



UvA-DARE (Digital Academic Repository)

Visual understanding of dynamic scenes using object relationships and open vocabularies

Ülger, O.

Publication date
2026

[Link to publication](#)

Citation for published version (APA):

Ülger, O. (2026). *Visual understanding of dynamic scenes using object relationships and open vocabularies*. [Thesis, fully internal, Universiteit van Amsterdam].

General rights

It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations

If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: <https://uba.uva.nl/en/contact>, or a letter to: Library of the University of Amsterdam, Secretariat, P.O. Box 19185, 1000 GD Amsterdam, The Netherlands. You will be contacted as soon as possible.

Auto-Vocabulary Segmentation for LiDAR Points

In Chapter 4, we introduced Auto-Vocabulary Semantic Segmentation for 2D images, where relevant class names are autonomously generated from the image alone. While this approach demonstrated the potential of removing the dependence on human-specified vocabularies, it relies entirely on image appearance. In challenging conditions such as poor lighting, dense rain or fog, this dependence can limit the robustness of vocabulary generation and segmentation, since image features of objects become unreliable and ambiguous.

To address this shortcoming, this chapter explores how geometric information from LiDAR can complement or even substitute visual cues. Unlike images, LiDAR captures the underlying 3D structure of a scene and remains reliable in settings where appearance cues are degraded. Building on the principles of Auto-Vocabulary Segmentation, we propose 3D-Auto-Vocabulary Segmentation (3D-AVS), a framework that autonomously identifies relevant target classes directly from LiDAR data and uses them for 3D semantic segmentation. By leveraging geometry, 3D-AVS extends open-ended segmentation to scenarios where image-based methods fall short, making vocabulary generation more robust and broadly applicable.

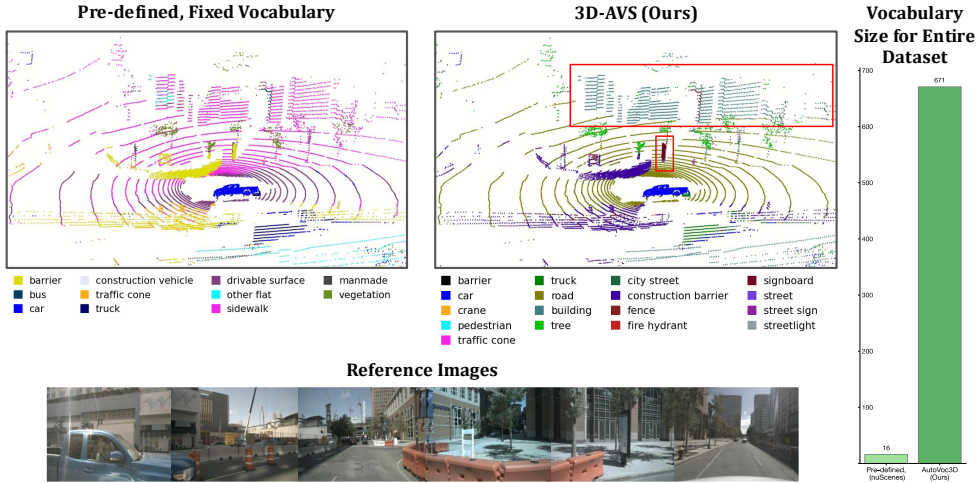


Figure 5.1: Pre-defined Vocabulary vs. Auto-Vocabulary. 3D-AVS automatically generates a vocabulary for which it predicts segmentation masks, offering greater semantic precision. Our predictions identify specific classes *e.g.*, *building* and *signboard* (highlighted in red boxes), which are annotated with ambiguous terms like *manmade*. Quantitatively, 3D-AVS recognizes 671 unique categories on the validation set of nuScenes [167], significantly surpassing nuScenes’s original 16 categories. Left and middle sections of the figure indicate a single scene, while the right plot concerns the entire dataset.

5.1 Introduction

Existing perception methods [159, 160, 161, 162, 163, 164] for autonomous driving often rely on an inclusiveness assumption that all potential categories of interest must exist in the training dataset. Nevertheless, public datasets often annotate instances with pre-defined categories, which can vary from three (*e.g.*, vehicle, cyclist and pedestrian) [165, 166] to several dozen types [167, 168], and fail to annotate rare objects with correct semantic labels. Failing to recognize atypical objects or road users poses a significant risk to the perception model’s adaptability to diverse real-life scenarios.

The development of Vision-Language Models (VLMs) strengthens the connection between vision and language modalities and promotes progress in multi-modal tasks, such as zero-shot image classification [22], image search and retrieval [169], image captioning [145], video understanding [170], and open-vocabulary learning [171]. Open-vocabulary learning methods often utilize pre-trained VLMs to find the correspondence between visual entities and a semantic vocabulary, thereby creating the potential to detect any category of interest [171, 172]. However, these methods rely on human-specified queries, and thus can not dynamically recognize all se-

semantic entities in a scene. Conversely, predefining everything is neither scalable nor practical in a dynamic world, as it is impossible to anticipate all the categories the model may encounter in advance. This shortcoming severely limits the real-life applicability of existing methods, as newly encountered object categories could still be unknown to the model or unaware to humans.

In this work, we propose 3D-AVS, a framework that automatically recognizes objects, generates a vocabulary for them and segments LiDAR points. We evaluate our method on indoor and outdoor datasets [167, 173, 174] and introduce a metric, TPSS, to assess the model performance based on semantic consistency in CLIP [22] space. Figure 5.1 compares the same segmenter, namely OpenScene [175] with different vocabularies. 3D-AVS generates convincing semantic classes as well as accurate point-wise segmentations. Moreover, when pre-defined categories are general and ambiguous, *e.g.*, *man-made*, 3D-AVS recognizes the semantically more precise categories, *e.g.*, *building* and *signboard*.

