# LangSplat: 3D Language Gaussian Splatting

Minghan Qin[1,*], Wanhua Li[2,*,†], Jiawei Zhou[1,*], Haoqian Wang[1,†], Hanspeter Pfister[2]
[1]Tsinghua University [2]Harvard University
qmh21@mails.tsinghua.edu.cn, wanhua@seas.harvard.edu, zhoujw22@mails.tsinghua.edu.cn
wanghaoqian@tsinghua.edu.cn, pfister@seas.harvard.edu

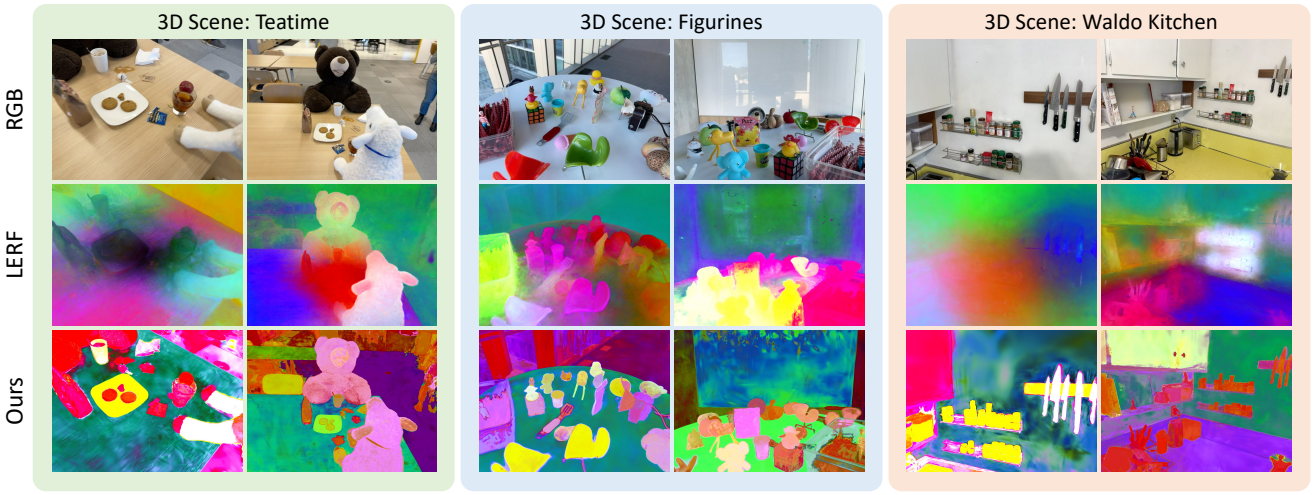


Figure 1. Visualization of learned 3D language features of the previous SOTA method LERF and our LangSplat. While LERF generates imprecise and vague 3D features, our LangSplat accurately captures object boundaries and provides precise 3D language fields without any post-processing. While being effective, our LangSplat is also **199** × faster than LERF at the resolution of 1440 × 1080.

## Abstract

*Human lives in a 3D world and commonly uses natural language to interact with a 3D scene. Modeling a 3D language field to support open-ended language queries in 3D has gained increasing attention recently. This paper introduces LangSplat, which constructs a 3D language field that enables precise and efficient open-vocabulary querying within 3D spaces. Unlike existing methods that ground CLIP language embeddings in a NeRF model, LangSplat advances the field by utilizing a collection of 3D Gaussians, each encoding language features distilled from CLIP, to represent the language field. By employing a tile-based splatting technique for rendering language features, we circumvent the costly rendering process inherent in NeRF. Instead of directly learning CLIP embeddings, LangSplat first trains a scene-wise language autoencoder and then learns language features on the scene-specific latent space, thereby alleviating substantial memory demands imposed by explicit modeling. Existing methods struggle with imprecise and vague 3D language fields, which fail to discern clear boundaries between objects. We delve into this issue and propose to learn hierarchical semantics using SAM, thereby eliminating the need for extensively querying the language field across various scales and the regularization of DINO features. Extensive experiments on open-vocabulary 3D object localization and semantic segmentation demonstrate that LangSplat significantly outperforms the previous state-of-the-art method LERF by a large margin. Notably, LangSplat is extremely efficient, achieving a 199 × speedup compared to LERF at the resolution of 1440 × 1080. We strongly recommend readers to check out our video results at https://langsplat.github.io/.*

## 1. Introduction

Language is the primary means of communication for human beings [3]. Modeling a 3D language field allows users

* Equal contribution. †Corresponding authors.

to interact with and query 3D worlds using open-ended language, which presents a promising avenue for human-computer interaction and understanding [1, 5, 13]. The field of open-ended language queries in 3D has attracted increasing attention due to its various applications such as robotic navigation [14] and manipulation [35], 3D semantic understanding [9, 49] and editing [20], autonomous driving [16], and augmented/virtual reality [23].

Due to the absence of large-scale and diverse 3D scene data with language annotations, the current prevailing approach like LERF [18] involves feature distillation from off-the-shelf vision-language models such as CLIP into a 3D scene. However, these methods [18, 23] suffer from significant limitations in both speed and accuracy, severely constraining their practical applicability. To address these two issues, we revisit two key aspects of 3D language field modeling: the 3D modeling approach that bridges the gap between 2D and 3D, and the rendering target, which determines what to learn for a 3D point. For the 3D modeling technology, most methods utilize neural radiance fields (NeRFs) to represent a 3D scene, where volume rendering techniques are employed to accumulate 3D points along a ray into a single pixel. While NeRF has been widely demonstrated for its powerful 3D representation capabilities [7, 31, 32, 47], the nature of volume rendering leads to its computationally expensive rendering speed [8, 30, 37], which imposes notable constraints on the potential applications within the NeRF-based language field.

Regarding the rendering target, learning a CLIP embedding for a 3D point could be ambiguous as CLIP embeddings are aligned with images rather than pixels. Employing CLIP embeddings from a cropped patch also raises the point ambiguity issue, as the same 3D position can be associated with semantic concepts of varying scales. For instance, a point located on a bear's nose should yield high response values for three distinct textual queries: *"bear's nose"*, *"bear's head"*, and *"bear"* given that this point contributes to all three hierarchical regions. To deal with this issue, current methods [18, 23] introduce an additional absolute scale input to NeRF, trains with patch-wise CLIP features at different scales, and densely renders 2D maps at multiple scales during querying to select the optimal one. However, this scale-based solution compromises both efficiency and effectiveness. It could increase query time by up to 30 times as it needs to render at multiple different scales. Moreover, most patches with varying scales often fail to accurately encompass objects, either frequently including other objects from the background or omitting portions of the target object. These inaccurate CLIP features lead to the trained 3D language field lacking clear boundaries and containing a significant amount of noise. Therefore, they often simultaneously learn pixel-aligned DINO features to mitigate this issue. However, the performance remains unsatisfactory. As shown in Figure 1, LERF still generates imprecise 3D language features.

In this paper, we propose the 3D Language Gaussian Splatting (LangSplat) to address the above issues. Instead of using NeRF to build 3D representations, we resort to 3D Gaussian Splatting, which represents a 3D scene as a collection of 3D Gaussians and uses tile-based splatting to achieve efficient rendering at high resolutions. Our LangSplat defines a set of 3D language Gaussians, with each Gaussian being enhanced by a language embedding. These language-enhanced Gaussians are supervised using CLIP embeddings extracted from image patches captured from multiple training views, ensuring multi-view consistency. As an explicit modeling method, directly storing the high-dimensional language embeddings for each 3D language Gaussian is memory-inefficient. To reduce the memory cost and further improve the rendering efficiency, we propose to first learn a scene-wise language autoencoder, which maps CLIP embeddings in a scene to a low-dimensional latent space. In this way, each language Gaussian only contains the low-dimensional latent language features and the final language embeddings are obtained with decoding of the rendered features. To address the point ambiguity issue, we propose to employ the semantic hierarchy defined by the Segment Anything Model (SAM) [19]. Specifically, for each 2D image, we obtain three well-segmented maps at different semantic levels with SAM. Then we extract the CLIP feature for each mask with precise object boundaries and assign this feature to every point on the corresponding mask. Learning with SAM-based masks not only endows each point with precise CLIP embeddings, resulting in higher model accuracy, but also enables direct querying at predefined three semantic scales. This circumvents the need for intensive searches across multiple absolute scales and the auxiliary DINO features, thereby effectively improving efficiency. We summarize the contributions of this paper as follows:

- We propose the LangSplat, which is the first 3D Gaussian Splatting-based method for 3D language fields. A scene-specific autoencoder is further introduced to alleviate the memory cost issue imposed by explicit modeling.
- We propose to learn the hierarchical semantics defined by SAM to address the point ambiguity issue for 3D language field modeling.
- Experimental results show that our method outperforms the state-of-the-art methods on open-vocabulary 3D object localization and semantic segmentation tasks while being 199 $\times$ faster than LERF at 1440 $\times$ 1080 resolution.

## 2. Related Work

**3D Gaussian Splatting.** Real-time rendering has always been a pursued objective for neural rendering. Recently, Kerbl *et al*. [17] proposed to represent the 3D scene with a set of 3D Gaussians, which attained real-time render-

ing for 1080p resolution while maintaining state-of-the-art visual quality. Encouraged by the success of 3D Gaussian Splatting on novel view synthesis, many works extend it to other tasks to fully exploit the efficient rendering process. To achieve real-time dynamic scene rendering, some studies [26, 40, 44] have extended the 3D Gaussian Splatting technique to dynamic scenes. Luiten *et al*. [26] proposed the Dynamic 3D Gaussians, which extended 3D Gaussians to dynamic scenes by explicitly modeling the 3D Guassians across different time steps. Yang *et al*. [43] presented a deformable 3D Gaussians Splatting method, which learned 3D Gaussians in canonical space and modeled the dynamic scenes with a deformation field. Meanwhile, some researchers have combined 3D Gaussian Splatting with diffusion models to achieve efficient text-to-3D generation [10, 38, 46]. For example, Tang *et al*. [38] introduced DreamGaussian for efficient 3D content generation with a generative 3D Gaussian Splatting model. Unlike these methods, our paper extends each 3D Gaussian with language embeddings for open-vocabulary 3D queries.

