59	45	14	363	42	15	9	0.9	8
Waldo	187	5	18	22	44	18	371	39	19	4	0.5	8
Ramen	131	7	14	71	40	9	227	37	14	11	1	6
Figurines	299	4	21	56	58	38	811	45	22	8	0.8	16

Table A. **Runtimes for evaluation on LERF Segmentation [19, 35].** The last column (Total) reports total time, which breaks down between i) feature map generation (Gen.) and uplifting (Up.) for all training images; ii) graph diffusion, divided between scene-specific (querying neighbors, defining S_f) and prompt-specific (defining P, running diffusion) operations for all text queries; iii) 2D segmentation with/without SAM for all text queries across test images. We also report the number of training and test images and the number of text queries across test images.

	SA-TensoRF	SA-GS	OmniSeg3D	SAGA	LUDVIG	
					SAM	DINOv2
2D feature extraction (sec.)	30	30	900[2]	900[1]	20	30
3D feature training (sec.)	*15*	*4*	900	1500	*4*	40+*2*

(a) **Multi-view segmentation (NVOS)**. *Italic* denote inference-time operations (after 2D scribbles are given). OmniSeg3D and SAGA avoid inference overhead but require longer feature extraction and training.

	LERF	LangSplat	OpenGaussian	LUDVIG	
				w/o diff.	w/ diff.
2D feature extraction (min)	-	50[1]	50[1]	4.5	4.5
3D feature training (min)	25	25	40	2	2.5
Inference (s/query)	30.9	0.26	~ 0.2	0.3	1.3

(b) **Open-vocabulary segmentation (Ramen scene) .**

Table B. **Training and inference times.** All methods build on top of a GS pretraining stage ($\sim$ 10min). Reported values are paper-sourced and indicative, as hardware setups may differ.

dent from the downstream task.

- **Prompt**: operations that are specific to the downstream task. This includes defining the regularization P (*e.g.* training logistic regression model(s)) and running the diffusion. The time taken depends on dimension of the diffused features (*e.g.* number of text queries): 1 to 2 seconds for foreground/background segmentation (a single mask) and 18 seconds on average for LERF segmentation (14 to 21 text queries).

Post-uplifting: segmentation. Our evaluation on LERF involves 2D segmentation with SAM based on 3D relevancy scores. The runtime depends on the number of test images and on the total number of 2D text queries, as it involves one call to the SAM backbone per test image, and multiple calls to the SAM prediction head per text query, as detailed in Appendix A.4. Our total inference time per scene is of $8s$ on average, against $0.8s$ when not using SAM. In contrast, Langsplat does not rely on SAM at inference time, but relies on a computationally expensive feature map generation process, with more than 1 hour runtime.

B.2. Runtime comparisons

Table B reports runtime comparisons for the multi-view (Sec. 5.2) and open-vocabulary (Sec. 5.3) segmentation tasks. Compared to approaches relying on SAM's automatic mask generation, such as OmniSeg3D [52], SAGA [5], LangSplat [35], and OpenGaussian [46], our method offers significantly faster feature generation and uplifting, at the cost of a slightly higher inference time. In particular, the graph diffusion step adds approximately 1 second per query at inference (reported as 14 seconds for Ramen's 14 queries in supplementary Table A).

Our experiments in Sec. 5.4 show that our uplifting process can serve as a drop-in replacement for the training phase in methods optimized for fast inference, such as OpenGaussian [46], which learns from 2D feature maps generated using LangSplat's approach. In these experiments, we adopt the same 2D feature map generation and evaluation protocols as OpenGaussian, replacing only their quantization-based training phase with our fast feature aggregation. This results in significantly faster uplifting while achieving notable accuracy gains.

C. Additional results

C.1. Per-scene multi-view segmentation results

In this section, we present per-scene segmentation results on NVOS and SPIn-NeRF in Tables C, D and E, along with an extended analysis of these results.

Segmentation on SPIn-NeRF. We report our segmentation

	MVSeg	SA3D-GS	SAGA	OmniSeg3D	LUDVIG (Ours)		
3D representation:	NeRF	GS	GS	NeRF	GS		
Uplifting:		SAM	SAM	SAM	DINOv2	SAM	SAM2
Orchids	92.7	84.7	-	92.3	92.6	92.2	91.0
Leaves	94.9	97.2	-	96.0	96.2	96.3	96.3
Fern	94.3	96.7	-	97.5	96.3	97.0	97.0
Room	95.6	93.7	-	97.9	95.7	96.5	96.1
Horns	92.8	95.3	-	91.5	95.1	94.5	94.8
Fortress	97.7	98.1	-	97.9	97.5	98.3	98.3
Fork	87.9	87.9	-	90.4	85.0	86.8	86.7
Pinecone	93.4	91.6	-	92.1	93.2	88.8	90.7
Truck	85.2	94.8	-	96.1	95.6	94.9	93.9
Lego	74.9	92.0	-	90.8	91.1	92.7	92.9
Average	90.9	93.2	93.4	94.3	93.8	93.8	93.8

Table C. Segmentation (IoU) on SPIn-NeRF [32] with DINOv2, SAM and SAM2.

