LidarCLIP or: How I Learned to Talk to Point Clouds

Georg Hess†,1,2 Adam Tonderski†,1,3 Christoffer Petersson1,2 Kalle Åström3 Lennart Svensson2
1Zenseact 2Chalmers University of Technology 3Lund University
{first.last}@zenseact.com lennart.svensson@chalmers.se karl.astrom@math.lth.se

Abstract

Research connecting text and images has recently seen several breakthroughs, with models like CLIP, DALL·E 2, and Stable Diffusion. However, the connection between text and other visual modalities, such as lidar data, has received less attention, prohibited by the lack of text-lidar datasets. In this work, we propose LidarCLIP, a mapping from automotive point clouds to a pre-existing CLIP embedding space. Using image-lidar pairs, we supervise a point cloud encoder with the image CLIP embeddings, effectively relating text and lidar data with the image domain as an intermediary. We show the effectiveness of LidarCLIP by demonstrating that lidar-based retrieval is generally on par with image-based retrieval, but with complementary strengths and weaknesses. By combining image and lidar features, we improve upon both single-modality methods and enable a targeted search for challenging detection scenarios under adverse sensor conditions. We also explore zero-shot classification and show that LidarCLIP outperforms existing attempts to use CLIP for point clouds by a large margin. Finally, we leverage our compatibility with CLIP to explore a range of applications, such as image captioning [28], image retrieval [2, 16], semantic segmentation [45], text-to-image generation [31, 33], and referring image segmentation [21, 39].

While most works trying to bridge the gap between NLP and CV have focused on a single visual modality, namely images, other visual modalities, such as lidar point clouds, have received far less attention. Existing attempts to connect NLP and point clouds are often limited to a single application [5, 34, 44] or designed for synthetic data [43]. This is a natural consequence due to the lack of large-scale text-lidar datasets required for training flexible models such as CLIP in a new domain. However, it has been shown that the CLIP embedding space can be extended to new languages [4] and new modalities, such as audio [40], without the need for huge datasets and extensive computational resources. This raises the question if such techniques can be applied to point clouds as well, and consequently open up a body of research on point cloud understanding, similar to what has emerged for images [39, 45].

1. Introduction

Connecting natural language processing (NLP) and computer vision (CV) has been a long-standing challenge in the research community. Recently, OpenAI released CLIP [30], a model trained on 400 million web-scraped text-image pairs, that produces powerful text and image representations. Beside impressive zero-shot classification performance, CLIP enables interaction with the image domain in a diverse and intuitive way by using human language. These capabilities have resulted in a surge of work building upon CLIP embeddings within multiple applications, such as image captioning [28], image retrieval [2, 16], semantic segmentation [45], text-to-image generation [31, 33], and referring image segmentation [21, 39].
We propose LidarCLIP, a method to connect the CLIP embedding space to the lidar point cloud domain. While combined text and point cloud datasets are not easily accessible, many robotics applications capture images and point clouds simultaneously. One example is autonomous driving, where data is both openly available and large scale. To this end, we supervise a lidar encoder with a frozen CLIP image encoder using pairs of images and point clouds from the large-scale automotive dataset ONCE [25]. This way, the image encoder’s rich and diverse semantic understanding is transferred to the point cloud domain. At inference, we can compare LidarCLIP’s embedding of a point cloud with the embeddings from either CLIP’s text encoder, image encoder, or both, enabling various applications.

While conceptually simple, we demonstrate LidarCLIP’s fine-grained semantic understanding for a wide range of applications. LidarCLIP outperforms prior works applying CLIP in the point cloud domain [17, 43] on both zero-shot classification and retrieval. Furthermore, we demonstrate that LidarCLIP can be combined with regular CLIP to perform targeted searches for rare and difficult traffic scenarios, e.g., a person crossing the road while hidden by water drops on the camera lens, see Fig. 1. Finally, LidarCLIP’s capabilities are extended to point cloud captioning and lidar-to-image generation using established CLIP-based methods [9, 28].

In summary, our contributions are the following:

- We propose LidarCLIP, a new method for embedding lidar point clouds into an existing CLIP space.
- We demonstrate the effectiveness of LidarCLIP for retrieval and zero-shot classification in automotive data, where it outperforms existing CLIP-based methods.
- We show that LidarCLIP is complementary to its CLIP teacher and even outperforms it in certain retrieval categories. By combining both methods, we further improve performance and enable retrieval of safety-critical scenes in challenging sensing conditions.
- Finally, we show that our approach enables a multitude of applications off-the-shelf, such as point cloud captioning and lidar-to-image generation.

