Online Segmentation of LiDAR Sequences: Dataset and Algorithm

Romain Loiseau[1,2], Mathieu Aubry[1], and Loïc Landrieu[2]

[1] LIGM, Ecole des Ponts, Univ Gustave Eiffel, CNRS, France
[2] LASTIG, Univ Gustave Eiffel, IGN, ENSG

Abstract. Roof-mounted spinning LiDAR sensors are widely used by autonomous vehicles. However, most semantic datasets and algorithms used for LiDAR sequence segmentation operate on 360° frames, causing an acquisition latency incompatible with real-time applications. To address this issue, we first introduce HelixNet, a 10 billion point dataset with fine-grained labels, timestamps, and sensor rotation information necessary to accurately assess the real-time readiness of segmentation algorithms. Second, we propose Helix4D, a compact and efficient spatio-temporal transformer architecture specifically designed for rotating LiDAR sequences. Helix4D operates on acquisition slices corresponding to a fraction of a full sensor rotation, significantly reducing the total latency. Helix4D reaches accuracy on par with the best segmentation algorithms on HelixNet and SemanticKITTI with a reduction of over 5× in terms of latency and 50× in model size. The code and data are available at: `https://romainloiseau.fr/helixnet`

Keywords: LiDAR, Transformer, Autonomous Driving, Real-Time, Online Segmentation

1 Introduction

Due to their low acquisition latency and high precision, rotating LiDAR sensors are among the most prevalent sensors for autonomous vehicles [38]. The acquired sequences of 3D points exhibit a complex structure in which the temporal and spatial dimensions are entangled through the rotation of the sensor around a reference point in motion; see Figure 1. However, this structure is often not reflected in the formatting of open-access LiDAR datasets [3,25,28], which are discrete sequences of range images, or frames, each corresponding to a 360° degree arc around the sensor. Consequently, most LiDAR semantic segmentation methods operate on one or several such frames at the same time, in the image [12] or point cloud [50,47,42] format. However, waiting for an entire frame to be acquired introduces an unavoidable latency of more than 100ms on top of the processing time, excluding applications for high-speed or urban driving. In this paper, we address this issue by introducing (i) HelixNet, the largest available LiDAR dataset, and whose fine-grained point information allows for the realistic real-time evaluation of segmentation methods, and (ii) Helix4D, a spatio-temporal transformer designed for the efficient segmentation of LiDAR sequences.

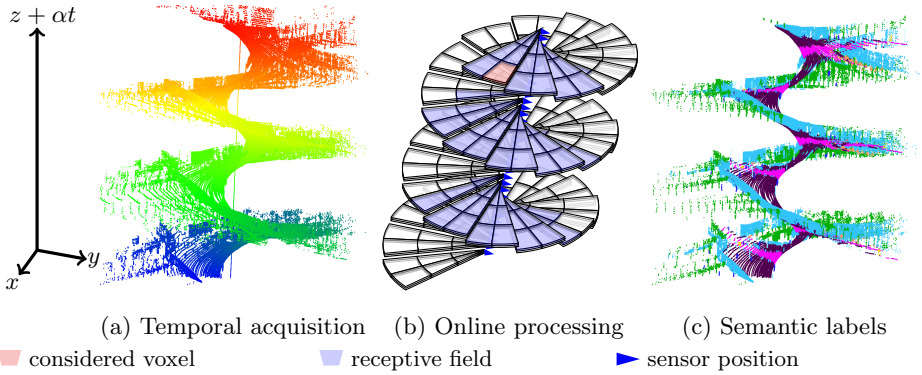


Fig. 1: **Online LiDAR Segmentation**. The 3D point sequences of rotating LiDAR data of our proposed dataset HelixNet follow a complex helix-like structure in space and time, represented in a by using the vertical axis for both time and elevation. We propose an efficient spatio-temporal transformer to process angular slices of data centered on the sensor's position. The slices are partitioned into voxels, each attending other voxels from past slices to build a large spatio-temporal receptive field b. Our proposed model can segment the LiDAR point stream c with state-of-the-art accuracy and in real-time.

Our dataset HelixNet, has several key advantages compared to standard datasets such as SemanticKITTI [3], see Table 1. By organizing points with respect to sensor rotation and reporting their precise release times, we can accurately benchmark the real-time readiness of leading state-of-the-art LiDAR sequence segmentation algorithms. Furthermore, the pointwise sensor orientation allows us to split the data into slices of acquisition corresponding to a fraction of the sensor's rotation. These slices can be processed sequentially by our proposed network Helix4D, resulting in a lower acquisition latency and a more realistic scenario for autonomous driving. Based on a spatio-temporal transformer designed explicitly for LiDAR sequences, Helix4D is more than 50 times smaller than the current best semantic segmentation architectures and reaches state-of-the-art performance with significantly reduced latency.

2 Related work

We present an overview of the existing LiDAR datasets related to autonomous driving, and a summary of the recent developments in 3D semantic segmentation.

Autonomous Driving 3D Datasets. As autonomous driving becomes an increasingly realistic prospect, multiple datasets have been proposed to evaluate the performance of perception algorithms [33,11]. In addition to cameras, rotating LiDARs have become one of the most prevalent sensors mounted on autonomous vehicles due to their high accuracy, low latency, and steadily decreasing prices [38]. ApolloScape [23], DublinCity [51], and TerraMobilita/iQmulus [43] have been

Table 1: **Embarked LiDAR Datasets with Semantic Point Annotations.** With over 8.8B annotated 3D points, HelixNet is 70% larger than SemanticKITTI, and includes more diverse scenes spanning 6 different French cities. Contrary to other datasets, HelixNet arranges points with respect to the sensor rotation and contains fine-grained information about their release time.

Dataset	labels	frames	classes	span	format
HelixNet (Ours)	8.85B	78k	9	6 cities	sensor rotation
SemanticKITTI [16,3]	5.2B	43k	19	1 city	frame
Rellis3D [25]	1.5B	13k	16	1 city	frame
KITTI-360 [28]	1.0B	81k	37	1 city	frame
A2D2 [17]	387M	41k	38	3 cities	frames
Paris-Lille-3D [37]	143M	N/A	50	2 cities	multi-frame
Toronto3D [41]	78M	N/A	8	1 city	multi-frame

acquired with LiDAR setups that offer scans of urban environments with high precision and density. However, the vertical orientation of the emitters is not compatible with real-time road perception. Several prominent datasets such as NuScene [6] or the very large ONCE dataset [31] provide only object-level annotations (*i.e.* boxes).

