

Explicit Memory through Online 3D Gaussian Splatting Improves Class-Agnostic Video Segmentation

Anthony Opipari¹, Aravindhan K Krishnan², Shreekanth Gayaka², Min Sun²
Cheng-Hao Kuo², Arnie Sen², Odest Chadwicke Jenkins¹

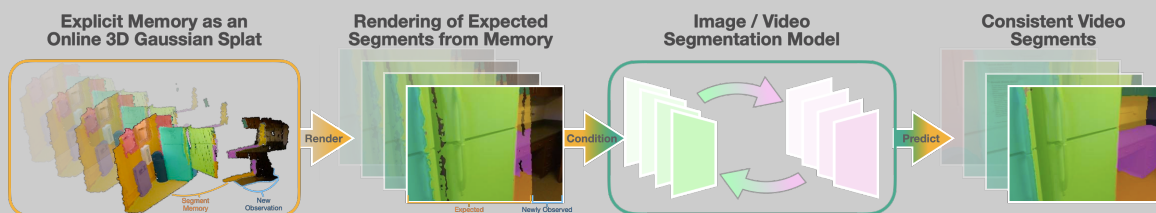


Fig. 1: Illustration of this paper’s core insight: Using explicit memory in the form of an online 3D Gaussian splat to condition image and video segmentation models for improved video segmentation consistency.

Abstract—Remembering where object segments were predicted in the past is useful for improving the accuracy and consistency of class-agnostic video segmentation algorithms. Existing video segmentation algorithms typically use either no object-level memory (e.g. FastSAM) or they use implicit memories in the form of recurrent neural network features (e.g. SAM2). In this paper, we augment both types of segmentation models using an explicit 3D memory and show that the resulting models have more accurate and consistent predictions. For this, we develop an online 3D Gaussian Splatting (3DGS) technique to store predicted object-level segments generated throughout the duration of a video. Based on this 3DGS representation, a set of fusion techniques are developed, named FastSAM-Splat and SAM2-Splat, that use the explicit 3DGS memory to improve their respective foundation models’ predictions. Ablation experiments are used to validate the proposed techniques’ design and hyperparameter settings. Results from both real-world and simulated benchmarking experiments show that models which use explicit 3D memories result in more accurate and consistent predictions than those which use no memory or only implicit neural network memories.

Project Page: <https://topipari.com/projects/FastSAM-Splat>

Index Terms—Object Detection, Segmentation and Categorization; RGB-D Perception

I. INTRODUCTION

OBJECT-level semantic mapping is a crucial challenge in deploying autonomous robots in domestic human environments [1]. In order to construct useful maps in these settings, robots must detect and segment objects from any category, including those that were unknown before deployment. This challenge is compounded by the substantial variability of real-world environments, where occlusion, low illumination, duplicated, and dynamic objects are common [2]. Even

Manuscript received: May, 15, 2025; Revised August, 13, 2025; Accepted September, 20, 2025.

This paper was recommended for publication by Editor Markus Vincze upon evaluation of the Associate Editor and Reviewers’ comments.

¹University of Michigan, {topipari, ocj}@umich.edu

²Amazon Inc., {krsar, sgayaka, minnsun, chkuo, senarnie}@amazon.com

Digital Object Identifier (DOI): see top of this page.

©2026 IEEE

the robot’s own embodiment—its camera placement, sensor modality, motion profile, etc.—is a source of variability that limits the generalization of state-of-the-art segmentation algorithms in the open-world [3]. To address these challenges, recent research has focused on the potential for class-agnostic vision foundation models, such as SAM [4] and FastSAM [5] to contribute to semantic mapping. While foundation models like these have shown excellent performance in class-agnostic image segmentation, their predictions are not temporally consistent [6]. Meanwhile, advances in mapping systems like Clío [7] have identified the consistency of image-level segments as a key enabler of open-world semantic maps. This observation motivates the use of *video segmentation* in place of static image segmentation algorithms for robotic mapping, where the consistency of object segments is critical [8]. An expanding body of work has proposed techniques to ensure these models predict consistent segments across space [9]–[12]. Building on this emerging area of research, this paper aims to explore the following question: **Can estimated image-level segments be combined efficiently over time to enhance the consistency of downstream video segments (Fig. 1)?**

In particular, we set out to improve the temporal consistency and accuracy of state-of-the-art class-agnostic video segmentation. Unlike traditional segmentation approaches that are trained to segment objects from a predefined closed-set of categories, class-agnostic methods enable robots to track objects from categories not seen during training. The recent SAM2 [6] video segmentation model achieved state-of-the-art performance on this task without using an explicit 3D memory of its past predictions. We hypothesize that models like SAM2 would benefit from access to a dense 3D memory of past predictions (i.e. a robot’s object map). The present study sets out to evaluate this hypothesis by developing class-agnostic video segmentation models which use explicit 3D memories to inform their predictions.

This paper proposes an approach that uses a robot’s estimated semantic map to improve foundation segmentation

model predictions in video segmentation. Specifically we develop two models, referred to as **FastSAM-Splat** and **SAM2-Splat**, which build upon the FastSAM and SAM2 foundation models respectively, by integrating explicit spatial memory in the form of a 3DGS. The proposed approach is inspired by recent online 3DGS techniques [13]–[15], which construct high-fidelity, explicit maps of a robot’s environment in real-time using videos as input. Our proposed model builds a 3DGS memory alongside its video segmentation, allowing the 3DGS to store a spatial history of past predictions. At each step in the process, the 3DGS map can then be used to refine the foundation model’s prediction either with a segment-matching algorithm (FastSAM-Splat) or a re-prompting strategy (SAM2-Splat).

This paper makes the following contributions:

- 1) **FastSAM-Splat**: An extension of FastSAM—which originally lacks temporal memory—that integrates a 3DGS memory to improve segmentation consistency.
- 2) **SAM2-Splat**: A 3DGS-based re-prompting strategy to improve SAM2—which originally uses only implicit memories—by incorporating explicit 3D memories to reduce inconsistent predictions.
- 3) **Experimental** results showing models which use explicit 3D memories produce more consistent video segments than those with no or only implicit memories.

II. RELATED WORK

Video segmentation tasks typically fall into one of two types: semantic or class-agnostic segmentation. In video semantic segmentation, algorithms predict pixel-level segments for each object in a video along with a classification of which categories the objects belong while video instance segmentation requires models distinguish between multiple objects of the same category [16]. Many approaches for semantic video segmentation specialize to specific categories of interest [17]–[22]. For instance, Video K-Net [18] learns a specialized kernel for each class and each instance. Similarly, PAOT [21] and Tube-Link [22] are transformer-based architectures that associate segments throughout the video based on which category the segments belong to. **Class-agnostic video segmentation** tasks on the other hand, require that objects be segmented and tracked *regardless of their semantic class* [23]. When only a few specific objects are of interest, video object segmentation is considered [24]–[28], whereas class-agnostic video instance segmentation sets out to segment and track every object of every class [3]. Within class-agnostic video segmentation, Siam et al. proposed using motion cues in the form of optical flow to separate object segments [23]. The present paper builds on these ideas to consider whether motion and depth cues can improve video segmentation in open-world robotic settings.

