UNIT: Unsupervised Online Instance Segmentation through Time

Supplementary Material

In Appendix A, we detail the construction of pseudo-labels. In Appendix B, we provide details on the architecture and training of UNIT. In Appendix C, we provide details on our improvement of the baselines. Finally, in Appendix D, we detail the experiments on low density point clouds and provides more qualitative results (visualizations).

A. Pseudo-label construction

In this section, we describe the practical implementation details of the construction of pseudo-labels: for our 4D-Seg (Appendix A.1), for TARL-Seg (Appendix A.2) regarding other datasets than the one used in [26], and for the case of PandaSet (Appendix A.3), which does not contain ground-truth instance information.

A.1. 4D-Seg implementation details

Ground point segmentation. To construct our 4D segments, we first apply Patchwork [17] or Patchwork++ [16] on each individual scan to segment ground points.

For SemanticKITTI, we used Patchwork with the set of parameters used by TARL [26]. For nuScenes, which uses a 32-beam Lidar, we used Patchwork++ with a sensor height of 1.840, thresholds for seeds and distance respectively of 0.5 and 0.25, and a minimum range of 2. PandaSet being provided in world coordinates, we register each Lidar scan to its local reference frame before applying Patchwork++ with default parameters. The respective quality of Patchwork++ for nuScenes and PandaSet-GT varies little with the parameters.

Temporal aggregation. Points are then aggregated on a common reference frame, along with their temporal index (an integer), for 40 consecutive scans on SemanticKITTI and PandaSet-GT, and 80 on the less dense nuScenes dataset, that is however captured at 20 Hz while both SemanticKITTI and PandaSet-GT are scanned at 10 Hz.

A grid sampling with a voxel size of 5 cm along the spatial axis and 5 time steps along the temporal axis is then applied to reduce the computation burden. While this grid sampling may reduce the overall quality of the segmentation, it considerably speeds up clustering, enabling longer temporal windows compared to TARL [26].

Clustering. Points that have little chances to belong to an object instance are put aside before clustering. It concerns estimated ground points as well as points extremely close to the sensor, which are likely to be outliers or part of the roof of the ego-vehicle.

For the HDBSCAN spatio-temporal clustering, the temporal dimension is multiplied by 0.03 to reduce its importance relative to the spatial dimensions. On nuScenes, the z coordinate is also multiplied by 0.25 to alleviate a splicing issue occurring in HDBSCAN due to the large vertical distance between Lidar beams of the Velodyne HDL-32E sensor. HDBSCAN's minimum number of samples and minimum cluster size are respectively set to 1 and 300 to obtain a large set of clusters, with very few discarded points.

Instance ID assignment. Each resulting segment is assumed to correspond to a different object instance and is given a separate index. Ground points and discarded points are given special instance indexes. Points culled by the grid sampling process are added back, with the same index as non-culled points in their voxel.

Processing and output. This segmentation is computed a single time, as a pre-processing for our method, and provides temporal windows of consecutive scans with consistent segments, sharing IDs between scans, and special ground and unknown segments.

A.2. Implementing TARL-Seg on more datasets

TARL [26] was implemented by its authors on SemanticKITTI solely. For comparison purposes, we reimplemented TARL-Seg for PandaSet-GT and nuScenes.

We used Patchwork++ [17] for ground point segmentation as it requires less tuning that Patchwork [17], and has been tested with success on 4D-Seg (see Appendix A.1).

As in the original TARL-Seg segment construction for SemanticKITTI, we do not apply any grid sampling. We thus use context windows of 12 time steps for PandaSet-GT and 24 for nuScenes.

Unlike in 4D-Seg, we do not use time as extra information for clustering. However, on nuScenes, we apply the splicing trick described previously (see Appendix A.1), as otherwise results are unfairly poor. The HDBSCAN parameters are the same as those used in TARL for SemanticKITTI.

A.3. Instance ground-truth on PandaSet-GT

The PandaSet dataset does not come with panoptic labels. However, it comes with semantically-labeled 3D bounding boxes, as well as semantic segmentation for all Lidar points, with matching semantic classes between both modalities.

To obtain instance labels, we look for all points inside a box that share the same class as the box; we then assign to these points an instance ID based on their associated

box. We then relax slightly this point assignment by also incorporating into instance segments neighboring points of the same class, on the condition that their distance to the closest point of the instance is less than 1 m. In case a point is less than 1 m away from several instances, it is assigned to the closest one.

