



# OverlapNet: a siamese network for computing LiDAR scan similarity with applications to loop closing and localization

Xieyuanli Chen<sup>1</sup> · Thomas Läbe<sup>1</sup> · Andres Milioto<sup>1</sup> · Timo Röhling<sup>1</sup> · Jens Behley<sup>1</sup> · Cyrill Stachniss<sup>1</sup>

Received: 30 January 2021 / Accepted: 29 June 2021 / Published online: 5 August 2021  
© The Author(s) 2021

## Abstract

Localization and mapping are key capabilities of autonomous systems. In this paper, we propose a modified Siamese network to estimate the similarity between pairs of LiDAR scans recorded by autonomous cars. This can be used to address both, loop closing for SLAM and global localization. Our approach utilizes a deep neural network exploiting different cues generated from LiDAR data. It estimates the similarity between pairs of scans using the concept of image overlap generalized to range images and furthermore provides a relative yaw angle estimate. Based on such predictions, our method is able to detect loop closures in a SLAM system or to globally localize in a given map. For loop closure detection, we use the overlap prediction as the similarity measurement to find loop closure candidates and integrate the candidate selection into an existing SLAM system to improve the mapping performance. For global localization, we propose a novel observation model using the predictions provided by OverlapNet and integrate it into a Monte-Carlo localization framework. We evaluate our approach on multiple datasets collected using different LiDAR scanners in various environments. The experimental results show that our method can effectively detect loop closures surpassing the detection performance of state-of-the-art methods and that it generalizes well to different environments. Furthermore, our method reliably localizes a vehicle in typical urban environments globally using LiDAR data collected in different seasons.

**Keywords** Loop closing · Localization · SLAM

## 1 Introduction

Over the past decades, Light Detection and Ranging (LiDAR) sensors became key components of the sensor suite of autonomous vehicles that allows to perceive and navigate the world. Especially mapping and localization systems can leverage the geometric information provided by LiDAR sensors covering the 360° surroundings of the vehicle. Accurate odometry estimation allows to build locally consistent maps, and a reliable loop closure detection enables SLAM systems to correct accumulated drift and to build globally consistent

maps. These globally consistent maps can then be used to localize the vehicle. Both tasks, the loop closure detection and localization, need to determine the similarity between pairs of laser range scans. The similarity of laser range scans of the same scene should be high regardless of the sensor locations used to capture them, but should be low if the sensor observes different parts of the environment.

In this article, we show that our prior work (Chen et al. 2020a, b) can be formulated as a general approach, which exploits a neural network to estimate the similarity between laser range scans produced by a rotating 3D LiDAR sensor mounted on a wheeled robot or autonomous vehicle to tackle loop closing and global localization as shown in Fig. 1. The proposed network predicts both, a so-called overlap defined on range images corresponding to the similarity between the 3D LiDAR scans and a yaw angle offset between the two scans. The concept of overlap is used in photogrammetry to describe the configuration of image blocks, e.g., of aerial surveys, and we extend this to LiDAR range images. It is a useful tool for estimating the similarity between pairs of LiDAR scans, which can be used to find loop closure candidates and

---

This work has partially been funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy, EXC-2070 - 390732324 - PhenoRob, and by the Chinese Scholarship Committee.

This is one of the several papers published in *Autonomous Robots* comprising the Special Issue on Robotics: Science and Systems 2020.

---

✉ Xieyuanli Chen  
xieyuanli.chen@igg.uni-bonn.de

<sup>1</sup> University of Bonn, Bonn, Germany

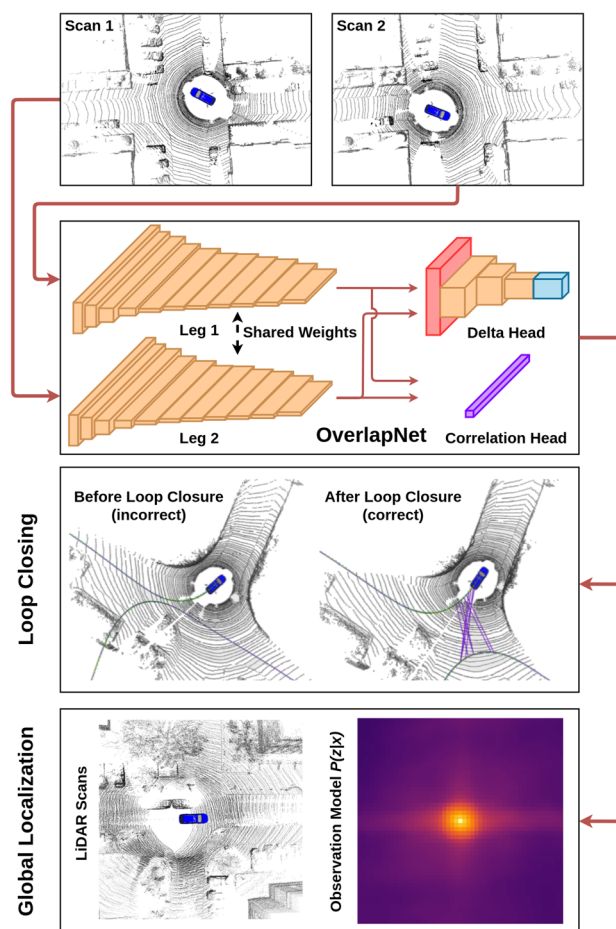
estimate the likelihood of an observation given the sensor position. The yaw estimate can serve as an initial guess for a subsequent application of the iterative closest point (ICP) algorithm (Besl and McKay 1992) to determine the precise relative pose between scans to derive loop closures constraints for pose graph-based optimization in SLAM. During localization, the yaw estimate can be used to estimate the heading likelihood of the current observation.

The main contribution of this article is a deep neural network that exploits different types of information generated from LiDAR scans to provide overlap and relative yaw angle estimates between pairs of 3D scans. This information includes depth, normals, and intensity/remission values. We additionally exploit a probability distribution over semantic classes, which can be computed for each laser beam. Our approach relies on a spherical projection of LiDAR scans rather than the raw point clouds, which makes the proposed OverlapNet comparably lightweight. We furthermore test our method in different applications for autonomous vehicles and mobile robots. We integrate our OverlapNet into a state-of-the-art SLAM system (Behley and Stachniss 2018) for loop closure detection and evaluate its performance also with respect to generalization to different environments. We also integrate our OverlapNet as the observation model in a Monte-Carlo localization (MCL) approach for updating the importance weights of the particles.

To test LiDAR-based loop closing, we train the proposed OverlapNet on parts of the KITTI odometry dataset and evaluate it on unseen data. We thoroughly evaluate our approach, provide ablation studies using different modalities, and test the integrated SLAM system in an online manner. Furthermore, we provide results for the Ford campus dataset, which was recorded using a different sensor setup in a different country and a differently structured environment. The experimental results suggest that our method outperforms other state-of-the-art baseline methods and is able to generalize well to unseen environments.