Our contributions can be summarized as follows: **1)** we introduce auto-vocabulary segmentation for point clouds, aiming to label all points using a rich and scene-specific vocabulary. Unlike methods that rely on predefined vocabularies, we address an *unknown* vocabulary setting by dynamically generating vocabulary per input; **2)** we propose 3D-AVS, a framework that automatically identifies objects, either through an image-free point-based captioner or an off-the-shelf image-based captioner; **3)** we propose a point captioner for 3D-AVS-LiDAR that decodes text from point-based CLIP features, achieving image independence and enhanced object diversity through a sparse masked attention pooling (SMAP) module; and **4)** we introduce the Text-Point Semantic Similarity score, a novel CLIP-based, annotation-free metric that evaluates semantic consistency, accounting for synonyms, hierarchies, and similarity in unknown vocabularies, enabling scalable auto-vocabulary evaluation without human input.

5.2 Related Work

Open-Vocabulary Segmentation (OVS). OVS aims to perform segmentation based on a list of arbitrary text queries. CLIP [22] achieves this in 2D by aligning vision and language in a shared latent space. However, no comparable large-scale point cloud dataset exists for similar training in 3D. Additionally, captions in point cloud datasets are typically much sparser. Therefore, existing methods usually freeze the text encoder and image encoder, and align point features to vision-language feature space [176, 175, 177, 178, 179]. ULIP [176] distills vision-language knowledge into a point encoder via contrastive learning on text-image-point triplets. CLIP2Scene [177] adopts self-supervised learning, aligning point-text features using spatial-temporal cues. OpenScene [175] supervises the point encoder with CLIP-based image features through point-pixel

projection. While these OVS approaches show promising results, they require user-defined categories as prompts. Conversely, our approach automatically generates categories that potentially appear in the scene without any human in the loop.

Auto-Vocabulary Segmentation (AVS). AVS differs from OVS in that it segments entities directly from perceptual data rather than relying on a human-defined vocabulary as input. Relevant target categories are directly inferred from the image - usually without any additional training, finetuning, data sourcing or annotation effort. Zero-Guidance Segmentation (ZeroSeg) [143] achieve this by using clustered DINO [144] embeddings to obtain binary object masks. These masks were used to guide the attention of CLIP, resulting in embeddings that are more accurately targeted to individual segments, and a trained large language model was tasked to output texts closest to said embeddings. While this required switching between three different latent representations, AutoSeg [180] proposed a more direct approach based on BLIP [23] embeddings only. They introduced a procedure in which multi-scale BLIP embeddings are enhanced through clustering, alignment and denoising. The embeddings are then captioned using BLIP’s decoder and parsed into a noun set used by an OVS model for segmentation. CaSED [181] retrieves captions from an external database and then integrates parsed texts with different segmentation methods. Despite these attempts in 2D domain, AVS in 3D domain remains unexplored. Concurrently and independently, Meng *et al.* [182] have proposed vocabulary-free 3D instance segmentation and a method PoVo for this task. While PoVo first obtains 3D clusters and then matches the generated semantic categories to the clusters, our work focuses more on target category generation and seamless integration with existing OVS methods.

Challenges of AVS Evaluation. AVS presents additional challenges linked to evaluation. Since generated categories can be open-ended and outside of the fixed dataset vocabulary, one needs to bridge the gap between the two to assess the segmentation performance. ZeroSeg [143] exploits subjective assessment. In AutoSeg [180], the LLM-based mapper, LAVE, is introduced to address this challenge. However, the mapping targets are typically limited in size, causing the auto-generated categories - often more semantically rich and precise - to be discarded. To overcome these limitations, we propose the TPSS metric, which enables the evaluation of the generated categories while preserving their open-ended nature.

Captioning 2D and 3D Data. Captioning is the process of generating a concise and meaningful description from data modalities such as images, videos or point clouds. Notable works in 2D combine image-based templates with extracted attributes [183, 184], combine deep learning models like convolutional neural networks with RNN, LSTM or transformer-based generators [185, 186, 187], or leverage pre-trained vision-language embeddings such as CLIP or BLIP [145, 146, 147, 23]. BLIP [23], known for its effective but somewhat generic captions, often focus only

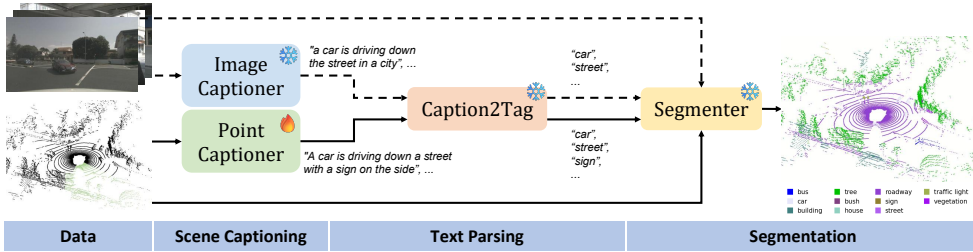


Figure 5.2: Overview of 3D-AVS. A point cloud and corresponding images are fed to respective point captioner and image captioner to generate captions. Then, Caption2Tag excludes irrelevant words in the captions. The remaining nouns are passed to a text encoder and eventually assigned to points through a segmenter. The dashed lines indicate that the entire images branch is optional. The point captioner is the only trainable component in 3D-AVS. Note that, the example point caption is generated based on observing the green points.