	Fern	Flower	Fortress	HornsC	HornsL	Leaves	Orchids	Trex	Average
NVOS	-	-	-	-	-	-	-	-	70.1
SA3D	82.9	94.6	98.3	96.2	90.2	93.2	85.5	82.0	90.3
OmniSeg3D	82.7	95.3	98.5	97.7	95.6	92.7	84.0	87.4	91.7
SA3D-GS	-	-	-	-	-	-	-	-	92.2
SAGA	-	-	-	-	-	-	-	-	92.6
Ours-DINOv2	84.5	95.6	97.5	97.3	93.4	96.3	91.7	84.7	92.4
Ours-SAM	85.5	97.6	98.1	97.9	94.1	96.4	73.1	88.0	91.3
Ours-SAM2	84.8	97.3	98.3	97.7	93.4	96.7	73.1	89.1	91.3

Table D. Segmentation (IoU) on NVOS [39] with DINOv2, SAM and SAM2.

	Geometry only	Single view		Uplifting		Graph diffusion
Model:	Reference mask	DINOv2	SAM2	DINOv2	SAM2	DINOv2
Orchids	71.3	91.5	78.4	91.5	91.0	92.6
Leaves	72.4	89.3	96.6	94.1	96.3	96.2
Fern	93.9	95.1	96.7	96.7	97.0	96.3
Room	77.4	95.4	95.6	97.3	96.1	95.7
Horns	80.7	90.9	93.0	94.2	94.8	95.1
Fortress	94.3	96.8	97.7	98.6	98.3	97.5
Fork	67.5	85.6	80.5	88.8	86.7	85.0
Pinecone	56.5	92.8	67.8	89.6	90.7	93.2
Truck	60.1	83.6	90.9	95.2	93.9	95.6
Lego	57.3	64.4	89.0	69.9	92.9	91.1
Average	73.1	88.5	88.6	91.6	93.8	93.8

Table E. **Segmentation (IoU) on SPIn-NeRF [32].** We compare purely geometrical reference mask uplifting and reprojection, single-view prediction, feature/mask uplifting, and graph diffusion leveraging DINOv2 or SAM2.

results for the SPin-NeRF dataset [32] in Table C. Our results are comparable to the state of the art while not relying on optimization-based approaches. Surprisingly, our segmentation with DINOv2 using graph diffusion also gives results on par with models leveraging SAM masks. Our lower segmentation results compared to OmniSeg's can be partly attributed to poor Gaussian Splatting reconstruction of highly specular scenes such as the Fork, in which semi-transparent Gaussians floating over the object try to represent reflections or surface effects that are difficult to capture with standard rasterization techniques [16].

Segmentation on NVOS. We report our segmentation results for the NVOS dataset [39] in Table D. Our results are comparable to those obtained by prior work. Again, DINOv2 performs surprisingly well while not having been trained on billions of labeled images like SAM. Compared to SAM, DINOv2 better captures complex objects, but sometimes also captures some background noise. This can be seen in Appendix Fig. B with the example of Trex: while SAM misses out the end of the tail as well as the end of

the ribs, DINOv2 captures the whole Trex, but also captures part of the stairs behind. Visualisations of Orchids in Appendix Fig. B also explain the lower performance of SAM on this scene: the two orchids SAM is missing are not covered by the positive scribbles, which makes the task ambiguous.

Ablation study. In Table E, we compare our segmentation protocol using DINOv2 and SAM2 to multiple simpler variants. More precisely, we evaluate i) a purely geometrical variant that does not use SAM2 or DINOv2, ii) single-view segmentation in 2D based on SAM2 or DINOv2 2D predictions, iii) uplifting DINOv2 features or SAM2 masks into 3D then rendering them for segmentation, as described in Sec. A.1 and A.2, and iv) segmenting using graph diffusion over DINOv2 3D feature similarities.

The purely geometrical approach works well on the forward-facing LLFF scenes (Orchids to Fortress). In these scenes, the reference mask is accurately uplifted and reprojected as the viewing direction changes only a little between each frame. However, it fails on the 360-degree scenes (Fork, Pinecone, Truck, Lego). This points to a suboptimal 3D reconstruction of the scene, likely due to overfitting on the limited numbers of available views [7].

