Supplementary Material for 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

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 [8, 14] 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 inter-

est). 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 [9] for SAM 2.

A.2. Multiple-view segmentation with DINOv2

DINOv2 [11] 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 [5] 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 [11], 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:

$$S_f(f_i, f_j) = \exp\left(-\frac{||f_i - f_j||_2^2}{bs_f^2}\right) \tag{1}$$

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

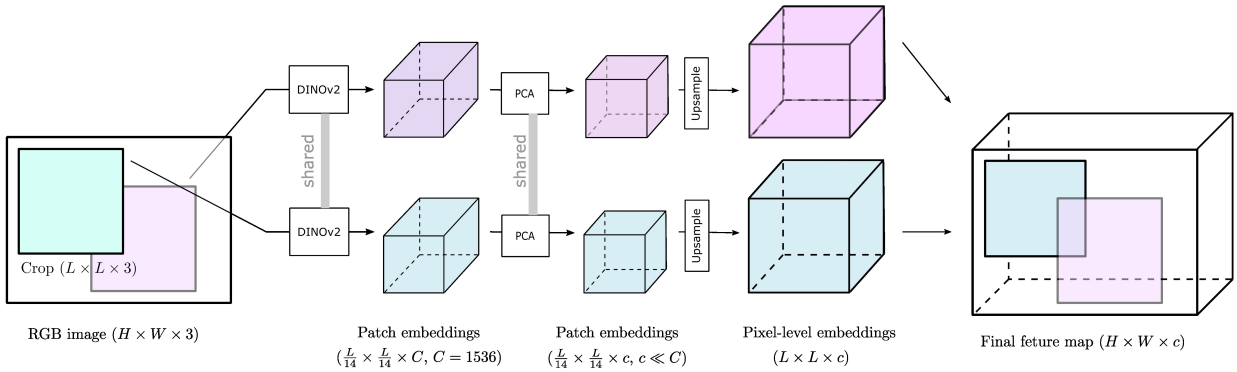


Figure A. Sliding windows for construction of DINOv2 feature maps.

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 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 [7] and LangSplat [13] 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:

$$r_{i,q} = \min_k \frac{\exp(T \cdot f_i \cdot \phi_q)}{\exp(T \cdot f_i \cdot \phi_q) + \exp(T \cdot f_i \cdot \phi^k_{cano})}, \quad (2)$$

where T is a temperature parameter set to 10 by [7] and ϕ^k_{cano} is the CLIP embedding of a predefined canonical phrase chosen from "object," "things," "stuff," and "texture." Note that [13] 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 [13] 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 [12].

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 [12].

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. 1 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 [12] 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 [13], that queries SAM over a grid of points on the image at various resolutions to generate full image segmentation masks.[2] 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 [7, 13, 20]. 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 [1]) or dimensionality-reduced features (LangSplat [13] 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 [20] 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:

[2] 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 [7, 13].** 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[1]	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.

- **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 [17]. Defining S_f based on DINOv2 features is also independent 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 [19], SAGA [2], LangSplat [13], and OpenGaussian [18], 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 [18], 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.

	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 [10] 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 [15] 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 [10]**. We compare purely geometrical reference mask uplifting and reprojection, single-view prediction, feature/mask uplifting, and graph diffusion leveraging DINOv2 or SAM2.

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 results for the SPin-NeRF dataset [10] 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 [6].

Segmentation on NVOS. We report our segmentation results for the NVOS dataset [15] 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 [4].

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 [15] 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 [16] 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 [16]. 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 [18]. Following the approach of LangSplat [13], 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 [3] for uplifting 2D binary masks to a 3D Gaussian splatting

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

Figure B. Segmentation results on NVOS [15] with DINOv2 and SAM.

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, [3] propose to normalize their uplifted masks based on the total count of view/pixel pairs (d, p) contributing to the mask of 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 [3] and our approach. The aggregation by [3] 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. Combining 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 [7], LangSplat [13] 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.