2. Related work

CLIP and its applications. CLIP [30] is a model with a joint embedding space for images and text. The model consists of two encoders, a text encoder \( F_T \) and an image encoder \( F_I \), both of which produce a single feature vector describing their input. Using contrastive learning, these feature vectors have been supervised to map to a common language-visual space where images and text are similar if they describe the same scene. By training on 400 million text-image pairs collected from the internet, the model has a diverse textual understanding.

The shared text-image space can be used for many tasks. For instance, to do zero-shot classification with \( K \) classes, one constructs \( K \) text prompts, e.g., "a photo of a (class name)". These are individually embedded by the text encoder, producing a feature map \( Z_T \in \mathbb{R}^{K \times d} \). The logits for an image \( I \) are calculated by comparing the image embedding, \( z_I \in \mathbb{R}^d \), with the feature map for the text prompts, \( Z_T \), and class probabilities \( p \) are found using the softmax function, \( \text{softmax}(Z_T z_I) \). In theory, any concept encountered in the millions of text-image pairs could be classified with this approach. Further, by comparing a single prompt to multiple images, CLIP can also be used for retrieving images from a database.

Multiple works have built upon the CLIP embeddings for various applications. DALL-E 2 [31] and Stable Diffusion [33] are two methods that use the CLIP space to condition diffusion models for text-to-image generation. Other works have recently shown how to use text-image embeddings to generate single 3D objects [35] and neural radiance fields [38] from text. In [45], CLIP is used for zero-shot semantic segmentation without any labels. Similarly, [39] extracts pixel-level information for referring semantic segmentation, i.e., segmenting the part of an image referred to via a natural linguistic expression. We hope that LidarCLIP can spur similar applications for 3D data.

CLIP outside the language-image domain. Beside new applications, multiple works have aimed to extend CLIP to new domains, and achieved impressive performance in their respective domains. For videos, CLIP has been used for tasks like video clip retrieval [22, 24] and video question answering [42]. In contrast to our work, these methods rely on large amounts of text-video pairs for training. Meanwhile, WAV2CLIP [40] and AudioCLIP [14] extend CLIP to audio data for audio classification, tagging, and retrieval. Both methods use contrastive learning, which typically requires large batch sizes for convergence [6]. The scale of automotive point clouds would require extensive computational resources for contrastive learning, hence we supervise LidarCLIP with a simple mean squared error, which works well for smaller batch sizes and has been shown to promote the learning of richer features [4].

Point clouds and natural language. Recently, there has been increasing interest in connecting point clouds and natural language, as it enables an intuitive interface for the 3D domain and opens possibilities for open-vocabulary zero-shot learning. In [8] and [26], classifiers are supervised with pre-trained word embeddings to enable zero-shot learning. Parts2Words [36] explores 3D shape retrieval by mapping scans of single objects and descriptive texts to a joint embedding space. However, a key limitation of these approaches is their need for dense annotations [8, 26] or...
Figure 2. Overview of LidarCLIP. We use existing CLIP image and text encoders (top left) and learn to embed point clouds into the same feature space (bottom left). To that end, we train a lidar encoder to match the features of the frozen image encoder on a large automotive dataset with image-lidar pairs. This enables a wide range of applications, such as scenario retrieval (top right), zero-shot classification, as well as lidar-to-text and lidar-to-image generation (bottom right).

detailed textual descriptions [36], making them unable to leverage the vast amount of raw automotive data considered in this paper.

Other methods, such as PointCLIP [43] and CLIP2Point [17], use CLIP to bypass the need for text-lidar pairs entirely. Given a point cloud, they render it from multiple viewpoints and apply CLIP’s image encoder to these renderings. While this works well with dense point clouds of a single object, the approach is not feasible for sparse and large-scale automotive data with heavy occlusions. In contrast, our method relies on an encoder specifically designed for the point cloud domain, avoiding the overhead introduced by multiple renderings and allowing for more flexibility in the model choice.