Open-world segmentation tasks consider the scenario in which robots must segment categories of objects that were unknown during training. Many approaches have been proposed for open-world image segmentation [4], [5], [29]–[33] and interest in open-world video segmentation has been growing [3], [6], [21], [23], [28], [34]–[37]. For distinguishing foreground and background objects in zero-shot settings, object motion through optical flow has been used [23], [28], [35]. Wang

et al. [38], [39] and Li et al. [40] propose using language features in open-world video segmentation. SegGPT [34] proposed using an image-level foundation model and in-context coloring for open-world video segmentation. More recently, Ravi et al. [6] extended the image-level Segment Anything Model (SAM [4]) with recurrent neural network features to improve video segmentation efficiency and consistency. Similarly, GLEE [41] proposed using bipartite feature matching to extend an image-level foundation model to video segmentation tasks. We are inspired by the open-world generalization of these foundation segmentation models but observe that they do not use explicit memory to reduce segment inconsistencies. Ding et al. [37] observed that SAM2 suffers from “error accumulation” on long videos, particularly when objects are occluded and re-appear, and proposed a temporal memory tree for video object segmentation. In contrast, we set out to segment and track *every object* and focus on robotic use cases, where embodiment (i.e. depth and camera pose) may be used to form memories. For these settings, FastSPAM [3] was proposed to extend the image-level FastSAM [5] model to class-agnostic video segmentation by tracking a set of sparse object centroids for recursive self-prompting. Building on these ideas, this paper considers the potential for a dense 3D memory of object segments to be used to condition segment predictions and further improve video segmentation consistency.

3D Gaussian Splatting is a powerful representation for dense 3D scene reconstruction and fast novel-view synthesis [42]. While the original 3DGS representation was limited to reconstructing scene geometry and appearance, a growing body of work has focused on encoding image-level segments in 3DGS reconstructions [11], [43]–[48]. Gaussian grouping [44] and Gaga [11] assign learnable identity embeddings to each Gaussian and decode instance segments with a scene-level neural network. CoSSegGaussians [45] assigns pre-trained foundation model features to the Gaussians and trains a decoder to generate instance segments. LangSplat [46] uses pre-trained language features for each Gaussian; however, these are limited to category-level segmentation. SAGA [48] and EgoLifter [47] use contrastive learning to model semantic similarity between Gaussians, while FlashSplat [43] uses linear programming to assign segment identities, assuming a fixed 3DGS reconstruction as input. A related line of work has explored segmenting 3D scenes using neural radiance fields (NeRF) rather than 3DGS [10], [49]–[51]. These are generally more computationally expensive than 3DGS-based methods due to their use of per-pixel ray-marching [14]. Both 3DGS and NeRF-based segmentation methods rely on global reconstruction, making them unsuitable for video segmentation, where new objects must be integrated incrementally over time. In contrast, recent work on simultaneous localization and mapping has demonstrated how 3DGS models can be constructed incrementally from video input [13]–[15]. However, these incremental 3DGS reconstruction approaches do not encode object segments. This paper proposes augmenting online 3DGS reconstruction with instance-level segments to form an explicit 3D memory which can improve the consistency of predicted video segments.

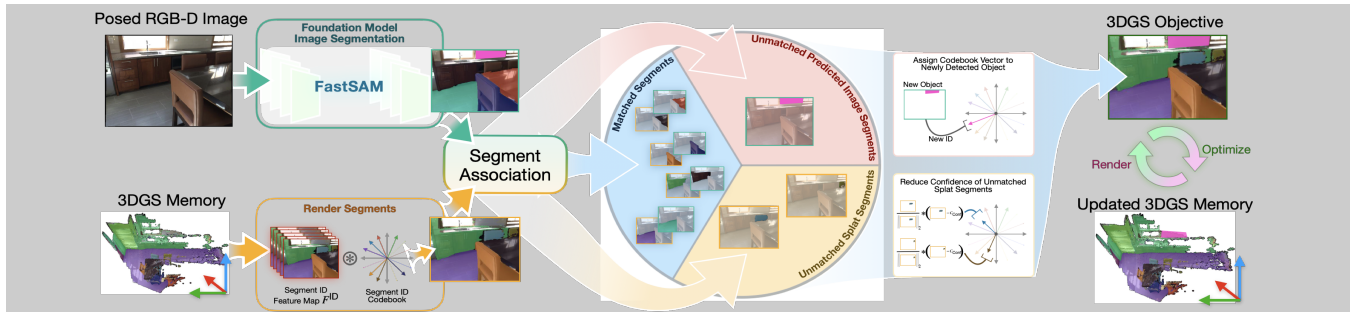


Fig. 2: Illustration of the FastSAM-Splat model. Image-level ‘Predicted image segments’ refers to an image-level segment output from the foundation image segmentation model (FastSAM).

III. APPROACH

A. Preliminaries

Problem Definition: The objective of *class-agnostic video instance segmentation* is to detect, segment and track every visible object instance within a video, regardless of the objects’ semantic class. Given a video sequence with T image frames, $I^{1:T} = \{I^1, \dots, I^T\}$, we set out to predict one segment tube for each object. A predicted segment tube is defined as a sequence of per-frame instance masks, represented as $\hat{M}_{z_i} = \{\hat{m}^1, \dots, \hat{m}^T\}_{z_i} \in \{0, 1\}^{H \times W \times T}$ where z_i represents a unique identifier for a single object. Ground truth segment tubes are defined similarly, based on annotations for each frame in the video. In addition to the input RGB images, we assume models have access to an observed depth image and camera pose for each frame.

3DGS Representation: We adopt a 3DGS representation to model a robot’s dense 3D memories as a collection of anisotropic Gaussians. Each Gaussian primitive is parameterized by its 3D position ($\mu \in \mathbb{R}^3$), orientation ($q \in \mathbb{R}^4$), scale ($s \in \mathbb{R}^3$), opacity ($\sigma \in [0, 1]$), and color ($c \in \mathbb{R}^3$). In order to store semantic memories, we extend this 3DGS parameterization [42] by associating each Gaussian with a segment ID feature vector, $f^{\text{ID}} \in \mathbb{R}^{D_{\text{ID}}}$, which encodes the Gaussian’s segment identity in a continuous feature space. Based on this representation and given a calibrated camera pose, the 3DGS splatting process enables each Gaussian to be projected onto the camera’s image space and rasterized using depth-order blending to render an output image [42]. Following the same rendering process, our 3DGS can also be rendered into a segment ID feature map $F^{\text{ID}} \in \mathbb{R}^{H \times W \times D_{\text{ID}}}$.