References

[1] 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. 4

[2] 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. 5

[3] 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. 7, 8, 13

[4] 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. 7

[5] 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

[6] 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. 6

[7] 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. 3, 4, 5, 9

[8] 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

[9] Chun Hung Li and CK Lee. Minimum cross entropy thresholding. *Pattern recognition*, 26(4):617–625, 1993. 2

[10] 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. 6

[11] 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. 2

[12] Nobuyuki Otsu et al. A threshold selection method from gray-level histograms. *Automatica*, 11(285-296):23–27, 1975. 4

[13] 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. 3, 4, 5, 7, 9

[14] 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. 1

[15] 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

[16] 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. 7, 12

[17] 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. 5

[18] 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. 5, 7

[19] 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. 5

[20] 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. 4

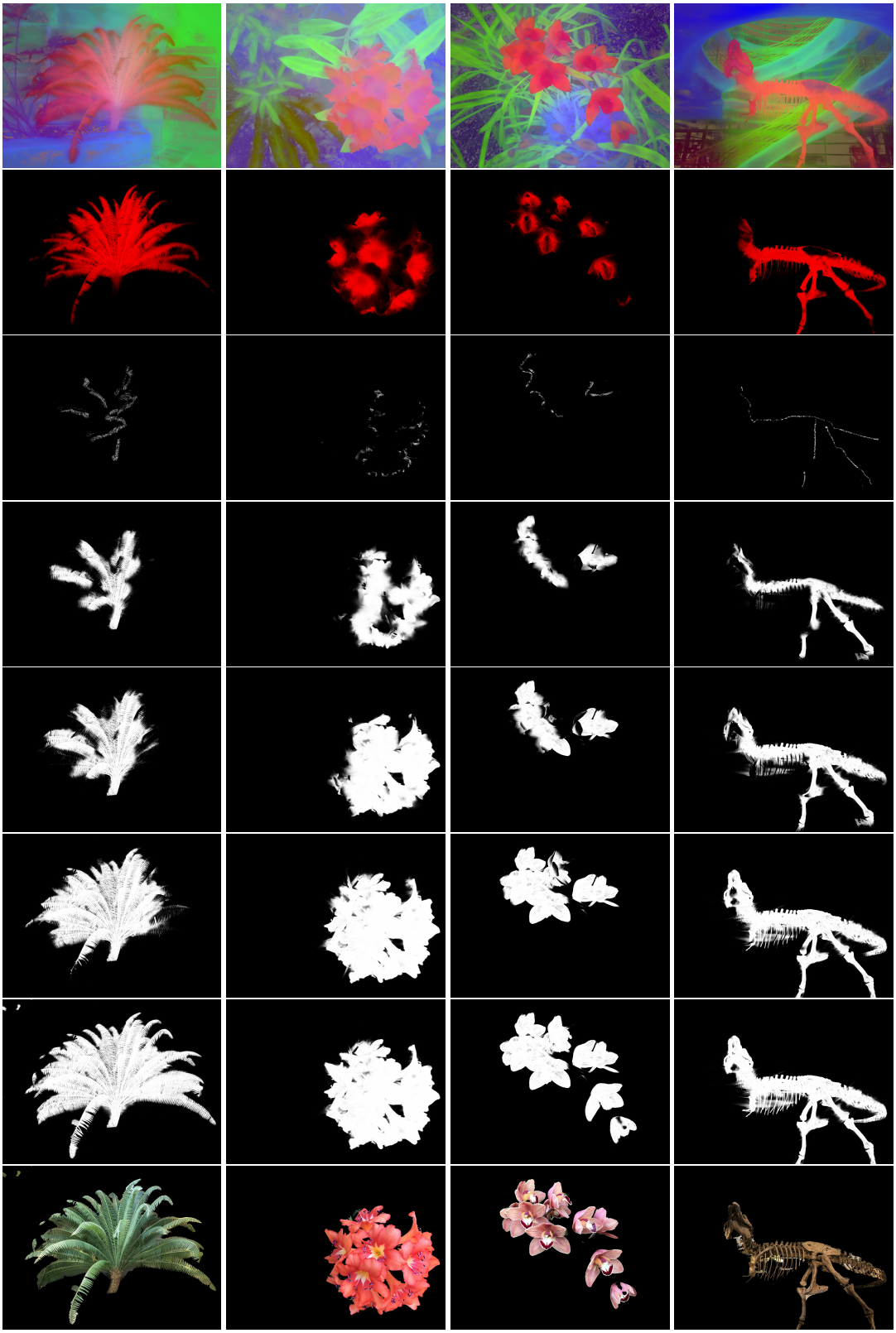
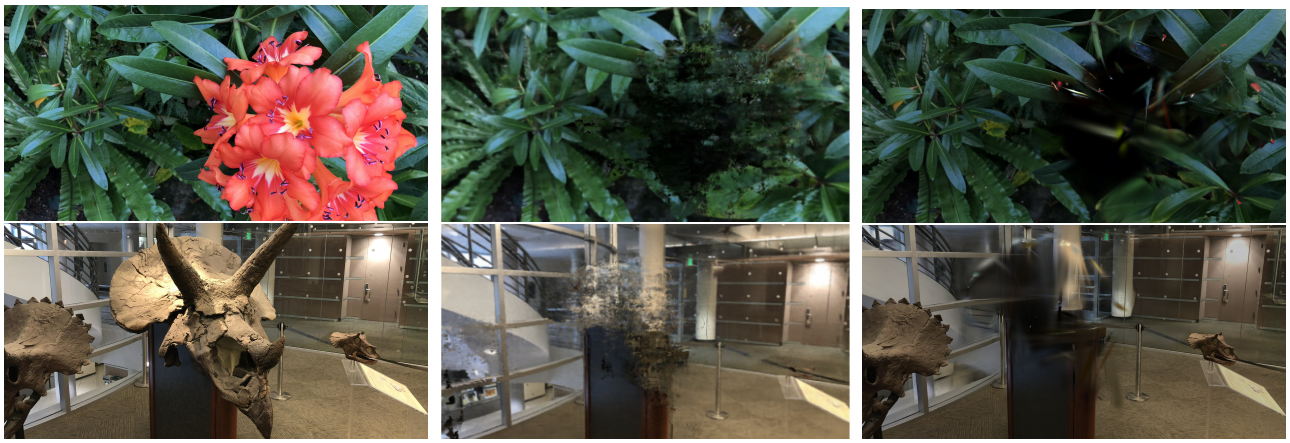