The single-view variants use a similar process for constructing the features and using them for segmentation as in Sec. A.1 and A.2 but without uplifting and rendering. It improves from a purely geometrical approach and performs reasonably well on average, the foreground being well isolated from the rest of the scene. However, as illustrated in Fig. 3, the semantic features are at a much lower resolution than those resulting from 3D uplifting, leading to a coarser segmentation.

3D uplifting considerably boosts results compared to single-view approaches. However, performing segmentation in 2D based on the uplifted DINOv2 features does not benefit from the 3D spatial information and typically fails on the 360-degree scenes (Pinecone, Truck and Lego) which have higher variability across different views. Introducing 3D spatial information through 3D graph diffusion results in a boosted performance on these scenes.

D. Additional visualizations

D.1. Segmentation tasks

Segmentation on NVOS. Fig. B shows our segmentation masks from SAM and DINOv2 for the three most challenging scenes of the NVOS dataset: Fern, Orchids and Trex.

Diffusion process. Fig. C illustrates different steps of the diffusion process for Fern, Leaves, Flower and Trex from the NVOS [39] dataset. Starting from the reference scribbles, the diffusion rapidly spreads through the large neighboring Gaussians. Covering the entire object takes more time for complex structures such as Fern, or for masks with disconnected components such as Orchids. As illustrated in the case of Flower, the last diffusion steps allow spreading to the smaller Gaussians on the flowers' edges, yielding a refined segmentation mask. For Trex, the parts being reached the latest are the head and tail. Their features are further away from the reference features (defined as the average feature over 3D reference scribbles), and therefore the regularization for diffusion is stronger in these regions. Overall once the object has been fully covered, the regularization is very effective at preventing leakage, which allows diffusion to run for an arbitrary number of steps.

Object removal. Fig. D shows comparative visualizations of object removal with N3F [43] and LUDVIG. For rendering the edited RGB image, N3F sets to zero the occupancy for all 3D points belonging to the object. For LUDVIG, we remove all Gaussians pertaining to the 3D semantic mask resulting from graph diffusion. We observe that the regions behind to segmented object are much smoother for LUDVIG than for N3F. Regions unseen from any viewpoint are black for LUDVIG (no gaussians) and result in a background partially hallucinated by NeRF for N3F.

D.2. Visual comparisons of uplifted features

Fig. E show a comparison of LUDVIG's 3D DINOv2 features with learned 3D DINO features of N3D [43]. Their figures are taken from their work. Compared to N3D, LUDVIG produces a more fine-grained reconstruction of the background (notably in the trex and horns scenes) and smoother features across all scenes.

D.3. 3D features for ScanNet semantic segmentation

Fig. F presents visualizations of 3D semantic features obtained by uplifting 2D feature maps generated by OpenGaussian [46]. Following the approach of LangSplat [35], the 2D features are computed by assigning object-level CLIP embeddings to segmentation masks produced by SAM in *everything* mode.

Surprisingly, despite the very constrained training conditions used during Gaussian Splatting reconstruction (frozen positions and disabled densification process), the uplifting still yields coherent and meaningful 3D semantic features.

D.4. Comparison to GaussianEditor's uplifting

Our aggregation procedure in Eq. (3) from the main paper, illustrated in Fig. 2, bears similarity with the one from [6] for uplifting 2D binary masks to a 3D Gaussian splatting scene. In their method, uplifted masks are thresholded to create 3D binary masks that are used for semantic tracing. Specifically, they rely on rough 3D segmentation masks to selectively optimize Gaussians that are relevant for an editing task. Unlike in Eq. (3) and (5) from the main paper, [6] propose to normalize their uplifted masks based on the total count of view/pixel pairs (d, p) contributing to the mask of

(a) Reference image (b) Ground truth mask (c) DINOv2 mask (d) SAM mask

Figure B. Segmentation results on NVOS [39] with DINOv2 and SAM.

a Gaussian i, i.e. $\sum_{d,p\in\mathcal{S}_i} 1$, without taking the rendering weight $w_i(d,p)$ into account. Consequently, the uplifted features tend to have larger values for large, opaque Gaussians. Fig. G shows a qualitative comparison between 3D DINOv2 features obtained using the aggregation proposed by [6] and our approach. The aggregation by [6] fails to assign the right semantics to large gaussians, which is particularly visible in scenes with high specularity such as Room. This showcases the importance of defining 3D features as *convex combinations* of 2D pixel features.