**SAM.** The Segment Anything Model [19], released by Meta in 2023, has attracted considerable attention [15, 27, 28]. SAM is trained on over 1 billion masks from 11 million images and has achieved impressive zero-shot performance. It has become the foundational model for image segmentation. It supports flexible prompts including point, box, mask, and text. SAM has been used for many computer vision tasks such as image inpainting [48], super-resolution [25], image matting [45], object tracking [11, 42], medical image segmentation [29], image editing [12], and so on. Many efforts have also been made to utilize SAM in the 3D domain. Liu *et al*. [24] proposed Seal to explore the potential of VFMs including SAM for point cloud segmentation. SA3D [6] generalized SAM to 3D objects by leveraging NeRF to connect 2D images and 3D space. Anything-3D [34] proposed to elevate objects to 3D, where SAM is used to segment the interested object and then a single-view 3D reconstruction pipeline was performed. Different from these works, we use SAM to obtain accurate object masks with three well-defined hierarchical semantics to train a 3D language field.

**3D Langugae Fields.** Some early attempts to construct 3D feature fields included Distilled Feature Fields [20] and Neural Feature Fusion Fields [39]. They learned 3D-consistent features by distilling LSeg [21] or DINO [4] features across multiple views into a NeRF. Shen *et al*. [35] further adopted distilled feature fields for few-shot language-guided robotic manipulation by distilling CLIP feature into a NeRF. There are also some efforts [36, 49] that embed semantic information into NeRFs. For example, Semantic NeRF [49] jointly encoded semantics with appearance and geometry within a NeRF for novel semantic view synthesis. LERF [18] was the first to embed CLIP features into NeRF, enabling open-vocabulary 3D queries leveraging the powerful CLIP representation. DINO features were also used for supervising LERF to improve its performance. Liu *et al*. [23] also utilized CLIP and DINO features to train a NeRF model for 3D open-vocabulary segmentation. While these methods use NeRF for 3D modeling and suffer from the costly rendering process, we propose the 3D language Gaussian Splatting to obtain efficient 3D language fields.

## 3. Proposed Approach

In this section, we first revisit the challenges of modeling 3D language fields and identify the key factors for inaccuracy and inefficiency. Subsequently, we then elaborate on how our proposed LangSplat addresses these issues. Figure 2 depicts the framework of our proposed LangSplat.

### 3.1. Revisiting the Challenges of Language Fields

We denote an input image as $\boldsymbol{I} \in \mathbb{R}^{3\times H\times W}$, where $H$ and $W$ represent the height and weight of the image size. We take a set of calibrated images $\{\boldsymbol{I}_t|t = 1, 2, ...T\}$ as input and train a 3D language field $\Phi$ with these images. Most existing methods [18, 23, 33] employ the CLIP image encoder $V$ to extract image features and utilize the extracted CLIP embeddings to supervise the 3D language field $\Phi$, leveraging the well-aligned text-image latent space provided by CLIP, thus facilitating open-vocabulary queries. However, CLIP embeddings are image-aligned rather than pixel-aligned. In other words, simply computing $V(\boldsymbol{I}_t) \in \mathbb{R}^D$ only obtains an image-level feature, whereas what we need is a pixel-aligned language embedding $\boldsymbol{L}_t \in \mathbb{R}^{D\times H\times W}$, where $D$ represents the CLIP feature dimension. Meanwhile, modeling pixel-aligned language features faces the issue of point ambiguity, as a single point on an object contributes to multiple semantic levels of regions. For instance, a point on a cat's ear simultaneously contributes to the cat's ear, the cat's head, and the entire cat, and should be activated by all three types of textual queries.

To address these issues, most existing methods [18, 23] extract a hierarchy of CLIP features from cropped image patches. Specifically, for a pixel with coordinates $v \in \{1, ..., H\} \times \{1, ..., W\}$, the corresponding CLIP features are obtained from image patches centered around $v$ at different absolute physical scales $s$, with the expectation that at a certain scale $s$, the patch can fully encompass the object. However, this multi-scale approach has two limitations. Firstly, patch features are imprecise because they often include additional contextual object information, leading to overly smoothed language fields with indistinct object boundaries. To alleviate the patchy issue, most methods [18, 23] leverage additional pixel-aligned DINO features to supervise the network. However, the learned 3D language features are still imprecise, as illustrated in Figure 1. Secondly, it requires simultaneous rendering at multiple

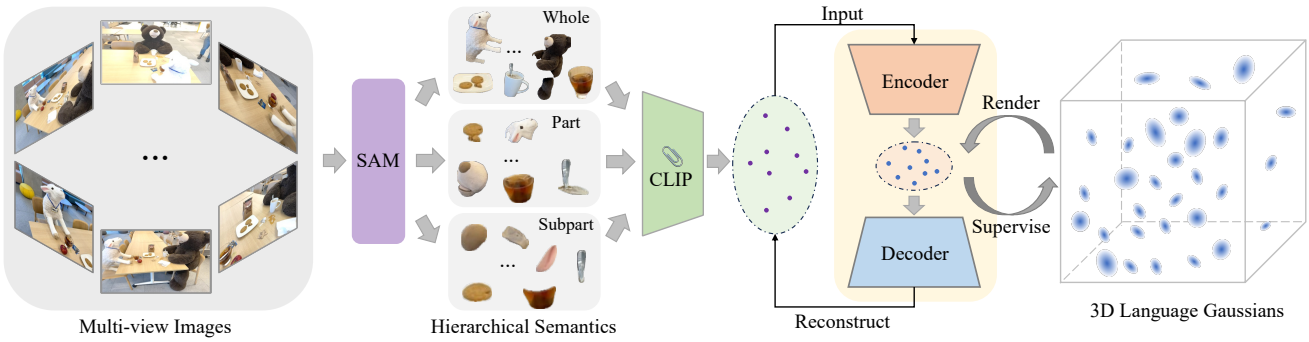


Figure 2. The framework of our LangSplat. Our LangSplat leverages SAM to learn hierarchical semantics to address the point ambiguity issue. Then segment masks are sent to the CLIP image encoder to extract the corresponding CLIP embeddings. We learn an autoencoder with these obtained CLIP embeddings. Our 3D language Gaussian learn language features on the scene-specific latent space to reduce the memory cost. During querying, the rendered language embeddings are sent to the decoder to recover the features on the CLIP space.

scales during inference to find the optimal scale. With the number of scales $s$ potentially reaching as high as 30 [18], this significantly diminishes the inference speed.

Besides the rendering target, another key design space is the 3D modeling approach. Most existing methods [2, 39] employ NeRFs for 3D representation, where they learn a language feature at each 3D point and subsequently render the language feature onto an image, similar to color rendering. However, NeRF-based methods are constrained by their time-consuming rendering process, even though the most advanced NeRF techniques currently available cannot achieve real-time rendering in high-resolution, unrestricted scenes [17]. Meanwhile, there is a high demand for efficient open vocabulary querying in practical applications, especially in fields such as intelligent robotics.

### 3.2. Learning Hierarchical Semantics with SAM

As a foundation model for image segmentation, SAM [19] can accurately group a pixel with its surrounding pixels belonging to the same object, thereby segmenting the image into many object masks with clear boundaries. Furthermore, SAM addresses point ambiguity by generating three different masks for a point prompt, namely, *whole*, *part*, and *subpart*, representing three hierarchical levels of semantics. In this paper, we propose leveraging SAM to obtain precise object masks, which are then used to acquire pixel-aligned features. We also explicitly model the semantic hierarchy defined by SAM to address the point ambiguity issue. With SAM, we can capture the semantic hierarchy of objects in 3D scenes, providing accurate and multi-scale segmentation maps for each input image.

Specifically, we feed a regular grid of $32 \times 32$ point prompts into SAM to obtain the masks under three different semantic levels: $\boldsymbol{M}_0^s, \boldsymbol{M}_0^p, \boldsymbol{M}_0^w$, where $\boldsymbol{M}_0^s, \boldsymbol{M}_0^p$, and $\boldsymbol{M}_0^w$ represent the masks at subpart, part, and whole levels, respectively. Then we remove redundant masks for each of the three mask sets based on the predicted IoU score, stability score, and overlap rate between masks. Each filtered mask set independently performs a comprehensive full-image segmentation based on its respective semantic level, resulting in three segmentation maps: $\boldsymbol{M}^s, \boldsymbol{M}^p, \boldsymbol{M}^w$. These segmentation maps precisely delineate the boundaries of objects at their hierarchical levels, effectively partitioning the scene into semantically meaningful regions. With the obtained segmentation maps, we proceed to extract CLIP features for each segmented region. These features capture the semantic context of the objects at various levels within the scene. Mathematically, the obtained pixel-aligned language embeddings are:

$$\boldsymbol{L}_t^l(v) = V(\boldsymbol{I}_t \odot \boldsymbol{M}^l(v)), l \in \{s, p, w\}, \tag{1}$$

where $\boldsymbol{M}^l(v)$ represents the mask region to which pixel $v$ belongs at the semantic level $l$.

Each pixel rendered from the 3D language scene now possesses a CLIP feature that aligns with its precise semantic context. This alignment reduces ambiguity and enhances the accuracy of language-based queries. We can learn an accurate 3D language field even without the commonly used DINO regularization. Another advantage of our SAM-based approach is the predefined semantic scales. Since we have distinct segmentation maps for "whole," "part," and "subpart" levels, we can directly query the 3D language field at these predefined scales. This eliminates the need for intensive searches across multiple absolute scales, making the querying process more efficient. By incorporating SAM's semantic hierarchy into our approach, we not only improve the accuracy of our 3D language field but also streamline the querying process, making it more efficient and effective for a wide range of applications.