The code to generate these instance labels for PandaSet-GT will be supplied with the code for UNIT.

B. UNIT implementation details

B.1. Architecture

Our architecture is largely inherited from [33]. The backbone is a MinkUnet34 [9] with a voxel size of 15 cm. To save some computation resources, we use mixed precision using bfloat16, as well as flash attention [10].

We apply two random augmentations during scan-wise training: a random scaling of $\pm 10\%$ and a random rotation 0-360°. No augmentation is applied when training in a temporal setup.

The input features are the Euclidean distance to the sensor and the returned Lidar intensity.

B.2. Training protocol

In a first phase, we pretrain our network in a single-scan setting. The training is done in mixed precision using AdamW [18], a cosine annealing scheduler for the learning rate with initial value of 10^{-4}, a weight decay of 10^{-2}, and a batch size of 3. Due to differences in dataset sizes, we pretrain the network for 50 epochs on SemanticKITTI, 150 on PandaSet-GT and 4 epochs on nuScenes, which corresponds to roughly 300k training iterations for SemanticKITTI and nuScenes, and 200k iterations for PandaSet-GT.

In the second phase, we finetune the network using pairs of consecutive scans as input. We use the same hyperparameters for the optimizer. We use the same number of epochs, and a batch size of 4 for SemanticKITTI and 5 for other methods, which corresponds to roughly 200k training iterations for SemanticKITTI and nuScenes, and 100k training iterations for PandaSet-GT.

C. Baseline improvement details

All the baselines methods that we initially considered either operate on single scans (3DUIS) or are consistent over a limited time window (TARL-Seg, 4D-Seg). We extended those baselines to stitch objects between two scans with separate ID predictions. For 3DUIS, we stich all pairs of successive scans; for TARL-Seg and 4D-Seg, we stich every pair of non-overlapping successive windows. This results in our improved versions of the baselines (3DUIS++, TARL-Seg++, 4D-Seg++).

The process is as follows. Given the last scan ℓ and the new scan f, registered in a common reference frame, we compute the convex hulls of all predicted instances in f and in ℓ. For each hull in f, we compute the IoU with all hulls in ℓ using a Monte Carlo method, and associate the two instances if the IoU is greater than 0.5.

This process is extremely slow, and to speed it up, we decimate the instances to at most 200 points before the computation of the convex hull, estimate the IoU using 1000 samples, and stop the computation at the first hull found greater than 0.5. An interesting observation is that this criterion is not strictly unique, meaning that multiple instances in f can be associated to a single instance in ℓ. This alone explains why those improved baselines performs usually poorer in scan-wise metrics.

Method	Road removal	Clustering	Temporal	Voxels
SegContrast	RANSAC	DBSCAN	✗	✗
TARL	Patchwork	HDBSCAN	✓	✗
UNIT	Patchwork++*	HDBSCAN	✓	✓

*denotes that UNIT uses Patchwork when comparing to TARL

Table 6. **Comparison of the main ingredients in the clustering process.**

D. Experiments

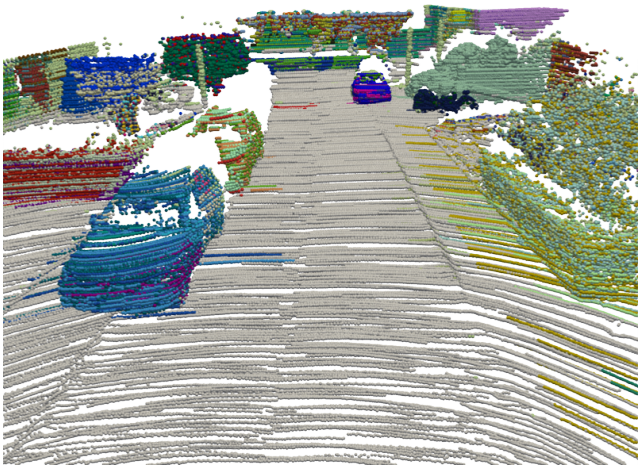
D.1. Datasets

SemanticKITTI [3] and PandaSet-GT [38] are described in the main paper. We give here an overview of nuScenes [5].