D.5. Visualization of CLIP segmentation results

In this section, we present illustrations of the impact of the diffusion process (Fig. H), and comparative visualizations of localization heatmaps with LangSplat and LERF (Fig. I).

D.5.1. DINOv2-guided graph diffusion

Fig. H shows 2D segmentation masks colored by CLIP relevancy scores, obtained with and without leveraging SAM and/or DINOv2-guided graph diffusion for refining 3D relevancy scores.

Direct segmentation from raw 3D relevancy scores. Isolating a specific object in the scene directly based on CLIP relevancy scores is challenging: the segmentation masks obtained by automatic thresholding include parts of other objects with similar features, like for the *sheep* (segmentation of the bear nose) and the *spoon*. The segmentation might also cover surroundings of the object of interest simply due to the low resolution of CLIP visual features, such as in the *knife* example.

2D segmentation with SAM. SAM delivers a precise 2D segmentation of the object covered by points with the highest relevancy scores. However, point prompts may not span the entire object, resulting in undersegmentation, like for the *sheep*. In some cases, point prompts with the highest relevancy may even be located on the wrong object, resulting in an entirely wrong segmentation (*e.g.*, the bowl segmented instead of the *spoon*).

Relevancy score refinement with graph diffusion based on 3D DINOv2 features. The graph diffusion process starts with positive weights for Gaussians with the highest relevancy scores, and propagates their weight to neighbors with similar DINOv2 features. However in cases where the object of interest consists of multiple subparts (e.g. for the *sheep*), the final distribution of weights may be inhomogeneous and the automatic thresholding may select only one subpart. Also, if multiple close objects are to be segmented (e.g. with the *knife*), the final weights may cover surrounding Gaussians and the final thresholding might not clearly isolate the objects.

3D graph diffusion with 2D SAM segmentation. Com-

bining 3D graph diffusion and 2D SAM segmentation helps solving the aforementioned problems observed when using either of the two approaches individually. The diffusion process allows selecting a large set of point prompts for SAM spanning the object of interest without covering other object with similar features, resulting in an accurate segmentation.

D.5.2. Comparisons for open-vocabulary localization

Fig. I illustrates open-vocabulary object localization with LERF [19], LangSplat [35] and LUDVIG. Both LangSplat and LUDVIG correctly localize all four example objects. For queries such as the chopsticks, LangSplat's localization is more precise, as the CLIP features are constructed by generating full image segmentation masks with SAM. This process is computationally expensive, as constructing a full segmentation mask requires querying SAM over a grid of points on the image and takes about 23s for a single image (on a GPU RTX 6000 ADA), which amounts to an average of 80 minutes for a scene from the LERF dataset. However, it yields coherent instance-level CLIP representations, which is desirable for downstream segmentation tasks.