### 3.3. 3D Gaussian Splatting for Language Fields

Having obtained the language embeddings on a set of 2D images $\{\boldsymbol{L}_t^l, |t = 1, ..., T\}$, we can learn a 3D language scene by modeling the relations between 3D points and

2D pixels. Most existing methods [2, 39] suffer from the costly rendering process as they adopt NeRFs for 3D modeling. To address this issue, we present the first 3D Gaussian Splatting-based method for 3D language field modeling.

3D Gaussian Splatting explicitly represents a 3D scene as a collection of anisotropic 3D Gaussians, with each Gaussian $G(x)$ characterized by a mean $\mu \in \mathbb{R}^3$ and a covariance matrix $\Sigma$:

$$G(x) = \exp(-\frac{1}{2}(x-\mu)^\top \Sigma^{-1}(x-\mu)). \tag{2}$$

To optimize the parameters of 3D Gaussians, they are rendered into 2D image planes [50], and a tile-based rasterizer is used to improve the rendering efficiency:

$$C(v) = \sum_{i\in\mathcal{N}} c_i \alpha_i \prod_{j=1}^{i-1}(1-\alpha_j), \tag{3}$$

where $c_i$ is the color of the $i$-th Gaussian, $\mathcal{N}$ denotes the Gaussians in the tile, $C(v)$ is the rendered color at pxiel $v$, and $\alpha_i = o_i G_i^{2D}(v)$. Here $o_i$ is the opacity of the $i$-th Gaussian and $G_i^{2D}(\cdot)$ represents the function of the $i$-th Gaussian projected onto 2D.

In this paper, we proposes the 3D language Gaussian Splatting, which augments each 3D Gaussian with three language embeddings $\{\boldsymbol{f}^s, \boldsymbol{f}^p, \boldsymbol{f}^w\}$. These embeddings are derived from CLIP features, which capture the hierarchical semantics provided by SAM. The augmented Gaussians are named as 3D language Gaussians. We also adopt the tile-based rasterizer to retain the rendering efficiency:

$$\boldsymbol{F}^l(v) = \sum_{i\in\mathcal{N}} \boldsymbol{f}_i^l \alpha_i \prod_{j=1}^{i-1}(1-\alpha_j), l \in \{s,p,w\}, \tag{4}$$

where $\boldsymbol{F}^l(v)$ represents the language embedding rendered at pixel $v$ with the semantic level $l$. By incorporating language information directly into the Gaussians, we enable the 3D language field to respond to language-based queries.

As an explicit modeling approach, our LangSplat may create millions of 3D points to model a complex 3D scene. As CLIP embeddings are high-dimensional features, directly learning $\boldsymbol{f}^l$ on the CLIP latent space significantly increases memory consumption. Compared to learning RGB colors without spherical harmonics coefficients, learning 512-dimensional CLIP features increases the memory requirements for storing 3D Gaussians by over 35 times, easily leading to the "out of memory" issue. To reduce memory consumption and improve efficiency, we introduce a scene-wise language autoencoder. This autoencoder maps CLIP embeddings in a scene to a lower-dimensional latent space, reducing memory requirements. The CLIP model is trained using 400 million (image, text) pairs, and its $D$-dimensional latent space could be highly compact, as it needs to align arbitrary text and images in this space. However, the language field $\Phi$ we train here is scene-specific, meaning we can leverage scene priors to compress CLIP features. In fact, for each input image, we will obtain hundreds of masks segmented by SAM, which is significantly smaller than the number of images used in CLIP training. Therefore, all the segmented regions in a scene are sparsely distributed in the CLIP latent space, allowing us to further compress these CLIP features using a scene-specific autoencoder.

Specifically, we use the collections of CLIP features of SAM segmented masks $\{\boldsymbol{L}_t^l | l \in \{s,p,w\}, 1 \le t \le T\}$ to train a lightweight autoencoder. An encoder $E$ maps the $D$-dimensional CLIP features $\boldsymbol{L}_t^l(v) \in \mathbb{R}^D$ to $\boldsymbol{H}_t^l(v) = E(\boldsymbol{L}_t^l(v)) \in \mathbb{R}^d$, where $d \ll D$. Then we learn a decoder $\Psi$ to reconstruct the original CLIP embeddings from the compressed representation. The autoencoder is trained with a reconstruction objective on the CLIP embeddings $\{\boldsymbol{L}_t^l\}$:

$$\mathcal{L}_{ae} = \sum_{l\in\{s,p,w\}} \sum_{t=1}^{T} d_{ae}(\Psi(E(\boldsymbol{L}_t^l(v))), \boldsymbol{L}_t^l(v)), \tag{5}$$

where $d_{ae}()$ denotes a distance function used for the autoencoder. Here we adopt both $\mathcal{L}_1$ and a cosine distance loss.

After training the autoencoder, we transform all CLIP embeddings $\{\boldsymbol{L}_t^l\}$ into scene-specific latent features $\{\boldsymbol{H}_t^l\}$. We let our 3D language Gaussians learn language embeddings in the scene-specific latent space instead of the CLIP latent space. Therefore, we have $\boldsymbol{f}^l \in \mathbb{R}^d$. In practice, we choose $d = 3$ as it yields excellent model efficiency and accuracy. Compared to directly modeling the $D$-dimensional CLIP embeddings, our method significantly reduced the memory cost by incorporating scene priors. We optimized the language embeddings with the objective:

$$\mathcal{L}_{lang} = \sum_{l\in\{s,p,w\}} \sum_{t=1}^{T} d_{lang}(\boldsymbol{F}_t^l(v), \boldsymbol{H}_t^l(v)), \tag{6}$$

where $d_{lang}()$ denotes the distance function used for our 3D Language Gaussians.

During inference, we follow the Eq. (4) to render the language embeddings from 3D to 2D, and then we use the trained scene-specific decoder $\Psi$ to recover the CLIP image embeddings $\Psi(\boldsymbol{F}_t^l) \in \mathbb{R}^{D\times H\times W}$, which enable open-vocabulary queries with the CLIP text encoder.

By enhancing 3D Gaussians with language embedding and employing a scene-wise language autoencoder, our proposed LangSplat presents a powerful and efficient solution for building 3D language fields. This approach not only preserves the rendering efficiency of Gaussian Splatting but also mitigates the catastrophic memory explosion associated with explicit modeling.

### 3.4. Open-vocabulary Querying

Due to the well-aligned latent space between images and text provided by the CLIP model, our learned 3D language field can easily support open-vocabulary 3D queries, including open-vocabulary 3D object localization and open-vocabulary 3D semantic segmentation. Many existing open-vocabulary 3D semantic segmentation methods [23] usually select the category from a category list, which includes the categories present in the images. However, obtaining a comprehensive category list for in-the-wild scenes is challenging. Different from them, our method generates precise object masks given an arbitrary text query.

Following LERF [18], we compute the relevancy score for each text query. Specifically, for each rendered language embedding $\phi_{img}$ and each text query $\phi_{qry}$, the relevancy score is defined as $\min_i \frac{\exp(\phi_{img}\cdot\phi_{qry})}{\exp(\phi_{img}\cdot\phi_{qry})+\exp(\phi_{img}\cdot\phi^i_{canon})}$, where $\phi^i_{canon}$ is the CLIP embeddings of a predefined canonical phase chosen from *"object"*, *"things"*, *"stuff"*, and *"texture"*. Hence, for each text query, we will obtain three relevancy maps, each representing results at a specific semantic level. We follow the strategy used in LERF [18] and choose the semantic level that yields the highest relevancy score. For the 3D object localization task, we directly choose the point with the highest relevance score. For the 3D semantic segmentation task, we filter out points with relevancy scores lower than a chosen threshold, and predict the object masks with remaining regions. Please refer to the appendix for additional details.

## 4. Experiments

### 4.1. Settings

**Datasets.** We employ two datasets for evaluation. The LERF dataset [18] is captured using the iPhone App Polycam, which consists of complex in-the-wild scenes. The LERF dataset is designed for 3D object localization tasks, here we extend the LERF dataset by annotating ground truth masks for textual queries, enabling the evaluation of the open-vocabulary 3D semantic segmentation on the LERF dataset. As the original LERF annotations for 3D object localization are relatively simple, performance in some scenarios has already approached saturation. Therefore, we further manually annotated additional challenging localization samples to better evaluate method performance. We report localization accuracy for the 3D object localization task following LERF [18], and report the IoU results for the 3D semantic segmentation task. We also employ the 3D-OVS dataset [23], which comprises a collection of long-tail objects captured in diverse poses and backgrounds. This dataset is developed for open-vocabulary 3D semantic segmentation, where a full list of categories is provided. While other methods use the full list to generate the predicted masks, we only use the query category to generate the corresponding masks. The mIoU metric is used for this dataset.

| Test Scene | LSeg [21] | LERF [18] | LangSplat |
|---|---|---|---|
| ramen | 14.1 | 62.0 | **73.2** |
| figurines | 8.9 | 75.0 | **80.4** |
| teatime | 33.9 | 84.8 | **88.1** |
| waldo_kitchen | 27.3 | 72.7 | **95.5** |
| overall | 21.1 | 73.6 | **84.3** |

Table 1. Localization accuracy (%) comparisons on LERF dataset.

| Test Scene | LSeg [21] | LERF [18] | LangSplat |
|---|---|---|---|
| ramen | 7.0 | 28.2 | **51.2** |
| figurines | 7.6 | 38.6 | **44.7** |
| teatime | 21.7 | 45.0 | **65.1** |
| waldo_kitchen | 29.9 | 37.9 | **44.5** |
| overall | 16.6 | 37.4 | **51.4** |

Table 2. Quantitative comparisons of 3D semantic segmentation on the LERF dataset. We report the average IoU scores (%).