Online 3DGS Construction: To use the 3DGS as a memory in video segmentation, it must be constructed incrementally as the video proceeds (online). While the original 3DGS [42] is constructed entirely offline, our approach for incremental construction is based on the Gaussian initialization and densification logic proposed in SplatAM [13], with added segment ID features for modeling semantic segments. At the first frame of a video, each pixel is back-projected into 3D using the depth image and camera pose, and a new Gaussian is initialized at each of these 3D points. These Gaussians are assigned the corresponding image color, an opacity of 1, and an isotropic scale to match the size of each pixel [13] given by $s = [\frac{D}{f}, \frac{D}{f}, \frac{D}{f}]$ where D is a pixel’s depth and f is the camera focal length. The Gaussians’ segment ID features are initialized based on the output of a segmentation model (e.g. FastSAM or SAM2): each detected segment is assigned

a unique vector from a predefined codebook, which is used as the initial segment ID feature for any Gaussians that project into the segment mask. In subsequent frames, new Gaussians are created for pixels that either (1) are not covered by existing Gaussians after projecting the 3DGS into the image, or (2) have a large depth disparity ($> 0.15\text{m}$) between the observed and rendered 3DGS depth maps.

Segment ID Codebook: Each segment is represented by a real-valued vector chosen from a predefined codebook $C = \{c_1, \dots, c_N\} \subset \mathbb{R}^{D_{\text{ID}}}$, where N is the maximum number of object segments and D_{ID} is the embedding space dimension. We set out to build this codebook such that the IDs are well separated to ensure that distinct object segments are not merged during the 3DGS optimization. We formulate this segment ID codebook generation as an optimization problem and choose a contrastive loss as the objective function. The chosen vectors represent segment ID using the vector direction, and model confidence using vector magnitude. Each vector is randomly initialized then optimized with stochastic gradient descent using the following contrastive objective to maximize inter-vector distance:

$$L = - \min_{1 \leq i \leq N} \left(\min_{1 \leq j \leq N, j \neq i} \|c_i - c_j\| \right) \quad (1)$$

All codebook vectors are normalized to have unit norm.

Since the 3DGS stores both RGB and segment ID features, it can be rendered into both an RGB image, I , as well as a segment ID feature map $F^{\text{ID}} \in \mathbb{R}^{H \times W \times D_{\text{ID}}}$. To convert this feature map into a discrete segment map $M^{\text{ID}} \in \{1, \dots, N\}^{H \times W}$, we compute the inner product of each pixel’s feature vector with all codebook vectors and select the argument of the codebook vector which has the largest inner product and whose similarity exceeds a threshold:

$$m_{x,y} = \arg \max \{d(c_1, F_{x,y}^{\text{ID}}), \dots, d(c_N, F_{x,y}^{\text{ID}})\} \quad (2)$$

$$\text{where } d(c_i, F_{x,y}^{\text{ID}}) = \langle c_i, F_{x,y}^{\text{ID}} \rangle * \mathbf{1}_{\langle c_i, F_{x,y}^{\text{ID}} \rangle > 0.5} \quad (3)$$

In effect, this operation applies the codebook vectors as a 2d-convolutional filter over the input feature map and filters low-confidence or low-similarity pixels.

B. FastSAM-Splat

Inspired by online 3DGS algorithms, we propose an explicit memory mechanism based on the 3DGS representation to enhance segmentation consistency across frames. To this end, we develop FastSAM-Splat, an extension of the FastSAM image segmentation model that uses an explicit 3DGS-based

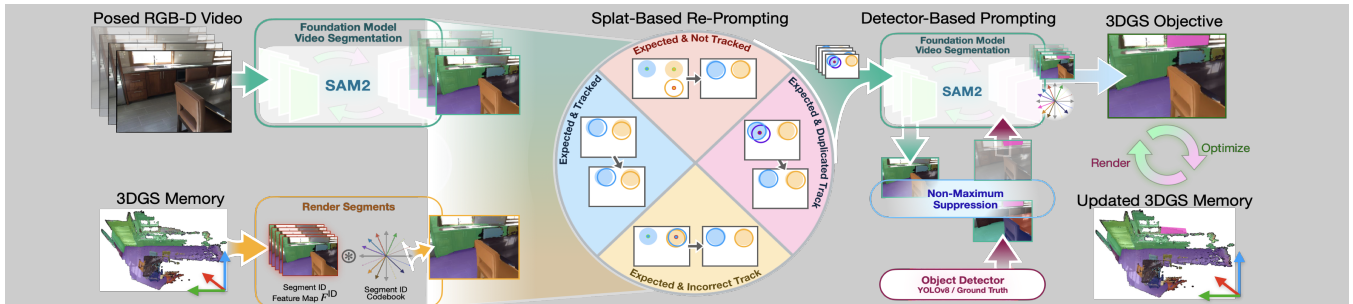


Fig. 3: Illustration of the SAM2-Splat model. Splat-segment is a segment stored and rendered by Gaussian splat. Predicted image segments-output from the FastSAM/SAM2 models.

memory. FastSAM-Splat incrementally builds a 3DGS memory of the scene and embeds predicted segments into its memory. With past segments stored, the 3DGS memory is rendered into a segment feature map representing memories that can be used to condition future predictions.

As illustrated in Fig. 2, FastSAM-Splat first renders the 3DGS memory into a set of 2D segments. These rendered segments are then matched with predicted image segments from FastSAM using the Hungarian algorithm, which solves for a pairwise matching to maximize the sum total F-score across all matched segment pairs. This association process yields three categories of segments:

- Matched segments: A segment from the 3DGS which matches to a predicted image segment from FastSAM.
- Unmatched predicted image segments: A segment predicted by FastSAM with no associated 3DGS segment.
- Unmatched splat segment: A decoded segment from the 3DGS memory that has no corresponding image segment predicted by FastSAM.