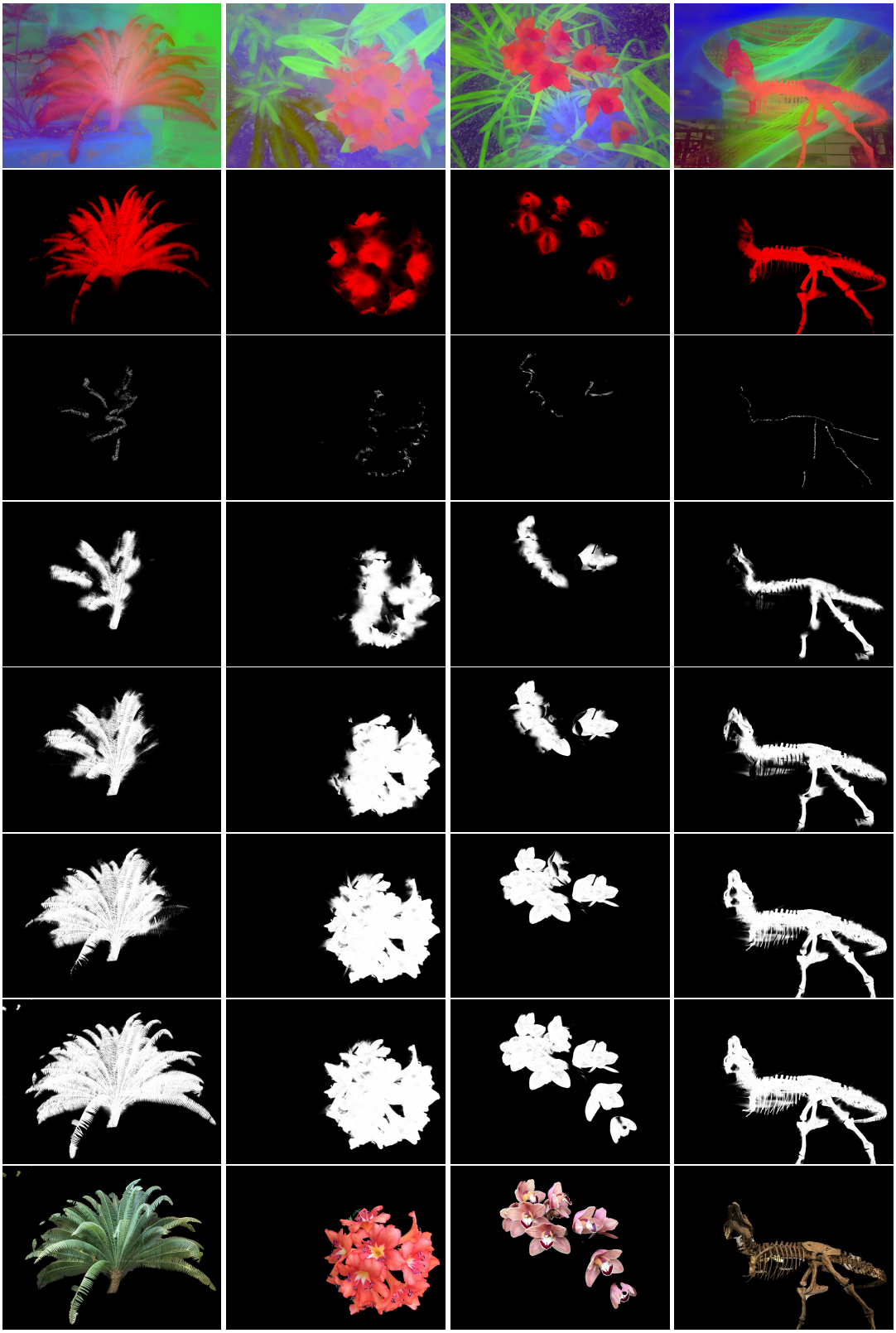
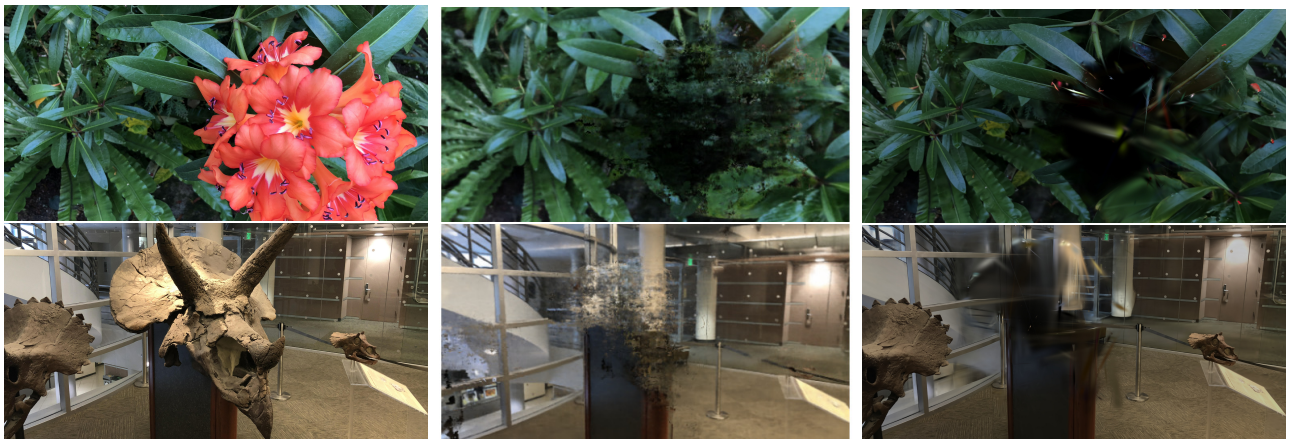


Figure C. **Illustration of the graph diffusion process.** 2D projections of i) first three PCA components of DINOv2 3D features, ii) unary regularization term (red), iii) weight vector g_t at timesteps $t \in \{0, 3, 5, 10, 100\}$, iv) RGB segmentation obtained using a mask based on the 2D projection of g_{100}.



(a) RGB image (b) N3F [43] (c) LUDVIG (ours)

Figure D. **Object removal.** 3D segmentation, removal and rendering for LUDVIG and N3F [43]. For N3F, figures are taken from [43].