**Implementation Details.** To extract the language features of each image, we utilize the OpenCLIP ViT-B/16 model. For SAM, we use the ViT-H model to segment 2D masks. For each scene, we first use 3D Gaussian Splatting to train an RGB scene. We train it for 30,000 iterations, and in the end, each scene comprises around 2,500,000 points. We follow the default parameter setting as in [17] to train the RGB scene. Then we train our 3D language Gaussians by fixing all other parameters of 3D Gaussians such as mean and opacity. Only the language features are learnable during this stage. We train the language features for 30,000 iterations. Our autoencoder is implemented by MLPs, which compress the 512-dimensional CLIP features into 3-dimensional latent features. For a scene with 1080p resolution, our model is trained for 25 minutes on an NVIDIA RTX-3090 GPU and takes roughly 4GB of memory in total.

### 4.2. Results on the LERF dataset

**Quantitative Results.** We first compare our method with other methods on the LERF dataset. Table 1 shows the localization results. We observe that our method achieves an overall accuracy of 84.3%, significantly outperforming LERF. Table 2 further shows the IoU results of 3D semantic segmentation, our method outperforms LERF by 14.0%, which illustrates the superiority of our proposed LangSplat.

**Visualization Results.** To show the learned 3D language field, we visualize the learned features by computing 3-dimensional PCA components of learned language features following [20]. The results are shown in Figure 1. We see that the LERF learned features fail to generate clear boundaries between objects while our method gives pre-

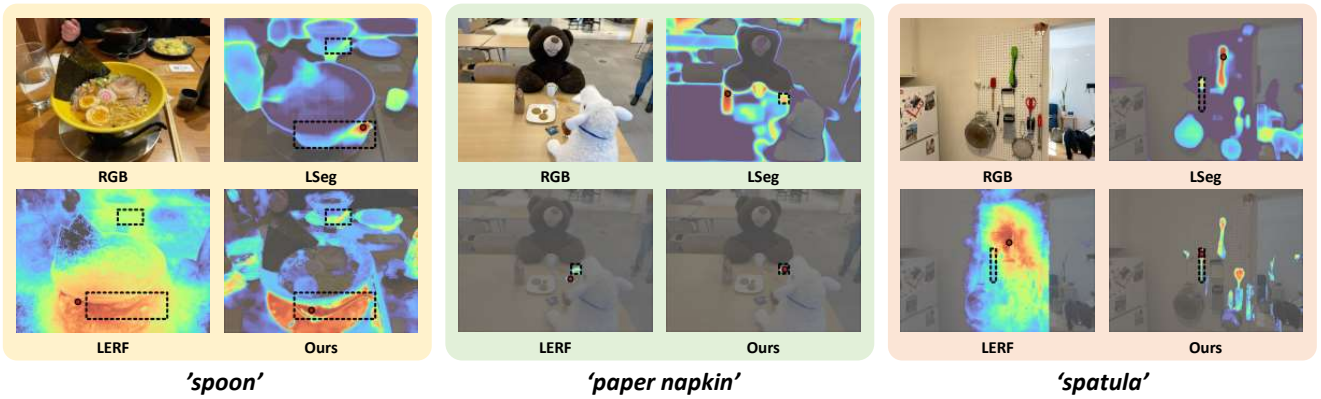


Figure 3. Qualitative comparisons of open-vocabulary 3D object localization on the LERF dataset. The red points are the model predictions and the black dashed bounding boxes denote the annotations.

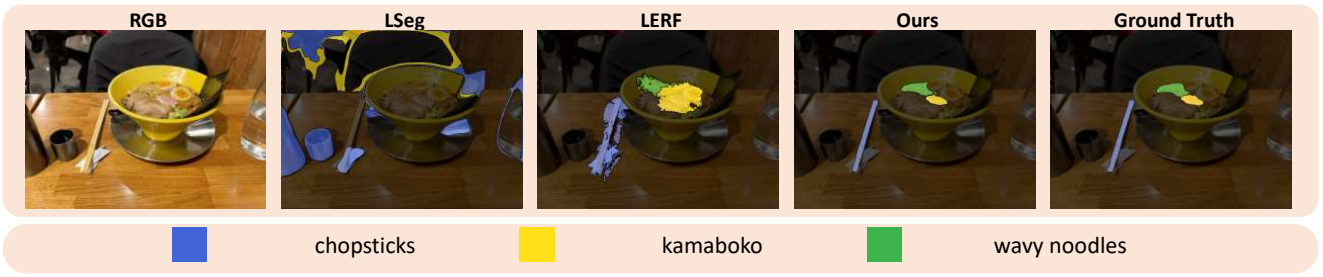


Figure 4. Qualitative comparisons of open-vocabulary 3D semantic segmentation on the LERF dataset.

| Component | | | Performance | |
|---|---|---|---|---|
| AE | 3D-GS | SAM | IoU (%) | Speed (s/q) |
| | | | 28.20 | 30.93 |
| | | ✓ | 46.74 | 7.77 |
| | ✓ | ✓ | OOM | OOM |
| ✓ | ✓ | ✓ | **51.15** | **0.26** |

Table 3. Ablations. The results are obtained on the ramen scene.

| Component | | | Performance | |
|---|---|---|---|---|
| AE | 3D-GS | SAM | mIoU (%) | Speed (s/q) |
| | | | 53.2 | 55.7 |
| | | ✓ | 85.7 | 18.4 |
| | ✓ | ✓ | OOM | OOM |
| ✓ | ✓ | ✓ | **94.2** | **0.28** |

Table 4. Ablations result on the bench scene of the 3D-OVS dataset. The image resolution is $1440 \times 1080$.

cise object shapes solely using CLIP features. We further show the visualization results of object localization and semantic segmentation in Figure 3 and Figure 4, respectively. We observe that the activation regions generated by LERF are more dispersed, while ours are more concentrated, and our activation regions can better align with the ground truth shape compared to those produced by LERF.

**Ablation Study.** We conduct ablations on the ramen scene and report the semantic segmentation results in Table 3. We test the query speed on an NVIDIA RTX-3090 GPU. Here AE represents autoencoder and 3D-GS denotes 3D Gaussian Splatting. Without any of our proposed components, our baseline equals LERF, which has a speed of 30.93 seconds per text query at the resolution of 988 $\times$ 731. Using SAM to replace the scale-based solution significantly increases the IoU by 18.54%, showing our SAM-based solution effectively addresses the point ambiguity issue, leading to accurate 3D language features. Simply replacing NeRF with 3D Gaussian Splatting leads to the out-of-memory issue as explicitly modeling CLIP features poses huge memory demands. Incorporating a scene-specific autoencoder effectively addresses this issue and results in further improvements in both accuracy and efficiency. In the end, our LangSplat achieved a 119 $\times$ speedup over LERF while significantly surpassing LERF in terms of accuracy.

We further conducted the ablations on the 3D-OVS dataset, which has a higher image resolution of $1440 \times 1080$. Table 4 lists the results on the bench scene. We also tested the query speed on an NVIDIA RTX-3090 GPU. We observed that with the increase in image resolution, the speedup over LERF further improved to 199 $\times$, which demonstrates the huge potential of our method. Actu-

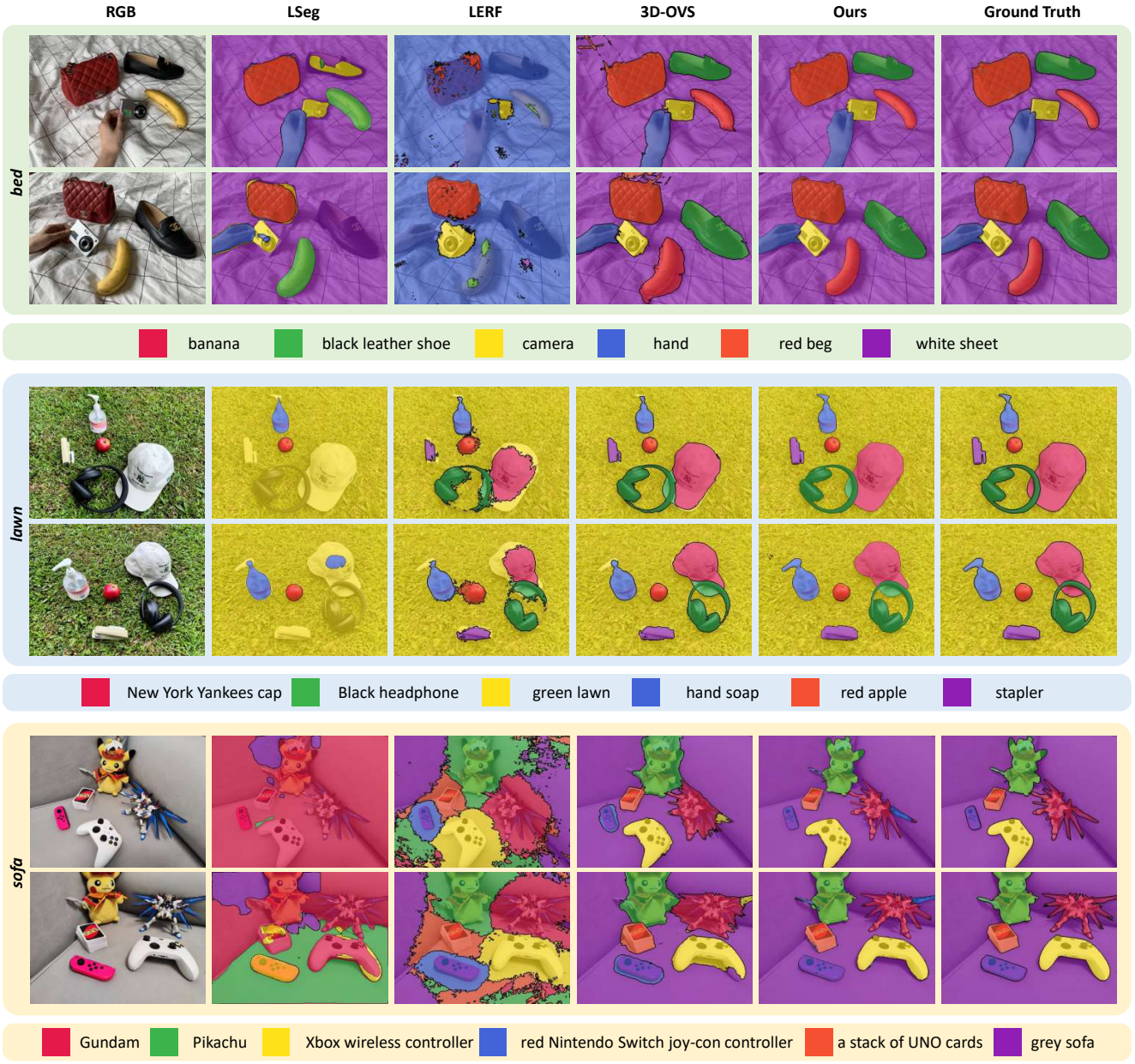


Figure 5. Qualitative comparisons of different methods on the 3D-OVS dataset. We visualize the segmentation results in 3 scenes. We observe that our method gives the most accurate segmentation maps.

ally, the rendering process in our method is highly efficient. Most of the computational time is allocated to the decoder rather than the rendering process. Therefore, as the resolution increases, the computational cost of our method only experiences a slight increase. We believe that a higher speedup can be achieved when dealing with higher-resolution scenes.