3. LidarCLIP

In this work, we encode lidar point clouds into the existing CLIP embedding space. As there are no datasets with text-lidar pairs, we cannot rely on the same contrastive learning strategy as the original CLIP model to directly relate point clouds to text. Instead, we leverage that automotive datasets contain millions of image-lidar pairs. By training a point cloud encoder to mimic the features of a frozen CLIP image encoder, the images act as intermediaries to connect text and point clouds; see Fig. 2.

Each training pair consists of an image $x_I$ and the corresponding point cloud $x_L$. Regular CLIP does not perform alignment between pairs, but some preprocessing is needed for point clouds. To align the contents of both modalities, we transform the point cloud to the camera coordinate system and drop all points that are not visible in the image. As a consequence, we only perform inference on frustums of the point cloud, corresponding to a typical camera field of view. We note that this preprocessing is susceptible to errors in sensor calibration and time synchronization, especially for objects along the edge of the field of view. Furthermore, the preprocessing does not handle differences in visibility due to sensor mounting positions, e.g., lidars are typically mounted higher than cameras in data collection vehicles, thus seeing over some vehicles or static objects. However, using millions of training pairs reduces the impact of such noise sources.

The training itself is straightforward. An image is passed through the frozen image encoder $F_I$ to produce the target embedding $z_I$ whereas the lidar encoder $F_L$ embeds a point cloud creating embedding $z_L$,

$$z_I = F_I(x_I), \quad z_L = F_L(x_L).$$

We train $F_L$ to maximize the similarity between features $z_I, z_L \in \mathbb{R}^d$, using the mean squared error (MSE)

$$L_{\text{MSE}}(z_L, z_I) = \frac{1}{d} (z_I - z_L)^T (z_I - z_L).$$

We also run ablations using the cosine similarity loss,

$$L_{\text{cos}}(z_L, z_I) = -\frac{z_I^T z_L}{||z_I|| ||z_L||},$$
By using a similarity loss that only considers positive pairs, as opposed to using a contrastive loss, we avoid the need for large batch sizes \([6, 7]\) and the accompanying computational requirements. Furthermore, the benefits of contrastive learning are reduced in our setting, as we aim to map a new modality into an existing feature space, rather than learning an expressive feature space from scratch.

### 3.1. Joint retrieval

Retrieval is one of the most successful applications of CLIP and is highly relevant for the automotive industry. By retrieval, we mean the process of finding samples that best match a given natural language prompt out of all the samples in a large database. In an automotive setting, it is used to sift through the abundant raw data for valuable samples. Although CLIP works well for retrieval out of the box, it inherits the fundamental limitations of the camera modality, such as poor performance in darkness, glare, or water spray. LidarCLIP can increase robustness by leveraging the complementary properties of lidar.

Relevant samples are retrieved by computing the similarity between a text query and each sample in the database, in the CLIP embedding space, and identifying the samples with the highest similarity. These calculations may seem expensive, but the embeddings only need to be computed once per sample, after which they can be cached and reused for every text query. Following prior work \([30]\), we compute the retrieval score using cosine similarity for both image and lidar

\[
{s_I} = \frac{z_T^Tz_I}{\|z_T\|\|z_I\|}, \quad s_L = \frac{z_T^Tz_L}{\|z_T\|\|z_L\|}.
\]  

(4)

where \(z_T\) is the text embedding. If a database only contains images or point clouds, we use the corresponding score \((s_I\text{ or } s_L)\) for retrieval. However, if we have access to both images and point clouds, we can jointly consider the lidar and image embeddings to leverage their respective strengths.

We consider various methods of performing joint retrieval. When providing both modalities with the same text prompt, we find simply adding the features, \(z_{I+L} = z_I + z_L\), to give the best performance. For separate prompts per modality, we instead add their similarity scores \(s_{I+L} = s_I + s_L\). We also explore methods to aggregate independent rankings for each modality. One such approach is to consider the joint rank to be the mean rank across the modalities. Inspired by \([27]\) we also consider a two-step re-ranking process, where one modality selects a set of candidates which are then ranked by the other modality.

One of the most exciting aspects of joint retrieval is the possibility of using different queries for each modality. For example, imagine trying to find scenes where a large white truck is almost invisible in an image due to extreme sun glare. In this case, one can search for scenes where the image embedding matches "an image with extreme sun glare" while considering the lidar embeddings' similarity to "a scene containing a large truck". This kind of scene would be almost impossible to retrieve using a single modality.