Figure C. **Illustration of the graph diffusion process.** 2D projections of i) first three PCA components of DINOv2 3D features, ii) unary regularization term (red), iii) weight vector g_t at timesteps $t \in \{0, 3, 5, 10, 100\}$, iv) RGB segmentation obtained using a mask based on the 2D projection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



(a) RGB image (b) N3F [16] (c) LUDVIG (ours)

Figure D. **Object removal.** 3D segmentation, removal and rendering for LUDVIG and N3F [16]. For N3F, figures are taken from [16].

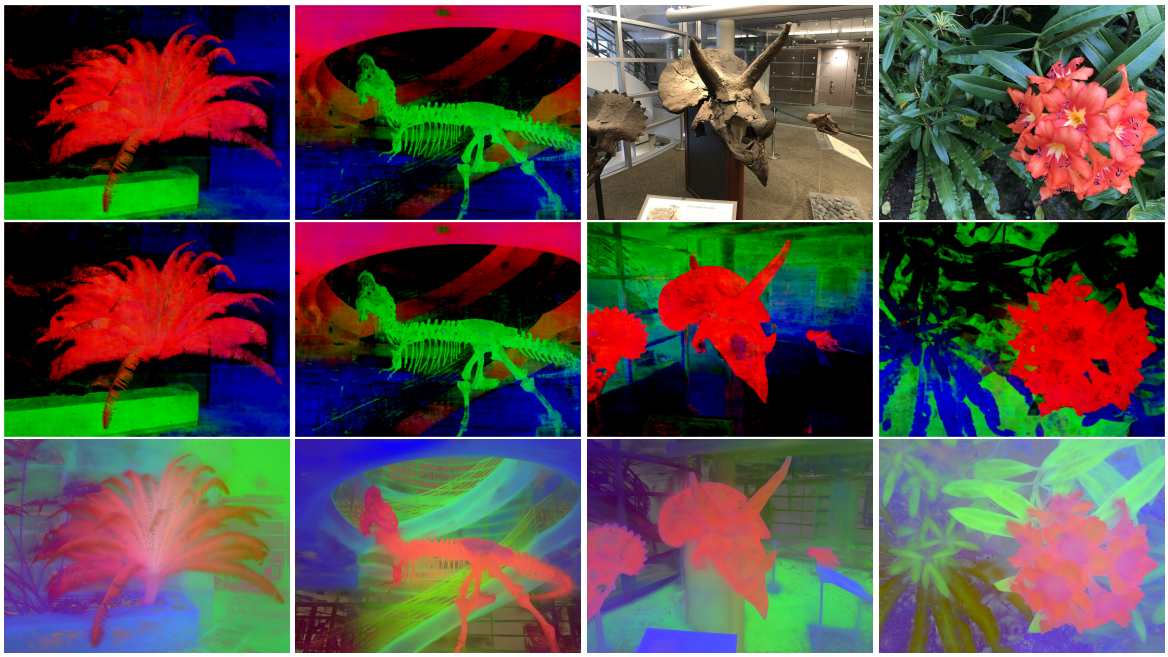


Figure E. Comparison between LUDVIG's uplifted DINOv2 features (bottom) and N3F's [16] learned DINO features (top). For N3F, figures are taken from [16].

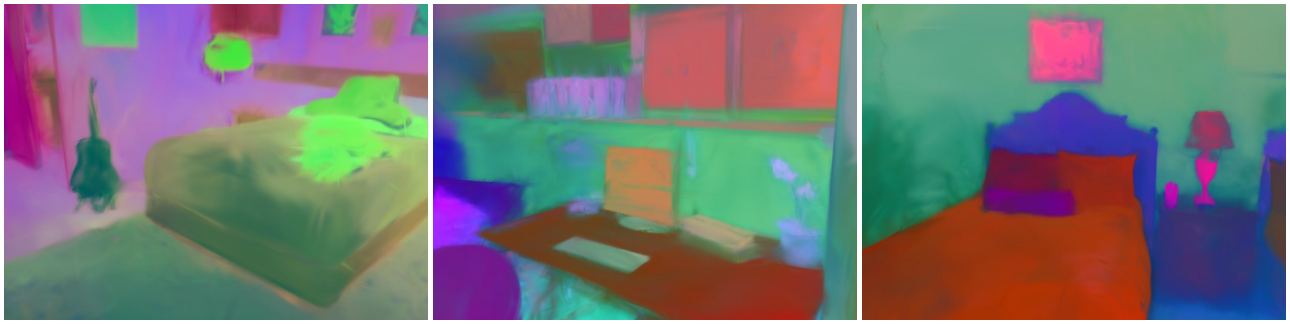
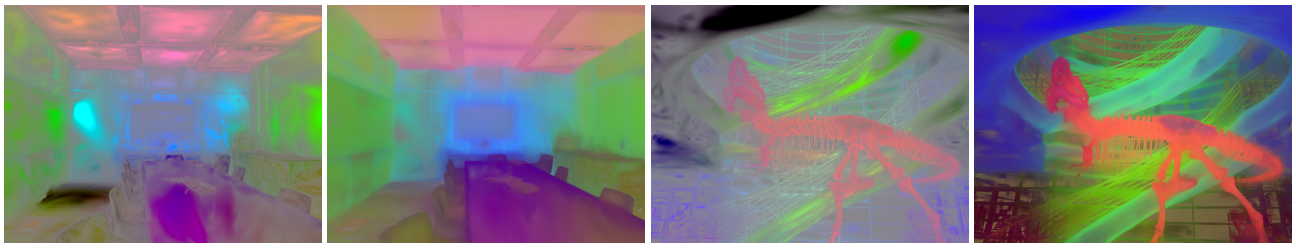


Figure F. PCA of uplifted semantic features on ScanNetv2, obtained by assigning object-level CLIP features to SAM segments.



(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 [3].