5

on the 2-3 most prominent entities in an image. BBoost [180] addresses this limitation by enhancing BLIP tokens through unsupervised semantic clustering in the latent space, enabling cluster-wise captioning and resulting in more comprehensive and detailed captions. More recently, xGen-MM (BLIP-3) [188] was introduced, building on BLIP with two improvements: an expanded and more diverse set of training data, and a scalable vision token sampler for flexible input resolutions. While this task is broadly explored in the 2D domain, it is yet to be solved in the 3D domain. Existing approaches focus on describing a single object, *e.g.*, CAD models [176, 189] and scanned shapes [190, 189, 191, 192, 193], or dense contextual indoor scenarios [194, 195, 196, 197, 198, 199, 193], but fail to caption sparse outdoor scenes due to sparsity and lack of colour information. LidarCLIP [200] encodes a sparse point cloud to a CLIP feature vector and then decodes it to a caption via ClipCap [145]. However, LidarCLIP only provides a global caption per scene, leading to limited coverage of semantic entities. Instead, our proposed point captioner copes with flexible receptive fields and offers a controllable number of captions with various granularity.

5.3 Method

5.3.1 Preliminaries

CLIP and CLIP-aligned Encoder. CLIP [22] is believed to properly align visual and text features due to its superior performance on vision-language tasks. It comprises a text encoder h_{tx} and an image encoder h_{im} , both of which map a data modality, *e.g.*, text and image, to a vision-language latent space, also known as the CLIP space. Many works [125, 128, 126, 121, 122] increase the

output resolution of CLIP image encoder, yielding high-resolution features $h_{\text{im}}^{\text{hr}}$, while preserving alignment within the original CLIP space. Furthermore, some 3D methods [176, 175, 179, 200] distill features from h_{im} or its high-resolution variant $h_{\text{im}}^{\text{hr}}$ into 3D backbones, yielding CLIP-aligned 3D encoder h_{pt} . In this work, we leverage such aligned 3D encoders and bypass the time-consuming training process whenever possible.

Problem Definition. Given a point cloud $\mathbf{P} = \{p_n\}_{n=1}^N \in \mathbb{R}^{N \times 3}$ with N points, the aim is to assign a semantic class label $l \in \mathbb{S}$ to every point, where \mathbb{S} indicates a vast semantic space. Different to closed-set or open-vocabulary segmentation for which the vocabulary is *known* either via a user-specified prompt or by pre-defined labels from dataset, the class set in auto-vocabulary segmentation is *unknown* and automatically generated for each input scene.

5.3.2 3D Auto-Vocabulary Segmentation

This section introduces 3D-AVS for which an overview of its major components is shown in Figure 5.2. Given a point cloud and a set of corresponding images, 3D-AVS first utilizes a point captioner and an image captioner to describe points and images in detail. The generated captions are parsed in the Caption2Tag module, resulting in a list of tags indicating semantic entities. Eventually, each point is assigned a semantic tag, forming segmentation results. These key components are elaborated in the following paragraphs.

Scene Captioning. A key step of our approach is the auto-generation of a vocabulary for the given scene, which is performed by a scene captioner that is either based on input images or on the input point cloud. Image captioning is a well-explored task with a variety of accessible multi-modality large-language models (MLLMs) [188, 23, 201]. We adopt xGen-MM [188] as the image captioner because of its architectural flexibility and enhanced semantic coverage, Given a set of K images $\mathbf{I} \in \mathbb{R}^{K \times H \times W \times 3}$ capturing a scene, and an instruction prompt (details in supplementary material), the image captioner generates a list of captions

$$\mathbf{D} = \{d_{\text{im}}^{(k)} \in \mathbb{R}^{w_k} \mid k = 1, \dots, K\}. \quad (5.1)$$

where w_k is the number of words in the caption for the k -th image. To ensure a diverse enough set of coherent captions, we opt for beam search in the generation process. Implementation details are in the supplementary material. Following caption generation, each caption is parsed and validated with Caption2Tag, as described in the section below.

LiDAR point cloud captioning remains an underexplored area in existing research despite the potential of such captions for applications. While images collected alongside LiDAR point clouds can be used to generate a target vocabulary, relying solely on images proves inadequate under

challenging conditions such as low light or adverse weather, where visual data becomes unreliable. To address this, we introduce a novel Point Captioner trained via transfer learning, which provides captions directly from color-independent LiDAR data. Our approach, detailed in Section 5.3.3, takes a point cloud \mathbf{P} as input and outputs captions \mathbf{d}_{pt} . Unlike image captioning, which requires extensive contextual information and sophisticated vision models to produce detailed captions, the Point Captioner provides robust descriptions by relying solely on geometric features. This color independence is particularly beneficial in low-visibility environments, such as nighttime scenes where image-based captioning often falls short. Combining both modalities ultimately yields the best results, uniting the diversity of image captions with the resilience of point-based captions.

Text Parsing. Captions generated by the image and point captioner are scene-specific sentences in natural language which we then parse into individual object nouns for semantic segmentation. To this end, we filter the sentence on (compound) nouns (*i.e.*, general entities) and proper nouns (*i.e.*, named entities) using spaCy [158] and transform them to their singular form through lemmatization. Lastly, we verify each category against the WordNet dictionary, resulting in a set of M scene-specific tags, denoted as $\mathbf{L} = \{l_m\}_{m=1}^M$.