### 4. Experiments

**Datasets.** Training, and most of the evaluation, is done on the large-scale ONCE dataset \([25]\), with roughly 1 million scenes. Each scene consists of a 360° lidar sweep and seven camera images, resulting in \(\sim7\) million unique training pairs. We withhold the validation and test sets and use these for the results presented below.

**Implementation details.** We use the official CLIP package and models, specifically the most capable vision encoder, ViT-L/14, which has a feature dimension \(d = 768\). As our lidar encoder, we use the Single-stride Sparse Transformer (SST) \([11]\) (randomly initialized). Due to computational constraints, our version of SST is down-scaled and contains about 8.5M parameters, which can be compared to the \(~85\)M and \(~300\)M parameters of the text and vision encoders of CLIP. The specific choice of backbone is not key to our approach; similarly to the variety of CLIP image encoders, one could use a variety of different lidar encoders. However, we choose a transformer-based encoder, inspired by the findings that CLIP transformers perform better than CLIP ResNets \([30]\).

SST is trained for 3 epochs, corresponding to \(~20\) million training examples, using the Adam optimizer and the one-cycle learning rate policy. For full details, we refer to our code.

**Retrieval ground truth & prompts.** One difficulty in quantitatively evaluating the retrieval capabilities of LidarCLIP is the lack of direct ground truth for the task. Instead, automotive datasets typically have fine-grained annotations for each scene, such as object bounding boxes, segmentation masks, etc. This is also true for ONCE, which contains annotations in terms of 2D and 3D bounding boxes for five classes, and metadata for the time of day and weather. We leverage these detailed annotations and available metadata to create as many retrieval categories as possible. For object retrieval, we consider a scene positive if it contains one or more instances of that object. To probe the spatial understanding of the model, we also propose a "nearby" category, searching specifically for objects closer than 15 m. We verify that the conclusions hold for thresholds between 10 m and 25 m. Finally, to minimize the effect of prompt engineering, we follow \([13]\) and average multiple text embeddings to improve results and reduce variability. For object retrieval, we use the same 85 prompt templates as in \([13]\), and for the other retrieval categories, we use similar patterns to generate numerous relevant prompts templates. The exact prompts are provided in the source code.
4.1. Zero-shot classification

We would like to study zero-shot classification on ONCE, where scenes may contain many objects and classes. We construct this task by treating each annotated object in ONCE as a separate classification sample. Typically, LidarCLIP outputs a set of voxel features that are pooled into a single, global CLIP feature. For object classification, we instead generate object embeddings by only pooling features for voxels inside the corresponding bounding box, without any object-specific training/fine-tuning.

We compare our performance with PointCLIP \[43\] and CLIP2Point \[17\]. To transfer CLIP to 3D, both methods render point clouds from multiple “virtual” viewpoints, apply the CLIP image encoder, and pool the features from different views. Although these methods work without any additional training of the CLIP model, CLIP2Point proposes to fine-tune the image encoder on rendered point clouds with supervision from image embeddings of the same scene, similar to our training. We evaluate their provided ShapeNet weights and a version we train on ONCE \[1\] using the same scene-level data as LidarCLIP. To evaluate, we follow their proposed protocol and render only points within each annotated bounding box. Although this differs from LidarCLIP’s global processing, it is analogous to the methods’ original single-object setting and greatly improves their performance. Further, we use the prompts proposed in \[43\] instead of our prompt ensembling, as we find them to work better.

We report top-1 accuracy in Tab. 1, both averaged equally over all instances and classwise, as the data contains a few majority classes. LidarCLIP convincingly outperforms its lidar counterparts \[17,43\], demonstrating the gain of training a modality-specific encoder rather than transferring point clouds to the image domain. For completeness, we also extract image crops corresponding to each object bounding box and classify them using CLIP. While performing very well, CLIP, as well as PointCLIP and CLIP2Point, are given clear advantages over Lidar-

