— Technical Report —
Contrastive Instance Association for 4D Panoptic Segmentation

Rodrigo Marcuzzi Lucas Nunes Louis Wiesmann Ignacio Vizzo
Jens Behley Cyrill Stachniss
University of Bonn

Abstract

We propose a novel approach that builds on top of an arbitrary single-scan panoptic segmentation network and extends it to the temporal domain by associating instances across time. We propose a contrastive aggregation network that leverages the point-wise features from a panoptic network. It generates an embedding space in which encodings of the same instance at different timesteps lie close together and far from encodings belonging to other instances. The training is inspired by contrastive learning techniques for self-supervised metric-learning. Our association module combines appearance and motion cues to associate instances across scans, allowing us to perform temporal perception. We evaluate our proposed method on the SemanticKITTI benchmark[1] *and achieve state-of-the-art results even without relying on pose information. We plan to publish the source code of our approach.*

1. Our Approach

Fig. 1 gives an overview of our approach. The first element is a frozen panoptic segmentation network to obtain semantic predictions, instance predictions, and point-wise features. Given a point cloud $\mathcal{P} = \{\boldsymbol{p}_1^C, \ldots, \boldsymbol{p}_N^C\}$ with point coordinates $\boldsymbol{p}_i^C \in \mathbb{R}^3$, we apply the backbone B and obtain point-wise semantic classes $\mathcal{P}^S = \{p_1^S, \ldots, p_N^S\}$, where $p_i^S \in \{1, \ldots, n_{\text{classes}}\}$ and point-wise instance IDs $\mathcal{P}^I = \{p_1^I, \ldots, p_N^I\}$ with $p_i^I \in \mathbb{N}$ from the task specific heads. From the feature extractor, we obtain the point-wise features $\mathcal{P}^F = \{\mathbf{p}_1^F, \ldots, \mathbf{p}_N^F\}$, where $\mathbf{p}_i^F \in \mathbb{R}^{D_B}$ with feature dimension D_B. We use the ID of each point to select for each instance $j \in \{1, \ldots, M\}$, its points $I_j^P = \{p_i^C \in \mathcal{P} \mid p_i^I = j\}$ and its point-wise features I_j^F, which we use as input of our contrastive aggregation network.

1.1. Contrastive Aggregation Network

Our contrastive aggregation network (CA-Net) generates consistent appearance features over time for instance associations. Given the input points I_j^P and point-wise features I_j^F for all M instances in the current point cloud $\mathcal{P}$, the output of the network are instance-wise features $I^F = \{\mathbf{f}_1, \ldots, \mathbf{f}_M\}, \mathbf{f}_j \in \mathbb{R}^{D_A}$ with feature dimension D_A, i.e., a single feature vector for each instance in the scan.

To learn from the features of the backbone as well as from the shape of the instances, we first apply three convolutions. Then, a pooling layer computes a single instance feature from all the features belonging to the instance points I_j^P. This is followed by two linear blocks and a projection head to obtain the final instance-wise feature $\mathbf{f_j}$.

1.2. Association Module

After computing instance-wise embeddings I^F using our CA-Net, we maintain consistent instance-wise IDs by associating the instances $\{I_1 \ldots I_M\}$ in the current scan with the corresponding instances $\{I_1 \ldots I_K\}$ in the previous ones. We compute a cost matrix $\mathbf{C} \in \mathbb{R}^{M \times K}$ between all M instances in the current scan and all K instances identified in previous scans, by means of the similarity between their features and determine the associations using the Hungarian method [11].

To handle new targets, we add detected objects in the current frame only if the feature similarity with any of the already active instances is lower than a threshold T_{new}. For re-identification of objects, we store the deactivated trajectories for a fixed number of scans n_{old} and compare with the deactivated targets. We re-identify objects if the similarity is higher than a threshold T_{old}.