To test LiDAR-based global localization, a dataset has been collected in different seasons with multiple sequences repeatedly exploring the same crowded urban area using our own car setup. Based on our novel observation model, MCL achieves global localization using 3D LiDAR scans over different seasons with a comparably small number of particles.

In sum, our approach is able to (i) predict the overlap and relative yaw angle between pairs of LiDAR scans by exploiting multiple cues without using relative poses, (ii) combine odometry information with overlap predictions to detect correct loop closure candidates, (iii) improve the overall pose estimation results in a state-of-the-art SLAM system yielding more globally consistent maps, (iv) initialize ICP using the OverlapNet predictions yielding correct scan matching results, (v) build a novel observation model and achieve



**Fig. 1** Overview: We use a siamese network with two LiDAR scans as input. The result of OverlapNet consisting of the overlap percentage and yaw angle offset can be used for loop closing and/or global localization

global localization. The source code of the implementation of our approach is publicly available (see Sect. 6.1).

## 2 Related work

Various sensor modalities have been used in autonomous cars and wheeled robots for loop closing and localization (Thrun et al. 2005; Stachniss et al. 2016). Here, we mainly concentrate on related work addressing 3D LiDAR-based approaches. Since our method address both loop closing and global localization, we discuss the related work in both topics separately and concentrate on LiDAR-based approaches.

### 2.1 Loop closing for SLAM

Loop closing is the problem of correctly identifying that a robot has returned to a previously visited place and is a key component in SLAM systems. For example, Steder et al.

(2011) propose a place recognition system operating on range images generated from 3D LiDAR data that uses a combination of bag-of-words and NARF features (Steder et al. 2010) for relative poses estimation. Röhling et al. (2015) present an efficient method for detecting loop closures through the use of similarity measures on histograms extracted from 3D LiDAR scans of an autonomous ground vehicle and an optimization-based SLAM system. The work by He et al. (2016) presents M2DP, which projects a LiDAR scan into multiple reference planes to generate a descriptor using a density signature of points in each plane. Besides using pure geometric information, there are also methods (Cop et al. 2018; Guo et al. 2019) exploiting the remission information, i.e., how well LiDAR beams are reflected by a surface, to create descriptors for localization and loop closure detection with 3D LiDAR data. Other researchers link LiDAR observations with background knowledge about the environment to create global constraints, similar to a loop closure (Vysotska and Stachniss 2017).

Recently, deep learning-based methods that yield features in an end-to-end fashion have been proposed for loop closing to overcome the need for hand-crafting features. For example, Dubé et al. (2017) investigated an approach that matches segments extracted from a scan to find loop closures via segment-based features. A geometric test via RANSAC is used to verify a potential loop closure identified by the matching procedure. Based on such segments, Cramariuc et al. (2018) train a CNN to extract descriptors from segments and use it to retrieve near-by place candidates. Uy and Lee (2018) propose PointNetVLAD which is a deep network used to generate global descriptors for 3D point clouds. It aggregates multiple LiDAR scans and end-to-end encodes a global descriptor to tackle the retrieval and place recognition task. Yin et al. (2019) develop LocNet, which uses semi-handcrafted features learning based on a siamese network to solve LiDAR-based place recognition. Schaupp et al. (2019) propose a system called OREOS for place recognition. They use a convolutional neural network to extract compact descriptors from LiDAR scans and use the features to retrieve near-by place candidates from a map and to estimate the yaw discrepancy. Most recently, Zaganidis et al. (2019) proposed a normal distributions transform histogram-based loop closure detection method, which is assisted by semantic information. Kong et al. (2020) also use semantic graphs for place recognition for 3D point clouds. Their network is capable of capturing topological and semantic information from the point cloud and also achieves rotational invariance.

Contrary to the above-mentioned methods, our method exploits multiple types of information extracted from 3D LiDAR scans, including depth, normal information, intensity/remission and probabilities of semantic classes generated by a semantic segmentation system developed and

reported by Milioto et al. (2019) and Milioto and Stachniss (2019). Our method furthermore uses a siamese network to learn features and yield predictions end-to-end, which directly provides estimates for overlap and the relative yaw angle between pairs of LiDAR scans. Based on that, our method can be used not only for detecting loop closure candidates but also provides an estimate of the matching quality in terms of the overlap.

## 2.2 Global localization

Besides loop closing, our method can be further used for LiDAR-based localization. Localization is a classical topic in robotics (Dellaert et al. 1999; Thrun et al. 2005). For localization given a map, one often distinguishes between pose tracking and global localization. In pose tracking, an approach needs to locally localize the vehicle from a known pose and the pose is updated over time. In global localization, no pose prior is available and an approach needs to localize the vehicle globally. In this work, we address global localization using 3D LIDAR sensors without assuming any pose prior from GPS or other sensors.

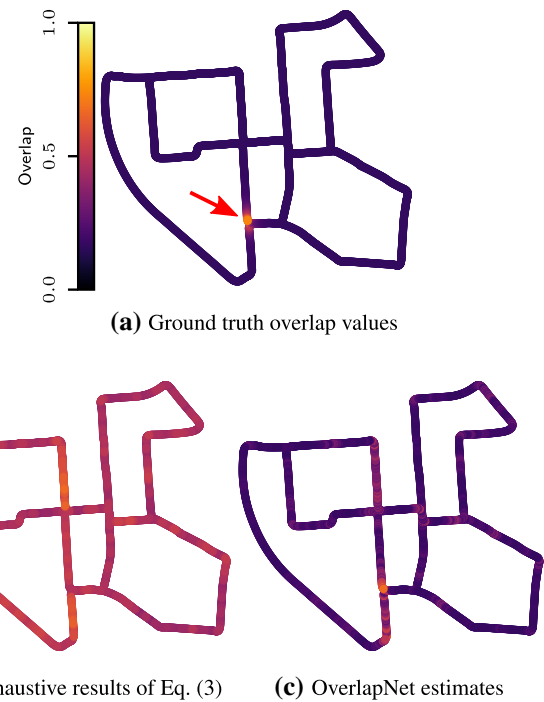
A popular traditional framework for robot localization relies on probabilistic state estimation techniques is Monte-Carlo localization (Dellaert et al. 1999; Fox et al. 1999), which uses a particle filter to estimate the pose of the robot.