Segmentation. The proposed pipeline separates the vocabulary generation and segmentation, enabling seamless integration with an open-vocabulary point segmenter. The segmenter consists of three encoders, namely a text encoder $h_{\text{tx}} : \mathbb{R}^1 \rightarrow \mathbb{R}^C$, a high-resolution image encoder $h_{\text{im}}^{\text{hr}} : \mathbb{R}^{H \times W \times 3} \rightarrow \mathbb{R}^{H \times W \times C}$ and a point encoder $h_{\text{pt}} : \mathbb{R}^{N \times 3} \rightarrow \mathbb{R}^{N \times C}$, that are pre-aligned with the CLIP vision-language latent space. Following the inference procedure of CLIP [22], namely similarity-based label assignment, we first compute the embeddings as follows:

$$\mathbf{E}_{\text{tx}} = \{e_m\}_{m=1}^M \leftarrow h_{\text{tx}}(\mathbf{L}) \quad (5.2)$$

$$\mathbf{F}_{\text{im}} = \{f_k\}_{k=1}^K \leftarrow h_{\text{im}}(\mathbf{I}) \quad (5.3)$$

$$\mathbf{F}_{\text{pt}} = \{f_n\}_{n=1}^N \leftarrow h_{\text{pt}}(\mathbf{P}) \quad (5.4)$$

where \mathbf{E}_{tx} , \mathbf{F}_{im} and \mathbf{F}_{pt} indicate text embeddings, image features, and point features. $e_m \in \mathbb{R}^{1 \times C}$, $f_k \in \mathbb{R}^{H \times W \times C}$ and $f_n \in \mathbb{R}^{1 \times C}$ represent per-label, per-image and per-point features. Then, the image features are lifted to 3D and assign each point a pixel feature if the point is visible in the images. In other words, given a point, we calculate its 2D coordinates by point-to-pixel mapping $\Gamma : \mathbb{R}^3 \rightarrow \mathbb{R}^2$ and then copy the corresponding pixel feature to the point, denoted as $f_n^{\text{im}} \in \mathbb{R}^{1 \times C}$. Eventually, each point is assigned a semantic label as follows:

$$\hat{l}_n = \underset{m}{\operatorname{argmax}} \left(\max \left(\operatorname{SIM}(f_n, e_m) \parallel \operatorname{SIM}(f_n^{\text{im}}, e_m) \right) \right) \quad (5.5)$$

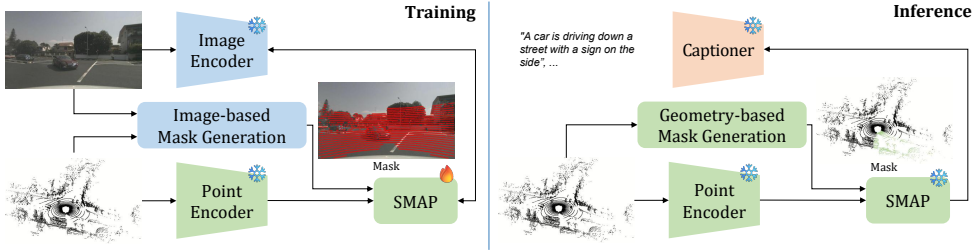


Figure 5.3: Point Captioner Overview. The image encoder and point encoder are pre-aligned in the CLIP latent space. During training (**left**), Sparse Masked Attention Pooling (SMAP) aggregates features from points visible in the image (highlighted in red) and is supervised using CLIP image features. During inference (**right**), neither the image nor camera intrinsic parameters are available. To address this, a group of masks are generated based solely on geometric information. The SMAP output is then decoded into a group of captions. For simplicity, only one image (left) and one sector (right) are shown.

where \hat{l}_n denotes the predicted label for point p_n , $\text{SIM}(\cdot, \cdot)$ is a similarity metric, for which we employ dot product, producing a tensor $\in \mathbb{R}^{1 \times M}$ and \parallel indicates concatenation when image features are available. $\max(\cdot)$ takes a tensor $\in \mathbb{R}^{2 \times M}$ as input, performs a column-wise maximum operation, and outputs a tensor $\in \mathbb{R}^{1 \times M}$.

5.3.3 Point Captioner

Inspired by LidarClip [200], we develop the Point Captioner that first encodes points to CLIP latent space and then decodes CLIP features to captions. However, LidarClip only provides a global caption per point cloud, leading to limited coverage of semantic entities. Therefore, we propose a sparse masked attention pooling (SMAP) that can increase the receptive field and output a controllable number of feature vectors, making it possible to train the network with a varying number of images. We detail the training stage, the inference stage and the SMAP in the following paragraphs.

Training. The training of the Point Captioner is essentially a 2D-to-3D distillation that transfers knowledge from the 2D vision foundation model to the 3D backbone. We utilize the CLIP image encoder [22] $h_{\text{im}}^{\text{clip}} : \mathbb{R}^{H \times W \times 3} \rightarrow \mathbb{R}^{1 \times 1 \times C}$ and a CLIP-aligned point encoder $h_{\text{pt}} : \mathbb{R}^{N \times 3} \rightarrow \mathbb{R}^{N \times C}$ to encode images and points. However, $h_{\text{im}}^{\text{clip}}(\cdot)$ outputs a global feature vector that does not match the per-point features obtained from $h_{\text{pt}}(\cdot)$. Therefore, we add SMAP to pool point-wise features. As shown in Figure 5.3 (left), during training, a point cloud and a point-to-pixel mapping function (visualized as an image) are fed to the image-based mask generation. The output is a point-wise binary mask, where *true* indicates the point is visible in the image. We visualize the point mask by projecting the point to the image. The mask and the point features obtained from $h_{\text{pt}}(\cdot)$ are