To add information about the movement of the targets, we use a constant velocity motion model independent of the sensor ego-motion. Each entry of $\mathbf{C}$ is an association cost between the new instance m and the previous instance k:

$$\mathbf{C}_{m,k} = \alpha_f \cdot (1 - sim(\mathbf{f}_m, \mathbf{f}_k)) + \alpha_d \cdot \|\mathbf{c}_m, \mathbf{c}_k\|, \quad (1)$$

where $\mathbf{f}_m$ is the feature and $\mathbf{c}_m \in \mathbb{R}^3$ the center coordinates of instance m, $sim(\cdot)$ is the cosine similarity function $sim(\boldsymbol{f}_m, \boldsymbol{f}_k) = \boldsymbol{f}_m^\top \boldsymbol{f}_k / \|\boldsymbol{f}_m, \boldsymbol{f}_k\|$, and α_f, $\alpha_d \in \mathbb{R}$ are importance weights for the individual feature and distance costs. For re-identification, the motion model is applied continuously also to the deactivated targets (not associated with any other instance) to estimate their position in case of occlussions or missing detections.

[1] See https://competitions.codalab.org/competitions/26369#results with username `rmarcuzzi`.

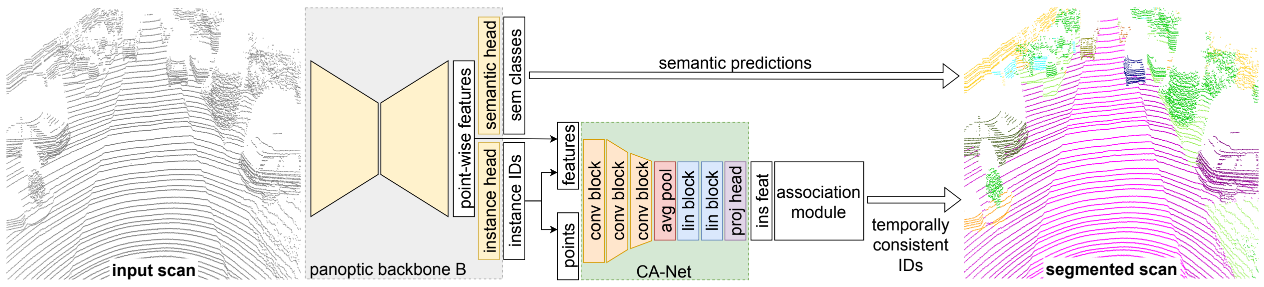

Figure 1: Overview of our method. Given the current 3D LiDAR scan, we use a panoptic segmentation backbone B to obtain semantic classes, instance IDs, and features for each point. We select the points and features for each instance using the IDs and input them into the CA-Net to obtain instance-wise features. These features are used to perform instance associations using our association module to assign temporally consistent instance IDs. Combining this with the semantic predictions, we obtain 4D panoptic segmentation.

1.3. Input Features

The point-wise features $\mathcal{P}^F$ from the panoptic segmentation backbone are used by the task-specific heads to perform segmentation of the points. They are similar for instances belonging to the same semantic class and thus cannot be directly used to perform instance associations for objects of the same semantic class.

Inspired by the recent findings in natural language processing [13], we seek to enhance the point-wise features with spatial knowledge using positional encodings . We compute for each instance point $\mathbf{p}_l \in I_j^P$ a fixed positional encoding $\mathbf{e}_l \in \mathbb{R}^{D_B}$. We follow [9] and use the point coordinates $\mathbf{p}_l^c \in \mathbb{R}^3$ to generate an encoding with the same dimension as the point-wise features and add them together.

Due to their similarity, using the point-wise features as the only input to our CA-Net does not provide enough information to learn distinct instance-wise features to do the associations. We add positional encodings as extra information to exploit the spatial relations between the instances and create more distinct features.

1.4. Instance Point Extraction

As the input of the CA-Net, we select from all the points and features (P, P^F) in the current point cloud $\mathcal{P}$, the ones belonging to the different instances (I^P, I^F). During training, we rely on the instance labels $\hat{\mathcal{P}}^I$ to group the points into instances and select their corresponding features. At inference time the instance points are selected using the predictions P^I from the backbone. These predictions are not perfect due to problems like wrong semantic predictions or errors in the clustering algorithm used to separate instances in the backbone after the instance head. The inputs for the CA-Net are then different at train and test time.