This paper focuses on semantic segmentation algorithms for roof-mounted rotating LiDAR sensors. In Table 1, we report several key characteristics of such datasets [25,16,3,28,17,37,41]. Our proposed dataset HelixNet is 70% larger than SemanticKITTI [16,3], and spans 6 cities and various environments. In contrast to previously released datasets, the 3D points of HelixNet are given with respect to the sensor rotation and in the order in which they are made available. This last point proves crucial for evaluating the precision and latency of segmentation algorithms in a setting that is compatible with real-time inference.

Deep Semantic Segmentation of 3D Point Clouds. The development of specific deep architectures for the semantic segmentation of 3D point clouds has led to a tremendous increase in performance [21]. The first set of methods that operate on rotating LiDAR sequences processes data in range image format [27,40,12]. Taking advantage of advances in the implementation of sparse convolutions [18,9], a second set of methods uses fine grids in polar [47], Cartesian [42,8] or cylindrical [50,22] coordinates. A third kind of approach proposes exploiting the temporal dimension of LiDAR acquisitions by *stacking* contiguous frames [9,2]. Observing that cylindrical partitions better capture the geometry of rotating LiDAR acquisition, our proposed Helix4D builds on the idea of Cylinder3D [50] and adds a temporal component to the architecture.

Due to their remarkable performance and scalability, transformers [44] have quickly been adapted from text processing to images [13,30,39,7], videos [1], or meshes [29]. Transformers are also well suited to handle unordered sets, such as 3D point clouds [20,48]. In particular, their scalability can be leveraged to achieve large receptive fields [32,34] and more discriminative features [5]

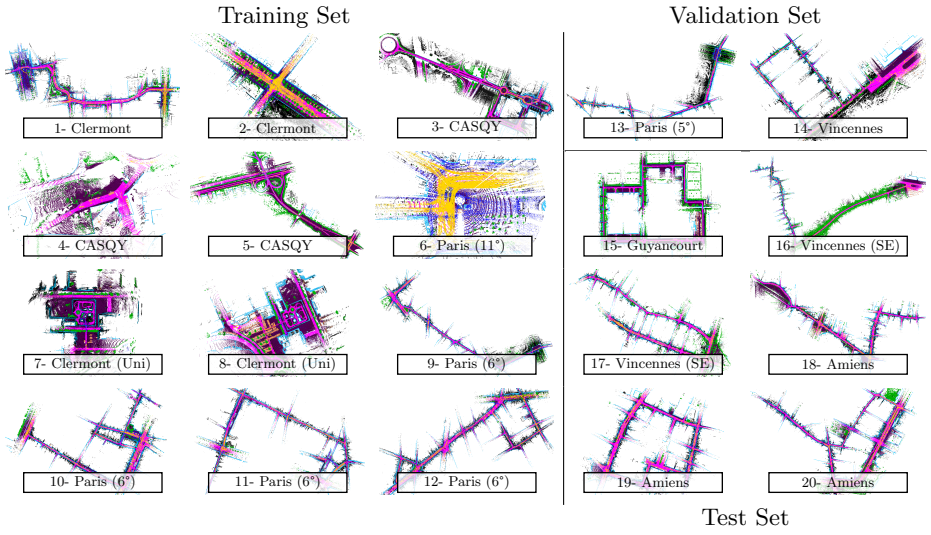


Fig. 2: **Coverage from HelixNet.** We split the acquisitions into 12 training, 2 validation, and 6 testing sequences. HelixNet contains diverse scenes in various urban environments from static or mobile sensors.

than purely convolutional approaches. Transformers can also efficiently process complex temporal [26,45] and spatio-temporal [15] sequences. In the wake of hybrid convolution-transformer models [19,10,14], our proposed Helix4D combines efficient cylindrical convolutions with a simplified spatio-temporal transformer architecture operating at low resolution.

3 HelixNet: A Dataset for Online LiDAR Segmentation

We introduce HelixNet, a new large-scale and open-access LiDAR dataset intended for the evaluation of real-time semantic segmentation algorithms. In contrast to other large-scale datasets, HelixNet includes fine-grained data on sensor rotation and position, as well as point release time.

General Characteristics. As seen in Figure 2, HelixNet contains 20 sequences of 3D points, each corresponding to 6 to 7 minutes of continuous acquisition, for a total of 129 minutes. Scanning was performed by an HDL-64E Velodyne rotating LiDAR [24] mounted on a mobile platform [35]. As shown in Figure 3, HelixNet covers multiple cities and a wide variety of environments such as a university campus, dense historical centers, and a highway interchange. With a total of 10 billion points across 78 800 frames and 8.85 billion individual labels, HelixNet is the largest densely annotated open-access rotating LiDAR dataset by a factor of 1.7 as shown in Table 1. HelixNet follows the file format of SemanticKITTI [3], allowing researchers to evaluate existing code with minimal effort.

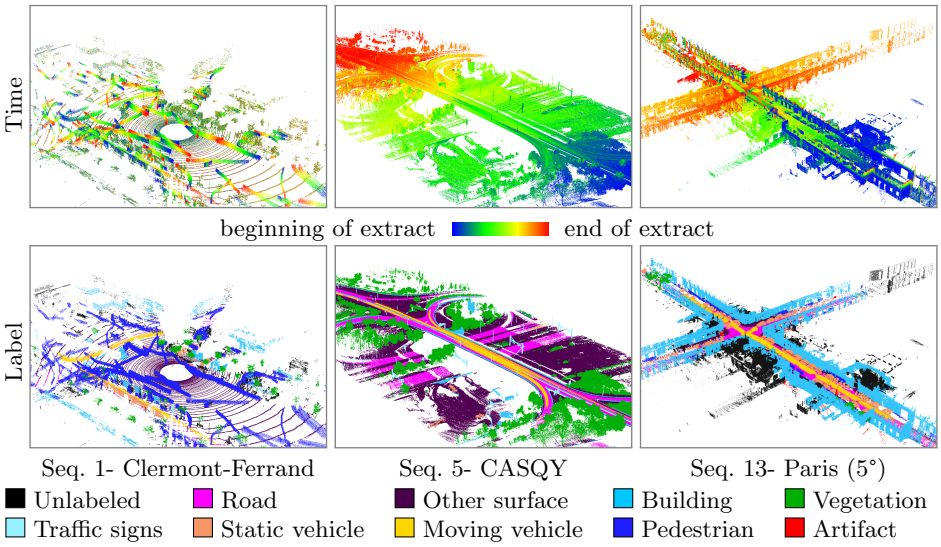
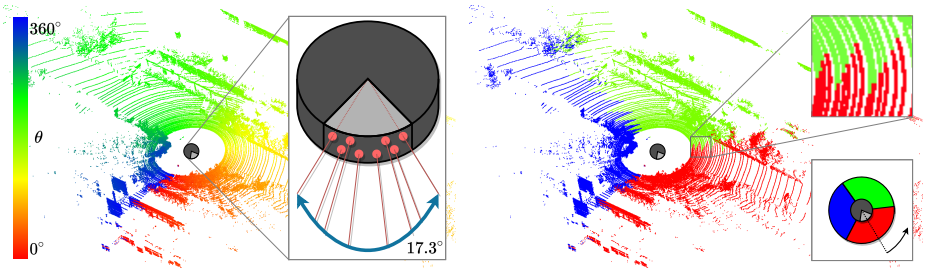