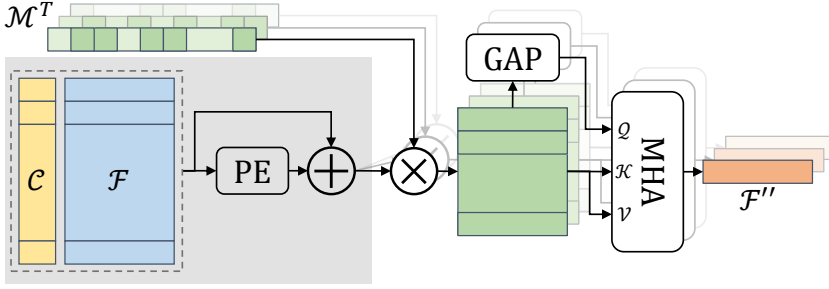


Figure 5.4: Sparse Masked Attention Pooling (SMAP). Given the coordinates and features of all points, a relative positional encoding (PE) is applied, followed by a residual connection. Masks are applied to the points, creating groups of point subsets. Global Average Pooling (GAP) on each subset produces a mean feature as a query. Finally, multi-head attention (MHA) is applied within each group to generate one feature per subset.

5

input to SMAP. SMAP integrates features of points that are visible in the image and is supervised by the output feature of $h_{\text{im}}^{\text{clip}}(\cdot)$. Note that only one image is visualized in Figure 5.3 for clarity but all images corresponding to the point cloud are processed in parallel during training.

Inference. Our goal is to generate diverse captions that comprehensively cover all semantic categories without requiring the intrinsic parameters of cameras. To achieve this, we propose a **geometry-based mask generation** strategy that efficiently partitions the point cloud into multiple regions, followed by individual captions for each region. Given the differences in point cloud distributions, we adopt cylindrical sector-based partitioning for outdoor scenes and square pillar-based partitioning for indoor scenes. In the remainder of this paragraph, we illustrate our approach using outdoor point clouds as an example, while details on indoor partitioning are provided in the supplementary materials. The point cloud is first transformed from a Cartesian coordinate system $\{p_n = (x_n, y_n, z_n)\}_{n=1}^N$ to a polar coordinate system $\{p_n = (\rho_n, \varphi_n, z_n)\}_{n=1}^N$ and then split into T sectors according to its polar angle φ . The binary masks $\mathcal{B} = \{b_n^t\} \in \mathbb{R}^{N \times T}$ are obtained as follows:

$$b_n^t = \begin{cases} \text{true}, & \text{if } \frac{t}{T}2\pi \leq \varphi < \frac{t+1}{T}2\pi \\ \text{false}, & \text{otherwise.} \end{cases} \quad (5.6)$$

where $t \in \{0, 1, \dots, T-1\}$. This way, SMAP generates mask-wise features that are further decoded into captions in the caption module. The merit of this method is that the number of captions is controllable by changing T .

Sparse Masked Attention Pooling (SMAP). SMAP takes as input 1) an entire point cloud with its per-point coordinates $\mathcal{C} \in \mathbb{R}^{N \times 3}$ and features $\mathcal{F} = \mathbf{F}_{\text{pt}} \in \mathbb{R}^{N \times C}$, and 2) J binary point-wise

masks $\mathcal{B} \in \mathbb{R}^{J \times N}$, where $J = K$ during training and $J = T$ during inference. SMAP first conducts a relative positional encoding and then applies the masks to the encoded point features:

$$\mathcal{F}' = \mathcal{B} * (\mathcal{F} + \text{PE}(\mathcal{C}, \mathcal{F})) \quad (5.7)$$

where PE indicates a relative positional encoding as in [202] and $*$ denotes matrix multiplication. The masks essentially divide a point cloud into several subsets, allowing replacement. Therefore, the feature $\mathcal{F}' = \{f'_j\}_{j=1}^J \in \mathbb{R}^{N_j \times C}$ has a variable length per mask. After multiplication, features \mathcal{F}' go through two paths: 1) zero-padded to the same length and then delivered to multi-head attention (MHA) as key \mathcal{K} and value \mathcal{V} , and 2) passed to a global average pooling and then input to MHA as query \mathcal{Q} . Eventually, we obtain pooled features $\mathcal{F}'' \in \mathbb{R}^{J \times C}$.

5.4 Evaluation

Auto-vocabulary segmentation introduces a novel setting without a standardized benchmark, making it challenging to compare methods directly. In this section, we introduce the challenges of evaluation this novel task and propose two strategies to evaluate segmentation accuracy and the semantic consistency between points and text labels.

5.4.1 Challenges

In open-vocabulary segmentation, which is a similar but simpler task, evaluation can be performed on conventional segmentation datasets by using the categories present in the annotations as pre-defined queries. However, this inherently means the model has prior knowledge of the classes it is expected to predict. In auto-vocabulary segmentation, however, no such information is available beforehand, presenting a unique challenge for evaluation. Moreover, natural language introduces ambiguities [180, 154], creating complex relationships between classes, such as synonymy, hyponymy and hypernymy. For instance, *road* could be labeled as *drivable surface*, *street*, or *roadway*, while a *tire* might be classified independently or as part of a wheel or vehicle. This makes it challenging to determine whether an instance is appropriately tagged with a precise semantic label. Given these nuances, evaluating the quality of generated labels and segmentation accuracy becomes complex, as the model must align with the varying language used in annotations, even when sometimes only general categories are provided in the ground truth.