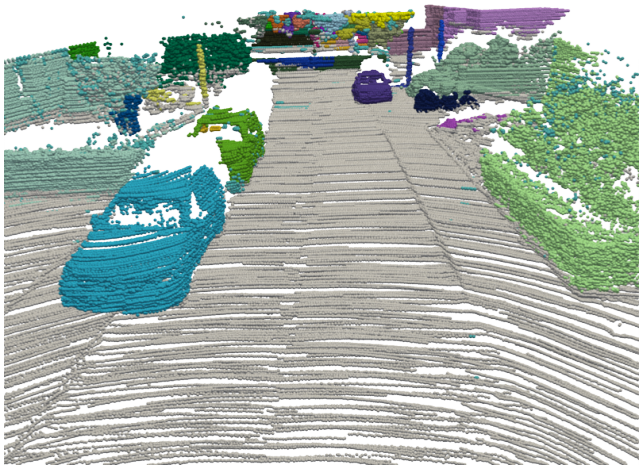
nuScenes is a Lidar dataset acquired in Boston and Singapore, with a 32-beam Lidar capturing scans at 20 Hz. We use the official train/val split with 700 sequences for (unsupervised) training and 150 sequences for validation. We leverage all the lidar scans acquired at 20 Hz during training and inference, for all methods. However, all metrics are computed on a subset of the validation scans as the point-wise annotations are provided at 2 Hz only. For evaluation, we use the official panoptic annotations provided for the instances of: barrier, bicycle, bus, car, construction vehicle, motorcycle, pedestrian, traffic cone, trailer and truck.

D.2. Qualitative results: more visualizations

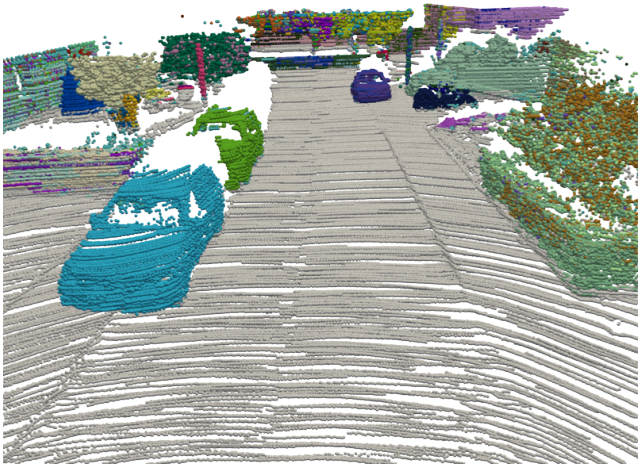
For qualitative inspection, we provide in the following figures more visualizations of the segments of 3DUIS++, TARL-Seg 4D-Seg, UNIT and the ground truth, both on SemanticKITTI and PandaSet-GT.



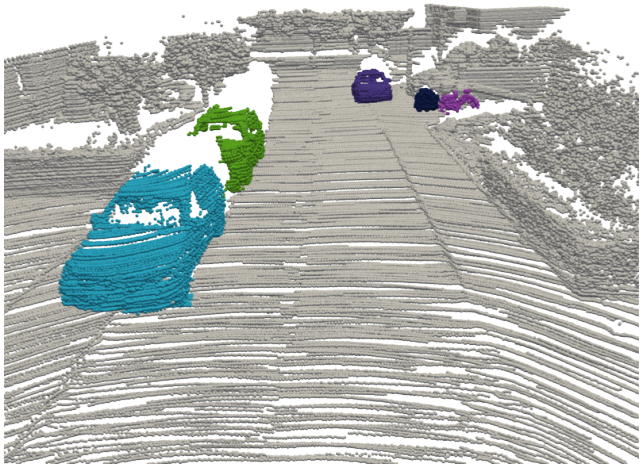
3DUIS++: offline stitching (‘++’) of online 3DUIS single-scan segments