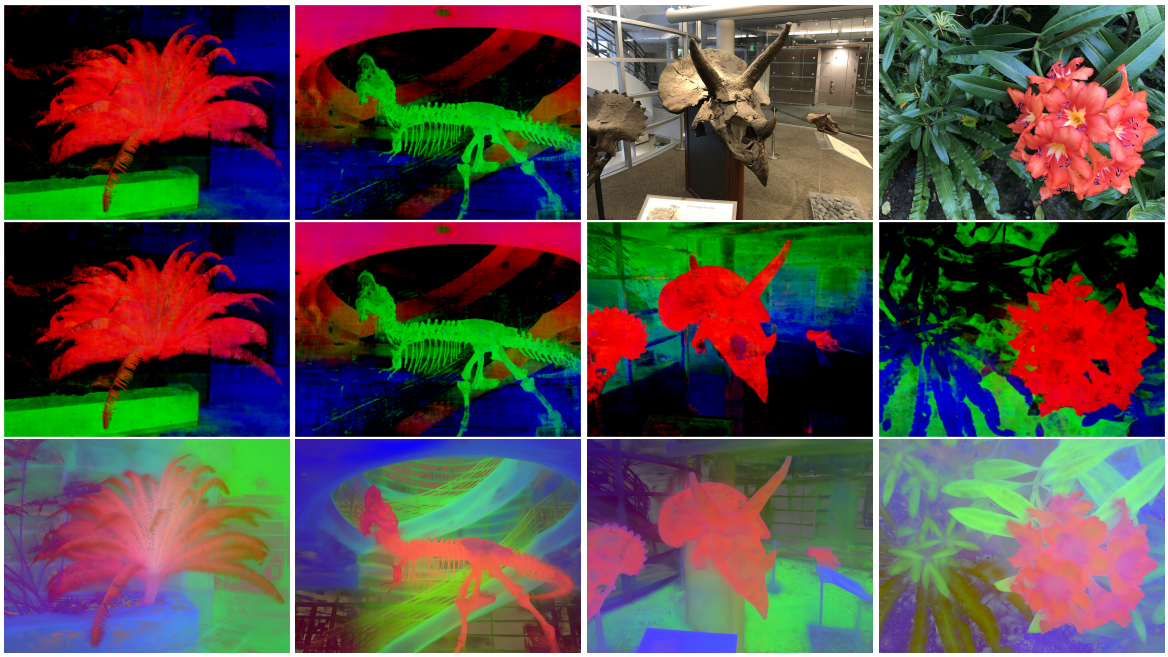


Figure E. Comparison between LUDVIG's uplifted DINOv2 features (bottom) and N3F's [43] learned DINO features (top). For N3F, figures are taken from [43].

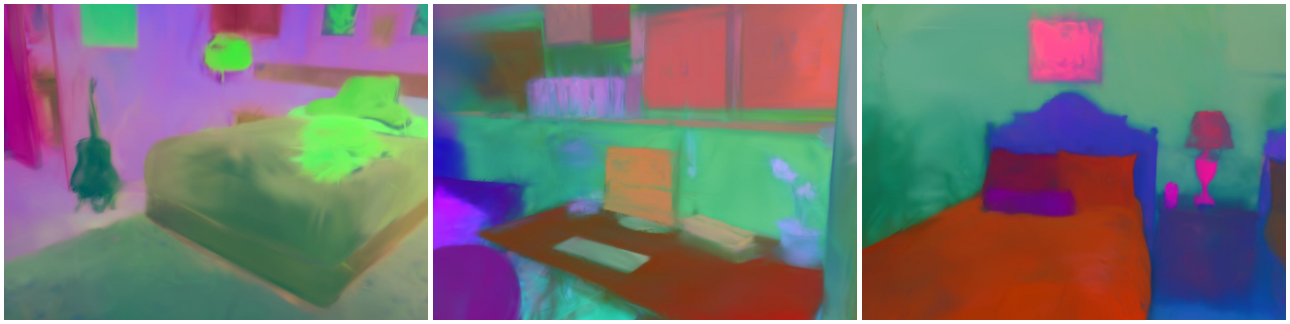
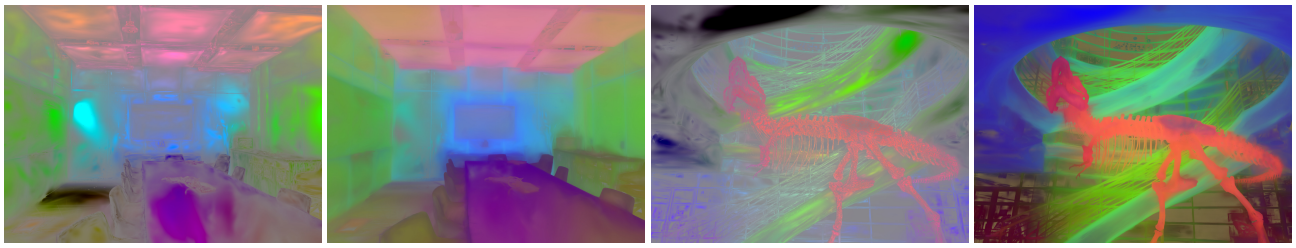


Figure F. PCA of uplifted semantic features on ScanNetv2, obtained by assigning object-level CLIP features to SAM segments.