Fig. 3: **Extracts from HelixNet.** Our proposed dataset contains various urban scenes from motorway to pedestrian plazas and historical centers. In the first row, we represent extracts of 15 to 30s of acquisition colored according to the point release time. In the second row, we represent the point semantic labels.

We use a 9-classes nomenclature: *road* (16.4% of all points), *other surface* (22.0%), *building* (31.3%), *vegetation* (8.5%), *traffic signs* (1.6%), *static vehicle* (4.9%), *moving vehicle* (2.1%), *pedestrian* (0.9%), and *acquisition artifact* (0.05%). Points without labels correspond to either *un-annotated* (6.2%) parts of the clouds due to their ambiguity, or point without echos (6.1%). Compared to fine-grained classes such as the ones used by SemanticKITTI [3] or Paris-Lille3D's [37], our focused nomenclature limits class imbalance and makes macro-averaged metrics more stable.

Each point is associated with the 9 following values: (1-3) Cartesian coordinates in a fixed frame of a reference, (4-6) cylindrical coordinate relative to the sensor at the time of acquisition, (7) intensity, (8) fiber index, and (9) packet output time. As detailed in the next paragraph, the last two features are not typically available in large-scale datasets and cannot be inferred.

Sensor-Based Timing and Grouping. A rotating LiDAR consists of a set of lasers—or fibers—arranged on a rotating sensor head. The lasers send periodic pulses of light whose return times give the position of the impact points relative to the sensor. In the context of autonomous driving, these sensors are typically deployed on a moving platform and capture 3D points with centimetric accuracy. The sensor releases the data stream as a discrete temporal sequence of *packets* of 3D points. For an HDL-64E LiDAR, each packet contains 6×64 points, corresponding to around $1°$ rotation of the sensor. To represent the real-time



(a) Rotation of the sensor head (b) Slices covering 120°

Fig. 4: **Sensor Acquisition Geometry.** We represent in a the acquisition of a rotating sensor, which is split into ⅓ turn slices in b. As the Laser emitters position forms an angle of over 17.3° around the sensor head, taking slices with respect to the sensor rotation θ results in a jagged profile.

operational setting of autonomous driving, we associate with each point the timestamp of its *packet output event*, *i.e.* the instant the packet is available and not the acquisition time of the point. The latency between the acquisition of the first point and the complete transfer of its packet is 278μs. Although small compared to acquisition and inference times, this more rigorous timing constitutes a step towards a more realistic evaluation setting of segmentation algorithms of LiDAR sequences.

On top of its absolute position, we associate with each individual point its cylindrical coordinates relative to the position of the sensor at the exact time of its acquisition. This differs from other datasets such as SemanticKITTI [3], which gives the relative position of all points but the absolute position of the sensor only once per frame. While sensor movement can be interpolated, the vehicle trajectory might not be linear and the sensor head rotates. For comparison, at 50km/h, the sensor moves more than 1.4m during each rotation.

LiDAR sequences are typically split into frames containing points that cover a 360° degree arc around the sensor. However, the acquisition geometry makes this grouping artificial. Indeed, the fibers (*i.e.* the individual lasers) do not all face the same direction: they are arranged around the sensor's heads at different angles, with a range of more than 17.3°. This means that the points within a packet are not vertically aligned but present a jagged profile as seen in Figure 4. In order to obtain frames with *straight edges* such as those of SemanticKITTI [3], we would have to consider an acquisition over a sensor rotation of 377°, adding a further 5ms of latency. Contrary to other datasets, HelixNet contains the index of the emitter of each point and organizes the points with respect to the angle of the sensor itself This allows us to easily build frames or frame portions that are directly consistent with the rotation *of the sensor head itself*. This is important for measuring the real latency of segmentation methods and, as described in the next section, contributes to the efficiency of our proposed network.

4 Helix4D: Fast LiDAR Segmentation with Transformers

We consider a sequence of 3D points acquired by a rotating LiDAR on a mobile platform, which we split into chronologically ordered slices of acquisition. As represented in Figure 5, we process each slice with a U-Net architecture [36] with cylindrical convolutions [50]. At the lowest resolution, a spatio-temporal transformer network connects neighboring voxels in space and time, resulting in a large receptive field. We first describe the construction of slices, then our cylindrical U-Net, and finally the transformer module.

4.1 Temporal Slicing

Instead of processing the data frame-by-frame, we propose to split the sequence into slices covering a fixed portion of the sensor rotation, resulting in a shorter acquisition time and a lower latency. Each point i of the sequence is characterized by the angular position θ_i *of the sensor head* at its exact time of acquisition. The points are sorted in chronological acquisition order *i.e.* $\theta_i \leq \theta_j$ if $i < j$. We partition the sequence into groups of contiguous points called slices, acquired during a portion $\Delta\theta \in]0, 2\pi]$ of a full rotation of the sensor itself. Choosing $\Delta\theta = 2\pi$ corresponds to the classic frame-by-frame setting and implies an acquisition latency of 104ms in HelixNet or SemanticKITTI [3]. A slice size of $\Delta\theta = 2\pi/5$ leads to an acquisition latency of 21ms, which is more conducive to real-time processing of driving data.