4D-Seg: our offline 4D segments computed on aggregated scans



UNIT (trained on 4D-Seg): overlay of our successive online single-scan segments

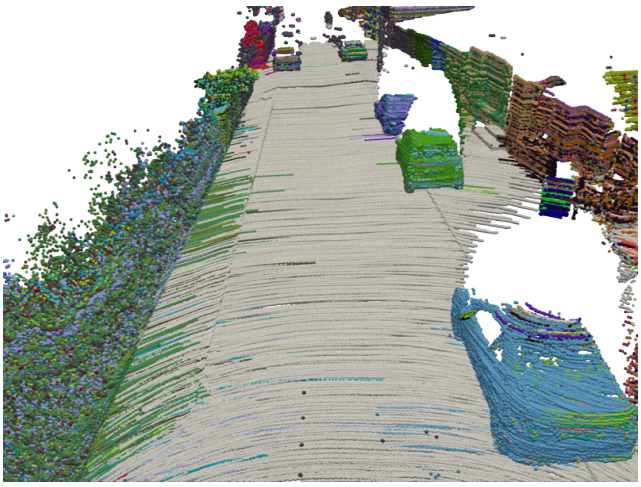


GT: overlay of ground-truth single-scan segments

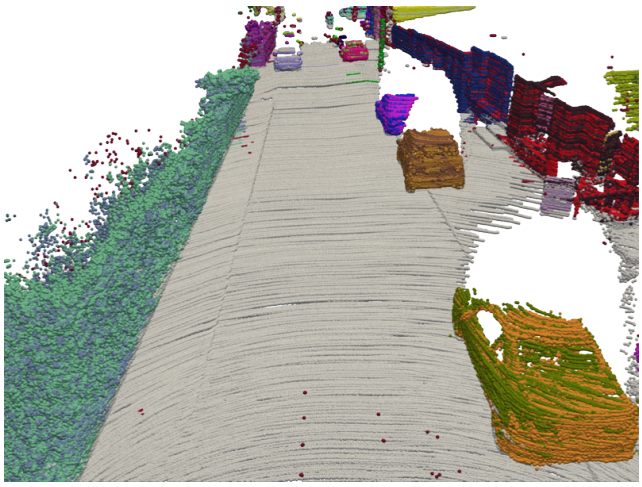
Figure 5. Visualization of object instances across time, obtained on a sample scene of SemanticKITTI. Different instances are assigned colors that are random but (tentatively) consistent over time, forming segments in offline aggregated scans or overlaid online scans.

In this sample, which features a static scene, 4D-Seg is significantly cleaner than 3DUIS++ where single objects, such as the foreground car, have points belonging to several instances and leaking into the ground. UNIT is online (it inputs and outputs one scan at a time) while 4D-Seg is offline (operating on an aggregation of scans), and is here slightly better than 4D-Seg on the cars, but not as good on “stuff” such as vegetation or walls.

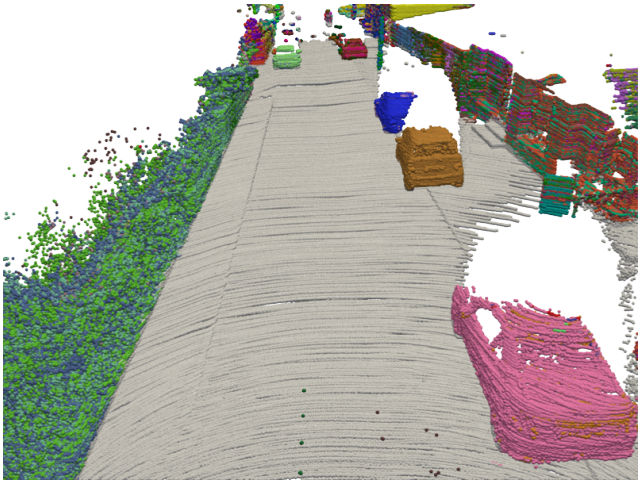
Please note that we (4D-Seg and UNIT) obtain more labels than in the ground truth as our class-agnostic segmentation include all objects and stuff, such as trees or buildings, while the ground truth is restricted to a few selected object classes.