### 4.3. Results on the 3D-OVS dataset

**Quantitative Results.** We compare our method with other 2D and 3D state-of-the-art methods on the 3D-OVS dataset in Table 5. We observe that our method not only outperforms 2D-based methods such as ODISE [41] and OV-Seg [22], but also achieves better results than 3D-based methods including LERF [18] and 3D-OVS [23] by a large margin. Note that in this dataset, we generate object masks only based on the query text while others, such as 3D-OVS, require the complete category list. In the end, our method achieves an overall mIoU of 93.4%, which demonstrates that our method effectively learns a precise 3D language field.

**Qualitative Results.** We present the qualitative results in

| Method | *bed* | *bench* | *room* | *sofa* | *lawn* | overall |
|---|---|---|---|---|---|---|
| LSeg [21] | 56.0 | 6.0 | 19.2 | 4.5 | 17.5 | 20.6 |
| ODISE [41] | 52.6 | 24.1 | 52.5 | 48.3 | 39.8 | 43.5 |
| OV-Seg [22] | 79.8 | 88.9 | 71.4 | 66.1 | 81.2 | 77.5 |
| FFD [20] | 56.6 | 6.1 | 25.1 | 3.7 | 42.9 | 26.9 |
| LERF [18] | 73.5 | 53.2 | 46.6 | 27 | 73.7 | 54.8 |
| 3D-OVS [23] | 89.5 | 89.3 | 92.8 | 74 | 88.2 | 86.8 |
| LangSplat | **92.5** | **94.2** | **94.1** | **90.0** | **96.1** | **93.4** |

Table 5. Quantitative comparisons of 3D semantic segmentation on the 3D-OVS dataset. We report the mIoU scores (%).

Figure 5. As LERF suffers from the patchy issue and learns over-smoothed features, it fails to find accurate object boundaries. Among all state-of-the-art methods, our methods give the most accurate segmentation maps, which further demonstrates the effectiveness of our LangSplat.

## 5. Conclusion

In this paper, we have presented LangSplat, a method for constructing 3D language fields that enables precise and efficient open-vocabulary querying within 3D spaces. By extending 3D Gaussian Splatting with language features and learning a scene-specific language autoencoder, LangSplat circumvents slow rendering speed associated with NeRF-based methods. Furthermore, we propose to learn the semantic hierarchy defined by SAM, which effectively resolves the point ambiguity problem, enabling more precise and reliable 3D language fields. The experimental results clearly demonstrate LangSplat's superiority over existing state-of-the-art methods like LERF, particularly in terms of its remarkable 199 $\times$ speed improvement and enhanced performance in open-ended 3D language query tasks.

## References


[1] Daichi Azuma, Taiki Miyanishi, Shuhei Kurita, and Motoaki Kawanabe. Scanqa: 3d question answering for spatial scene understanding. In *CVPR*, pages 19129–19139, 2022. 2

[2] Yash Bhalgat, Iro Laina, João F Henriques, Andrew Zisserman, and Andrea Vedaldi. Contrastive lift: 3d object instance segmentation by slow-fast contrastive fusion. *arXiv preprint arXiv:2306.04633*, 2023. 4, 5

[3] Nancy Bonvillain. *Language, culture, and communication: The meaning of messages*. Rowman & Littlefield, 2019. 1

[4] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *ICCV*, pages 9650–9660, 2021. 3

[5] Paola Cascante-Bonilla, Hui Wu, Letao Wang, Rogerio S Feris, and Vicente Ordonez. Simvqa: Exploring simulated environments for visual question answering. In *CVPR*, pages 5056–5066, 2022. 2

[6] Jiazhong Cen, Zanwei Zhou, Jiemin Fang, Chen Yang, Wei Shen, Lingxi Xie, Xiaopeng Zhang, and Qi Tian. Segment anything in 3d with nerfs. 2023. 3

[7] Eric R Chan, Marco Monteiro, Petr Kellnhofer, Jiajun Wu, and Gordon Wetzstein. pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis. In *CVPR*, pages 5799–5809, 2021. 2

[8] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *ECCVn*, pages 333–350. Springer, 2022. 2

[9] Boyuan Chen, Fei Xia, Brian Ichter, Kanishka Rao, Keerthana Gopalakrishnan, Michael S Ryoo, Austin Stone, and Daniel Kappler. Open-vocabulary queryable scene representations for real world planning. In *ICRA*, pages 11509–11522. IEEE, 2023. 2

[10] Zilong Chen, Feng Wang, and Huaping Liu. Text-to-3d using gaussian splatting. *arXiv preprint arXiv:2309.16585*, 2023. 3

[11] Yangming Cheng, Liulei Li, Yuanyou Xu, Xiaodi Li, Zongxin Yang, Wenguan Wang, and Yi Yang. Segment and track anything. *arXiv preprint arXiv:2305.06558*, 2023. 3

[12] Shanghua Gao, Zhijie Lin, Xingyu Xie, Pan Zhou, Ming-Ming Cheng, and Shuicheng Yan. Editanything: Empowering unparalleled flexibility in image editing and generation. In *ACM MM, Demo track*, 2023. 3

[13] Daniel Gordon, Aniruddha Kembhavi, Mohammad Rastegari, Joseph Redmon, Dieter Fox, and Ali Farhadi. Iqa: Visual question answering in interactive environments. In *CVPR*, pages 4089–4098, 2018. 2

[14] Chenguang Huang, Oier Mees, Andy Zeng, and Wolfram Burgard. Visual language maps for robot navigation. In *ICRA*, pages 10608–10615. IEEE, 2023. 2

[15] Siyuan Huang, Zhengkai Jiang, Hao Dong, Yu Qiao, Peng Gao, and Hongsheng Li. Instruct2act: Mapping multi-modality instructions to robotic actions with large language model. *arXiv preprint arXiv:2305.11176*, 2023. 3

[16] Krishna Murthy Jatavallabhula, Alihusein Kuwajerwala, Qiao Gu, Mohd Omama, Tao Chen, Shuang Li, Ganesh Iyer, Soroush Saryazdi, Nikhil Keetha, Ayush Tewari, et al. Conceptfusion: Open-set multimodal 3d mapping. *arXiv preprint arXiv:2302.07241*, 2023. 2

[17] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *TOG*, 42(4):1–14, 2023. 2, 4, 6

[18] Justin Kerr, Chung Min Kim, Ken Goldberg, Angjoo Kanazawa, and Matthew Tancik. Lerf: Language embedded radiance fields. In *ICCV*, pages 19729–19739, 2023. 2, 3, 4, 6, 8, 9, 1

[19] 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 *ICCV*, 2023. 2, 3, 4

[20] Sosuke Kobayashi, Eiichi Matsumoto, and Vincent Sitzmann. Decomposing nerf for editing via feature field distillation. *NeurIPS*, 35:23311–23330, 2022. 2, 3, 6, 9, 1

[21] Boyi Li, Kilian Q Weinberger, Serge Belongie, Vladlen Koltun, and Rene Ranftl. Language-driven semantic segmentation. In *ICLR*, 2022. 3, 6, 9, 1

[22] Feng Liang, Bichen Wu, Xiaoliang Dai, Kunpeng Li, Yinan Zhao, Hang Zhang, Peizhao Zhang, Peter Vajda, and Diana Marculescu. Open-vocabulary semantic segmentation with mask-adapted clip. In *CVPR*, pages 7061–7070, 2023. 8, 9, 1

[23] Kunhao Liu, Fangneng Zhan, Jiahui Zhang, Muyu Xu, Yingchen Yu, Abdulmotaleb El Saddik, Christian Theobalt, Eric Xing, and Shijian Lu. Weakly supervised 3d open-vocabulary segmentation. In *NeurIPS*, 2023. 2, 3, 6, 8, 9, 1

[24] Youquan Liu, Lingdong Kong, Jun Cen, Runnan Chen, Wenwei Zhang, Liang Pan, Kai Chen, and Ziwei Liu. Segment any point cloud sequences by distilling vision foundation models. *arXiv preprint arXiv:2306.09347*, 2023. 3

[25] Zhihe Lu, Zeyu Xiao, Jiawang Bai, Zhiwei Xiong, and Xinchao Wang. Can sam boost video super-resolution? *arXiv preprint arXiv:2305.06524*, 2023. 3

[26] Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. *arXiv preprint arXiv:2308.09713*, 2023. 3

[27] Jun Ma and Bo Wang. Segment anything in medical images. *arXiv preprint arXiv:2304.12306*, 2023. 3

[28] Gengchen Mai, Weiming Huang, Jin Sun, Suhang Song, Deepak Mishra, Ninghao Liu, Song Gao, Tianming Liu, Gao Cong, Yingjie Hu, et al. On the opportunities and challenges of foundation models for geospatial artificial intelligence. *arXiv preprint arXiv:2304.06798*, 2023. 3