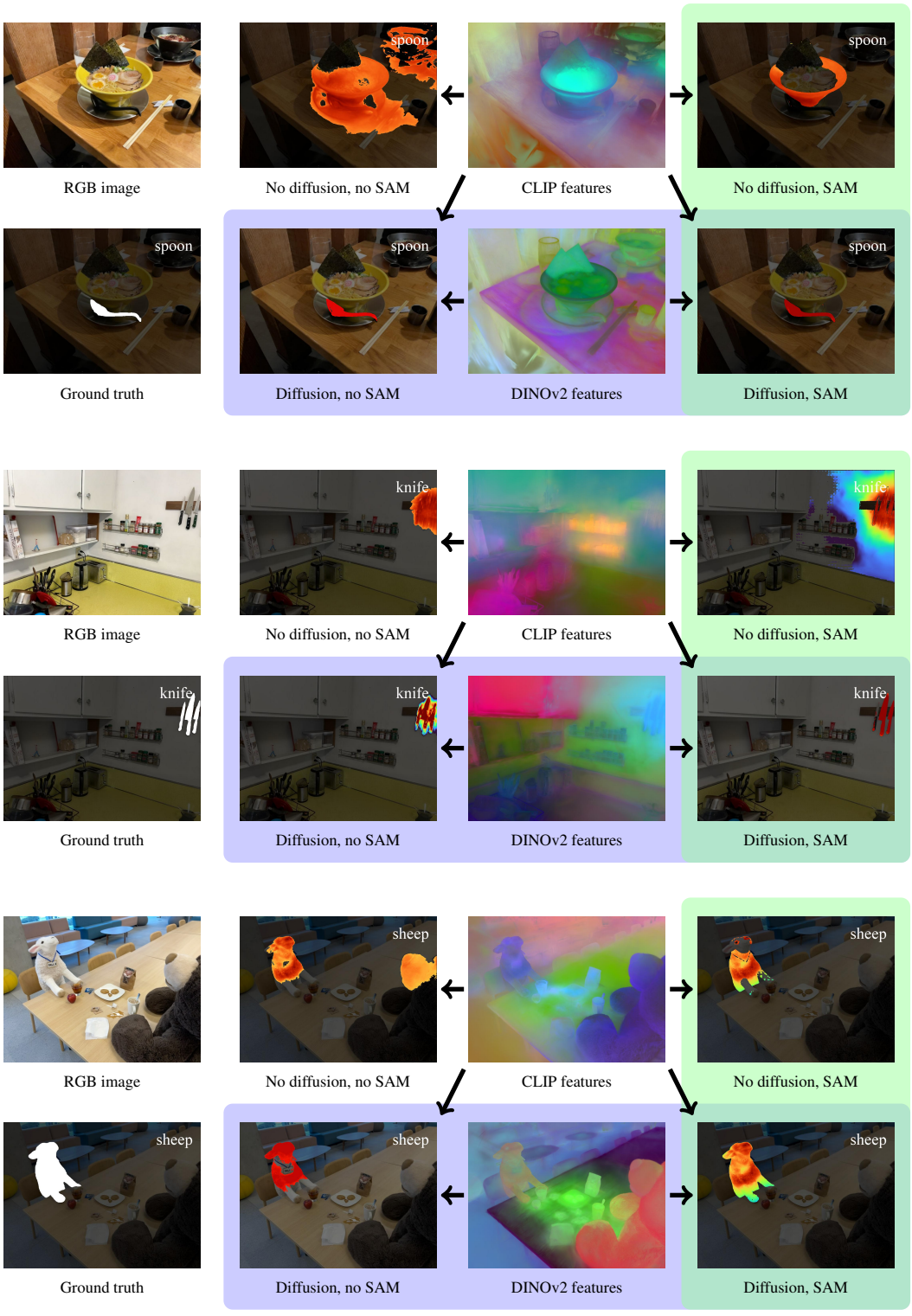


Figure H. **Open-vocabulary object segmentation with and without using 3D graph diffusion (blue) and/or 2D SAM segmentation (green).** Projections of 3D CLIP and DINOv2 features colored by three main PCA components and 2D segmentation masks colored by relevancy scores.