<table>
<thead>
<tr>
<th>Fine-tuned on</th>
<th>Cls.</th>
<th>Obj.</th>
</tr>
</thead>
<tbody>
<tr>
<td>PointCLIP [43]</td>
<td>-</td>
<td>29.1%</td>
</tr>
<tr>
<td>CLIP2Point [17]</td>
<td>-</td>
<td>31.1%</td>
</tr>
<tr>
<td>CLIP2Point [17]</td>
<td>ShapeNet</td>
<td>29.8%</td>
</tr>
<tr>
<td>CLIP2Point [17]</td>
<td>ONCE</td>
<td>21.4%</td>
</tr>
<tr>
<td>Image</td>
<td>-</td>
<td>58.6%</td>
</tr>
<tr>
<td>LidarCLIP (ours)</td>
<td>ONCE</td>
<td>43.6%</td>
</tr>
<tr>
<td>Joint (ours) (see above)</td>
<td></td>
<td><strong>60.8%</strong></td>
</tr>
</tbody>
</table>

Table 1. Zero-shot classification on ONCE val, top-1 accuracy averaged over classes/object instances.

4.2. Retrieval

To evaluate retrieval, we report the commonly used Precision at rank K (P@K) \[12,23,32\], for K = 10, 100, which measures the fraction of positive samples within the top K predictions. Recall at K is another commonly used metric \[12,32\], however, it is hard to interpret when the number of positives is in the thousands, as is the case here. We evaluate the performance of three approaches: lidar-only, camera-only, and the joint approach proposed in Sec. 3.1. We compare our performance to PointCLIP \[43\] and CLIP2Point \[17\], for which we render scene-level point clouds (details in the supplementary). Tab. 2 shows that PointCLIP and CLIP2Point are poorly suited for the large-scale point clouds considered here, even though fine-tuning on ONCE greatly improves the CLIP2Point performance. We also include a version of LidarCLIP supervised by ViT-B/32 for direct comparison to existing methods.

Object-level. Interestingly, LidarCLIP performs slightly better than image CLIP for object retrieval despite being trained to mimic the image features. We hypothesize some classes’ features to be more similar across instances in the point cloud than in the image. A class breakdown (see supplementary material), for instance, shows large gains for LidarCLIP in the cyclist class, where we believe the lidar encoder generalizes to cyclists that go undetected by the image encoder. Simultaneously, upon qualitative inspection, we find that the lidar encoder confuses trucks with buses, as these appear more similar in lidar data than in images. We also attempt to retrieve scenes where objects of a given class are close to the ego vehicle. Here, we can see that joint retrieval truly shines. One interpretation is that the lidar is more reliable at determining distance, while the image can be leveraged to distinguish between classes (such as trucks and buses) based on textures and other fine details only visible in the image.

Scene-level. Object-level retrieval focuses on local details of a scene and should trigger even for a single occluded pedestrian on the side of the road. Therefore, we run another set of experiments focusing on global properties such as weather, time of day, and general ‘crowdedness’ of the
Table 2. Retrieval for scenes corresponding to various categories. We report precision at ranks 10 and 100. B and L correspond to the CLIP version (ViT-B/32 or ViT-L/14). † ONCE fine-tuning. Interestingly, Lidar-B outperforms Image-B and Lidar-L, but Joint-L strongly outperforms all other approaches. Detailed results are available in the supplementary material.

<table>
<thead>
<tr>
<th>Category</th>
<th>Objects</th>
<th>Nearby objects</th>
<th>Time of Day</th>
<th>Weather</th>
<th>Crowdness</th>
<th>Overall</th>
</tr>
</thead>
<tbody>
<tr>
<td>PointCLIP</td>
<td>0.30</td>
<td>0.29</td>
<td>0.06</td>
<td>0.51</td>
<td>0.40</td>
<td>0.244</td>
</tr>
<tr>
<td>CLIP2Point</td>
<td>0.24</td>
<td>0.24</td>
<td>0.08</td>
<td>0.50</td>
<td>0.45</td>
<td>0.288</td>
</tr>
<tr>
<td>CLIP2Point†</td>
<td>0.58</td>
<td>0.60</td>
<td>0.46</td>
<td>0.85</td>
<td>0.75</td>
<td>0.625</td>
</tr>
<tr>
<td>Image-B</td>
<td>0.84</td>
<td>0.82</td>
<td>0.76</td>
<td>0.95</td>
<td>0.75</td>
<td>0.834</td>
</tr>
<tr>
<td>LidarCLIP-B</td>
<td>0.92</td>
<td>0.84</td>
<td>0.80</td>
<td>0.60</td>
<td>0.90</td>
<td>0.869</td>
</tr>
<tr>
<td>Joint-B</td>
<td>0.94</td>
<td><strong>0.85</strong></td>
<td>0.84</td>
<td>0.76</td>
<td>0.85</td>
<td>0.881</td>
</tr>
<tr>
<td>Image-L</td>
<td>0.84</td>
<td>0.81</td>
<td>0.76</td>
<td>0.95</td>
<td>0.75</td>
<td>0.856</td>
</tr>
<tr>
<td>LidarCLIP-L</td>
<td>0.92</td>
<td>0.82</td>
<td>0.88</td>
<td>0.60</td>
<td>0.65</td>
<td>0.813</td>
</tr>
<tr>
<td>Joint-L</td>
<td><strong>0.96</strong></td>
<td><strong>0.90</strong></td>
<td><strong>0.81</strong></td>
<td><strong>1.00</strong></td>
<td><strong>0.90</strong></td>
<td><strong>0.944</strong></td>
</tr>
</tbody>
</table>