Based on this segment association, FastSAM-Splat fuses the 3DGS segments with FastSAM’s predicted image segments to form a final segment map used as its output prediction and as its online objective to update the 3DGS with new segment features. The fusion process is as follows:

- Matched segments inherit the existing codebook vector of their associated 3DGS segment.
- Unmatched predicted image segments are randomly assigned an unused codebook vector.
- Unmatched splat segments retain their feature ID’s direction with linearly reduced magnitude according to:

$$f^{ID'} = \frac{f^{ID}}{\|f^{ID}\|} * (\|f^{ID}\| - C_{conf}) \quad (4)$$

where C_{conf} is a scalar constant used to decrement the 3DGS confidence when segments are not consistently detected by FastSAM ($C_{conf} = 0.1$ in this paper). Once each segment is assigned a feature ID, the binary segment representations are converted into segment feature maps. These feature maps are aggregated into a final fused segment feature map $\hat{F}^{ID} \in \mathbb{R}^{H \times W \times D_{ID}}$. This fused feature map is decoded using the codebook for the model’s final prediction and serves as an optimization objective to update the 3DGS memory. The 3DGS segment features are updated using stochastic gradient descent for $N_{opt} = 20$ steps with the following loss:

$$L = \lambda_{mag} \text{MSE}(\|\hat{F}^{ID}\|, \|F^{ID}\|) + \lambda_{dir} (1 - S_C(\hat{F}^{ID}, F^{ID}))$$

where MSE is mean-squared error, S_C denotes cosine similarity, \hat{F}^{ID} is the fused objective map, and F^{ID} is the rendered 3DGS feature map. Hyperparameters λ_{mag} and λ_{dir} control the relative contribution of magnitude and direction loss terms ($\lambda_{mag} = 50.0$, $\lambda_{dir} = 1.0$ in this paper). This objective ensures that the 3DGS memory is aligned with the current frame’s predictions for use at the next time step.

C. SAM2-Splat

In addition to image-based models like FastSAM, we explore the potential for explicit 3D memory to improve video segmentation models. To this end we extend SAM2 with a 3DGS-based memory mechanism, resulting in the SAM2-Splat model. The core insight of SAM2-Splat is to use the 3DGS memory to identify objects that SAM2 has either failed to track or assigned an incorrect track ID (i.e. ”error accumulation” [37]), and to then re-prompt SAM2 using the 3DGS memory as a correction.

SAM2 requires object prompts—as points, bounding boxes, or masks—in order to begin segmenting and tracking an object. In this work, we use either ground truth object masks or FastSAM detections as a source of mask prompts. On the first frame, all detected masks are used as prompts. On subsequent frames, non-maximum suppression is applied (Sec. IV) to avoid tracking redundant objects. Unlike FastSAM, SAM2 inherently predicts track IDs for each object it is tracking. As illustrated in Fig. 3, SAM2-Splat uses these predicted track IDs to associate SAM2’s predictions with corresponding 3DGS segments, in-place of the Hungarian matching. At each frame, SAM2 predicts a set of tracked segments. SAM2-Splat compares the segments from SAM2 with those rendered from the 3DGS memory as shown in Fig. 4. This comparison sets out to identify:

- 1) Segments that were expected but not tracked by SAM2.
- 2) Segments that were tracked but with an incorrect ID.
- 3) Segments that were predicted as a duplicated track.

For each of these errors, SAM2-Splat then re-prompts SAM2 by sampling a positive and/or negative point-prompt from the relevant region of the image. The re-prompts are

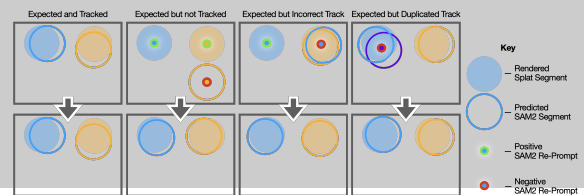


Fig. 4: Illustration of the SAM2-Splat re-prompting strategy.

designed to correct SAM2’s predictions for a given object. For example, in the third column of Fig. 4, SAM2 has assigned an incorrect track ID to its predicted blue segment. This error can be corrected by re-prompting SAM2 with a positive point sampled from the rendered 3DGS memory of the blue object and a negative point sampled from the region that SAM2 incorrectly assigned to the blue ID. After re-prompting, SAM2 is updated with any new object detections and the 3DGS memory is updated using the same process described for FastSAM-Splat. The final segment prediction for SAM2-Splat is the updated segments produced by SAM2 after it is re-prompted and incorporates new detections.

IV. EXPERIMENTS

Datasets: We conduct experiments on both the ScanNet [52] and MVPd [3] datasets. ScanNet includes 1,201 RGB-D videos for training and 312 videos for validation. All videos in ScanNet are collected from real-world environments where a domestic robot would be expected to operate. The 312 validation videos are used for testing in this paper. We follow AnyView [53] and Memory-Based Adapters [54] to downsample the videos from ScanNet into the ScanNet-MV subset. Videos in ScanNet-MV are downsampled to a resolution of 240x320 and at 1/20th the original frame rate.

In addition to ScanNet, we use the MVPd dataset as a large-scale benchmark designed for class-agnostic video segmentation. This dataset provides 16,200 videos for training and 1,741 videos for testing. Each video in MVPd has RGB-D camera data at 480x640 resolution simulated from 180 scenes with ground truth camera pose and depth. Notably, MVPd videos vary between 300 and 600 frames and contain annotations for more than 94 objects per video, on average. Thus MVPd provides a rigorous benchmark covering a very large number of objects per video in a domestic robot setting.

Metrics: For experiments using MVPd, the Video Segmentation Quality (VSQ) metric [3] is used to evaluate all models. VSQ is designed for the class-agnostic video segmentation task and sets out to measure the overlap between predicted and ground truth segment tubes. More specifically, VSQ^k measures the overlap between each ground truth and predicted segment tube of length k frames as follows:

$$VSQ^k = \frac{\sum_{(u, \hat{u}) \in TP} IoU(u, \hat{u})}{|TP| + \frac{1}{2}|FP| + \frac{1}{2}|FN|} \quad (5)$$

where the Hungarian algorithm is used to match each predicted tube to at most one ground truth tube to determine the true positives (TP), false positives (FP) and false negatives (FN). The aggregate VSQ score is the average VSQ^k over a set of window sizes, K . This paper uses $K = \{1, 5, 10, 15\}$ and VSQ^k is calculated using a window stride of 15 frames.

For benchmarking experiments on ScanNet-MV, we use both VSQ and the Segmentation and Tracking Quality (STQ) metric [55]. STQ is computed as the geometric mean between an association quality (AQ) component and a segmentation quality (SQ) component. AQ measures the accuracy and consistency of a model’s track assignment predictions while SQ measures the semantic segmentation accuracy of a models predictions. Importantly, in the class-agnostic setting SQ reduces

to the IoU between all predicted and labeled pixels. While both VSQ and STQ set out to measure segment accuracy and consistency, only VSQ penalizes models that incorrectly predict overlapping segments while only STQ measures segmentation consistency across the entire video.

Implementation Details: FastSAM is trained on a single RTX A6000 GPU using a batch size of 16 images until convergence. All other hyperparameters of FastSAM are left unchanged. For all experiments using SAM2, the SAM2.1 Hiera-B+ model is used. To avoid prompting SAM2 with redundant objects detected by YOLOv8, we use non-maximum suppression which ignores any YOLOv8 detections that have an IoU above 0.1 with any of SAM2’s segments.