To address these challenges, we propose two solutions. Firstly, we introduce a novel, objective and annotation-independent metric in Section 5.4.2 that assesses how accurately a label - either auto-generated or selected from a fixed vocabulary - fits a given 3D point. This metric allows for

flexible, any-to-any class evaluation. Secondly, we leverage an LLM-based mapping approach to align auto-generated vocabulary classes with the ground-truth classes, enabling us to effectively evaluate both the quality of the segmentation mask and the relevance of the predicted labels (Section 5.4.3).

5.4.2 Text-Point Semantic Similarity Metric

We introduce the Text-Point Semantic Similarity (TPSS) metric, a measure independent of dataset annotations and subjective assessment. TPSS draws inspiration from inference with CLIP [22], where the best label out of a set of target classes $\{m_0, \dots, m_M\}$ is assigned to an image:

$$\hat{l} = \underset{m}{\operatorname{argmax}}(\operatorname{SIM}(f^{\text{im}}, e_m)) \quad (5.8)$$

where \hat{l} represents the predicted label, f^{im} is the image feature, and e_m denotes the text embeddings for class m . This equation identifies the label with the closest text embedding to the provided image feature in latent space, indicating the highest semantic alignment within CLIP’s language space. TPSS metric employs a similar approach, comparing pairs of individual point features with text features in this aligned space. This enables evaluation of how well any label corresponds to a specific point based on semantic similarity, making TPSS ideal for assessing both dynamic and fixed vocabularies. For further illustration, consider a scenario where a LiDAR point belongs to an object outside the nuScenes official classes, such as a “trash bin”, and is thus annotated as “background”. If our method predicts “garbage can” for this point, it should not be penalized for not predicting “background”, as the original prediction is semantically closer to “trash bin”. TPSS accounts for such cases, evaluating the predicted label based on the object’s visual appearance rather than annotation setting or potential bias. Formally, let $\mathbf{P} = \{p_n\}_{n=1}^N$ be a point cloud with N points and $\mathbf{L} = \{l_m\}_{m=1}^M$ be a set of M unique semantic labels generated for this point cloud. The text embeddings \mathbf{E} and the point features \mathbf{F} are obtained as follows:

$$\mathbf{E} = \{e_m\}_{m=1}^M \leftarrow g_{\text{tx}}(\mathbf{L}) \quad (5.9)$$

$$\mathbf{F} = \{f_n\}_{n=1}^N \leftarrow g_{\text{pt}}(\mathbf{P}) \quad (5.10)$$

where $g_{\text{tx}}(\cdot)$ and $g_{\text{pt}}(\cdot)$ are the frozen CLIP text encoder [22] and a CLIP-aligned point encoder, respectively. The TPSS score is calculated as follows:

$$S_n = \max_m (\operatorname{SIM}(f_n, e_m)) \quad (5.11)$$

$$\operatorname{TPSS}(\mathbf{P}, \mathbf{L}, g_{\text{tx}}, g_{\text{pt}}) = \operatorname{mean}_n(S_n) \quad (5.12)$$

where S_n is a point-wise similarity score for the point n . $\text{TPSS}(\mathbf{P}, \mathbf{L}, g_{\text{pt}}, g_{\text{tx}})$ measures the text-point semantic similarity between the point cloud \mathbf{P} and the label set \mathbf{L} . TPSS is encoder-agnostic as long as g_{pt} and g_{tx} are aligned. However, to reliably quantify which label set aligns better with a given point cloud, the point encoder and text encoder must remain unchanged across comparisons.

5.4.3 Mapping Auto-Vocabulary to Fixed Vocabulary

While TPSS effectively measures semantic similarity within the embedding space, evaluating the quality of the resulting segmentations is crucial for meaningful assessment. This requires establishing a correspondence between open-ended classes and the ground truth classes. To achieve this, we employ an evaluation scheme that leverages an LLM-based mapper, inspired by the LLM-based Auto-Vocabulary Evaluator (LAVE) [180]. LAVE maps each unique auto-vocabulary category to a fixed ground truth class in the dataset. After segmenting the LiDAR point cloud using auto-vocabulary categories, each classification is updated according to this mapping. For example, points labeled as *sedan* are reclassified under the *car* category. This mapping enables evaluation of segmentation quality using the widely accepted mean Intersection-over-Union (mIoU) metric based on fixed-vocabulary categories, facilitating comparison with prior methods. Our evaluation framework extends LAVE by integrating mappings with GPT-4o and SBERT [153]. While we provide detailed results of all methods in the supplementary material, GPT-4o is used throughout the main experiments due to its superior mapping accuracy compared to both SBERT and LAVE’s Llama-2-7B.

5.5 Experiments

5.5.1 Experimental Setup

Our method is evaluated on nuScenes [167], ScanNet [173] and ScanNet200 [174] datasets. nuScenes dataset [167] is a comprehensive real-world dataset for autonomous driving research, capturing diverse urban driving scenarios from Boston and Singapore. To increase the spatial density, we aggregate LiDAR points over a 0.5-second interval, focusing on the dataset’s LiDAR segmentation benchmark with 16 manually annotated categories. Given the homogeneity often found in autonomous driving scenarios, we also assess 3D-AVS on the ScanNet [173] and ScanNet200 [174]. ScanNet dataset is an indoor dataset with 20 annotated classes. ScanNet200 updates the annotations of ScanNet with more and finer-grained categories, *i.e.*, 200 categories, while keeping the input point clouds unchanged.