In the context of autonomous cars, there are many approaches building and using high-definition (HD) maps for localization. For example, Levinson et al. (2007) utilize GPS, IMU, and LiDAR data to build HD maps for localization. They generate a 2D surface image of ground reflectivity in the infrared spectrum and define an observation model that uses these intensities. Wolcott and Eustice (2015) propose a new scan matching algorithm that leverages Gaussian mixture maps to exploit the structure in the environment. The uncertainty in intensity values has been handled by building a prior HD map. Barsan et al. (2018) use a fully convolutional neural network to perform online-to-map matching for improving the robustness to dynamic objects and eliminating the need for LiDAR intensity calibration. Their approach requires a good GPS prior to achieve good performance. Based on this approach, Wei et al. (2019) proposed a learning-based compression method for HD maps, which compresses the intermediate representations of the neural network while retaining important information for downstream tasks. Mersels and Stachniss (2016) present an efficient chain graph-like pose graph for vehicle localization exploiting graph optimization techniques and different sensing modalities. Based on that work, Wilbers et al. (2019) propose a LiDAR-based localization system performing a combination of local data association between laser scans and HD maps, temporal data association smoothing, and a map matching approach for robustification. To compress the map for online localization,

Wiesmann et al. (2021) propose a deep network to compress and decompress the point cloud maps. Recently, Vizzo et al. (2021) also propose a light-weighted mesh mapping algorithm using Poisson surface reconstruction for LiDAR scans and later using the mesh maps for LiDAR-based localization (Chen et al. 2021).

Other approaches aim at performing LiDAR-based place recognition to initialize localization. For example, Kim et al. (2019) transform point clouds into scan context images and train a CNN based on such images. They generate scan context images for both the current frame and all grid cells of the map and compare them to estimate the current location as the cell presenting the largest score. Yin et al. (2019) propose a siamese network to first generate fingerprints for LiDAR-based place recognition and then use iterative closest points to estimate the metric poses. Cop et al. (2018) propose a descriptor for LiDAR scans based on intensity information. Using this descriptor, they first perform place recognition to find a coarse location of the robot, eliminate inconsistent matches using RANSAC (Fischler and Bolles 1981), and then refine the estimated transformation using iterative closest points.

Recently, several approaches exploiting semantic information for 3D LiDAR localization have been proposed. Zhang et al. (2018) utilize both ground reflectivity features and vertical features for localizing autonomous cars in rainy conditions. Both a histogram filter and a particle filter are integrated to estimate the posterior distributions of the vehicle poses. Using a similar idea, Ma et al. (2019) combine semantic information such as lanes and traffic signs in a Bayesian filtering framework to achieve accurate and robust localization within sparse HD maps. Yan et al. (2019) exploit buildings and intersections information from a LiDAR-based semantic segmentation system to localize on OpenStreetMap data using 4-bit semantic descriptors. Not targeting tiny descriptors but the exploitation of prior knowledge for localization has also been tackled by Silver and Stentz (2011). They use satellite imagery of outdoor scenes and propose a sensor model to link the observation of the robot with the aerial imagery. Schaefer et al. (2019) detect and extract pole landmarks from 3D LiDAR scans for long-term urban vehicle localization. The proposed pole detector considers both the laser ray endpoints and the free space in between the laser sensor and the endpoints, which demonstrates reliable and accurate localization performance. Tinchev et al. (2019) propose a learning-based method to match segments of trees and localize in both urban and natural environments. The approach learns a light feature space representation which can be deployed using only a CPU. In our previous work (Chen et al. 2019), we also exploit semantic information to improve the localization and mapping results by detecting and removing dynamic objects. Sun et al. (2020) use a deep-probabilistic model to accelerate the initialization of the Monte Carlo localization and achieve a fast



**Fig. 2** Overlap estimations of one frame to all others. **a** The red arrow points out the position of the query scan. **b** If we directly use Eq. (3) to estimate the overlap between two LiDAR scans without knowing the accurate relative poses, it is hard to decide which pairs of scans are true loop closures, since most evaluations of Eq. (3) show high values. **c** In contrast, our OverlapNet can predict the overlaps between two LiDAR scans without the relative transformation between the scans such that only the correct location get a high overlap

localization in outdoor campus environments. This hybrid LiDAR-based localization approach integrates a learning-based with a Markov-filter-based method, which makes it possible to effectively and efficiently provide global localization results.

Different to the above discussed methods (Barsan et al. 2018; Ma et al. 2019; Wei et al. 2019), which use GPS as prior for localization, our method only exploits LiDAR information to achieve global localization without using any GPS information. Moreover, our approach uses range scans without explicitly exploiting semantics or extracting landmarks. In contrast to approaches that perform place recognition (Kim et al. 2019; Cop et al. 2018; Yin et al. 2019), our approach relies on convolutional neural networks to predict the overlap between range scans and their yaw angle offset and we exploit this information as an observation model for Monte-Carlo localization.

### 3 OverlapNet

The idea of overlap that we are using here has its origin in the photogrammetry and computer vision community (Hussain



and Bethel 2004). To successfully match two images and to calculate their relative orientation, the images must overlap. This can be quantified by defining the overlap value as the percentage of pixels in the first image, which can successfully be projected back into the second image without occlusion. Note that this measure is not symmetric. If there is a large scale difference of the image pair, e.g., one image shows a wall and the other shows many buildings around that wall, the overlap percentage for the first to the second image can be large and from the second to the first image low. In this paper, we use the idea of overlap for range images and exploit the range information explicitly.

### 3.1 Definition of the overlap between LiDAR scans

We use spherical projections of LiDAR scans as input data, which is often used to speed up computations (Bogoslavskyi and Stachniss 2016; Behley and Stachniss 2018; Chen et al. 2019). We project the point cloud  $\mathcal{P} = \{\mathbf{p}_i\}, i \in \{1, \dots, N\}$  to a so-called vertex map  $\mathcal{V} : \mathbb{R}^2 \mapsto \mathbb{R}^3$ , where each pixel is mapped to the nearest 3D point. Each point  $\mathbf{p}_i = (x, y, z)$  is converted via the function  $\Pi : \mathbb{R}^3 \mapsto \mathbb{R}^2$  to spherical coordinates and finally to image coordinates  $(u, v)$  by

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \frac{1}{2} [1 - \arctan(y, x)\pi^{-1}] w \\ [1 - (\arcsin(zr^{-1}) + f_{\text{up}}) f^{-1}] h \end{pmatrix}, \quad (1)$$

where  $r = \|\mathbf{p}\|_2$  is the range,  $f = f_{\text{up}} + f_{\text{down}}$  is the vertical field-of-view of the sensor, and  $w, h$  are the width and height of the resulting vertex map  $\mathcal{V}$ .