We can circumvent this by applying augmentations to the point instances to emulate what may happen during inference.

The problem of splitted instances is due to the imperfect clustering applied in the backbone to obtain the instance predictions. It can wrongly cluster the points into instances, dividing one instance into two smaller ones, as shown in Fig. 2. We use the so-called *split* augmentation to separate the points on each side of a plane to split the instance. We sample a random normal vector $\mathbf{n} \in \mathbb{R}^3$ and a random instance point $\mathbf{p}_l \in I_j^P$ to generate a randomly oriented plane and compute the point to plane distance $d = \mathbf{n}^\top \mathbf{p}_l$. The remaining points after the augmentation are the query points $\mathbf{q}$ which lie in the half-space of the normal:

$$I_{aug} = \left\{ \mathbf{q}_l \in I_j^P \mid \mathbf{n}^\top \mathbf{q}_l^C - d > 0 \right\}, \qquad (2)$$

which leads to splitted instances.

The incomplete instances are caused by the wrong semantic class predicted for some instance points at the borders. Their semantic class assigns them to the background, and they are not considered as part of the instance, as can be seen in Fig. 2. We use the so-called *contour* augmentation to discard points in the contour of the instances. We first normalize the point coordinates to the range $[-1, 1]$, generate a random maximum coordinate value $\gamma \in \mathbb{R}$ and only keep the points with smaller absolute coordinates, i.e.,

$$I_{aug} = \left\{ \mathbf{p}_l \in I_j^P \mid |x_l| < \gamma \wedge |y_l| < \gamma \wedge |z_l| < \gamma \right\}, \quad (3)$$

where x_l, y_l, z_l are the coordinate components of l^{th} point.

We illustrate our augmentations applied to instances in Fig. 2. As an extra augmentation, we randomly drop cuboids of points belonging to the instance [14].

1.5. Pose Information

Both, the positional encoding and the constant velocity motion model rely on the positions of the instances in the current scan, i.e., positions in a local coordinates frame. As these are local coordinates, the positions are not consistent for scans at different timesteps and the ego-motion of the sensor must be compensated.

By adding the sensor pose estimates using a SLAM approach [3], we can improve the instance associations by leveraging this extra information. We use the pose estimates to transform the positions of previously observed objects

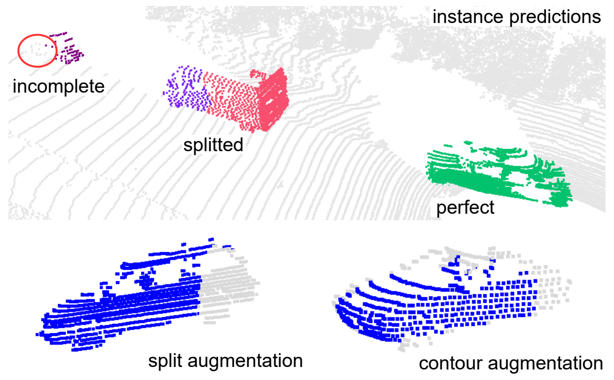

Figure 2: Problems in the instance predictions using the backbone. Incomplete, splitted and perfect instance (top). Proposed *split* augmentation (left) and *countour* augmentation (right). The gray points are discarded.

into the current local coordinates frame. This way, objects have consistent positions over time and the constant velocity motion model only predicts the motion of the other objects as the ego-motion is compensated. We recompute the features of the previously observed instances by updating their positional encoding using the consistent positions in the current coordinate frame.

1.6. Contrastive Training

For the multi-object tracking problem, we use the instance head to get the predicted instances $\{I_1, \ldots, I_M\}$ in the current point cloud $\mathcal{P}$ and perform tracking by associating them with previous instances across time.

Several works use metric learning to learn instance representations by comparing the embedding of one anchor object with one or a few positive and negative samples. At the same time, contrastive learning is used in self-supervised representation learning [12] to train a network, which is later fine-tuned on a downstream task. The main idea of those approaches is to use data augmentation to generate two versions of one anchor sample and train the network to learn to pull together the positive samples in the embedding space and push them apart from many negative samples.