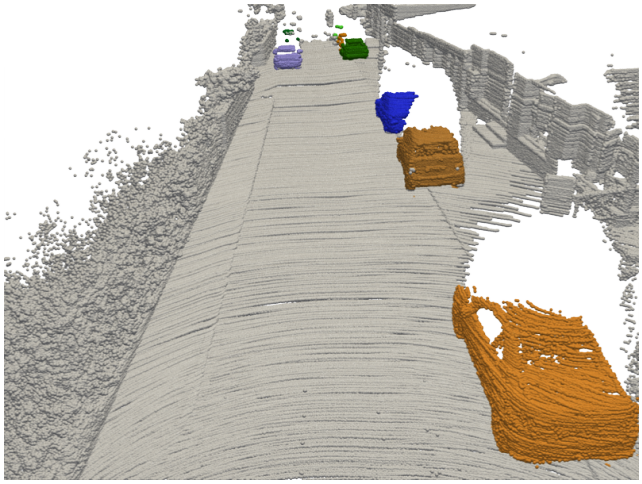
3DUIS++: offline stitching (‘++’) of online 3DUIS single-scan segments



4D-Seg: our offline 4D segments computed on aggregated scans



UNIT (trained on 4D-Seg): overlay of our successive online single-scan segments

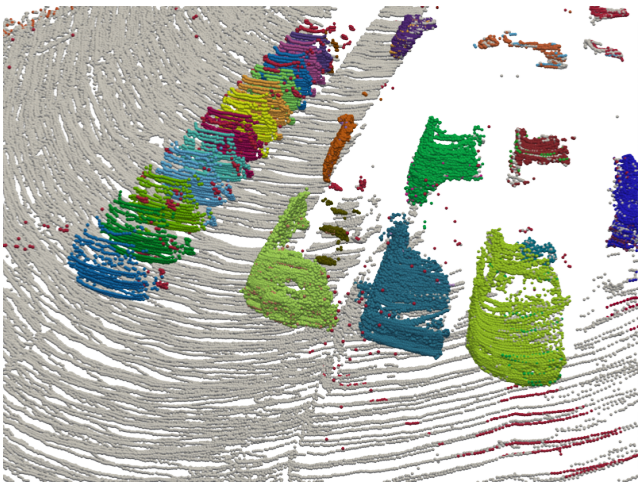


GT: overlay of ground-truth single-scan segments

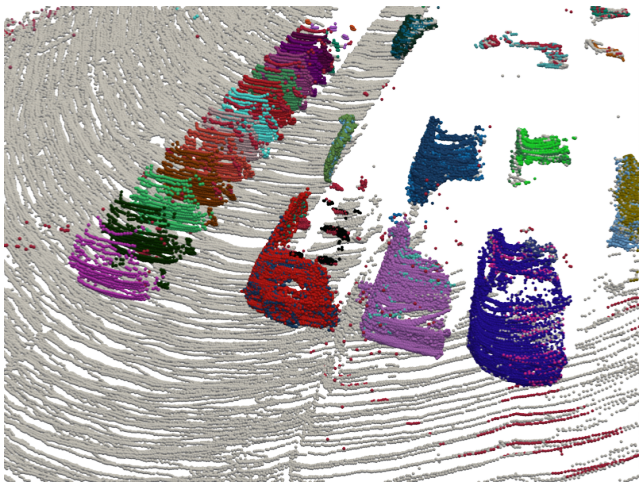
Figure 6. Visualization of object instances across time, obtained on a sample scene of SemanticKITTI. Different instances are assigned colors that are random but (tentatively) consistent over time, forming segments in offline aggregated scans or overlaid online scans.

In this sample, which features a static scene, 4D-Seg is not perfect but still better than 3DUIS++: there is no object segment leaking onto the ground, and the objects are assigned a smaller number of different IDs. For instance, the foreground car in 3DUIS++ is made of 6 different IDs while only 2 are used in 4D-Seg. It is all the more so when the objects are further away, e.g., concerning the cars at the end of the street. The training of UNIT on 4D-Seg, while it is harder because it works online on single scans, rather than offline on aggregated scans as does 4D-Seg, regularizes the segmentation. For instance, the foreground car with UNIT is “now” almost exclusively made of a single ID. Even further away objects benefit from that regularization, e.g., the cars at the end of the street. The instance ground truth shown here was obtained in a process detailed in Appendix A.3