For the LiDAR scans  $\mathcal{P}_1$  and  $\mathcal{P}_2$ , we generate the corresponding vertex maps  $\mathcal{V}_1, \mathcal{V}_2$ . We denote the sensor-centered coordinate frame at time step  $t$  as  $C_t$ . Each pixel in coordinate frame  $C_t$  is associated with the world frame  $W$  by a pose  $\mathbf{T}_{W C_t} \in \mathbb{R}^{4 \times 4}$ . Given the poses  $\mathbf{T}_{W C_1}$  and  $\mathbf{T}_{W C_2}$ , we can reproject scan  $\mathcal{P}_1$  into the other's vertex map  $\mathcal{V}_2$  and generate a reprojected vertex map  $\mathcal{V}'_1$ :

$$\mathcal{V}'_1 = \Pi \left( \mathbf{T}_{W C_1}^{-1} \mathbf{T}_{W C_2} \mathcal{P}_1 \right). \quad (2)$$

We calculate the absolute difference of all corresponding pixels in  $\mathcal{V}'_1$  and  $\mathcal{V}_2$ , considering only those pixels that correspond to valid range readings in both range images. The overlap is then calculated as the percentage of all differences in a certain distance  $\epsilon$  relative to all valid entries, i.e., the overlap of two LiDAR scans  $O_{C_1 C_2}$  is defined as:

$$O_{C_1 C_2} = \frac{\sum_{(u,v)} \mathbb{I} \left\{ \|\mathcal{V}'_1(u, v) - \mathcal{V}_2(u, v)\| \leq \epsilon \right\}}{\min(\text{valid}(\mathcal{V}'_1), \text{valid}(\mathcal{V}_2))}, \quad (3)$$

where  $\mathbb{I}\{a\} = 1$  if  $a$  is true and 0 otherwise. The function  $\text{valid}(\mathcal{V})$  refers to the number of valid pixels in  $\mathcal{V}$ , since

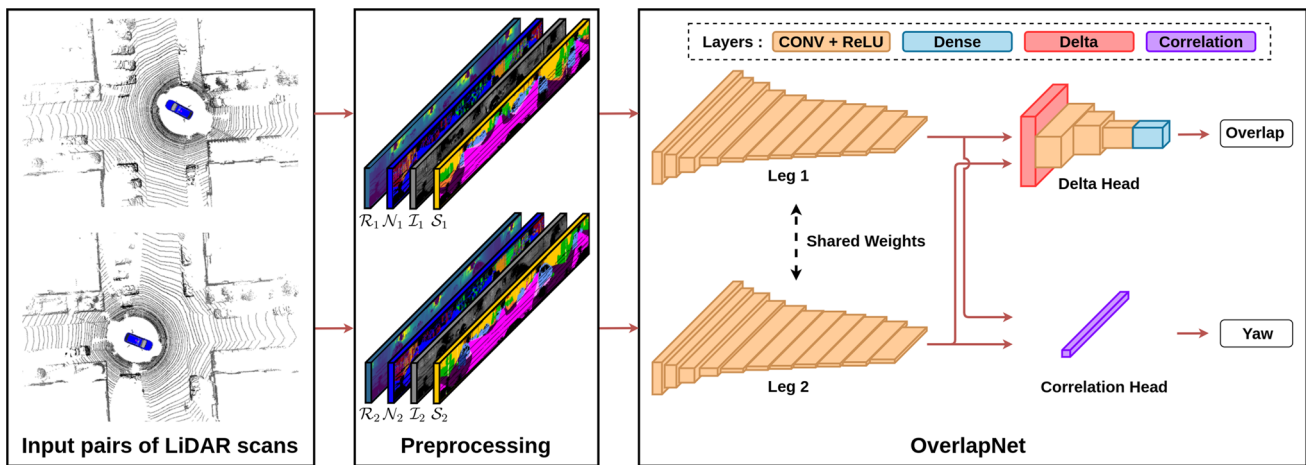
not all pixel might have a valid LiDAR measurement associated after the projection.

We use Eq. (3) only for *creating training data*, i.e., only positive examples of correct loop closures get a non-zero overlap assigned using the relative poses between scans, as shown in Fig. 2. Thus, the proposed approach can learn not only high overlap values for nearby scans, but also to estimate high overlap values for scans that correspond to the same part of the environment and lower overlap values for different parts. During test time, no (ground-truth) poses are available and not required by our proposed approach. When performing loop closure detection for online SLAM, the approximate relative poses computed by the SLAM system before loop closure are not accurate enough to calculate suitable overlaps by using Eq. (3) because of accumulated drift.

To verify that the estimated overlap captures more than just the similarity of the raw scans, we tried directly calculating overlaps using Eq. (3) assuming the relative pose to be the identity and applying different orientations, e.g., every 30 degrees rotation around the vertical axis, and using the maximum over all these overlaps as an estimate. Figure 2 shows the estimated overlaps for all scans using a query scan produced by this method and the result of the estimated overlap for all scans using OverlapNet. We leave out the 100 most recent scans because they will not be loop closure candidates. In the case of the exhaustive approach based on Eq. (3), many scans which are far away get high overlap values, which makes this method unsuitable for loop closing or global localization. Our approach, however, correctly identifies the correct similarity as it produces a highly distinctive peak around the correct location.

### 3.2 Overlap network architecture

A visual overview of our proposed OverlapNet is depicted in Fig. 3. We exploit multiple cues, which can be generated from a single LiDAR scan, including depth, normal, intensity, and semantic class probability information. The depth information is stored in the range map  $\mathcal{R}$ , which defines one input channel. We use neighborhood information of the vertex map to generate a normal map  $\mathcal{N}$ , which gives three channels encoding the normal coordinates. We directly obtain the intensity information, also called remission, from the sensor and represent the intensity information as a one-channel intensity map  $\mathcal{I}$ . The point-wise semantic class probabilities are computed using RangeNet++ by Milioto et al. (2019) and we represent them as a semantic map  $\mathcal{S}$ . The semantic segmentation network delivers probabilities for 20 different classes. For efficiency reasons, we reduce the 20 dimensional RangeNet++ output to a compressed 3 dimensional vector using principal component analysis. All the information is combined as the input of the OverlapNet with the size of  $64 \times 900 \times 8$ .



**Fig. 3** Pipeline overview of our proposed approach. The left-hand side shows the preprocessing of the input data which exploits multiple cues generated from a single LiDAR scan, including range  $\mathcal{R}$ , normal  $\mathcal{N}$ , intensity  $\mathcal{I}$ , and semantic class probability  $\mathcal{S}$  information. The right-

hand side shows the proposed OverlapNet which consists of two legs sharing weights and the two heads use the same pair of feature volumes generated by the two legs. The outputs are the overlap and relative yaw angle between two LiDAR scans

Our proposed OverlapNet is a siamese network architecture (Bromley et al. 1993), which consists of two legs sharing weights and two heads that use the same pair of feature volumes generated by the two legs. The trainable layers are listed in Table 1.

### 3.2.1 Legs

The proposed OverlapNet has two legs, which have the same architecture and share weights. Each leg is a fully convolutional network consisting of 11 convolutional layers. This architecture is quite lightweight, i.e., it only consists of 1.8 million parameters, and generates feature volumes of size  $1 \times 360 \times 128$ .

Note that our range images are cyclic projections and that a change in the yaw angle of the vehicle results in a cyclic column shift of the range image. Thus, the single row in the feature volume can represent a relative yaw angle estimate because a yaw angle rotation results in a pure horizontal shift of the input maps. As the fully convolutional network is translation-equivariant, the feature volume will be shifted horizontally in the same manner. The number of columns of the feature volume defines the resolution of the yaw estimation, which is  $1^\circ$  in the case of our leg architecture.