Table 3. Ablation of the LidarCLIP-B training loss. We report precision at ranks 10 and 100, averaged over all prompts. Training with MSE leads to better retrieval performance.

<table>
<thead>
<tr>
<th>Loss function</th>
<th>P@10</th>
<th>P@100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean squared error</td>
<td><strong>0.869</strong></td>
<td><strong>0.810</strong></td>
</tr>
<tr>
<td>Cosine similarity</td>
<td>0.781</td>
<td>0.748</td>
</tr>
</tbody>
</table>

Table 4. Ablation of joint retrieval methods. We report precision at ranks 10 and 100, averaged over all prompts. All methods improve upon the single-modality models, but averaging lidar and image features before normalization achieves the best performance.

<table>
<thead>
<tr>
<th>Method</th>
<th>P@10</th>
<th>P@100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Image only</td>
<td>0.856</td>
<td>0.810</td>
</tr>
<tr>
<td>LidarCLIP only</td>
<td>0.813</td>
<td>0.803</td>
</tr>
<tr>
<td>Mean feature</td>
<td><strong>0.944</strong></td>
<td><strong>0.876</strong></td>
</tr>
<tr>
<td>Mean norm. feature</td>
<td><strong>0.944</strong></td>
<td>0.875</td>
</tr>
<tr>
<td>Mean score</td>
<td>0.919</td>
<td>0.874</td>
</tr>
<tr>
<td>Mean ranking</td>
<td>0.888</td>
<td>0.854</td>
</tr>
<tr>
<td>Reranking - img first</td>
<td>0.925</td>
<td>0.867</td>
</tr>
<tr>
<td>Reranking - lidar first</td>
<td>0.875</td>
<td>0.860</td>
</tr>
</tbody>
</table>

Table 5. nuScenes val retrieval with different train sets. Performance is averaged over classes. LidarCLIP supports the joint retrieval, even when trained and evaluated on separate datasets.

<table>
<thead>
<tr>
<th>Train set</th>
<th>ONCE</th>
<th>nuScenes</th>
</tr>
</thead>
<tbody>
<tr>
<td>P@K</td>
<td>10</td>
<td>100</td>
</tr>
<tr>
<td>Image</td>
<td>0.69</td>
<td>0.65</td>
</tr>
<tr>
<td>LidarCLIP</td>
<td>0.46</td>
<td>0.40</td>
</tr>
<tr>
<td>Joint</td>
<td><strong>0.74</strong></td>
<td><strong>0.69</strong></td>
</tr>
</tbody>
</table>


performance on a different dataset than it was trained on, namely, the nuScenes dataset [3]. Compared to ONCE, the nuScenes lidar sensor has fewer beams (32 vs 40), lower horizontal resolution, and different intensity characteristics. Further, nuScenes is collected in Boston and Singapore, while ONCE is collected in Chinese cities. The challenge of transferring between these datasets has been shown in unsupervised domain adaptation [25]. Similar to ONCE, the retrieval ground truth is generated from object annotations.

We compare the model trained on ONCE with a reference model trained on nuScenes in Tab. 5. As expected, the differences in sensor characteristics hamper the ability to perform lidar-only retrieval on the target dataset. Notably, we find that the joint method is robust to this effect, showing almost no domain transfer gap, and outperforming camera-only retrieval even with the ONCE-trained lidar encoder.