4.2 Cylindrical U-Net

Inspired by the Cylinder3D model [50], we first discretize each slice along a fine cylindrical partition $\mathsf{grid}\,(1)$. Each point i is associated with a descriptor x_i^{point} based on its intensity, relative position with respect to the sensor in Cartesian and cylindrical coordinates, and its offset with respect to the center of its voxels in $\mathsf{grid}\,(1)$. We compute the point feature f_i^{point} by applying a shared Multi-Layer Perceptron (MLP) $\mathcal{E}^{\mathsf{point}}$ to x_i^{point} for all points i in the slice. The resulting f_i^{point} are then maxpooled with respect to the voxels of $\mathsf{grid}\,(1)$ to serve as input to a convolutional encoder $\mathcal{E}^{\mathsf{grid}}$. The network $\mathcal{E}^{\mathsf{grid}}$ is composed of sparse cylindrical convolutions [18] and strided convolutions for downsampling. $\mathcal{E}^{\mathsf{grid}}$ produces a set of L sparse feature maps $f^{\mathsf{grid}(1)}, \cdots, f^{\mathsf{grid}(L)}$ with decreasing resolutions:

$$f_i^{\mathsf{point}} = \mathcal{E}^{\mathsf{point}}\left(x_i^{\mathsf{point}}\right) \tag{1}$$

$$f^{\mathsf{grid}(1)}, \cdots, f^{\mathsf{grid}(L)} = \mathcal{E}^{\mathsf{grid}}\left(\text{maxpool}\left(f^{\mathsf{point}}\right)\right) \ , \tag{2}$$

where maxpool is performed with respect to $\mathsf{grid}\,(1)$. At the lowest resolution $\mathsf{grid}\,(L)$, we apply the transformer-based module $\mathcal{T}$ presented in the next subsection to the feature map $f^{\mathsf{grid}(L)}$ to obtain the coarse cylindrical map $g^{\mathsf{grid}(L)}$:

$$g^{\mathsf{grid}(L)} = \mathcal{T}\left(f^{\mathsf{grid}(L)}\right) \ . \tag{3}$$

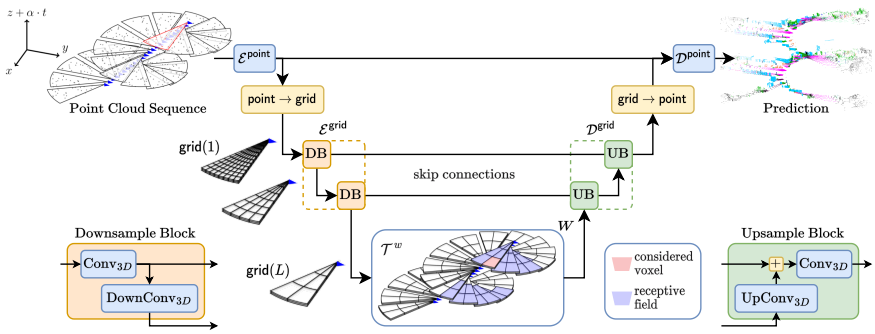


Fig. 5: **Helix4D Architecture.** A point sequence is split into angular slices, whose points are encoded by $\mathcal{E}^{\text{point}}$ and pooled along a fine-grained cylindrical partition. A convolutional encoder $\mathcal{E}^{\text{grid}}$ yields feature maps at lower resolutions. We apply W consecutive spatio-temporal transformer blocks $\mathcal{T}^w$ on the coarse voxels, with attention spanning across current and past slices. The resulting features are up-sampled to full resolution with a convolutional decoder $\mathcal{D}^{\text{grid}}$ using the encoder's maps at intermediate resolutions through skip connections. Finally, the grid features are allocated to the points, which are classified by $\mathcal{D}^{\text{point}}$.

The decoder $\mathcal{D}^{\text{grid}}$ combines cylindrical convolutions and strided transposed convolutions to map $g^{\text{grid}(L)}$ to a feature map $g^{\text{grid}(1)}$ at the highest resolution, and uses the maps $f^{\text{grid}(L-1)}, \cdots, f^{\text{grid}(1)}$ through residual skip connections. We concatenate for each point i the descriptor $g^{\text{grid}(1)}(i)$ of its voxel in $\text{grid}\,(1)$ and its point feature f_i^{point}. Finally, the point decoder $\mathcal{D}^{\text{point}}$ associates a vector of class scores c_i^{point} with each point i:

$$g^{\text{grid}(1)} = \mathcal{D}^{\text{grid}}\left(g^{\text{grid}(L)}, f^{\text{grid}(L-1)}, \cdots, f^{\text{grid}(1)}\right) \tag{4}$$

$$c_i^{\text{point}} = \mathcal{D}^{\text{point}}\left(\left[g^{\text{grid}(1)}(i), f_i^{\text{point}}\right]\right) , \tag{5}$$

where $[\,\cdot\,]$ is the channelwise concatenation operator. The network is supervised by the cross-entropy and Lovász-softmax [4] losses directly on the point prediction, without class weights.

Our approach differs from Cylinder3D [50] by relying on simple $3\times3\times3$ sparse cylindrical convolutions instead of asymmetrical convolutions and dimension-based context modeling. Furthermore, we do not use voxel-wise supervision.

Our simplified architecture results in a lighter computational and memory load, but can still learn rich spatio-temporal features thanks to the addition of the transformer module described below.

4.3 Spatio-Temporal Transformer

We denote by $\mathcal{V}$ the set of non-empty voxels at the lowest resolution $\text{grid}\,(L)$ *for all slices of the considered sequence*. We associate with each voxel v of $\mathcal{V}$ a

feature f_v^{voxel} defined as the value of $f^{\mathsf{grid}(L)}$ at v. We remark that f^{voxel} can be ordered as a non-strictly ordered time sequence, and propose to successively apply W independent transformer blocks $\mathcal{T}^1, \cdots, \mathcal{T}^W$ whose architecture is described below. We denote by g^{voxel} the resulting spatio-temporal voxel representation:

$$g^{\mathsf{voxel}} = \mathcal{T}^W \circ \cdots \circ \mathcal{T}^1(f^{\mathsf{voxel}}) \ . \tag{6}$$

We associate each voxel v of $\mathcal{V}$ with the absolute position (X_v, Y_v, Z_v) of its center, the release time T_v of its first point, and the index I_v of the sensor rotation of its corresponding slice. In order to use a sparse attention scheme, we define for each voxel v a spatio-temporal mask $M(v)$ characterized by a radius R and a set of rotation offsets $P \subset \mathbb{N}$:

$$M(v) = \{u \mid \|(X_v, Y_v, Z_v) - (X_u, Y_u, Z_u)\| < R \,,\, I_v - I_u \in P\} \ . \tag{7}$$

In the context of autonomous driving, we choose $R = 6$m and $P = \{0, 5, 10\}$. With a standard rotation speed of 10Hz, this corresponds to considering slices 0.5 and 1 seconds in the past along with the current one. See Figure 6 for an illustration of the receptive field and attention maps.