(a) GaussianEditor (b) LUDVIG (c) GaussianEditor (d) LUDVIG

Figure G. **Comparison to GaussianEditors's uplifting.** Comparison of PCA visualization of uplifted features between LUDVIG's and GaussianEditor's aggregation [6].

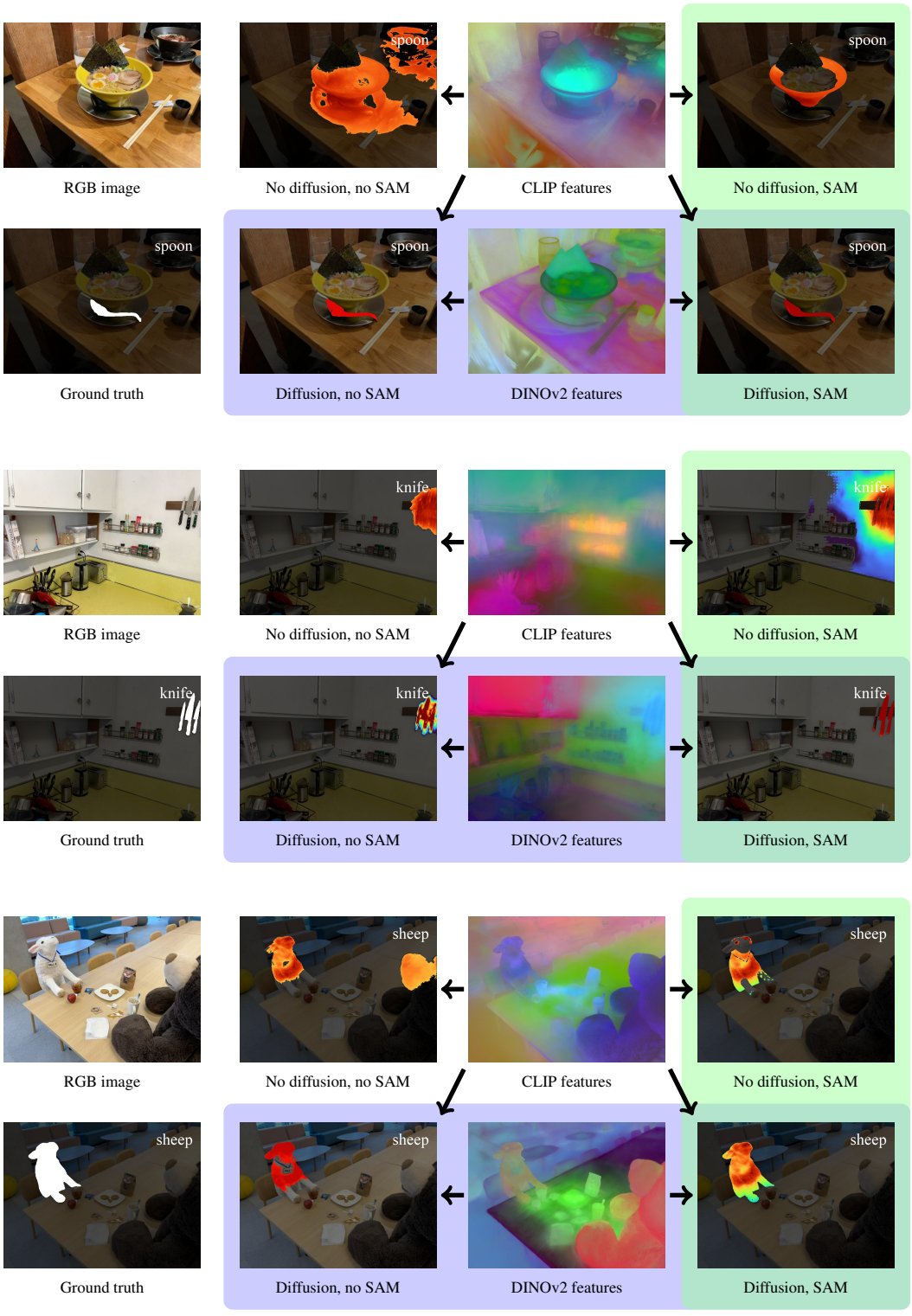


Figure H. **Open-vocabulary object segmentation with and without using 3D graph diffusion (blue) and/or 2D SAM segmentation (green).** Projections of 3D CLIP and DINOv2 features colored by three main PCA components and 2D segmentation masks colored by relevancy scores.