### 3.2.2 Delta head

The delta head is designed to estimate the overlap between two scans. It consists of a delta layer, three convolutional layers, and one fully connected layer.

The delta layer, which is shown in Fig. 4, computes all possible absolute differences of all pixels. It takes the out-

put feature volumes  $\mathbf{L}^l \in \mathbb{R}^{H \times W \times C}$  from the two legs  $l \in \{0, 1\}$  as input. These are stacked in a tiled tensor  $\mathbf{T}^l \in \mathbb{R}^{HW \times HW \times C}$  as follows:

$$\mathbf{T}^0(iW + j, k, c) = \mathbf{L}^0(i, j, c), \quad (4)$$

$$\mathbf{T}^1(k, iW + j, c) = \mathbf{L}^1(i, j, c), \quad (5)$$

with  $k = \{0, \dots, HW - 1\}$ ,  $i = \{0, \dots, H - 1\}$  and  $j = \{0, \dots, W - 1\}$ .

Note that  $\mathbf{T}^1$  is transposed with respect to  $\mathbf{T}^0$ , as depicted in the middle of Fig. 4. After that, all differences are calculated by element-wise absolute differences between  $\mathbf{T}^0$  and  $\mathbf{T}^1$ .

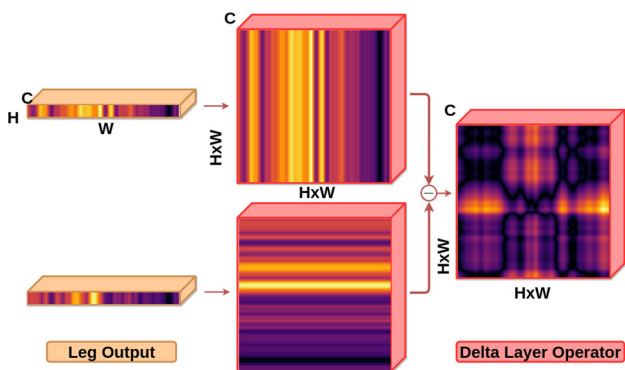
By using the delta layer, we can obtain a representation of the latent difference information, which can be later exploited by the convolutional and fully-connected layers to estimate the overlap. Different overlaps induce different patterns in the output of the delta layer.

### 3.2.3 Correlation head

The correlation head (Nagashima et al. 2007) is designed to estimate the yaw angle between two scans using the feature volumes of the two legs. To perform the cross-correlation, we first pad horizontally one feature volume by copying the same values (as the range images are cyclic projections around the yaw angle). This doubles the size of the feature volume. We then use the other feature volume as a kernel that is shifted over the first feature volume generating a 1D output of size 360. The argmax of these correlation values serves as the estimate of the relative yaw angle of the two input scans with a  $1^\circ$  resolution.

**Table 1** Layers of our network architecture

	Operator	Stride	Filters	Size	Output Shape
Legs	Conv2D	(2, 2)	16	(5, 15)	30 × 443 × 16
	Conv2D	(2, 1)	32	(3, 15)	14 × 429 × 32
	Conv2D	(2, 1)	64	(3, 15)	6 × 415 × 64
	Conv2D	(2, 1)	64	(3, 12)	2 × 404 × 64
	Conv2D	(2, 1)	128	(2, 9)	1 × 396 × 128
	Conv2D	(1, 1)	128	(1, 9)	1 × 388 × 128
	Conv2D	(1, 1)	128	(1, 9)	1 × 380 × 128
	Conv2D	(1, 1)	128	(1, 9)	1 × 372 × 128
	Conv2D	(1, 1)	128	(1, 7)	1 × 366 × 128
	Conv2D	(1, 1)	128	(1, 5)	1 × 362 × 128
	Conv2D	(1, 1)	128	(1, 3)	1 × 360 × 128
Delta Head	Conv2D	(1, 15)	64	(1, 15)	360 × 24 × 64
	Conv2D	(15, 1)	128	(15, 1)	24 × 24 × 128
	Conv2D	(1, 1)	256	(3, 3)	22 × 22 × 256
	Dense	–	–	–	1



**Fig. 4** The delta layer: Computation of pairwise differences is efficiently performed by concatenating the feature volumes and transposition of one concatenated feature volume

### 3.3 Loss function

We train our OverlapNet end-to-end to estimate the overlap and the relative yaw angle between two LiDAR scans at the same time. Typically, to train a neural network one needs a large amount of manually labeled ground truth data. In our case, this is  $(I_1, I_2, Y_O, Y_Y)$ , where  $I_1, I_2$  are two inputs and  $Y_O, Y_Y$  are the ground truth overlaps and the ground truth yaw angles respectively. We are, however, able to generate the input and the ground truth without any manual effort in a fully automated fashion given a dataset with pose information. From given poses (e.g., obtained using a GPS+IMU combination), we can calculate the ground truth overlap and relative yaw angles directly. We denote the legs part network with trainable weights as  $f_L(\cdot)$ , the delta head as  $f_D(\cdot)$  and the correlation head as  $f_C(\cdot)$ .

For training, we combine the loss  $L_O(\cdot)$  for the overlap and the loss  $L_Y(\cdot)$  for the yaw angle using a weight  $\alpha$ :

$$L(I_1, I_2, Y_O, Y_Y) = L_O(I_1, I_2, Y_O) + \alpha L_Y(I_1, I_2, Y_Y). \tag{6}$$

We treat the overlap estimation as a regression problem and use a weighted absolute difference of ground truth  $Y_O$  and network output  $\hat{Y}_O = f_D(f_L(I_1), f_L(I_2))$  as the loss function. For weighting, we use a scaled sigmoid function:

$$L_O(I_1, I_2, Y_O) = \text{sigmoid}\left(s\left(\left|\hat{Y}_O - Y_O\right| + a\right) - b\right), \tag{7}$$

with  $\text{sigmoid}(v) = (1 + \exp(-v))^{-1}$ , the variables  $a, b$  being offsets, and  $s$  being a scaling factor.

For the yaw angle estimation, we use a lightweight representation of the correlation head output, which leads to a one-dimensional vector of size 360. We take the index of the maximum, the argmax, as the estimate of the relative angle in degrees. As the argmax is not differentiable, we cannot treat this as a simple regression problem. The yaw angle estimation, however, can be regarded as a binary classification problem that decides for every entry of the head output whether it is the correct angle or not. Therefore, we use the binary cross-entropy loss given by

$$L_Y(I_1, I_2, Y_Y) = \sum_{i=\{1, \dots, N\}} H\left(Y_Y^i, \hat{Y}_Y^i\right), \tag{8}$$

where  $H(p, q) = -p \log(q) - (1 - p) \log(1 - q)$  is the binary cross entropy and  $N$  is the size of the output 1D vector.  $\hat{Y}_Y = f_C(f_L(I_1), f_L(I_2))$  is the relative yaw angle estimate. Note that we only train the network to estimate

the relative yaw angle of a pair of scans with overlap larger than 30%, since it is more uncertain and difficult to estimate the relative yaw angle if the pair of scans are less overlapping which is explained more detailed in Sect. 6.2.3. More uncertain estimations also decrease the accuracy of pose estimation building on top of OverlapNet and the corresponding experimental results can be found in Sect. 6.3.2.