Given a batch of $\mathcal{B}$ samples, the loss function seeks to discriminate between the positive pairs (augmented versions of the anchor) and the negatives (augmented versions of different anchors). In the self-supervised contrastive loss InfoNCE [12], an encoder is used to obtain the feature vectors $\mathbf{z}_j$ for each augmented sample j. Let $r(j)$ be the index of the other augmented sample from the same anchor j and $A(j)$ the set of all indices in the batch except j. Then, the loss function takes the form:

$$\mathcal{L}_{self} = -\sum_{j \in \mathcal{B}} \log \frac{\exp\left(\mathbf{z}_j^\top \mathbf{z}_{r(j)} / \tau\right)}{\sum\limits_{a \in A(j)} \exp\left(\mathbf{z}_j^\top \mathbf{z}_a / \tau\right)}, \tag{4}$$

where τ is a temperature parameter.

In our setup, the samples are all instances in the batch and their corresponding feature vectors $\{\mathbf{f}_1, \ldots, \mathbf{f}_b\}$ are computed using our CA-Net. We want to enforce that features of the same instance are consistent. The appearance of the instances changes over time as the viewpoint changes due to the instance and the sensor's motion. As the samples for the same instance are different across time but depict the same object, we can use this as an inherent augmentation. To obtain positive samples, we select the same instance in different scans over time instead of using one instance as an anchor and generating augmented versions of it.

The appearance can, however, change significantly in scans temporally far from each other. We define a temporal window of scans $\Delta \in \mathbb{N}$, in which we consider instances similar enough to perform the associations and from which the positive samples are drawn. As Δ is the number of scans from which we extract instances, it is the maximum number of positive samples to consider in the loss function. Using the labels, we select a set of positive samples $P(j)$ considering the instances with the same ID in consecutive scans in the temporal window. As negative samples, we use all other instances in the batch. To learn from many positive in addition to the many negative samples, we use the supervised contrastive loss [10]:

$$\mathcal{L}_{sup} = -\sum_{j \in \mathcal{B}} \frac{1}{|P(j)|} \sum_{p \in P(j)} \log \frac{\exp\left(\mathbf{f}_j^\top \mathbf{f}_p / \tau\right)}{\sum\limits_{a \in A(j)} \exp\left(\mathbf{f}_j^\top \mathbf{f}_a / \tau\right)}, \tag{5}$$

where $\mathbf{f}_j, \mathbf{f}_p, \mathbf{f}_a \in \mathbb{R}^{D_A}$ are respectively the feature for the anchor instance, the feature for the positive samples and, the feature for all the other instances in the batch.

1.7. Implementation Details

We build on top of DS-Net [7] as panoptic segmentation backbone. The feature extractor provides point-wise feature vectors $\mathbf{p}_i^F \in \mathbb{R}^{D_B}, D_B = 128$. For the instance clustering, we use mean shift [5].

For our CA-Net, the convolutional blocks are 3D sparse convolutions [4], followed by a batch normalization (BN) layer [8] and leaky ReLU as activation layer. The sparse linear blocks consist of a linear layer followed by a BN layer and a leaky ReLU as activation layer. The final projection head is a linear layer that projects the instance-wise embeddings into a feature space of dimension $D_A = 1024$.

We keep deactivated targets for $n_{old} = 8$ frames and use different thresholds to handle new targets and re-identification for the appearance and the motion model. For the feature similarity, we use $T_{new_f} = T_{old_f} = 0.7$ and for the center distance, we use $T_{new_d} = T_{old_d} = 2$. The weights for the feature cost and the center distance in Eq. (1) are $\alpha_f = 0.4$ and $\alpha_d = 0.7$. In the loss function, see Eq. (5) we use $\tau = 0.1$. At each step, the positive and negative samples are selected from a sequence of scans of random

Method	LSTQ	S_{assoc}	S_{cls}	IoU^{St}	IoU^{Th}
4DPLS[1]	56.89	56.36	57.43	66.86	51.64
Ours (without pose estimates)	60.04	59.49	60.60	66.88	51.98
Ours (with pose estimates[3])	63.11	65.71	60.60*	66.88*	51.98*

Table 1: 4D panoptic segmentation on SemanticKITTI test set. Numbers with * denote the same segmentation results. Adding pose information to our single-scan panoptic backbone does not influence the segmentation performance.