**Investigating lidar sensing capabilities.** Besides its usefulness for retrieval, LidarCLIP can offer more understanding of what concepts can be captured with a lidar sensor. While lidar data is often used in tasks such as object detection [41], panoptic/semantic segmentation [1, 19], and localization [10], research into capturing more abstract concepts with lidar data is limited and focused mainly on weather classification [15, 37]. However, we show that LidarCLIP can indeed capture complex scene concepts, as already demonstrated in Tab. 2.

Motivated by this, we explore the ability of LidarCLIP to extract color information, by retrieving scenes with "a (color) car". As illustrated in Figure 4, while LidarCLIP struggles to capture specific colors accurately, it consistently differentiates between bright and dark colors. Such partial color information may be valuable for systems fusing lidar and camera information. Additionally, LidarCLIP learns meaningful features for overall scene lighting conditions, as illustrated in Figure 5. It can retrieve scenes based on the time of day, and is even able to distinguish scenes with many headlights from regular night scenes. Notably, all retrieved scenes are sparsely populated, indicating that LidarCLIP does not rely on biases associated with street congestion at different times of the day.

### 4.3. Generative applications

To demonstrate the flexibility of LidarCLIP, we integrate it with two existing CLIP-based generative models.

![Figure 4. Top-5 retrieved examples from LidarCLIP for different colors. Note that images are only for visualization, point clouds were used for retrieval. LidarCLIP consistently differentiates black and white but struggles with specific colors.](image1)

![Figure 5. Top-5 retrieved examples from LidarCLIP for different lighting conditions (image only for visualization). LidarCLIP is surprisingly good at understanding the lighting of the scene, to the point of picking up on oncoming headlights with great accuracy.](image2)
For lidar-to-text generation, we utilize an image captioning model called ClipCap [28], and for lidar-to-image generation, we use CLIP-guided Stable Diffusion [33]. In both cases, we replace the expected text or image embeddings with our point cloud embedding.

We evaluate image generation with the widely used Fréchet Inception Distance (FID) [29]. For this, we randomly select ≈6000 images from ONCE val and generate images using CLIP-generated captions, CLIP features, or a combination of both. Notably, this setting not only evaluates the image generation performance but also serves as a proxy for assessing the captioning quality. While FID is widely used, it has been shown to sometimes align poorly with human judgment [20]. To complement this evaluation, we use CLIP-FID, with a different CLIP model to avoid any bias. We also implement pix2pix [18] as a baseline for lidar-to-image generation; details are in the supplementary material. Our results, presented in Tab. 6, demonstrate that incorporating captions significantly improves the photo-realism of the generated images. Interestingly, LidarCLIP with captions even outperforms image CLIP without captions, underscoring the effectiveness of our approach in generating high-quality images from point cloud data.

Some qualitative results are shown in Fig. 6. We find that both generative tasks work fairly well out of the box. The generated images are not entirely realistic, partly due to a lack of tuning on our side, but there are clear similarities with the reference images. This demonstrates that our lidar embeddings can capture a surprising amount of detail. We hypothesize that guiding the diffusion process locally, by projecting regions of the point cloud into the image plane, would result in more realistic images. We hope that future work can investigate this avenue. Similarly, the captions can pick up the specifics of the scene. However, we notice that more ‘generic’ images result in captions with very low diversity, such as “several cars driving down a street near to tall buildings”. This is likely an artifact of the fact that the captioning model was trained on COCO, which only contains a few automotive images and has a limited vocabulary.

5. Limitations

For the training of LidarCLIP, a single automotive dataset was used. While ONCE [25] contains millions of image-lidar pairs, they originate from about 1,000 densely sampled sequences, meaning that the dataset lacks diversity, such as “several cars driving down a street next to tall buildings”. This is likely an artifact of the fact that the captioning model was trained on COCO, which only contains a few automotive images and has a limited vocabulary.

Acknowledgements. This work was partially supported by the Knut and Alice Wallenberg Foundation. Computational resources were provided by the Swedish National Infrastructure for Computing at C3SE and NSC, partially funded by the Swedish Research Council, grant agreement no. 2018-05973.
References