## 4 OverlapNet for loop closing and SLAM

For loop closing, a threshold on the overlap percentage can be used to decide whether two LiDAR scans are taken from the same place. For finding loop closure candidates, this measure maybe even better than the commonly used distance between the recorded positions of a pair of scans, since the positions might be affected by drift and therefore unreliable. Furthermore, the overlap takes the scene into account, e.g., occlusions between the two scans, and is a direct measure for the number of corresponding points, which can be exploited by ICP that most SLAM systems are employing. The overlap predictions are independent of the relative poses and can be therefore used to find loop closures without knowing the correct relative pose between scans.

### 4.1 SLAM pipeline

We use the surfel-based mapping system called SuMa proposed by Behley and Stachniss (2018) as our SLAM pipeline and integrate OverlapNet in SuMa replacing its original heuristic loop closure detection method. We only summarize here the key steps of SuMa relevant to our approach and refer for more details to the original paper (Behley and Stachniss 2018).

SuMa uses the same vertex map  $\mathcal{V}_D$  and normal map  $\mathcal{N}_D$  as described in Sect. 3.1. Furthermore, SuMa uses projective ICP with respect to a rendered map view  $\mathcal{V}_M$  and  $\mathcal{N}_M$  at timestep  $t - 1$ , the pose update  $\mathbf{T}_{C_{t-1}C_t}$  and consequently  $\mathbf{T}_{WC_t}$  by chaining all pose increments. Therefore, each vertex  $\mathbf{u} \in \mathcal{V}_D$  is projectively associated to a reference vertex  $\mathbf{v}_u \in \mathcal{V}_M$ . Given this association information, SuMa estimates the transformation between scans by incrementally minimizing the point-to-plane error given by

$$E(\mathcal{V}_D, \mathcal{V}_M, \mathcal{N}_M) = \sum_{\mathbf{u} \in \mathcal{V}_D} \left( \mathbf{n}_u^\top \left( \mathbf{T}_{C_{t-1}C_t}^{(k)} \mathbf{u} - \mathbf{v}_u \right) \right)^2. \quad (9)$$

Each vertex  $\mathbf{u} \in \mathcal{V}_D$  is projectively associated to a reference vertex  $\mathbf{v}_u \in \mathcal{V}_M$  and its normal  $\mathbf{n}_u \in \mathcal{N}_M$  via

$$\mathbf{v}_u = \mathcal{V}_M \left( \Pi \left( \mathbf{T}_{C_{t-1}C_t}^{(k)} \mathbf{u} \right) \right) \quad (10)$$

$$\mathbf{n}_u = \mathcal{N}_M \left( \Pi \left( \mathbf{T}_{C_{t-1}C_t}^{(k)} \mathbf{u} \right) \right). \quad (11)$$

SuMa then minimizes the objective function given in Eq. (9) using Gauss-Newton and determines increments  $\delta$  by iteratively solving

$$\delta = \left( \bar{\mathbf{J}}_\delta^\top \mathbf{W} \mathbf{J}_\delta \right)^{-1} \bar{\mathbf{J}}_\delta^\top \mathbf{W} \mathbf{r}, \quad (12)$$

where  $\mathbf{W} \in \mathbb{R}^{n \times n}$  is a diagonal matrix containing weights  $w_u$ ,  $\mathbf{r} \in \mathbb{R}^n$  is the stacked residual vector, and  $\mathbf{J}_\delta \in \mathbb{R}^{n \times 6}$  the Jacobian of  $\mathbf{r}$  with respect to the increment  $\delta$ .

SuMa employs a loop closure detection module, which considers the nearest frame in the built map as the candidate for loop closure given the current pose estimate. Loop closure detection works well for small loops, but the heuristic fails in areas with only a few large loops. Furthermore, drifts in the odometry estimate can lead to large displacements, where a heuristic of taking the nearest frame in the already mapped areas into account does not yield correct candidates. This effect is shown in our experiments, see Sect. 6.3.

### 4.2 Covariance propagation for geometric verification

SuMa's loop closure detection uses a fixed search radius. In contrast, we additionally use the covariance of the pose estimate and error propagation to automatically adjust the search radius.

We assume a noisy pose  $\mathbf{T}_{C_{t-1}C_t} = \{\bar{\mathbf{T}}_{C_{t-1}C_t}, \mathbf{6}_{C_{t-1}C_t}\}$  with mean  $\bar{\mathbf{T}}_{C_{t-1}C_t}$  and covariance  $\mathbf{6}_{C_{t-1}C_t}$ . We can estimate the covariance matrix by

$$\mathbf{6}_{C_{t-1}C_t} = \frac{1}{K} \frac{E}{N - M} \left( \bar{\mathbf{J}}_\delta^\top \mathbf{W} \mathbf{J}_\delta \right)^{-1}, \quad (13)$$

where  $K$  is the correction factor of the Huber robustized covariance estimation (Huber 1981),  $E$  is the sum of the squared weighted point-to-plane errors (the sum of squared weighted residuals) given the pose  $\mathbf{T}_{C_{t-1}C_t}$ , see Eq. (9),  $N$  is the number of correspondences,  $M = 6$  is the dimension of the transformation between two 3D poses.

To estimate the propagated uncertainty during the incrementally pose estimation, we can update the mean and covariance as follows:

$$\bar{\mathbf{T}}_{WC_t} = \bar{\mathbf{T}}_{WC_{t-1}} \bar{\mathbf{T}}_{C_{t-1}C_t} \quad (14)$$

$$\mathbf{6}_{WC_t} \approx \mathbf{6}_{WC_{t-1}} + \mathbf{J}_{C_{t-1}C_t} \mathbf{6}_{C_{t-1}C_t} \mathbf{J}_{C_{t-1}C_t}^\top, \quad (15)$$

where  $\mathbf{J}_{C_{t-1}C_t}$  is the Jacobian of Eq. (14).

Since we use the Mahalanobis distance  $D_M$  as a probabilistic distance measure between two poses, we make use of Lie algebra to express  $\mathbf{T}$  as a 6D vector  $\xi \in \mathfrak{se}(3)$  using



$\xi = \log \mathbf{T}$ , yielding

$$D_M(\mathbf{T}_{C1}, \mathbf{T}_{C2}) = \sqrt{\Delta \xi_{C1C2}^\top \mathbf{6}_{C1C2}^{-1} \Delta \xi_{C1C2}}. \quad (16)$$

Using the scaled distance, we can now restrict the search space depending on the pose uncertainty to save computation time.