For experiments on ScanNet-MV, we crop a 5-pixel border out of each image due to missing RGB values along the image boundary that result from aligning the RGB and depth images. The resulting input resolution of images on ScanNet-MV is 230x310. For experiments on MVPd, the FastSAM-Splat and SAM2-Splat models use resized images at 240x320 resolution to reduce their memory consumption.

A. Real-world Benchmark

This experiment is designed to understand the effectiveness of the proposed models on a real-world dataset in which depth and camera pose represent the distribution a robot will expect in deployment. More specifically, we set out to test two hypotheses: 1) using explicit 3D memory will improve the consistency of both image and video segmentation models, and 2) the improvement will be more pronounced in the image segmentation model—which does not use any form of temporal memory—than the video segmentation model, which uses implicit neural network memories. For this test, we use the ScanNet-MV dataset, FastSAM as the baseline image segmentation model, and two variants of SAM2 as the baseline video segmentation model. One version of SAM2 uses YOLOv8 as a detector with non-maximum suppression (Sec. IV) and another version prompts SAM2 for each object on the frame they become visible using ground truth segments, following the original usage [6].

Results from this experiment are in Tab. I and Fig. 5, showing FastSAM-Splat, which uses a dense 3DGS memory, achieves an improvement of +5.82% VSQ and +10.22% STQ over the FastSAM baseline, which does not use any temporal memories. Moreover, the dense 3D memory used by the SAM2-Splat model leads to an improvement of +2.73% VSQ and +1.58% STQ over SAM2 which uses YOLOv8 detections and +4.00% VSQ and +5.36% STQ over the SAM2 baseline

Method	Backbone	VSQ ^k with k -Frame Window				VSQ	STQ	AQ	SQ
		$k = 1$	$k = 5$	$k = 10$	$k = 15$				
FastSAM	YOLOv8	46.98	39.19	34.28	31.21	37.92	28.17	19.74	40.19
FastSPAM	YOLOv8	50.56	42.78	37.33	34.03	41.18	30.33	21.86	42.09
SAM2	YOLOv8	45.53	41.41	39.28	37.89	41.03	33.43	27.47	40.68
SAM2	GT	59.32	52.68	49.86	47.81	52.42	33.83	35.96	31.82
FastSAM-Splat	YOLOv8	48.07	44.15	41.98	40.75	43.74	38.39	32.83	44.89
SAM2-Splat	YOLOv8	47.70	44.16	42.14	41.02	43.76	35.01	30.68	39.94
SAM2-Splat	GT	62.85	56.43	54.03	52.35	56.42	39.19	41.99	36.58

TABLE I: Comparing FastSAM, FastSPAM, and SAM2 with the proposed FastSAM-Splat and SAM2-Splat models on the ScanNet-MV dataset for class-agnostic video instance segmentation.

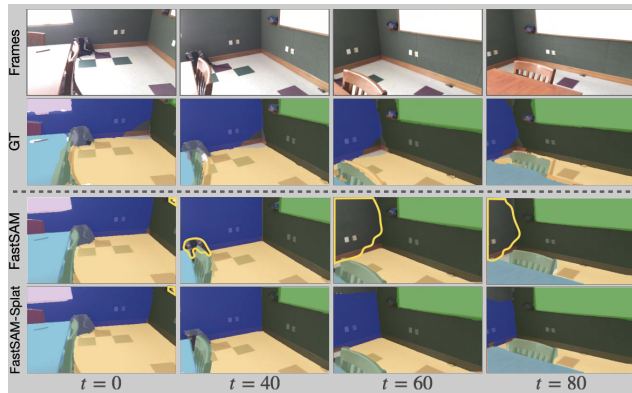


Fig. 5: Qualitative comparison of the FastSAM and FastSAM-Splat models showing FastSAM-Splat has fewer segment inconsistencies. False negative inconsistencies highlighted in yellow outline.

which uses ground truth detections. **These results confirm our first hypothesis that image and video segmentation models, which both lack an explicit memory representation, result in improved segmentation consistency once given access to an explicit 3DGS memory.**

Next, we tested the second hypothesis that including the explicit 3D memory would result in a more substantial improvement for the image segmentation baseline than the video segmentation baseline. This hypothesis stems from our observation that FastSAM uses no memories while SAM2 uses recurrent neural network features as an implicit memory. Comparing the improvement in VSQ and STQ that results from providing the 3DGS memory to FastSAM with the improvement that results from providing the memory to SAM2 shows that FastSAM-Splat led to larger improvements, even when controlling for the source of detections used by SAM2. In the case of YOLOv8 detections, FastSAM-Splat led to a larger improvement of +3.09% VSQ and +8.64% STQ, while in the case of ground truth detections FastSAM-Splat again led to a larger improvement by +1.82% VSQ and +4.86% STQ (Fig. 6). **These results confirm our second hypothesis and suggest that incorporating an explicit 3D memory during the training process for both image and video segmentation models could lead to further improvements in segmentation consistency and accuracy.**

B. Segment ID Feature Ablation

This experiment is designed to study the impact of the segment ID features used by FastSAM-Splat. In particular, we are interested in how the type (i.e. integer or floating point) and dimensionality of the segment ID feature vectors impacts downstream video segmentation accuracy and consistency. Results from this ablation are shown in Tab. II. These results show that using a single integer to represent each segment identity ($D_{ID} = 1$) results in the lowest VSQ

Method	$D_{ID} = 1$	$D_{ID} = 4$	$D_{ID} = 7$	$D_{ID} = 14$	$D_{ID} = 28$
FastSAM-Splat	20.00 / 15.47	42.29 / 36.97	43.08 / 37.93	43.66 / 38.38	43.74 / 38.39

TABLE II: Segment ID feature ablation experiment to evaluate the impact of feature vector dimension on VSQ/STQ measures of video segmentation performance. Vectors of dimension 1 ($D_{ID} = 1$) use a single integer to represent each segment while vectors of increasing dimension ($D_{ID} \in \{4, 7, 15, 28\}$) use real valued vectors.