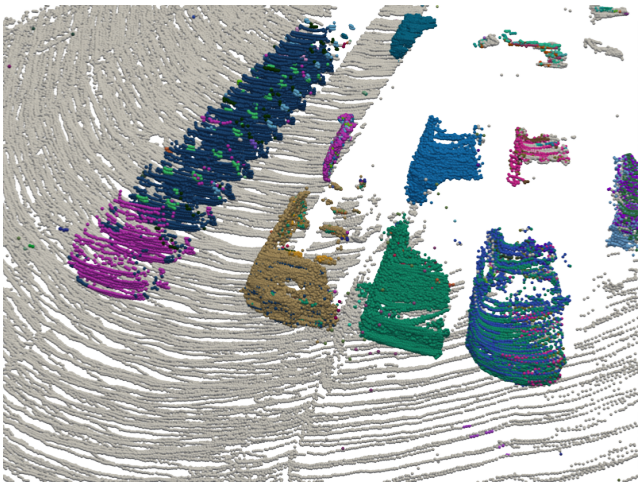
Please note that we (4D-Seg and UNIT) obtain more labels than in the ground truth as our class-agnostic segmentation include all objects and stuff, such as trees or buildings, while the ground truth is restricted to a few selected object classes.



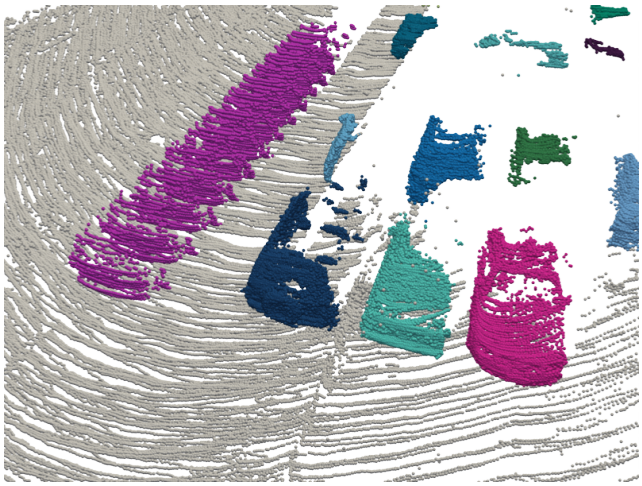
TARL-Seg: offline 4D segments from TARL computed on aggregated scans



4D-Seg: our offline 4D segments computed on aggregated scans



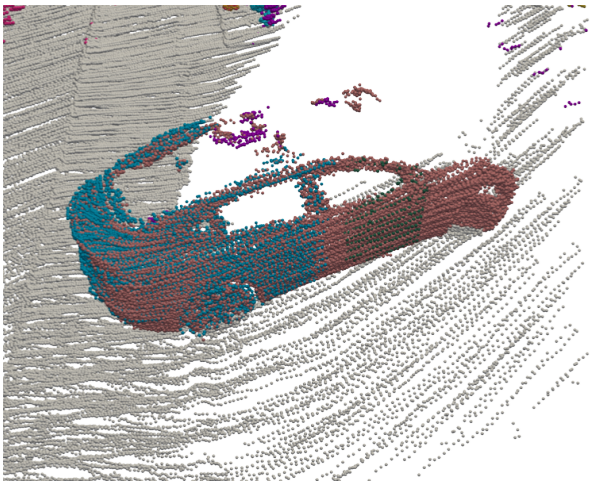
UNIT (trained on 4D-Seg): overlay of our successive online single-scan segments



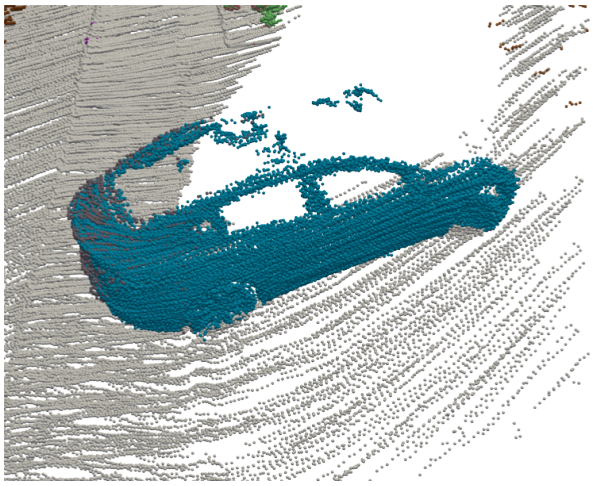
GT: overlay of ground-truth single-scan segments

Figure 7. Visualization of object instances across time, obtained on a sample scene of PandaSet-GT. Different instances are assigned colors that are random but (tentatively) consistent over time, forming segments in offline aggregated scans or overlaid online scans.