#	feature	encoding	motion model	poses	S_{assoc}
A	✓				59.9
B	✓	✓			70.4
C	✓	✓		✓	71.2
D			✓		68.0
E	✓	✓	✓		71.7
F	✓	✓	✓	✓	72.9

Table 2: Influence of the different components of the approach in S_{assoc} on SemanticKITTI validation set.

length $\Delta \in [2, 5]$. This way, at each step, the loss considers samples from sequences of variable length, which provides a different number of instances.

2. Experimental Evaluation

We evaluate our method on SemanticKITTI [2], which consist of 22 sequences from the KITTI odometry dataset [6]. Sequences 00 to 10 are used for training, leaving sequence 08 as validation set and the test set consists of sequences 11 to 21. To evaluate our performance, we use the LiDAR Segmentation and Tracking Quality (LTSQ) metric $LSTQ = \sqrt{S_{\text{cls}} \times S_{\text{assoc}}}$ [1]. It consists of two terms, the classification score S_{cls} and the association score S_{assoc}. As we use single-scan predictions of frozen panoptic segmentation backbone, our results do not change the segmentation results represented by the S_{cls} term. Thus, we focus instead on the S_{assoc} term to evaluate the quality of our associations.

2.1. 4D Panoptic Segmentation Results

We outperform all previous methods *without* relying on the pose estimations from a SLAM approach. See Tab. 1. Both experiments using our method show the same segmentation performance S_{cls}, IoU^{St}, and IoU^{Th} since we use the semantic predictions from the same single-scan backbone and the pose information does not modify semantic segmentation results of *a single scan*. More detailed results are shown in Tab. 1.

2.2. Ablation: Components

In this section, we illustrate the influence of the different components of our approach on the S_{assoc}. Tab. 2 shows the different performances on the validation set.

References

[1] M. Aygun, A. Osep, M. Weber, M. Maximov, C. Stachniss, J. Behley, and L. Leal-Taixe. 4D Panoptic LiDAR Segmentation. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[2] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall. SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*, 2019.

[3] J. Behley and C. Stachniss. Efficient Surfel-Based SLAM using 3D Laser Range Data in Urban Environments. In *Proc. of Robotics: Science and Systems (RSS)*, 2018.

[4] C. Choy, J. Gwak, and S. Savarese. 4D Spatio-Temporal ConvNets: Minkowski Convolutional Neural Networks. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[5] D. Comaniciu and P. Meer. Mean Shift: A Robust Approach Toward Feature Space Analysis. *IEEE Trans. on Pattern Analalysis and Machine Intelligence (TPAMI)*, 24:603–619, 2002.

[6] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[7] F. Hong, H. Zhou, X. Zhu, H. Li, and Z. Liu. LiDAR-Based Panoptic Segmentation via Dynamic Shifting Network. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[8] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint*, abs/1502.03167, 2015.

[9] A. Jaegle, F. Gimeno, A. Brock, A. Zisserman, O. Vinyals, and J. Carreira. Perceiver: General Perception with Iterative Attention. In *Proc. of the Int. Conf. on Machine Learning (ICML)*, 2021.

[10] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, and D. Krishnan. Supervised Contrastive Learning. In *Proc. of the Conference on Neural Information Processing Systems (NeurIPS)*, volume 33, pages 18661–18673, 2020.

[11] H. Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

[12] A. van den Oord, Y. Li, and O. Vinyals. Representation Learning with Contrastive Predictive Coding. *arXiv preprint:1807.03748*, 2019.

[13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, L. Kaiser, and I. Polosukhin. Attention Is All You Need. *Proc. of the Conference on Neural Information Processing Systems (NeurIPS)*, 2017.

[14] Z. Zhang, R. Girdhar, A. Joulin, and I. Misra. Self-Supervised Pretraining of 3D Features on any Point-Cloud. *arXiv preprint:2101.02691*, 2021.