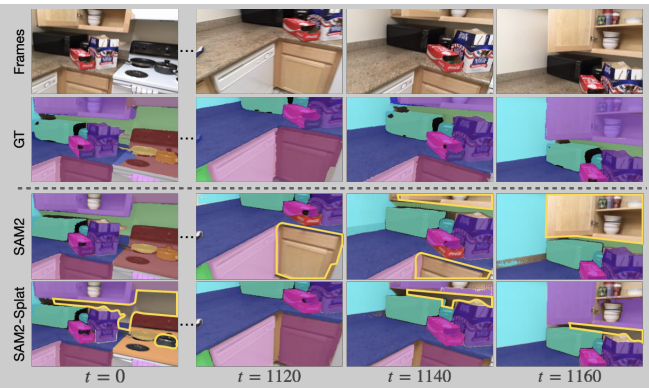


Fig. 6: Comparison of SAM2 and SAM2-Splat on a sequence from ScanNet-MV showing the re-prompting mechanism reduces false negative flickering. False negatives highlighted in yellow outline.

and STQ performance of 20.00% and 15.47% respectively. Moving to a real-valued vector representation ($D_{ID} = 4$) that is optimized using a contrastive loss leads to a substantial improvement of +22.29% VSQ and +21.50% STQ. Further increasing the dimensionality of the segment ID features correlates with improved VSQ and STQ performance, however the performance begins to saturate beyond $D_{ID} = 14$. For example, increasing the dimension from $D_{ID} = 4$ to $D_{ID} = 14$ results in an improvement of +1.37% VSQ and +1.41% STQ while moving from $D_{ID} = 14$ to $D_{ID} = 28$ results in only +0.08% VSQ and +0.01% STQ. **This shows that the optimized codebook representation outperforms a naive integer based representation for segment IDs and that increasing the dimensionality of segment ID features results in improved video segmentation consistency but saturation occurs beyond $D_{ID} = 14$.**

C. SAM2-Splat Re-Prompting Ablation

Next, we performed a set of ablation experiments to validate the re-prompting strategy used by SAM2-Splat and to understand the impact of key re-prompting hyperparameters on video segmentation accuracy consistency. The first hyperparameter studied is the logic used to formulate the re-prompting clicks. The SAM2-Splat model generates re-prompting clicks based on inconsistencies identified between the 3DGS memory and the predictions made by the underlying SAM2 model. These inconsistencies fall into three categories (Sec. III-C): 1) Segments that were expected in the 3DGS memory but not tracked by SAM2 (Not Tracked), 2) Segments that were expected by the 3DGS and tracked by SAM2 but with an incorrect ID (Incorrect Track), and 3) Segments that were expected by the 3DGS memory but predicted as

Method	Not Tracked	Incorrect Track	Duplicated Track	VSQ / STQ
SAM2				41.03 / 33.43
SAM2-Splat	✓			41.09 / 34.19
SAM2-Splat		✓		41.73 / 33.15
SAM2-Splat			✓	43.05 / 33.78
SAM2-Splat	✓	✓	✓	43.76 / 35.01

TABLE III: Ablation experiment to understand the effectiveness of each re-prompt category individually and together within the SAM2-Splat (YOLOv8) model. Models are evaluated on the class-agnostic video instance segmentation task on the ScanNet-MV validation set with VSQ / STQ metrics.

Method	Backbone	1-click	3-click	5-click
SAM2-Splat	YOLOv8	42.95 / 34.03	43.86 / 34.99	43.76 / 35.01
SAM2-Splat	GT	54.89 / 37.29	56.10 / 38.72	56.42 / 39.19

TABLE IV: Ablation experiment on click-based re-prompting of the SAM2-Splat model. Models are evaluated on the ScanNet-MV validation set with VSQ / STQ reported for each setting.

multiple duplicated tracks by SAM2 (Duplicated Track). In this ablation, SAM2-Splat was evaluated on the validation set of ScanNet-MV under varying re-prompting configurations. In each configuration, only a subset of the categories were used to formulate re-prompts and the resulting variants were compared to the default model, which uses all three categories of re-prompts, as well as the baseline SAM2, which uses no re-prompts. The results are included in Tab. III, showing that using only one category of re-prompt results in modest changes to the VSQ and STQ performance of SAM2-Splat when compared to the baseline SAM2 model. Notably, the results show that using all three categories of re-prompts leads to the highest performing SAM2-Splat model in terms of both VSQ and STQ with 43.76% and 35.01% respectively. **These results show that each category of SAM2-Splat’s re-prompts improves SAM2 inconsistencies, and that combining all re-prompting categories leads to the largest improvement in segmentation consistency.**

Another important hyperparameter when re-prompting SAM2 is the number of re-prompts used. To understand how this impacts video segmentation consistency, we evaluated SAM2-Splat on ScanNet-MV using a varying number of re-prompts per object and report the results in Tab. IV. These results show that for both SAM2-Splat models, using more than 1 re-prompt leads to better segmentation consistency as measured by VSQ and STQ. For example, the SAM2-Splat model which uses ground truth detections improved by 1.21/1.43% VSQ/STQ when moving from 1 to 3 re-prompts and by 0.32/0.47% when moving from 3 to 5 re-prompts. **These results indicate that using more re-prompts is generally better than fewer but improvement saturates when more than 3 re-prompts per object are used.**

D. Simulated Benchmark

This experiment sets out to establish the effectiveness FastSAM-Splat on a large-scale video instance segmentation dataset with additional baseline models. To address this question, we evaluated FastSAM-Splat on the MVPd benchmark. Quantitative results are included in Tab. V, showing FastSAM-Splat achieves the highest VSQ score of 56.76% which is +4.86% higher than the next-best model. Due to the high memory-consumption of SAM2 and the large number of objects in MVPd videos (up to 281), we were unable to benchmark SAM2 and SAM2-Splat models on this dataset. **These benchmarking results provide additional evidence that the dense 3D memory of FastSAM-Splat leads to enhanced video segmentation accuracy and consistency.**

E. Efficiency Benchmark

We evaluate the runtime and memory efficiency of each baseline model and report the results in Tab. VI. These

Method	Backbone	VSQ ^k with k -Frame Window				VSQ
		$k = 1$	$k = 5$	$k = 10$	$k = 15$	
Video K-Net	ResNet50	49.65	49.60	38.10	29.56	41.73
Video K-Net	Swin-base	50.83	50.50	38.71	30.05	42.52
Tube-Link	ResNet50	45.13	20.50	16.08	14.05	23.94
Tube-Link	Swin-large	48.17	21.14	16.60	14.55	25.12
OV2Seg	ResNet50	39.96	38.52	37.51	36.71	38.18
OV2Seg	Swin-base	40.22	38.58	37.43	36.52	38.19
FastSAM	YOLOv8	61.18	52.03	44.38	39.09	49.17
FastSPAM	YOLOv8	60.68	54.10	48.48	44.35	51.90
FastSAM-Splat	YOLOv8	60.91	57.68	55.20	53.26	56.76

TABLE V: Evaluating FastSPAM-Splat on the class-agnostic video instance segmentation benchmark of the MVPd dataset. Models are evaluated on the test videos held-out from training.