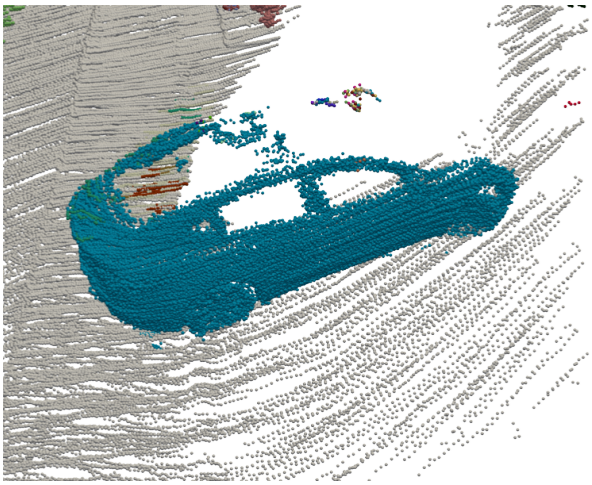
In this sample, which features a dynamic scene, both TARL-Seg and 4D-Seg fail to catch the moving vehicle as a single object. On static objects (stopped cars), TARL-Seg is a bit better than 4D-Seg. Yet, for most such objects, 4D-Seg catches with a single ID the majority of the object points. It provides enough regularization, after training UNIT (which is online) on offline 4D-Seg data, so that, for all static cars, points of an object instance mostly get a single ID. The moving car is still split into several IDs, but considerably less than in TARL-Seg or 4D-Seg. In fact, it seems that with UNIT, the car is more or less consistently tracked over time, but is split into different parts.



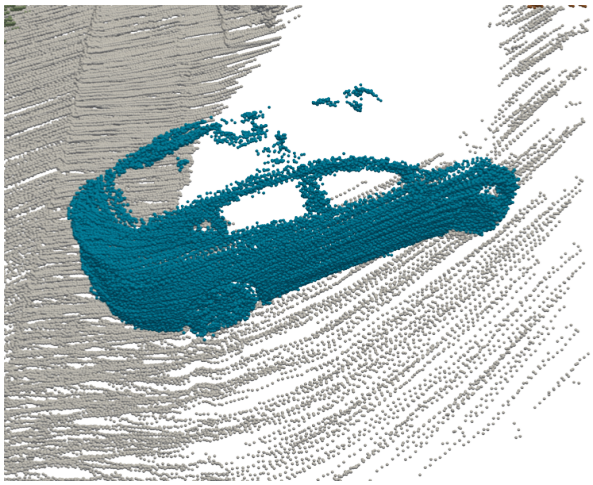
TARL-Seg: offline 4D segments from TARL computed on aggregated scans



4D-Seg: our offline 4D segments computed on aggregated scans



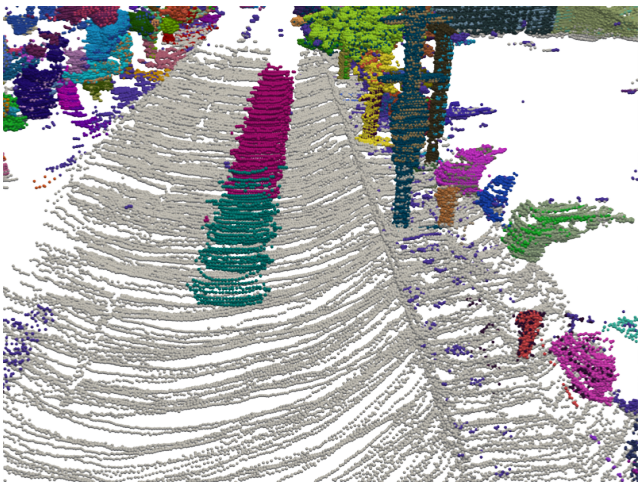
UNIT (trained on 4D-Seg): overlay of our successive online single-scan segments