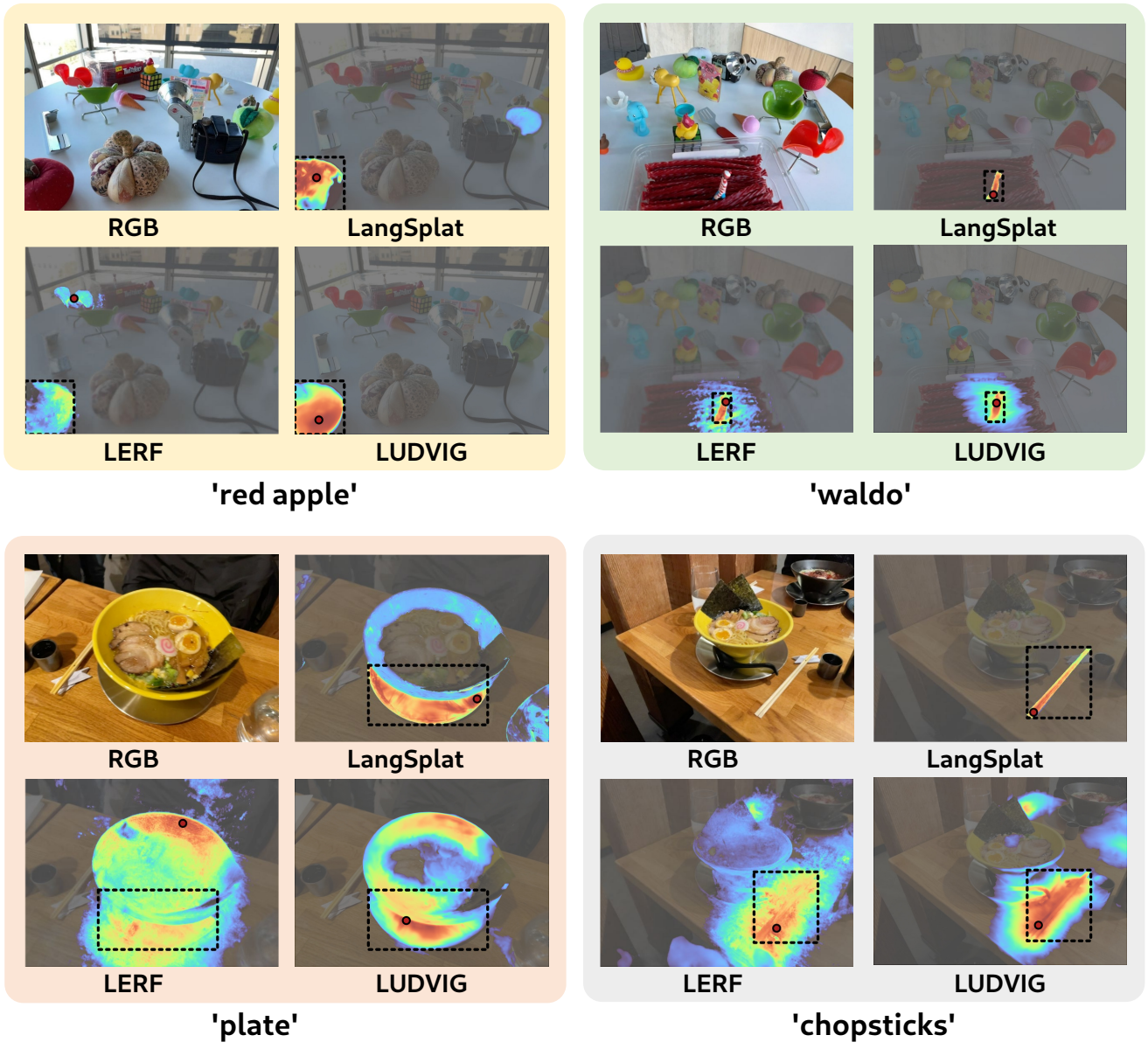


Figure I. **Qualitative comparisons of open-vocabulary 3D object localization on the LERF dataset.** The red points are the model predictions and the black dashed bounding boxes denote the annotations. This figure is taken and adapted from LangSplat's website (https://langsplat.github.io/), licensed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