results show that FastSAM-Splat runs at a 2.84 FPS frame rate, which is comparable to SAM2’s frame rate of 2.83 FPS but slower than each of the alternative baseline models. The SAM2-Splat model was measured to run at 1.46 FPS however we hypothesize this could be improved if the prompting API of SAM2 allowed for prompting multiple objects in-parallel. These results show that improving segmentation consistency with a 3DGS memory is possible but comes with a tradeoff in terms of higher processing latency. One limitation of both FastSAM-Splat and SAM2-Splat is their increased memory consumption—FastSAM-Splat used 28M parameters on average to store the explicit 3DGS memory for videos on the ScanNet-MV benchmark while SAM2-Splat used 26M parameters. **While the memory consumption can be improved with only minor decreases in video segmentation consistency by using feature ID vectors with fewer dimensions(Sec. IV-B), these results suggest that future work to optimize 3DGS storage would be beneficial for video segmentation approaches that seek to use 3DGS as a memory mechanism.**

V. CONCLUSION

This paper provides three core contributions: (1) a FastSAM-Splat model integrating 3DGS memories with FastSAM, (2) a SAM2-Splat model and re-prompting strategy to improve SAM2 video segmentation performance, and (3) empirical benchmarking experiments validating the performance of each model on simulated and real-world benchmarks. The experiments demonstrate that incorporating explicit memories stored in a 3DGS enables an image-based model like FastSAM to outperform state-of-the-art video segmentation models. Furthermore, the explicit memory mechanism is also shown to be useful for improving the consistency of video segmentation models that do not use explicit 3D memories. Based on these results, we envision future work to address remaining limitations of the proposed models, such as its reliance on input pose and depth observations, the lack of global optimization for the 3DGS memory, and implementing 3DGS optimizations for improved runtime.

REFERENCES

- [1] R. B. Rusu, Z. C. Marton, N. Blodow, M. Dolha, and M. Beetz, “Towards 3d point cloud based object maps for household environments,” *Robotics Auton. Syst.*, 2008.

Model	Video K-Net		Tube-Link		OV2Seg		FastSAM	FastSPAM	SAM2	FastSAM-Splat	SAM2-Splat
	ResNet50	Swin-base	ResNet50	Swin-large	ResNet50	Swin-base	YOLOv8	YOLOv8	Hiera-B+	YOLOv8	Hiera-B+
FPS	40.54	26.17	4.88	4.39	15.80	13.28	64.57	6.60	2.83	2.84	1.46
Params (M)	26	88	26	197	26	88	72	72	81	72+28	81+26

TABLE VI: Comparing models' runtime in frames per second (FPS) and complexity in parameter count (M). For FastSAM-Splat and SAM2-Splat, the parameter count is broken into backbone parameter count + 3DGS parameter count.