Once we find loop closure candidates, we try to align the current point cloud to the rendered view at the corresponding pose  $\mathbf{T}_{WC_j^*}$  using the frame-to-model ICP. For the ICP initialization, we use the yaw angle offset estimated from OverlapNet while keeping other setups the same as those used in SuMa. If we have found a loop closure candidate at timestep  $t$ , we try to verify it in the subsequent timesteps  $t + 1, \dots, t + \Delta_{verification}$ , which ensures that we only add consistent loop closures to the pose graph.

## 5 OverlapNet for global localization

For global localization in a Monte-Carlo localization framework (Dellaert et al. 1999), one of the key challenges lies in the design of the observation model. For the observation model, we need to compare the sensor data and the map. As we want to exploit the overlap and yaw angle predictions of OverlapNet for that, our map consists of virtual scans at discretized 2D locations on a grid. We assume that a point cloud of the environment is available, which we use to extract the map information. Then, we render virtual scans as a pre-processing step. We can then train the network completely self-supervised on the map of virtual scans.

Finally, we integrate an observation model using the overlap and a separate observation model for the yaw angle estimates in a particle filter to perform localization.

### 5.1 Map of virtual scans

OverlapNet requires two LiDAR scans as input. One is the current scan and the second has to be generated from the map point cloud. Thus, we build a map of virtual LiDAR scans given an aggregated point cloud by using a grid of locations with grid resolution  $\gamma$ , where we generate virtual LiDAR scans for each location. The grid resolution is a trade-off between the accuracy and storage size of the map. Instead of storing these virtual scans, we just need to use one leg of the OverlapNet to obtain a feature volume  $\mathbf{F}$  using the input tensor  $\mathbf{I}$  of this virtual scan. Storing the feature volume instead of the complete scan has two key advantages: First, it uses more than an order of magnitude less space than the original point cloud (roughly ours: 100 MB/km, raw scans: 1.7 GB/km). Second, we do not need to compute the  $\mathbf{F}$  during localization on the map. The features volumes of the virtual

scans can directly be used to compute overlap and yaw angle estimates with a query scan that is the currently observed LiDAR point cloud in our localization framework.

### 5.2 Monte-Carlo localization

Monte-Carlo localization or MCL is a localization algorithm based on the particle filter proposed by Dellaert et al. (1999). Each particle represents a hypothesis for the robot's or autonomous vehicle's 2D pose  $\mathbf{x}_t = (x, y, \theta)_t$  at time  $t$ . When the robot moves, the pose of each particle is updated with a prediction based on a motion model with the control input  $\mathbf{u}_t$ . The expected observation from the predicted pose of each particle is then compared to the actual observation  $\mathbf{z}_t$  acquired by the robot to update the particle's weight based on the observation model. Particles are resampled according to their weight distribution and resampling is triggered whenever the effective number of particles drops below 50% of the sample size, see Grisetti et al. (2007) for details. After several iterations of this procedure, the particles are likely to converge around the true pose.

MCL realizes a recursive Bayesian filtering scheme. The key idea of this approach is to maintain a probability density  $p(\mathbf{x}_t | \mathbf{z}_{1:t}, \mathbf{u}_{1:t})$  of the pose  $\mathbf{x}_t$  at time  $t$  given all observations  $\mathbf{z}_{1:t}$  up to time  $t$  and motion control inputs  $\mathbf{u}_{1:t}$  up to time  $t$ . This posterior is updated as follows:

$$p(\mathbf{x}_t | \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) = \eta p(\mathbf{z}_t | \mathbf{x}_t) \int p(\mathbf{x}_t | \mathbf{u}_t, \mathbf{x}_{t-1}) p(\mathbf{x}_{t-1} | \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t-1}) d\mathbf{x}_{t-1}, \quad (17)$$

where  $\eta$  is the normalization constant resulting from Bayes' rule,  $p(\mathbf{x}_t | \mathbf{u}_t, \mathbf{x}_{t-1})$  is the motion model, and  $p(\mathbf{z}_t | \mathbf{x}_t)$  is the observation model. This paper focuses only on the observation model. For the motion model, we follow a standard odometry model for vehicles (Thrun et al. 2005).

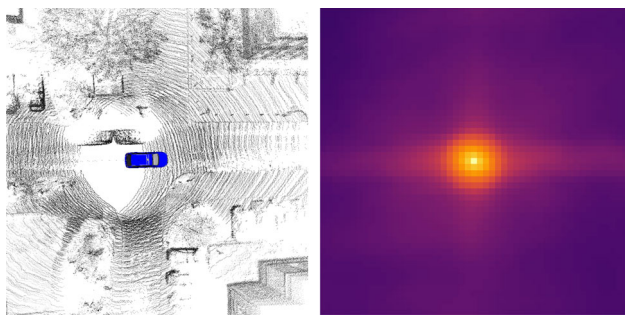
We split the observation model into two parts:

$$p(\mathbf{z}_t | \mathbf{x}_t) = p_L(\mathbf{z}_t | \mathbf{x}_t) p_O(\mathbf{z}_t | \mathbf{x}_t), \quad (18)$$

where  $\mathbf{z}_t$  is the observation at time  $t$ ,  $p_L(\mathbf{z}_t | \mathbf{x}_t)$  is the probability encoding the location  $(x, y)$  agreement between the current query LiDAR scan and the virtual scan at the nearest grid position and  $p_O(\mathbf{z}_t | \mathbf{x}_t)$  is the probability encoding the yaw angle  $\theta$  agreement between the same pairs of scans.

### 5.3 OverlapNet-based observation model

Given a particle  $i$  with the state estimate  $(x_i, y_i, \theta_i)$ , the overlap estimates encode the location agreement between the query LiDAR scan and virtual scans of the grid cells where



**Fig. 5** Overlap observation model describing  $p(\mathbf{z}_t | \mathbf{x}_t)$  for MCL. Local heatmap of the scan at the car's position with respect to the map. Lighter shades correspond to higher probabilities

particles are located. It can be directly used as the probability:

$$p_L(\mathbf{z}_t | \mathbf{x}_t) \propto f(\mathbf{z}_t, \mathbf{z}_i; \mathbf{w}), \quad (19)$$

where  $f$  corresponds to the neural network providing the overlap estimation between the input scans  $\mathbf{z}_t, \mathbf{z}_i$  and  $\mathbf{w}$  is the pre-trained weights of the network.  $\mathbf{z}_t$  and  $\mathbf{z}_i$  are the current query scan and a virtual scan of one  $(x, y)$  location respectively.

For illustration purposes, Fig. 5 shows the probabilities of all grid cells in a local area calculated by the overlap observation model. The blue car in the figure shows the current location of the car. The probabilities calculated by the overlap observation model can well represent the hypotheses of the current location of the car.

Typically, a large number of particles are used, especially when the environment is large. However, the computation time increases linearly with the number of particles. When applying the overlap observation model, particles can still obtain relatively large weights as long as they are close to the actual pose, even if not in the exact same position. This allows us to use fewer particles to achieve a high success rate of global localization.