Simplified Transformer Block. We now define a single transformer block $\mathcal{T}^w$ with H heads operating on a sequence of voxel features f^{voxel} of dimension D. For each head h and each voxel v, we apply the following operations:

(i) A single linear layer $\mathcal{L}^h$ generates both a key k_v^h of dimension K and a value val_v^h of dimension D/H.
(ii) For all voxels u in the mask $M(v)$, we define the compatibility score $y_{u,v}^h$ as the cross-product between keys and with a learned relative positional encoding $\mathrm{PE}^h(u, v)$.
(iii) The cross-voxel attention $a_{u,v}^h$ is obtained with a scaled softmax.
(iv) The values val_u^h of voxels in $M(v)$ are averaged into a vector $\tilde{f}_v^h$ using their respective cross-voxel attention as weights.
(v) The vectors $\tilde{f}_v^h$ are concatenated channelwise across heads and added to the input of the block to define its output.

These operations can be summarized as follows:

$$k_v^h, \mathrm{val}_v^h = \mathcal{L}^h\left(f_v^{\mathsf{voxel}}\right) \tag{8}$$

$$y_{u,v}^h = \left(k_v^h\right)^{\mathsf{T}} \left(k_u^h + \mathrm{PE}^h(u, v)\right) \quad \text{for } u \in M(v) \tag{9}$$

$$\left\{a_{u,v}^h\right\}_{u \in M(v)} = \mathrm{softmax}\left(\left\{y_{u,v}^h\right\}_{u \in M(v)} / \sqrt{K}\right) \tag{10}$$

$$\tilde{f}_v^h = \textstyle\sum_{u \in M(v)} a_{u,v}^h \mathrm{val}_u^h \tag{11}$$

$$\mathcal{T}^w(f^{\mathsf{voxel}})_v = f_v^{\mathsf{voxel}} + [\tilde{f}_v^1, \cdots, \tilde{f}_v^H] \ . \tag{12}$$

Our design is similar to the classical transformer architecture but uses keys as queries to save memory and computation. We also do not use feed-forward

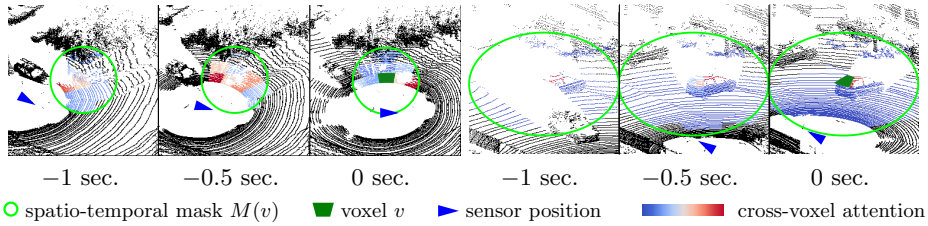


Fig. 6: **Spatio-Temporal Attention.** We represent the spatio-temporal mask and attention score of one head of the transformer for two different voxels. The network gathers information from different frame offsets P as the sensor moves.

networks after averaging the values: the only learnable part of a block $\mathcal{T}^w$ is its linear layers $\mathcal{L}^h$ and its relative positional encoding PE^h.

Since g^{voxel} only requires information about the voxels of the current and past slices, it can be computed sequentially for all slices in the order in which the sensor releases them. For a given slice, the voxel map $g^{\mathsf{grid}(L)}$ for non-empty voxels is given by the values of g^{voxel}, and set to zero otherwise. To save computation at inference time, we store in memory the keys, values, and absolute positions of the voxels in past slices with a fixed buffer of $\max(P)$ rotations. This allows us to allocate a large spatio-temporal receptive field to each voxel without supplementary computations.

Relative Positional Encoding. We propose to learn relative positional vectors $\mathrm{PE}^h(u, v)$ that encode the spatio-temporal offset $(X_u, Y_u, Z_u, T_u) - (X_v, Y_v, Z_v, T_v)$ between voxels u and v for each transformer block w independently. Inspired by the work of Wu *et al.* [46], we first discretize the offsets along each dimension $d \in \{X, Y, Z, T\}$ with B_d irregular bins. For each dimension d and head h, we learn B_d weight vectors of size K. We define the functions $\mathrm{PE}^h_d : \mathbb{R} \mapsto \mathbb{R}^K$ that map the d-dimension of an offset to the vector associated with its corresponding bin. The positional encoding between two voxels u and v is the sum of the vectors corresponding to their discretized offsets in each dimension:

$$\begin{aligned}\mathrm{PE}^h(u, v) = \mathrm{PE}^h_X(X_u - X_v) + \mathrm{PE}^h_Y(Y_u - Y_v) \\ + \mathrm{PE}^h_Z(Z_u - Z_v) \ + \mathrm{PE}^h_T(T_u - T_v)\ .\end{aligned} \tag{13}$$

Relative positional encoding vectors are used directly in the calculation of the compatibility score, as given in (9). Additional details on positional encoding are given in the supplementary material.

5 Evaluating Online Semantic Segmentation

We evaluate the performance and inference time of our approach and other state-of-the-art methods in both online and frame-by-frame settings. We use our proposed dataset HelixNet and the standard SemanticKITTI dataset.

Online Evaluation Setting We aim at evaluating the real-time readiness of rotating LiDAR semantic segmentation algorithms in the context of autonomous driving. The total latency of a model is determined by its inference speed and also the time it takes to acquire its input. Operating on full frames requires at least 104ms of acquisition, which is incompatible with realistic autonomous driving scenarios. Instead, we propose an online evaluation setting using the slices defined in Sec. 4.1. By default, we use a slice size of a fifth turn of the sensor head: $\Delta\theta = 2\pi/5$, corresponding to 21ms of acquisition.

Slices are processed sequentially. We define the inference latency of a segmentation method as the average time between the release of the last point of a slice and its segmentation. To meet the real-time requirement, inference must be faster than the acquisition of a slice. Slower processing would cause the classification to continuously fall behind. Although thinner slices directly reduce acquisition latency, they also make the real-time requirement more strict: as a full turn must be processed in less than 104ms, a fifth turn must be in at most 21ms.

Adapting SemanticKITTI. SemanticKITTI [3,16] contains 43 552 frames along 22 sequences of LiDAR scans densely annotated with 19 classes. In contrast to HelixNet, SemanticKITTI is not formatted with respect to the sensor rotation and only gives the acquisition time and sensor position once per frame. To measure the latency, we make the following approximation: (i) the fibers are assumed to be vertically aligned, meaning that the angle of the points is the same as the sensor's; (ii) we interpolate the acquisition time of points between frames from their angular positions; (iii) we use the acquisition time as release time. To obtain the absolute positions of the voxels, we assume that the sensor jumps between the positions given by the camera poses for each frame. In our open-source implementation, we provide an adapted dataloader allowing methods already running on SemanticKITTI to be evaluated in the online setting with minimal adaptation.