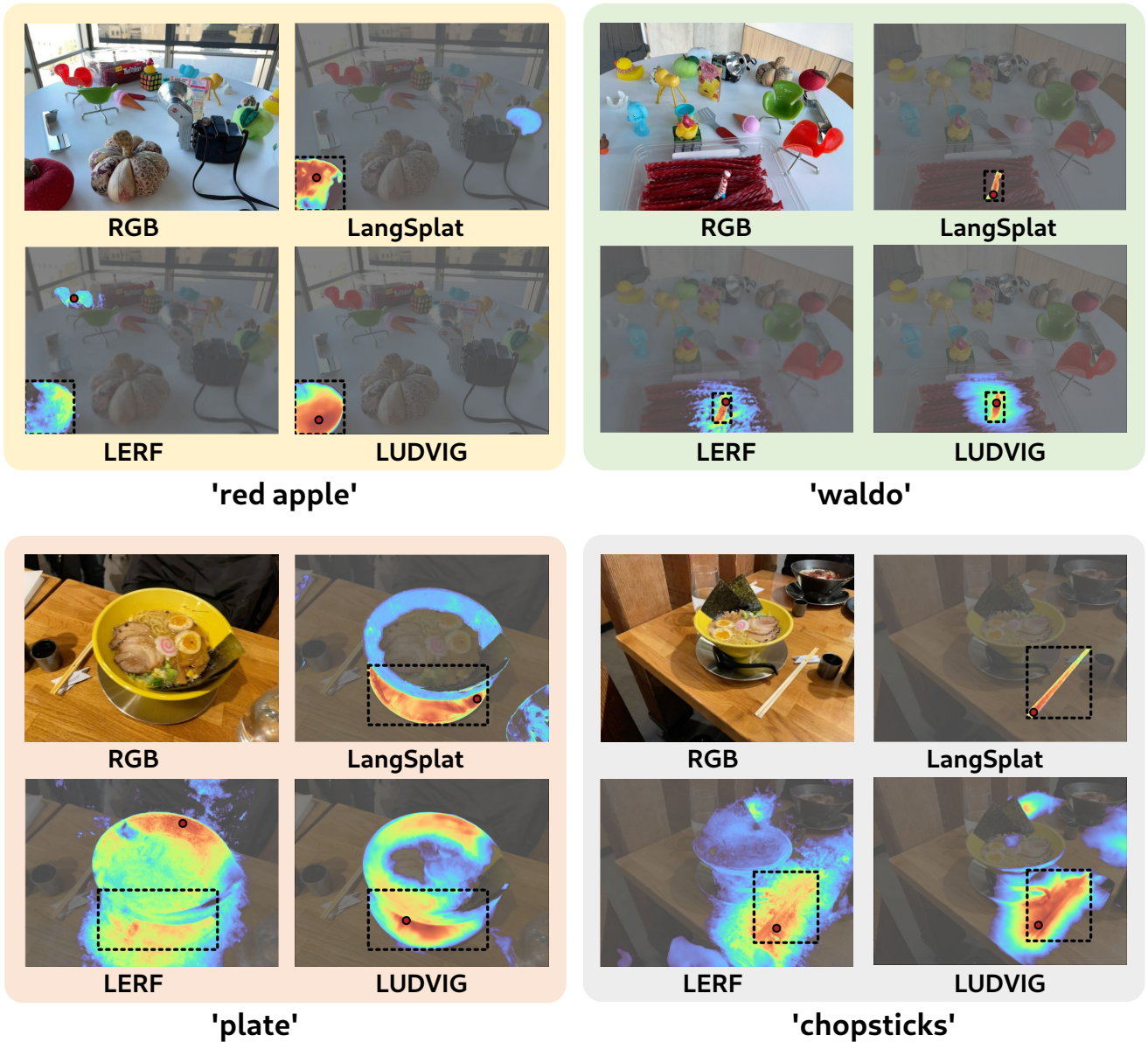


Figure I. **Qualitative comparisons of open-vocabulary 3D object localization on the LERF dataset.** The red points are the model predictions and the black dashed bounding boxes denote the annotations. This figure is taken and adapted from LangSplat's website (https://langsplat.github.io/), licensed under a Creative Commons Attribution-ShareAlike 4.0 International License.