- [2] K. Lai, L. Bo, X. Ren, and D. Fox, "Detection-based object labeling in 3d scenes," in *ICRA*, 2012.
- [3] A. Opipari, A. K. Krishnan, S. Gayaka, M. Sun, C.-H. Kuo, A. Sen, and O. C. Jenkins, "Configurable embodied data generation for class-agnostic rgb-d video segmentation," in *RA-L*, 2024.
- [4] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollár, and R. Girshick, "Segment anything," *arXiv:2304.02643*, 2023.
- [5] X. Zhao, W. Ding, Y. An, Y. Du, T. Yu, M. Li, M. Tang, and J. Wang, "Fast segment anything," *arXiv:2306.12156*, 2023.
- [6] N. Ravi, V. Gabeur, Y.-T. Hu, R. Hu, C. Ryali, T. Ma, H. Khedr, R. Rädle, C. Rolland, L. Gustafson, E. Mintun, J. Pan, K. V. Alwala, N. Carion, C.-Y. Wu, R. Girshick, P. Dollár, and C. Feichtenhofer, "SAM 2: Segment anything in images and videos," *arXiv:2408.00714*, 2024.
- [7] D. Maggio, Y. Chang, N. Hughes, M. Trang, D. Griffith, C. Dougherty, E. Cristofalo, L. Schmid, and L. Carlone, "Clío: Real-time task-driven open-set 3d scene graphs," *RA-L*, 2024.
- [8] L. Yang and H. Cai, "Unsupervised video object segmentation for enhanced slam-based localization in dynamic construction environments," *Automation in Construction*, 2024.
- [9] J. Cen, Z. Zhou, J. Fang, C. Yang, W. Shen, L. Xie, X. Zhang, and Q. Tian, "Segment anything in 3d with nerfs," in *NeurIPS*, 2023.
- [10] H. Ying, Y. Yin, J. Zhang, F. Wang, T. Yu, R. Huang, and L. Fang, "Omniseg3d: Omniversal 3d segmentation via hierarchical contrastive learning," in *CVPR*, 2024.
- [11] W. Lyu, X. Li, A. Kundu, Y.-H. Tsai, and M.-H. Yang, "Gaga: Group any gaussians via 3d-aware memory bank," *arXiv:2404.07977*, 2024.
- [12] X. Xu, H. Chen, L. Zhao, Z. Wang, J. Zhou, and J. Lu, "Embodiedsam: Online segment any 3d thing in real time," *arXiv:2408.11811*, 2024.
- [13] N. Keetha, J. Karhade, K. M. Jatavallabhula, G. Yang, S. Scherer, D. Ramanan, and J. Luiten, "Splatmap: Splat, track & map 3d gaussians for dense rgb-d slam," in *CVPR*, 2024.
- [14] H. Matsuki, R. Murai, P. H. J. Kelly, and A. J. Davison, "Gaussian Splatting SLAM," in *CVPR*, 2024.
- [15] Y. Li, J. Wang, L. Chu, X. Li, S.-h. Kao, Y.-C. Chen, and Y. Lu, "Streamsg: Online generalizable gaussian splatting reconstruction for unposed image streams," *arXiv:2503.06235*, 2025.
- [16] L. Yang, Y. Fan, and N. Xu, "Video instance segmentation," in *ICCV*, 2019.
- [17] J. Vertens, A. Valada, and W. Burgard, "Smsnet: Semantic motion segmentation using deep convolutional neural networks," in *IROS*, 2017.
- [18] X. Li, W. Zhang, J. Pang, K. Chen, G. Cheng, Y. Tong, and C. C. Loy, "Video k-net: A simple, strong, and unified baseline for video segmentation," in *CVPR*, 2022.
- [19] J. Miao, X. Wang, Y. Wu, W. Li, X. Zhang, Y. Wei, and Y. Yang, "Large-scale video panoptic segmentation in the wild: A benchmark," in *CVPR*, 2022.
- [20] S. Yang, X. Wang, Y. Li, Y. Fang, J. Fang, Liu, X. Zhao, and Y. Shan, "Temporally efficient vision transformer for video instance segmentation," in *CVPR*, 2022.
- [21] Y. Xu, Z. Yang, and Y. Yang, "Video object segmentation in panoptic wild scenes," in *IJCAI-23*, 2023.
- [22] X. Li, H. Yuan, W. Zhang, G. Cheng, J. Pang, and C. C. Loy, "Tube-link: A flexible cross tube framework for universal video segmentation," in *ICCV*, 2023.
- [23] M. Siam, A. Kendall, and M. Jagersand, "Video class agnostic segmentation benchmark for autonomous driving," in *CVPRW*, 2021.
- [24] X. Ren and J. Malik, "Tracking as repeated figure/ground segmentation," in *CVPR*, 2007.
- [25] P. Tokmakov, K. Alahari, and C. Schmid, "Learning motion patterns in videos," in *CVPR*, 2017.
- [26] F. Perazzi, A. Khoreva, R. Benenson, B. Schiele, and A. Sorkine-Hornung, "Learning video object segmentation from static images," in *CVPR*, 2017.
- [27] S. Caelles, K.-K. Maninis, J. Pont-Tuset, L. Leal-Taixé, D. Cremers, and L. Van Gool, "One-shot video object segmentation," in *CVPR*, 2017.
- [28] J. Cheng, Y.-H. Tsai, S. Wang, and M.-H. Yang, "Segflow: Joint learning for video object segmentation and optical flow," in *ICCV*, 2017.
- [29] M. Bucher, T.-H. VU, M. Cord, and P. Pérez, "Zero-shot semantic segmentation," in *NeurIPS*, 2019.
- [30] M. Danielczuk, M. Matl, S. Gupta, A. Li, A. Lee, J. Mahler, and K. Goldberg, "Segmenting unknown 3d objects from real depth images using mask r-cnn trained on synthetic data," in *IEEE ICRA*, 2019.
- [31] Y. Zheng, J. Wu, Y. Qin, F. Zhang, and L. Cui, "Zero-shot instance segmentation," in *CVPR*, 2021.
- [32] Y. Xiang, C. Xie, A. Mousavian, and D. Fox, "Learning rgb-d feature embeddings for unseen object instance segmentation," in *CoRL*, 2021.
- [33] E. P. Örnek, A. K. Krishnan, S. Gayaka, C.-H. Kuo, A. Sen, N. Navab, and F. Tombari, "Supergb-d: Zero-shot instance segmentation in cluttered indoor environments," *RA-L*, 2023.
- [34] X. Wang, X. Zhang, Y. Cao, W. Wang, C. Shen, and T. Huang, "Seggpt: Towards segmenting everything in context," in *ICCV*, 2023.
- [35] Y. Du, Y. Xiao, and V. Lepetit, "Learning to better segment objects from unseen classes with unlabeled videos," in *ICCV*, 2021.
- [36] H. K. Cheng, S. W. Oh, B. Price, A. Schwing, and J.-Y. Lee, "Tracking anything with decoupled video segmentation," in *ICCV*, 2023.
- [37] S. Ding, R. Qian, X. Dong, P. Zhang, Y. Zang, Y. Cao, Y. Guo, D. Lin, and J. Wang, "Sam2long: Enhancing sam 2 for long video segmentation with a training-free memory tree," *arXiv:2410.16268*, 2024.
- [38] H. Wang, C. Yan, S. Wang, X. Jiang, X. Tang, Y. Hu, W. Xie, and E. Gavves, "Towards open-vocabulary video instance segmentation," in *ICCV*, 2023.
- [39] H. Wang, C. Yan, K. Chen, X. Jiang, X. Tang, Y. Hu, G. Kang, W. Xie, and E. Gavves, "Ov-vis: Open-vocabulary video instance segmentation," *IJCV*, 2024.
- [40] X. Li, H. Yuan, W. Li, H. Ding, S. Wu, W. Zhang, Y. Li, K. Chen, and C. C. Loy, "Omg-seg: Is one model good enough for all segmentation?" in *CVPR*, 2024.
- [41] J. Wu, Y. Jiang, Q. Liu, Z. Yuan, X. Bai, and S. Bai, "General object foundation model for images and videos at scale," in *CVPR*, 2024.
- [42] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, "3d gaussian splatting for real-time radiance field rendering," in *TOG*, 2023.
- [43] Q. Shen, X. Yang, and X. Wang, "Flashsplat: 2d to 3d gaussian splatting segmentation solved optimally," in *ECCV*, 2024.
- [44] M. Ye, M. Danelljan, F. Yu, and L. Ke, "Gaussian grouping: Segment and edit anything in 3d scenes," in *ECCV*, 2024.
- [45] B. Dou, T. Zhang, Y. Ma, Z. Wang, and Z. Yuan, "Cosseggaussians: Compact and swift scene segmenting 3d gaussians," *arXiv:2401.05925*, 2024.
- [46] M. Qin, W. Li, J. Zhou, H. Wang, and H. Pfister, "Langsplat: 3d language gaussian splatting," in *CVPR*, 2024.
- [47] Q. Gu, Z. Lv, D. Frost, S. Green, J. Straub, and C. Sweeney, "Egolifter: Open-world 3d segmentation for egocentric perception," in *ECCV*, 2024.
- [48] J. Cen, J. Fang, C. Yang, L. Xie, X. Zhang, W. Shen, and Q. Tian, "Segment any 3d gaussians," in *AAAI*, 2025.
- [49] Y. Siddiqui, L. Porzi, S. R. Bulò, N. Müller, M. Nießner, A. Dai, and P. Kotschieder, "Panoptic lifting for 3d scene understanding with neural fields," in *CVPR*, 2023.
- [50] J. Kerr, C. M. Kim, K. Goldberg, A. Kanazawa, and M. Tancik, "Lerf: Language embedded radiance fields," in *ICCV*, 2023.
- [51] C. M. Kim, M. Wu, J. Kerr, M. Tancik, K. Goldberg, and A. Kanazawa, "Garfield: Group anything with radiance fields," in *CVPR*, 2024.
- [52] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, "ScanNet: Richly-annotated 3d reconstructions of indoor scenes," in *CVPR*, 2017.
- [53] Z. Wu, X. Xu, Z. Wang, C. Xia, L. Zhao, J. Lu, and H. Yan, "Anyview: generalizable indoor 3d object detection with variable frames," *arXiv:2310.05346*, 2023.
- [54] X. Xu, C. Xia, Z. Wang, L. Zhao, Y. Duan, J. Zhou, and J. Lu, "Memory-based adapters for online 3d scene perception," in *CVPR*, 2024.
- [55] M. Weber, J. Xie, M. Collins, Y. Zhu, P. Voigtlaender, H. Adam, B. Green, A. Geiger, B. Leibe, D. Cremers, A. Osep, L. Leal-Taixé, and L.-C. Chen, "Step: Segmenting and tracking every pixel," in *NeurIPS*, 2021.