Adapting Competing Methods. To evaluate the semantic segmentation performance and latency of other segmentation algorithms in the online setting, we process the point clouds corresponding to each slice independently and sequentially. This approach restricts the spatial receptive field to the extent of the slices. However, as the sensor moves, it is not straightforward to add past slices whose relative positions may no longer be valid. By explicitly modeling the spatio-temporal offset between voxels, Helix4D does not suffer from this limitation.

We selected five segmentation algorithms with open-source implementations and trained models for SemanticKITTI. SalsaNeXt [12] uses range images, PolarNet [47] and panoptic PolarNet [49] a bird's eye view polar grid, SPVNAS [42] a regular grid, and Cylinder3D [50] a cylindrical grid. We do not consider methods that stack frames as their structure and resulting latency is incompatible with the online setting. When using SemanticKITTI, we evaluate the provided pretrained models on the validation set. On HelixNet, we retrain the models from scratch

Table 2: **Semantic Segmentation Results.** Performance of Helix4D and competing approaches on HelixNet and on the validation set of SemanticKITTI⋆, in the frame-by-frame and online setting. We report the mean Intersection-over-Union (mIoU) and the inference time in ms. Methods meeting the real-time requirement are indicated with ✓and those who do not with ✗. ⋆ SemanticKITTI is denoted as SK. Measuring the latency on this dataset requires making non-realistic approximations about the fiber position.

Method	Size	Full frame ●		104ms	⅕ frame ◣		21ms
	$\times 10^6$	HelixNet	SK⋆	Inf. (ms)	HelixNet	SK⋆	Inf. (ms)
SalsaNeXt [12]	6.7	69.4	55.8	**23** ✓	68.2	55.6	**10** ✓
PolarNet [47]	13.6	73.6	58.2	49 ✓	72.2	56.9	36 ✗
Pan. PolarNet [49]	13.7	—	64.5	50 ✓	—	60.3	44 ✗
SPVNAS [42]	10.8	73.4	64.7	73 ✓	69.9	57.8	44 ✗
Cylinder3D [50]	55.9	76.6	**66.9**	108 ✗	75.0	65.3	54 ✗
Helix4D (Ours)	**1.0**	**79.4**	66.7	45 ✓	**78.7**	**66.8**	19 ✓

Table 3: **HelixNet Semantic Segmentation Scores.** We report the IoU for each class of HelixNet evaluated in the online setting with slices of 72°.

Method	Road	Other surface	Building	Vegetation	Traffic signs	Static vehicle	Moving vehicle	Pedestrian	Artifact	Avg
SalsaNeXt [12]	84.4	76.1	88.7	70.7	61.4	58.6	35.1	68.5	69.7	68.2
PolarNet [47]	86.2	77.9	91.2	77.9	63.2	64.8	35.4	68.1	84.8	72.2
SPVNAS [42]	80.5	77.1	93.0	81.8	68.0	60.9	36.9	71.7	59.0	69.9
Cylinder3D [50]	85.3	78.4	93.5	83.9	66.2	63.3	35.7	77.7	90.9	75.0
Helix4D (Ours)	**87.8**	**82.5**	**94.0**	**84.4**	**68.9**	**72.3**	**46.4**	**78.8**	**93.3**	**78.7**

using the procedure of their official repository. We removed all test-time augmentations that resulted in prohibitive inference time. All methods are evaluated on the same workstation using a NVIDIA TESLA V100 32Go GPU.

Analysis. In Table 2, we report performance in frame-by-frame and online setting with slices of 72°, for Helix4D and competing methods, for HelixNet and SemanticKITTI. We observe that Helix4D yields state-of-the-art accuracy, with mIoU scores only matched by Cylinder3D [50]. However, Cylinder3D is 50 times larger in terms of parameters and twice slower, not meeting the real-time requirement even in the full frame setting. As reported in Table 3, distinguishing moving vehicles in HelixNet is particularly difficult. Our approach even largely outperforms Panoptic PolarNet despite this method using instance annotation as supervision, preventing us from evaluating on HelixNet. Helix4D yields significantly improved scores thanks to its larger spatio-temporal receptive fields: 14m and 1000ms *vs*. 8m and 21ms for Cylinder3D for a fifth rotation.

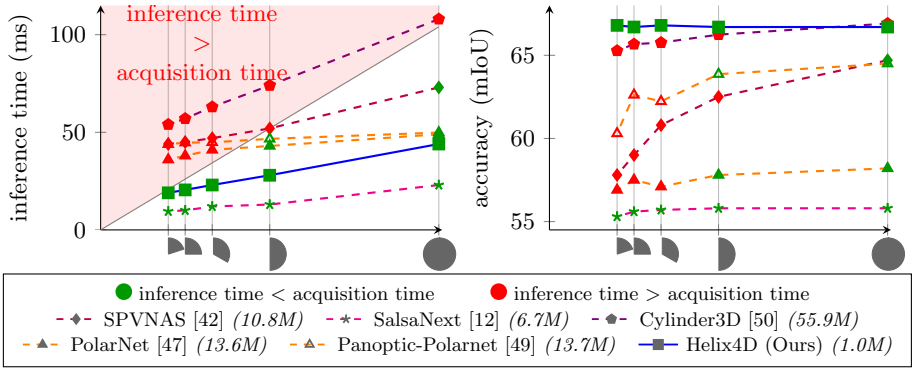


Fig. 7: **Influence of Slice Size.** We plot the processing time (left, in ms) and precision (right, in mIoU) of different methods with respect to the considered size of slices, estimated on the validation set of SemanticKITTI [3]. Methods whose inference time is slower than the acquisition time of the slice (red shaded area) do not meet the real time requirement.

Table 4: **Ablation Study.** We report the speed and accuracy of several modification of our Helix4D on the validation set of SemanticKITTI.