[29] Maciej A Mazurowski, Haoyu Dong, Hanxue Gu, Jichen Yang, Nicholas Konz, and Yixin Zhang. Segment anything model for medical image analysis: an experimental study. *Medical Image Analysis*, 89:102918, 2023. 3

[30] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *TOG*, 41(4):1–15, 2022. 2

[31] Keunhong Park, Utkarsh Sinha, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Steven M Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable neural radiance fields. In *ICCV*, pages 5865–5874, 2021. 2

[32] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. In *CVPR*, pages 10318–10327, 2021. 2

[33] Nur Muhammad Mahi Shafiullah, Chris Paxton, Lerrel Pinto, Soumith Chintala, and Arthur Szlam. Clip-fields: Weakly supervised semantic fields for robotic memory. *arXiv preprint arXiv:2210.05663*, 2022. 3

[34] Qiuhong Shen, Xingyi Yang, and Xinchao Wang. Anything-3d: Towards single-view anything reconstruction in the wild. *arXiv preprint arXiv:2304.10261*, 2023. 3

[35] William Shen, Ge Yang, Alan Yu, Jansen Wong, Leslie Pack Kaelbling, and Phillip Isola. Distilled feature fields enable few-shot language-guided manipulation. *arXiv preprint arXiv:2308.07931*, 2023. 2, 3

[36] Yawar Siddiqui, Lorenzo Porzi, Samuel Rota Bulò, Norman Müller, Matthias Nießner, Angela Dai, and Peter Kontschieder. Panoptic lifting for 3d scene understanding with neural fields. In *CVPR*, pages 9043–9052, 2023. 3

[37] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *CVPR*, pages 5459–5469, 2022. 2

[38] Jiaxiang Tang, Jiawei Ren, Hang Zhou, Ziwei Liu, and Gang Zeng. Dreamgaussian: Generative gaussian splatting for efficient 3d content creation. *arXiv preprint arXiv:2309.16653*, 2023. 3

[39] Vadim Tschernezki, Iro Laina, Diane Larlus, and Andrea Vedaldi. Neural feature fusion fields: 3d distillation of self-supervised 2d image representations. In *3DV*, pages 443–453. IEEE, 2022. 3, 4, 5

[40] Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 4d gaussian splatting for real-time dynamic scene rendering. *arXiv preprint arXiv:2310.08528*, 2023. 3

[41] Jiarui Xu, Sifei Liu, Arash Vahdat, Wonmin Byeon, Xiaolong Wang, and Shalini De Mello. Open-vocabulary panoptic segmentation with text-to-image diffusion models. In *CVPR*, pages 2955–2966, 2023. 8, 9, 1

[42] Jinyu Yang, Mingqi Gao, Zhe Li, Shang Gao, Fangjing Wang, and Feng Zheng. Track anything: Segment anything meets videos. *arXiv preprint arXiv:2304.11968*, 2023. 3

[43] Ziyi Yang, Xinyu Gao, Wen Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. Deformable 3d gaussians for high-fidelity monocular dynamic scene reconstruction. *arXiv preprint arXiv:2309.13101*, 2023. 3

[44] Zeyu Yang, Hongye Yang, Zijie Pan, Xiatian Zhu, and Li Zhang. Real-time photorealistic dynamic scene representation and rendering with 4d gaussian splatting. *arXiv preprint arXiv:2310.10642*, 2023. 3

[45] Jingfeng Yao, Xinggang Wang, Lang Ye, and Wenyu Liu. Matte anything: Interactive natural image matting with segment anything models. *arXiv preprint arXiv:2306.04121*, 2023. 3

[46] Taoran Yi, Jiemin Fang, Guanjun Wu, Lingxi Xie, Xiaopeng Zhang, Wenyu Liu, Qi Tian, and Xinggang Wang. Gaussiandreamer: Fast generation from text to 3d gaussian splatting with point cloud priors. *arXiv preprint arXiv:2310.08529*, 2023. 3

[47] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images. In *CVPR*, pages 4578–4587, 2021. 2

[48] Tao Yu, Runseng Feng, Ruoyu Feng, Jinming Liu, Xin Jin, Wenjun Zeng, and Zhibo Chen. Inpaint anything: Segment anything meets image inpainting. *arXiv preprint arXiv:2304.06790*, 2023. 3

[49] Shuaifeng Zhi, Tristan Laidlow, Stefan Leutenegger, and Andrew J Davison. In-place scene labelling and understanding with implicit scene representation. In *ICCV*, pages 15838–15847, 2021. 2, 3

[50] Matthias Zwicker, Hanspeter Pfister, Jeroen Van Baar, and Markus Gross. Ewa volume splatting. In *VIS*, pages 29–538. IEEE, 2001. 5

## A. Video Demo

In Figure 1 of our main paper, we have visualized the language features learned by LERF and our method. For a fair comparison, we perform PCA on the decoded feature $\Psi(\boldsymbol{F}_t^l) \in \mathbb{R}^{D\times H\times W}$ for our method. However, one benefit of our method is that we are able to directly visualize the learned language features in the encoded 3-dimensional latent space, which can ensure color consistency between frames.[1] Specifically, we normalize the encoded 3-dimensional latent features $\boldsymbol{H}_t^l(v) \in \mathbb{R}^{3\times H\times W}$ and visualize them by treating the 3-dimensional features as RGB channels.

We strongly recommend readers refer to our video demo to observe the learned 3D language fields in the scene-specific latent space. The video demonstrates that our method has acquired a 3D language representation that is both 3D-consistent and distinctly shaped, which significantly distinguishes it from existing methods that often only learn 3D language representations with blurred boundaries. Meanwhile, our approach achieves a speedup of 119 $\times$ compared to LERF at a resolution of $988 \times 731$ and further improves to 199 $\times$ faster at a resolution of $1440 \times 1080$.

## B. More Implementation Details

For each text query, we can obtain three relevancy maps with our trained 3D language Gaussians, each representing one semantic level defined by SAM. Then we use different strategies to choose the best semantic level and obtain the predictions for different tasks.

**3D Object Localization on LERF.** To mitigate the impact of outliers, we first employ a mean convolution filter with a size of 20 to smooth the values of three relevancy maps. For the smoothed relevancy maps, we select the one with the highest smoothed relevancy score and take the corresponding position as the final prediction.

**3D Semantic Segmentation on LERF.** Similarly, to mitigate the influence of outliers, we apply a mean filter with a size of 20 to smooth the three relevancy maps. Subsequently, we select the relevancy map with the maximum smoothed relevancy score for binary mask prediction. For the selected relevancy map, we first normalize its relevancy scores and then use a threshold to obtain a binary image as the final prediction mask.

**3D Semantic Segmentation on 3D-OVS.** For each class query, we obtain three relevancy maps. We apply a threshold of 0.4 to these relevancy maps, setting relevancy scores below 0.4 to 0 and relevancy scores above 0.4 to 1, resulting in three binary maps. We calculate the average relevancy scores within the mask region for each relevancy map and select the relevancy map with the highest average response as the final predicted binary map.

[1] The consistency of color in PCA visualizations across different frames is not ensured.

| Method | *bed* | *bench* | *room* | *sofa* | *lawn* | overall |
|---|---|---|---|---|---|---|
| LSeg [21] | 87.6 | 42.7 | 46.1 | 16.5 | 77.5 | 54.1 |
| ODISE [41] | 86.5 | 39.0 | 59.7 | 35.4 | 82.5 | 60.6 |
| OV-Seg [22] | 40.4 | 89.2 | 49.1 | 69.6 | 92.1 | 68.1 |
| FFD [20] | 86.9 | 42.8 | 51.4 | 9.5 | 82.6 | 54.6 |
| LERF [18] | 86.9 | 79.7 | 79.8 | 43.8 | 93.5 | 76.7 |
| 3D-OVS [23] | 96.7 | 96.3 | 98.9 | 91.6 | 97.3 | 96.2 |
| LangSplat | **99.2** | **98.6** | **99.3** | **97.9** | **99.4** | **98.9** |

Table 6. Quantitative comparisons of 3D semantic segmentation on the 3D-OVS dataset. We report the accuracy scores (%).

| $d$ | 1 | 2 | 3 | 8 |
|---|---|---|---|---|
| mIoU (%) | 6.46 | 91.93 | 94.19 | 95.20 |
| Speed (s/q) | 0.2770 | 0.2779 | 0.2788 | 0.2807 |

Table 7. The ablations of latent dimension $d$ for our scene-specific autoencoder. The results are obtained on the bench scene of the 3D-OVS dataset. The image resolution is $1440 \times 1080$.

## C. More Quantitative Results

In addition to the mIoU metric, the Accuracy metric is also employed on the 3D-OVS dataset in [23]. [2] Therefore, we also compare our method with other state-of-the-art methods on the 3D-OVS dataset using the Accuracy metric. The results are shown in Table 6. We observe that our method consistently outperforms other methods, which further illustrates the superiority of our method.

## D. More Ablation Study

To reduce the memory cost of our 3D language Gaussians, we proposed the scene-specific autoencoder to learn a latent feature. We show the ablation results of different latent dimensions $d$ on the bench scene of the 3D-OVS dataset in Table 7. We observed that as $d$ increases, the mIoU performance improves, with only a slight increase in the time cost. We chose $d = 3$ because it allows us to directly visualize the learned 3D language field in the latent space by treating the 3-dimensional features as the RGB channels. We also strongly encourage readers to refer to our video demo to observe how our learned language field accurately captures the precise 3D shape of objects in the scene-specific latent space.

[2] After carefully reviewing the codes and results, we discovered that the mAP results reported in [23] are, in fact, the Accuracy results.