Label Set	Human-free	nuScenes [167]	ScanNet [173]
Previous Label Sets			
Official Label Set	✗	7.39	3.44
Extended Label Set [175]	✗	8.70	-
This Work			
3D-AVS-Image	✓	8.78	3.49
3D-AVS-LiDAR	✓	8.80	3.71
3D-AVS	✓	9.65	3.78

Table 5.1: TPSS on the Validation Sets of nuScenes [167] and ScanNet [173]. Two datasets are created with 16 and 20 official categories, respectively. OpenScene [175] extends the nuScenes label set by manually defining 43 sub-categories. 3D-AVS outperforms these human-defined categories on both datasets, demonstrating its ability to generate a semantically more precise label set.

5

5.5.2 Implementation Details

Image Captioner. We generate the image-based vocabulary with the xGen-MM (BLIP-3) [188] model using a temperature of 0.05, number of beams set to 5 and top-p set to the default value, 1. Our prompt is “*Briefly describe all objects in the <image>. Be concise. Only name the object names.*”, where <image> refers to the image token.

Caption Module in Point Captioner. We follow LidarCLIP [200] to use the pre-trained caption model from ClipCap [145] as our captioning decoder. It decodes a CLIP feature vector to one caption.

Segmenter. We exploit OpenSeg [122] model and CLIP text encoder [22] as our image encoder h_{im}^{hr} and text encoder h_{tx} , respectively. We employ as our point encoder OpenScene [175] with its released OpenSeg pre-trained weights on nuScenes [167] and ScanNet [173], respectively. We also follow the inference phase of OpenScene where dot production is used as similarity metric SIM.

SMAP. We employ mean square error (MSE) as our loss function. The number of views J for SMAP is varying. During training, it is set the same as the number of images per point cloud, which is six for nuScenes [167] and variable for ScanNet [173]. During inference, it is set to 12 for nuScenes, indicating each point subset occupies a sector of 30 degrees. For ScanNet, each point cloud is divided into $0.5m \times 0.5m$ squares according to their x and y coordinates and then each square is treated as a view.

Training. To train SMAP, we use Adam [203] as the optimizer with an initial learning rate of $1e - 5$. The learning rate is decreased following the polynomial learning rate policy [204] with

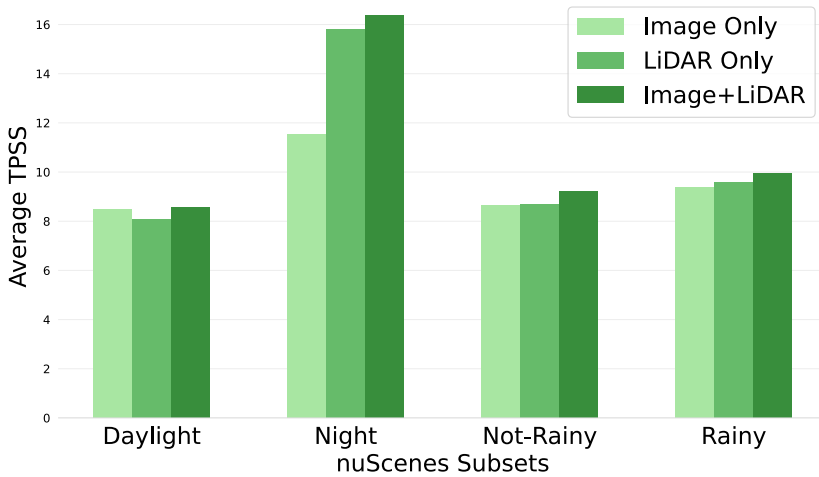


Figure 5.5: TPSS on nuScenes Subsets with Different Light Conditions. LiDAR-only 3D-AVS performs better during night and rainy scenes, suggesting its robustness across difficult conditions.

a decay of 0.9. The SMAP has trained 20 and 10 epochs for nuScenes [167] and ScanNet [173]. Our implementation is open-sourced on GitHub⁴, enabling reproduction and follow-up research.

5.5.3 Label Set Comparison

We compare the generated label set with the fixed, human-defined vocabulary classes in Table 5.1. OpenScene [175] manually create a more fine-grained vocabulary of 43 categories for the nuScenes [167] (originally 16 categories) dataset, boosting the TPSS performance on the dataset from 7.39 to 8.70. Although the performance gain is impressive, Table 5.1 demonstrates that 3D-AVS-generated labels are more semantically consistent with point clouds than manually defined labels, as 3D-AVS outperforms the predefined categories on both nuScenes [167] and ScanNet [173] datasets. Additionally, Table 5.1 demonstrates that combining text generation from both camera and LiDAR inputs, as done in 3D-AVS, improves text-point semantic similarity. This advantage stems from 3D-AVS’ ability to adapt to scenes where one modality struggles. For instance, the image captioner often faces challenges in night scenes due to limited color information, while the point captioner continues to accurately describe relevant objects. This is further reflected in Figure 5.5, which shows that the point captioner proves especially useful in visually challenging scenes where the Image Captioner falls short.