Method	Size $\times 10^3$	Full Frame ● 104ms		⅕ Frame ◣ 21ms	
		mIoU	Inf. (ms)	mIoU	Inf. (ms)
Helix4D	985	**66.7**	45 ✓	**66.8**	19 ✓
(a) Asymmetric Convolutions	1171	66.6	56 ✓	66.6	31 ✗
(b) Cylindrical U-Net	985	58.6	**22** ✓	60.2	**16** ✓
(c) Slice-by-Slice	985	62.9	29 ✓	62.6	19 ✓
(d) w. Queries	993	65.2	45 ✓	64.8	20 ✓
(e) w/o. Positional Encoding	983	64.3	41 ✓	64.1	18 ✓
(f) Helix4D Tiny	306	65.3	45 ✓	64.9	17 ✓

In the online setting, only two approaches meet the real-time requirement: SalsaNeXt [12] and Helix4D. Our approach outperforms SalsaNeXt by over 10 mIoU points in both the full frame and the on-line settings. In short, Helix4D is as accurate as the largest and slowest models with an inference speed comparable to that of the fastest and less accurate models. The total latency (acquisition plus inference time) of our model evaluated online is 40ms ($21 + 19$ms), and reaches the same performance as Cylinder3D evaluated on full frame with a latency of 212ms ($104 + 108$ms), an acceleration of more than 5 folds.

In Figure 7, we report the inference time and mIoU for different slice sizes. Due to various overheads, the inference time appears in an affine relationship with the size of slices, making the real-time requirement stricter for smaller slices. Due to its very design, the performance of Helix4D is not affected by the slice size. In contrast, competing methods perform worse with smaller slices.

Ablation Study We assess on SemanticKITTI the impact of different design choices by evaluating several alterations of our method, reported in Table 4.
(a) Asymmetric Convolutions: we replace the $3 \times 3 \times 3$ convolutions in our U-Net with the convolution design proposed by Cylinder3D [50]. We did not observe a significant change in performance and an increase in run-time of 50%, failing the real-time requirement for slices of 72°.
(b) Cylindrical U-Net: we replace the transformer by a $1 \times 1 \times 1$ convolution on the voxels of the lowest resolution. We observe a slight decrease in run-time and a significant drop of over 6 mIoU points. This result shows that the transformer is able to learn meaningful spatio-temporal features at low resolution.
(c) Slice-by-Slice: we restrict the mask $M(v)$ of each voxel to its current slice. This reduction in the temporal receptive field results in a drop of 4 mIoU points, without any appreciable acceleration.
(d) w. Queries: we modify our simplified transformer to associate a query for each voxel along with keys and values, and use key-queries compatibilities. This does not affect the run-time and slightly decreases the performance.
(e) w/o. Positional Encoding: we remove the relative positional encoding PE in the calculation of compatibilities in equation (9). This leads to a slightly decreased run time, but decreases performance by more than 2.5 points. This illustrates the advantage of explicitly modeling the spatio-temporal voxel offsets.
(f) Helix4D Tiny: we replace the learned pooling in our U-Net with maxpools and use narrower feature maps for a total of $306k$ parameters. This method only performs two points under Helix4D with a third of its parameters.

6 Conclusion

In this paper, we introduced a novel online inference setting for the semantic segmentation of sequences of rotating liDAR 3D point clouds. Our proposed large-scale dataset HelixNet contains specific sensor information that allows a rigorous evaluation of the performance and latency of segmentation methods in our online setting. We also introduced Helix4D, a transformer-based network specifically designed for online segmentation, achieving state-of-the-art results with a fraction of the latency and parameters of competing methods. We hope that our open-source dataset and implementation will encourage the evaluation of future semantic liDAR segmentation methods in more realistic settings and help to bridge the gap between academic work on 3D perception and the operational constraints of autonomous driving.

Acknowledgements This work was supported in part by ANR project READY3D ANR-19-CE23-0007 and was granted access to the HPC resources of IDRIS under the allocation 2022-AD011012096R1 made by GENCI. The point cloud sequences of HelixNet were acquired during the Stereopolis II project [35]. HelixNet was annotated by FUTURMAP. We thank Zenodo for hosting the dataset. We thank François Darmon, Tom Monnier, Mathis Petrovich and Damien Robert for inspiring discussions and valuable feedback.

References

1. Arnab, A., Dehghani, M., Heigold, G., Sun, C., Lučić, M., Schmid, C.: ViViT: A video vision transformer. ICCV (2021)
2. Aygun, M., Osep, A., Weber, M., Maximov, M., Stachniss, C., Behley, J., Leal-Taixé, L.: 4d panoptic LiDAR segmentation. CVPR (2021)
3. Behley, J., Garbade, M., Milioto, A., Quenzel, J., Behnke, S., Stachniss, C., Gall, J.: SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences. ICCV (2019)
4. Berman, M., Triki, A.R., Blaschko, M.B.: The Lovász-softmax loss: A tractable surrogate for the optimization of the intersection-over-union measure in neural networks. CVPR (2018)
5. Bhattacharyya, P., Huang, C., Czarnecki, K.: SA-Det3D: Self-attention based context-aware 3d object detection. ICCV Workshops (2021)
6. Caesar, H., Bankiti, V., Lang, A.H., Vora, S., Liong, V.E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., Beijbom, O.: nuscenes: A multimodal dataset for autonomous driving. CVPR (2020)
7. Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S.: End-to-end object detection with transformers. ECCV (2020)
8. Cheng, R., Razani, R., Taghavi, E., Li, E., Liu, B.: Af2-s3net: Attentive feature fusion with adaptive feature selection for sparse semantic segmentation network. In: CVPR (2021)
9. Choy, C., Gwak, J., Savarese, S.: 4d spatio-temporal convnets: Minkowski convolutional neural networks. CVPR (2019)
10. Coccomini, D., Messina, N., Gennaro, C., Falchi, F.: Combining EfficientNet and vision transformers for video deepfake detection. arXiv preprint arXiv:2107.02612 (2021)
11. Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B.: The cityscapes dataset for semantic urban scene understanding. CVPR (2016)
12. Cortinhal, T., Tzelepis, G., Aksoy, E.E.: SalsaNext: Fast, uncertainty-aware semantic segmentation of LiDAR point clouds for autonomous driving. arXiv:2003.03653 (2020)
13. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al.: An image is worth 16x16 words: Transformers for image recognition at scale. ICLR (2021)
14. d'Ascoli, S., Touvron, H., Leavitt, M.L., Morcos, A.S., Biroli, G., Sagun, L.: Convit: Improving vision transformers with soft convolutional inductive biases. ICML (2021)
15. Fan, H., Yang, Y., Kankanhalli, M.: Point 4d transformer networks for spatio-temporal modeling in point cloud videos. CVPR (2021)
16. Geiger, A., Lenz, P., Stiller, C., Urtasun, R.: Vision meets robotics: The KITTI dataset. International Journal of Robotics Research (2013)
17. Geyer, J., Kassahun, Y., Mahmudi, M., Ricou, X., Durgesh, R., Chung, A.S., Hauswald, L., Pham, V.H., Mühlegg, M., Dorn, S., et al.: A2D2: Audi autonomous driving dataset. arXiv preprint arXiv:2004.06320 (2020)
18. Graham, B., van der Maaten, L.: Submanifold sparse convolutional networks. arXiv preprint arXiv:1706.01307 (2017)
19. Guo, J., Han, K., Wu, H., Xu, C., Tang, Y., Xu, C., Wang, Y.: CMT: Convolutional neural networks meet vision transformers. arXiv preprint arXiv:2107.06263 (2021)