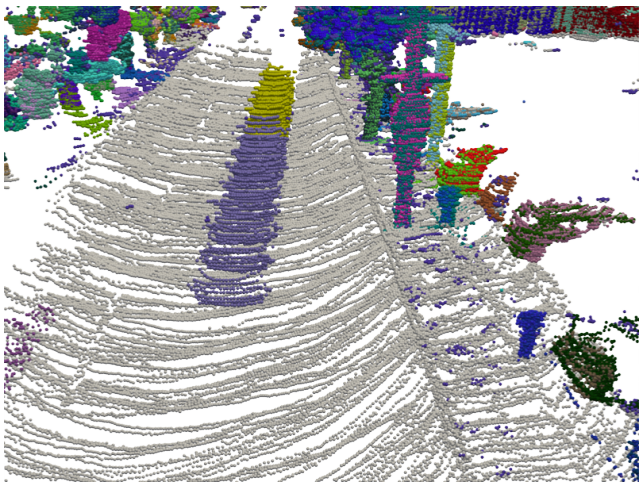
GT: overlay of ground-truth single-scan segments

Figure 8. Visualization of object instances across time, obtained on a sample scene of PandaSet-GT. Different instances are assigned colors that are random but (tentatively) consistent over time, forming segments in offline aggregated scans or overlaid online scans.

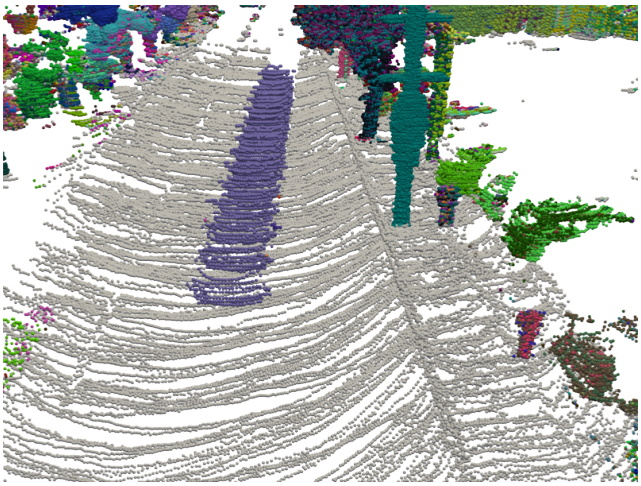
In this sample, which features a parked car, UNIT shows the highest precision of instance segmentation, with 4D-Seg showing very close results albeit. TARL-Seg shows the car instance divided into multiple IDs, showing the limits of its small context window.



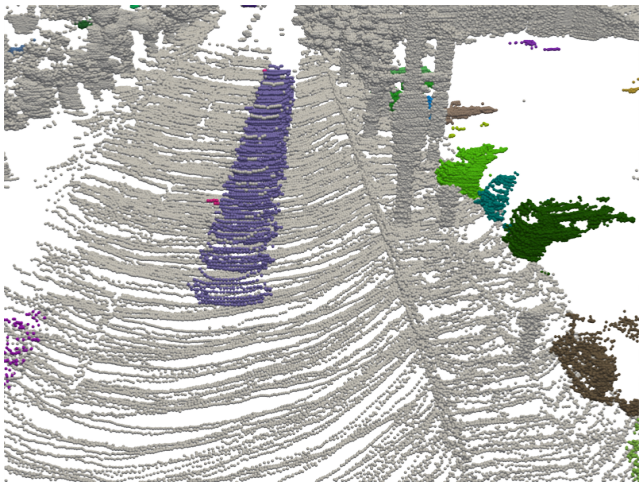
TARL-Seg: offline 4D segments from TARL computed on aggregated scans



4D-Seg: our offline 4D segments computed on aggregated scans



UNIT (trained on 4D-Seg): overlay of our successive online single-scan segments



GT: overlay of ground-truth single-scan segments

Figure 9. Visualization of object instances across time, obtained on a sample scene of PandaSet-GT (cf. Fig. 1). Different instances are assigned colors that are random but (tentatively) consistent over time, forming segments in offline aggregated scans or overlaid online scans.

In this sample, which features a dynamic scene, TARL-Seg and 4D-Seg have a similar qualitative performance: the background stuff and objects are a bit noisy while the moving car is segmented with two IDs instead of one. After training UNIT (which is online) on offline 4D-Seg data, a regularization operates and the moving car is now assigned a single ID over time, apart from just a few noisy points.

Please note that we (4D-Seg and UNIT) obtain more labels than in the ground truth as our class-agnostic segmentation include all objects and stuff, such as trees or buildings, while the ground truth is restricted to a few selected object classes.