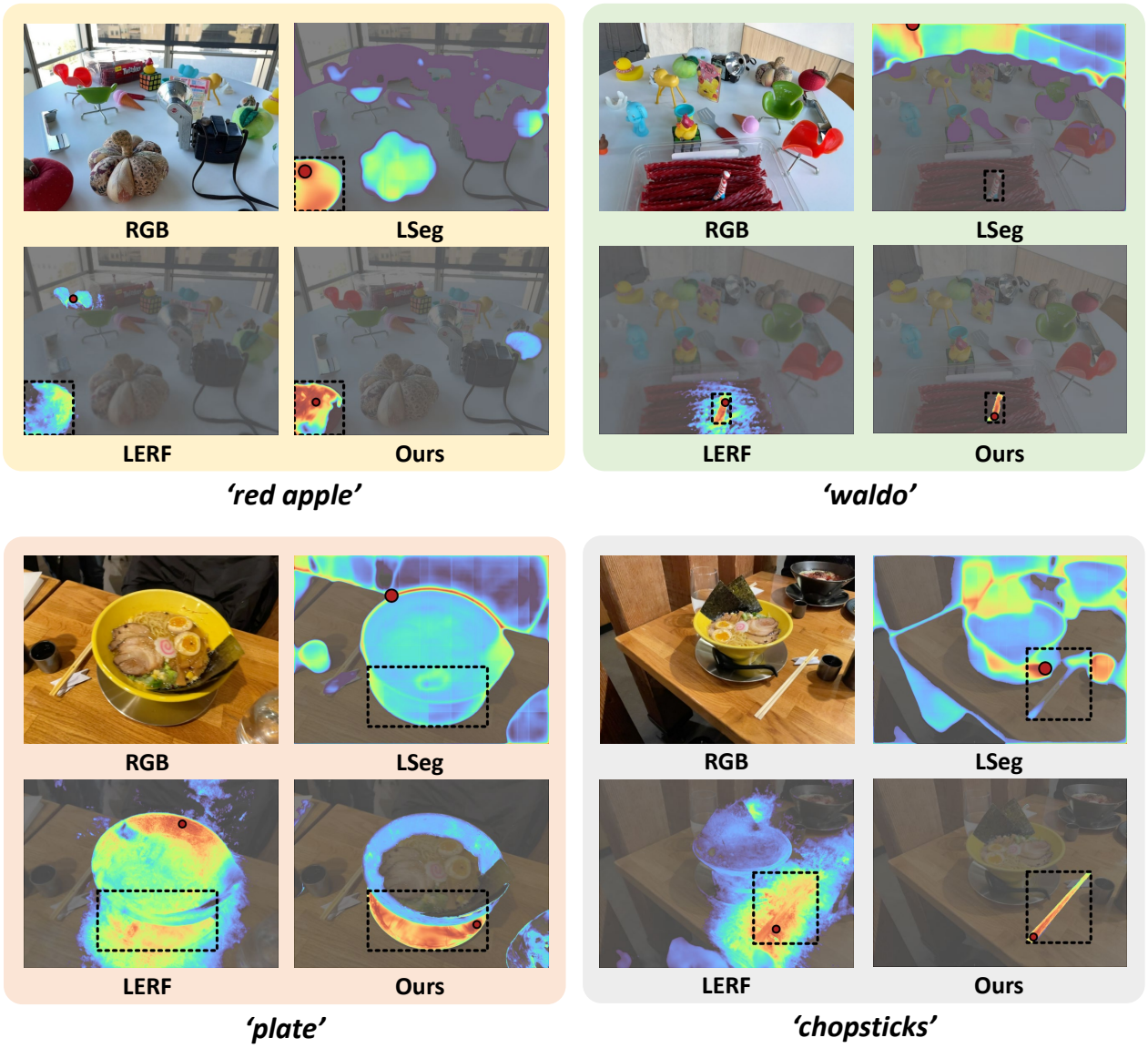


Figure 6. More qualitative comparisons of open-vocabulary 3D object localization on the LERF dataset. The red points are the model predictions and the black dashed bounding boxes denote the annotations.

## E. More Visualization Results

**3D Object Localization on LERF.** We visualize more examples on the LERF dataset for open-vocabulary 3D object localization in Figure 6. We found that for text queries such as "red apple" and "plate", LERF failed to correctly locate the 3D positions, whereas our method succeeded. For text queries like "waldo" and "chopsticks", although LERF could identify the correct location, its activation values were more dispersed, whereas our method was able to focus more precisely on the queried object.

**3D Semantic Segmentation on LERF.** We demonstrate more examples on the LERF dataset for open-vocabulary 3D semantic segmentation in Figure 7. We observed that the results produced by LERF were unable to provide the precise shape of the queried object and exhibited a significant amount of noise, whereas our method could accurately depict the object's shape. These results show the effectiveness of our proposed LangSplat.

**3D Semantic Segmentation on 3D-OVS.** We show more scenes on the 3D-OVS dataset for open-vocabulary 3D semantic segmentation in Figures 8, 9, 10, and 11, respectively. Compared to the previous state-of-the-art method 3D-OVS, our approach provides more precise object bound-

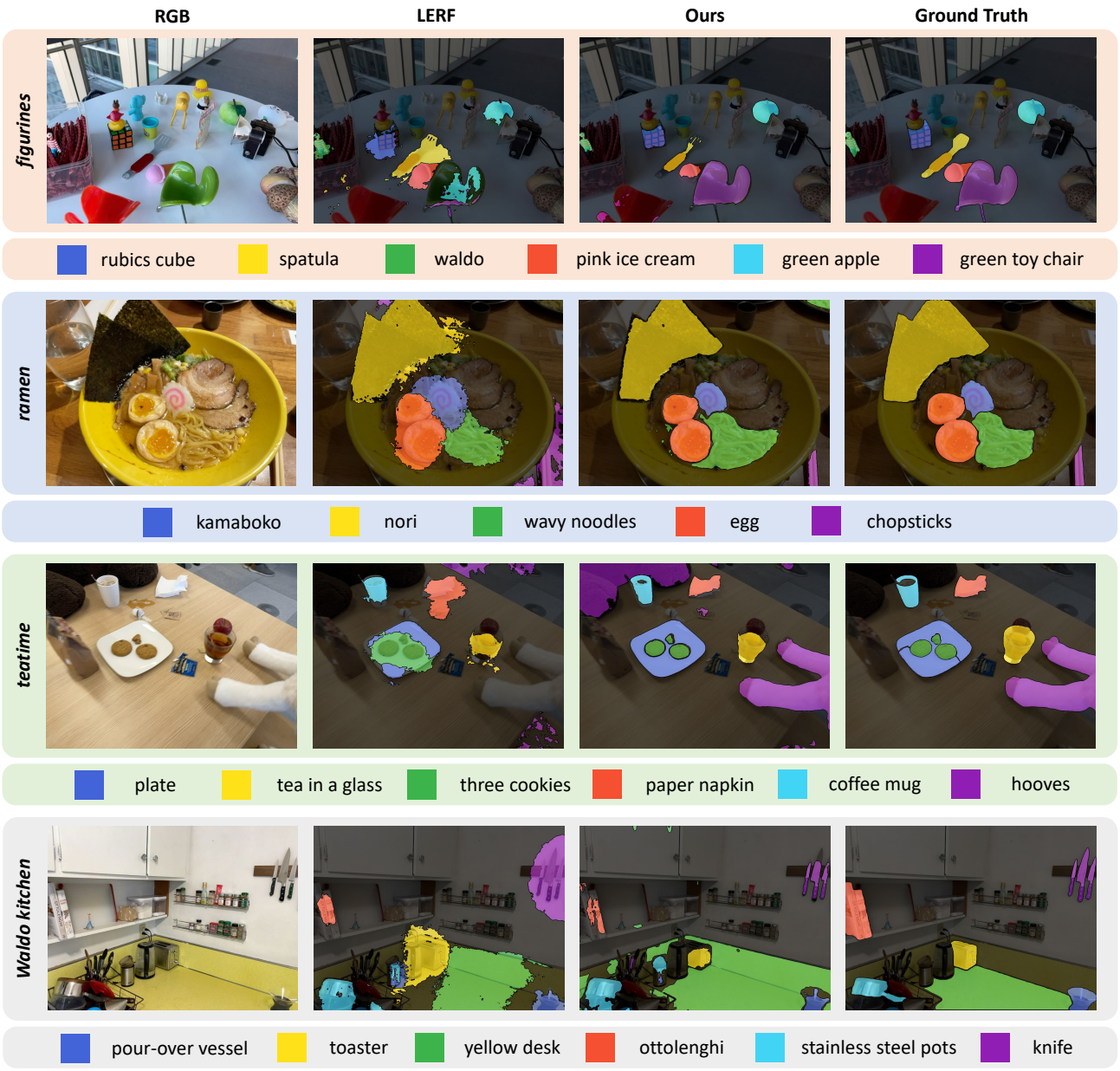


Figure 7. More qualitative comparisons of open-vocabulary 3D semantic segmentation on the LERF dataset.

aries and exhibits reduced noise, which illustrates that our LangSplat learns a more accurate 3D language field.