20. Guo, M.H., Cai, J.X., Liu, Z.N., Mu, T.J., Martin, R.R., Hu, S.M.: PCT: Point cloud transformer. Computational Visual Media (2021)
21. Guo, Y., Wang, H., Hu, Q., Liu, H., Liu, L., Bennamoun, M.: Deep learning for 3D point clouds: A survey. Transactions on pattern analysis and machine intelligence (2020)
22. Hong, F., Zhou, H., Zhu, X., Li, H., Liu, Z.: LiDAR-based panoptic segmentation via dynamic shifting network. CVPR (2021)
23. Huang, X., Cheng, X., Geng, Q., Cao, B., Zhou, D., Wang, P., Lin, Y., Yang, R.: The Apolloscape dataset for autonomous driving. CVPR Workshop (2018)
24. Inc., V.L.: HDL-64E User's Manual. Velodyne LiDAR Inc. 345 Digital Drive, Morgan Hill, CA 95037 (2008)
25. Jiang, P., Osteen, P., Wigness, M., Saripalli, S.: Rellis-3D dataset: Data, benchmarks and analysis. ICRA (2021)
26. Katharopoulos, A., Vyas, A., Pappas, N., Fleuret, F.: Transformers are RNNs: Fast autoregressive transformers with linear attention. ICML (2020)
27. Liang, Z., Zhang, Z., Zhang, M., Zhao, X., Pu, S.: Rangeioudet: Range image based real-time 3D object detector optimized by intersection over union. CVPR (2021)
28. Liao, Y., Xie, J., Geiger, A.: KITTI-360: A novel dataset and benchmarks for urban scene understanding in 2D and 3D. arXiv preprint arXiv:2109.13410 (2021)
29. Lin, K., Wang, L., Liu, Z.: End-to-end human pose and mesh reconstruction with transformers. CVPR (2021)
30. Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B.: Swin transformer: Hierarchical vision transformer using shifted windows. ICCV (2021)
31. Mao, J., Niu, M., Jiang, C., Liang, X., Li, Y., Ye, C., Zhang, W., Li, Z., Yu, J., Xu, C., et al.: One million scenes for autonomous driving: Once dataset. arXiv preprint arXiv:2106.11037 (2021)
32. Mao, J., Xue, Y., Niu, M., Bai, H., Feng, J., Liang, X., Xu, H., Xu, C.: Voxel transformer for 3D object detection. ICCV (2021)
33. Neuhold, G., Ollmann, T., Rota Bulo, S., Kontschieder, P.: The mapillary vistas dataset for semantic understanding of street scenes. CVPR (2017)
34. Pan, X., Xia, Z., Song, S., Li, L.E., Huang, G.: 3D object detection with pointformer. CVPR (2021)
35. Paparoditis, N., Papelard, J.P., Cannelle, B., Devaux, A., Soheilian, B., David, N., Houzay, E.: Stereopolis ii: A multi-purpose and multi-sensor 3d mobile mapping system for street visualisation and 3d metrology. Revue française de photogrammétrie et de télédétection (2012)
36. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. MICCAI (2015)
37. Roynard, X., Deschaud, J.E., Goulette, F.: Paris-Lille-3D: A large and high-quality ground-truth urban point cloud dataset for automatic segmentation and classification. The International Journal of Robotics Research (2018)
38. Royo, S., Ballesta-Garcia, M.: An overview of LiDAR imaging systems for autonomous vehicles. Applied Sciences (2019)
39. Strudel, R., Garcia, R., Laptev, I., Schmid, C.: Segmenter: Transformer for semantic segmentation. ICCV (2021)
40. Sun, P., Wang, W., Chai, Y., Elsayed, G., Bewley, A., Zhang, X., Sminchisescu, C., Anguelov, D.: Rsn: Range sparse net for efficient, accurate LiDAR 3D object detection. CVPR (2021)
41. Tan, W., Qin, N., Ma, L., Li, Y., Du, J., Cai, G., Yang, K., Li, J.: Toronto-3D: A large-scale mobile LiDAR dataset for semantic segmentation of urban roadways. CVPR Workshop (2020)

42. Tang, H., Liu, Z., Zhao, S., Lin, Y., Lin, J., Wang, H., Han, S.: Searching efficient 3D architectures with sparse point-voxel convolution. ECCV (2020)
43. Vallet, B., Brédif, M., Serna, A., Marcotegui, B., Paparoditis, N.: Terramobilita/iqmulus urban point cloud analysis benchmark. Computers & Graphics (2015)
44. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L.u., Polosukhin, I.: Attention is all you need. NeurIPS (2017)
45. Vyas, A., Katharopoulos, A., Fleuret, F.: Fast transformers with clustered attention. NeurIPS (2020)
46. Wu, K., Peng, H., Chen, M., Fu, J., Chao, H.: Rethinking and improving relative position encoding for vision transformer. ICCV (2021)
47. Zhang, Y., Zhou, Z., David, P., Yue, X., Xi, Z., Gong, B., Foroosh, H.: Polarnet: An improved grid representation for online LiDAR point clouds semantic segmentation. CVPR (2020)
48. Zhao, H., Jiang, L., Jia, J., Torr, P.H., Koltun, V.: Point transformer. CVPR (2021)
49. Zhou, Z., Zhang, Y., Foroosh, H.: Panoptic-polarnet: Proposal-free LiDAR point cloud panoptic segmentation. In: CVPR (2021)
50. Zhu, X., Zhou, H., Wang, T., Hong, F., Ma, Y., Li, W., Li, H., Lin, D.: Cylindrical and asymmetrical 3D convolution networks for LiDAR segmentation. CVPR (2021)
51. Zolanvari, S., Ruano, S., Rana, A., Cummins, A., da Silva, R.E., Rahbar, M., Smolic, A.: DublinCity: Annotated LiDAR point cloud and its applications. BMVC (2019)