⁴<https://github.com/ozzyou/3D-AVS>

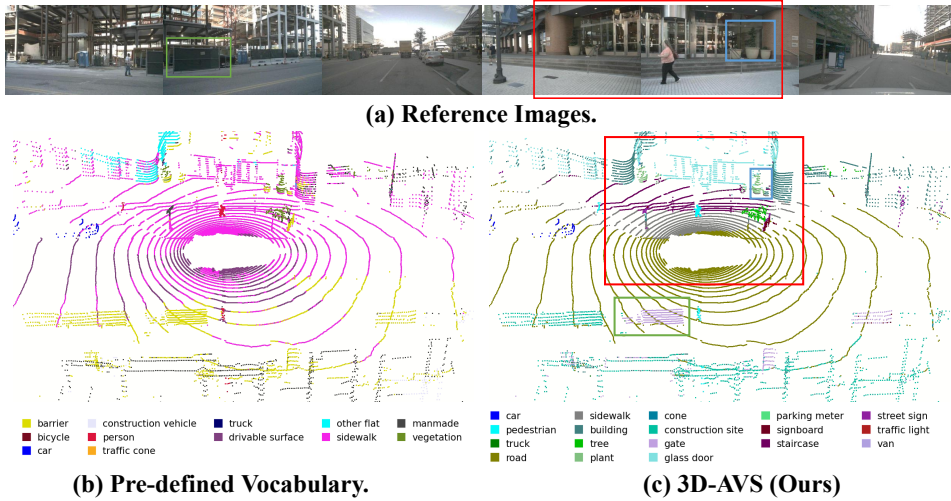


Figure 5.6: Qualitative Comparison between Inputting Pre-defined Vocabulary and 3D-AVS-generated Vocabulary to OpenScene [175] Segmentor. The (a) six-view images are presented for a better scene understanding. While general and ambiguous pre-defined vocabulary leads to large-area error (b). 3D-AVS segments regions with precise class names, e.g., *plant* (blue box), *gate* (green box), *road*, *sidewalk*, *staircase*, *building* and *glass door* (bottom-up in red box). These points are annotated as *vegetation*, *drivable surface*, *sidewalk* and *manmade* in the original dataset (not presented here) but are misclassified as *sidewalk* and *barrier* in (b).

5.5.4 Segmentation Comparison

For quantitative comparison, we employ LAVE [180] to map all generated novel categories back to predefined categories. Next, we calculate segmentation metrics, namely mean IoU (mIoU), on the validation sets of nuScenes [167], ScanNet [173], and ScanNet200 [174]. Note that 3D-AVS does not have any access to the predefined categories during testing, which makes the segmentation task much harder.

Outdoor Dataset. Table 5.2 shows 3D-AVS generates better segmentation results on nuScenes, confirming the effectiveness of 3D-AVS’ open-ended recognition capabilities. The segmentation performance mainly benefits from automatically generated categories for the ambiguous nuScenes categories, such as *driveable surface*, *terrain*, and *man-made*, achieving mIoU of 68.2, 41.4, and 55.4, respectively—substantially outperforming OpenScene [175] (see details in supplementary material). Such an increase is expected, as 3D-AVS is able to generate much more specific namings for these overly general categories which can easily introduce noise. Figure 5.6 highlights some of these generated categories, such as *man-made* being correctly recognized as *staircase*, *building* and *glass door*.

Method	Unknown Vocabulary	Label set	NUS [167] (16)	SN [173] (20)	SN200 [174] (200)
Previous Work (Official Label Set)					
CLIP2Scene [177]	✗	Official	20.8	25.1	-
ConceptFusion [205]	✗	Official	-	33.3	8.8
OpenMask3D [206]	✗	Official	-	34.0	10.3
HICL [207]	✗	Official	26.8	33.5	-
AdaCo [208]	✗	Official	31.2	-	-
CNS [209]	✗	Official	33.5	26.8	-
OpenScene [175]	✗	Official	30.1	47.0	11.7
Diff2Scene [210]	✗	Official	-	48.6	14.2
This Work (Unknown Vocabulary → Official Label Set via LAVE [180])					
3D-AVS (Ours)	✓	Image + LiDAR	36.2	40.5	14.6

Table 5.2: IoU Comparison on nuScenes (NUS) [167], ScanNet (SN) [173] and ScanNet200 (SN200) [174]. We map auto-classes from an unknown vocabulary to the official categories using LAVE [180].

Indoor Datasets. Table 5.2 shows that 3D-AVS achieves a lower mIoU on ScanNet [173] compared to using a fixed vocabulary. This is likely due to the extensive range and variety of objects, where the generated labels must be mapped to a small and coarse-grained set of 20 dataset categories. The state-of-the-art (SOTA) performance on ScanNet200 [174] further supports this argument. Notably, the predictions of 3D-AVS remain identical on ScanNet and ScanNet200, as the input data are the same. The only difference lies in the evaluation: mapping to 20 coarse vocabulary categories in ScanNet versus 200 fine-grained vocabulary categories in ScanNet200. This shift in evaluation granularity introduces a more challenging task while allowing for a more faithful and detailed assessment of segmentation performance. 3D-AVS achieves state-of-the-art results on ScanNet200, underscoring its effectiveness in open-ended 3D segmentation tasks.

5.5.5 Ablation Study

Ablation studies are conducted on the image captioner, point captioner and LAVE mapping to validate our design choices and hyperparameters. The corresponding results are provided in the supplementary material.

5.6 Conclusion

In this chapter, we presented 3D-AVS, the first method for auto-vocabulary LiDAR point segmentation, eliminating the need for human-defined target classes. In suboptimal image captioning conditions, our point captioner can capture missing semantics based on geometric information. To assess the quality of the generated vocabularies in relation to segmentations, we further proposed the TPSS metric. Our experiments show that our model's segmentations are semantically more aligned with the data than its annotations and achieves competitive masking accuracy. We believe 3D-AVS advances scalable open-ended learning for LiDAR point segmentation without human in the loop.