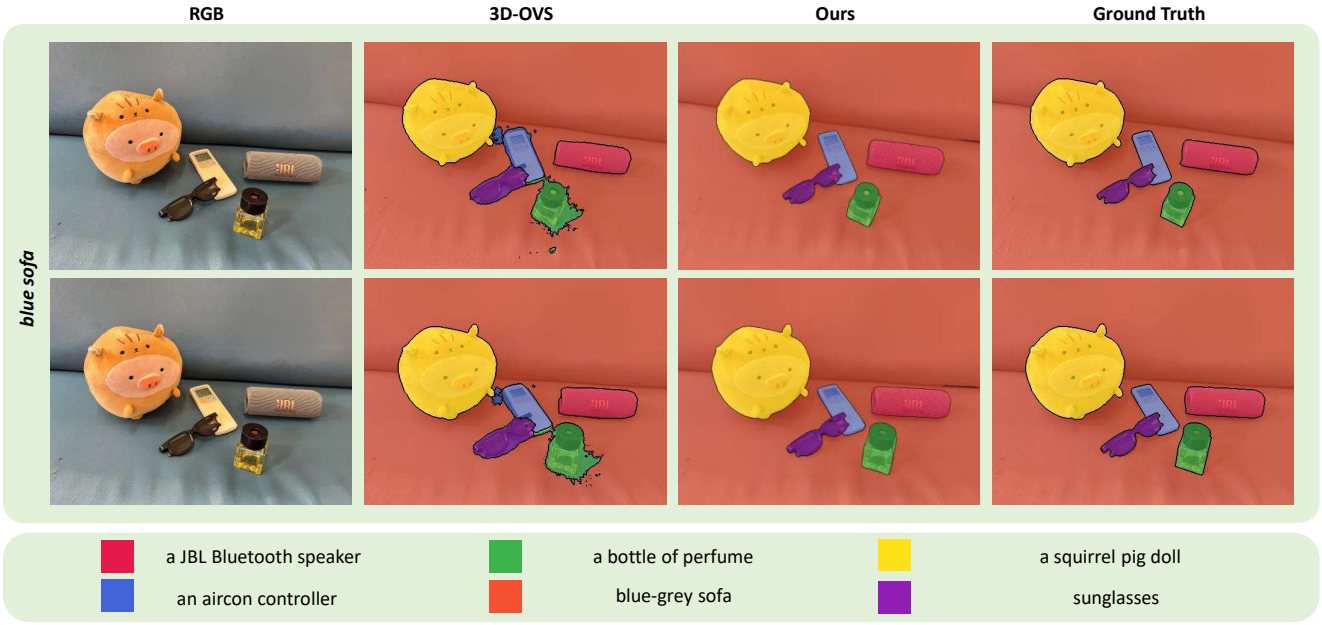


Figure 8. Qualitative comparisons on the blue sofa scene of the 3D-OVS dataset.

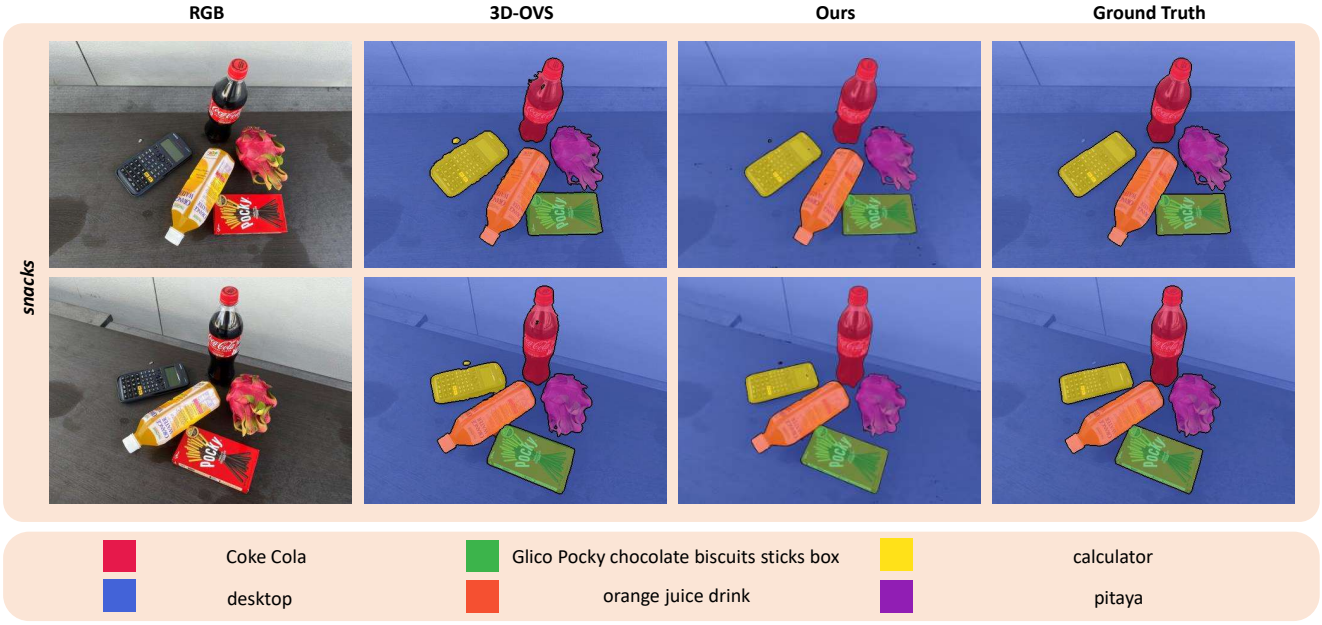


Figure 9. Qualitative comparisons on the snacks scene of the 3D-OVS dataset.

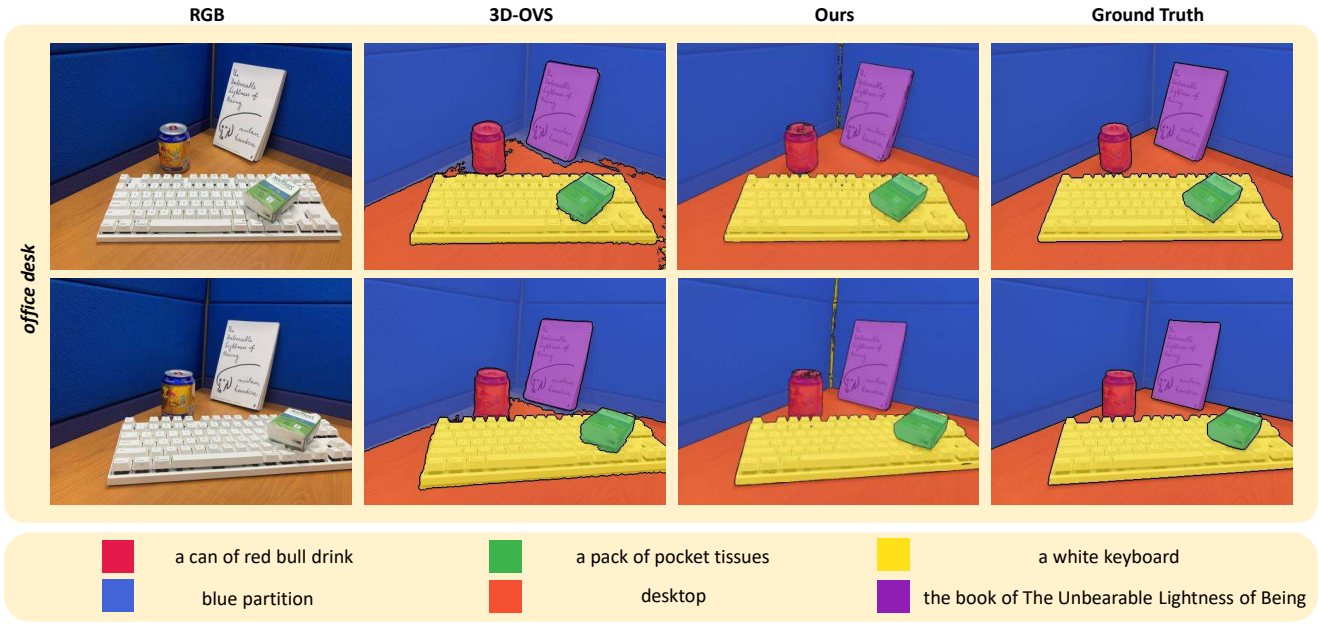


Figure 10. Qualitative comparisons on the office desk scene of the 3D-OVS dataset.

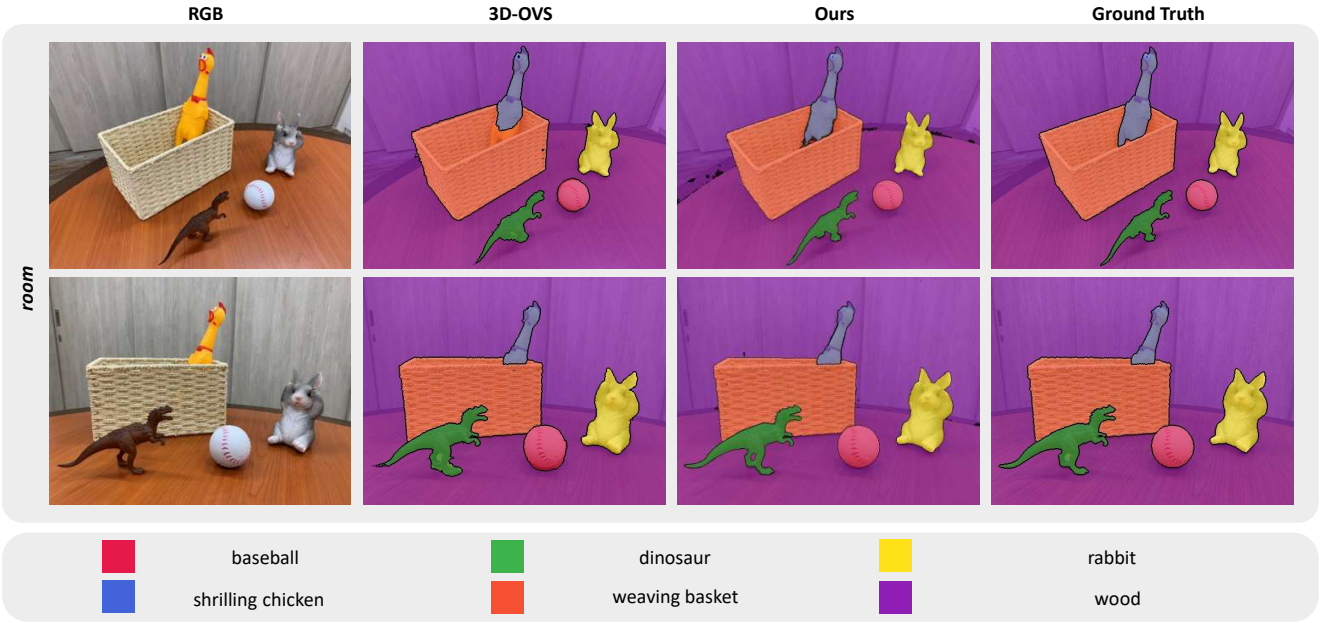


Figure 11. Qualitative comparisons on the room scene of the 3D-OVS dataset..