Furthermore, the overlap estimation only encodes the location hypotheses. Therefore, if multiple particles locate in the same grid area, only a single inference using the nearest virtual scan of the map needs to be done, which can further reduce the computation time.

Given a particle  $i$  with the state vector  $(x_i, y_i, \theta_i)$ , the yaw angle estimates encode the orientation agreement between the query LiDAR scan and virtual scans of the corresponding grids where particles are located. We formulate the orientation probability as follows:

$$p_O(\mathbf{z}_t | \mathbf{x}_t) \propto \exp\left(-\frac{1}{2} \frac{(g(\mathbf{z}_t, \mathbf{z}_i; \mathbf{w}) - \theta_i)^2}{\sigma_\theta^2}\right), \quad (20)$$

where  $g$  corresponds to the neural network providing the yaw angle estimation between the input scans  $\mathbf{z}_t, \mathbf{z}_i$  and  $\mathbf{w}$  is the pre-trained weights of the network.  $\mathbf{z}_t$  and  $\mathbf{z}_i$  are the current query scan and a virtual scan of one particle respectively.

When generating the virtual scans of the grid map, all virtual scans will be set facing along the absolute  $0^\circ$  yaw angle direction. By doing this, the estimated relative yaw angle between the query scan and the virtual scan indicates the absolute yaw angle of the current query scan. Eq. (20) assumes a Gaussian measurement error in the heading.

By combining overlap and yaw angle estimation, the proposed observation model will correct the weights of particles considering agreements between the query scan and the map with the full pose  $(x, y, \theta)$ .

## 6 Experimental results

The experimental evaluation is designed to evaluate our approach and support the claims we made in the introduction of this article, which are that our approach is able to: (i) predict the overlap and relative yaw angle between pairs of LiDAR scans by exploiting multiple cues without using relative poses, (ii) combine odometry information with overlap predictions to detect correct loop closure candidates, (iii) improve the overall pose estimation results in a state-of-the-art SLAM system yielding more globally consistent maps, (iv) initialize ICP using the OverlapNet predictions yielding correct scan matching results, (v) build a novel observation model and achieve global localization.

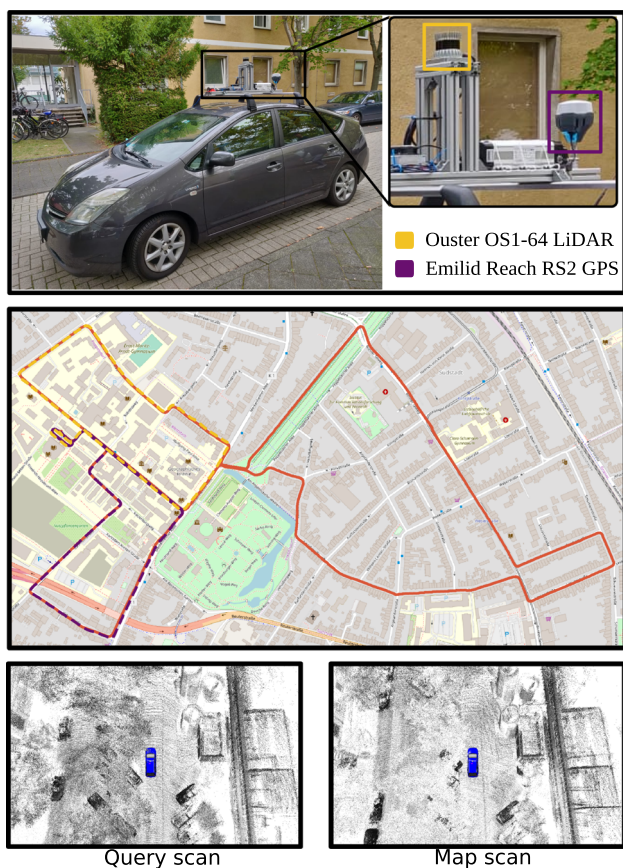
### 6.1 Implementation, datasets, and experimental setups

We implemented OverlapNet based on Python and TensorFlow (Abadi et al. 2016). The open source implementation is available in form of a stand-alone library, released under the MIT license and can be obtained from GitHub.<sup>1</sup> The source code is well documented with multiple demos to show the functionalities including a loop closing example and can be easily integrated into any other framework. Besides the network, we also released the implementation of the proposed OverlapNet-based global localization method.<sup>2</sup>

For SLAM and loop closing, we train and evaluate our approach on the KITTI odometry benchmark (Geiger et al. 2012). It provides LiDAR scans recorded with a Velodyne HDL-64E showing urban areas around Karlsruhe in Germany. It provides 11 sequences (00–10) with ground truth poses covering different types of environment, e.g., urban, country, and highway. We follow the experimental setup

<sup>1</sup> <https://github.com/PRBonn/OverlapNet>.

<sup>2</sup> [https://github.com/PRBonn/overlap\\_localization](https://github.com/PRBonn/overlap_localization).



**Fig. 6** Upper: sensor setup used for data recording. Middle: trajectories of the dataset used in this paper, overlaid on OpenStreetMap. The orange trajectory represents the sequence used to generate a map for localization. The yellow and purple trajectories represent two different test sequences. Bottom: LiDAR scans of the same place, once during mapping and once during localization. Since the LiDAR data was collected in different seasons, the appearance of the environment changed quite significantly due to changes in the vegetation but also due to parked vehicles at different places (Color figure online)

proposed by Schaupp et al. (2019) and use sequence 00 for evaluation. Sequences 03–10 are used for training and sequence 02 is used for validation.

To evaluate the generalization capabilities of our method, we also test it on the Ford campus dataset (Pandey et al. 2011), which was recorded on the Ford research campus in downtown Dearborn in Michigan using a different version of the Velodyne HDL-64E. In the case of the Ford campus dataset, we test our method on sequence 00, which contains several large loops. Note that we never trained our approach on the Ford campus dataset, only on the KITTI dataset.

For global localization, we use our own IPB-Car dataset, collected with our self-developed sensor platform illustrated in Fig. 6. The KITTI dataset and Ford Campus dataset do not perfectly fulfill our needs for evaluating a localization system, because there are no sequences of the same place but from different seasons available. We therefore col-

lected a large-scale dataset in different seasons with multiple sequences repeatedly exploring the same crowded urban area of Bonn city in Germany using an Ouster OS1-64. For our car dataset, we performed a 3D LiDAR SLAM, SuMa (Behley and Stachniss 2018), combined with a high-precision and SAPOS correction GPS information to create near ground truth poses. During localization, we only use LiDAR scans for global localization *without* using any GPS.

The dataset has three sequences that were collected at different times of the year, sequence 00 in September 2019, sequence 01 in November 2019, and sequence 02 in February 2020. The whole dataset covers a distance of over 10 km. We use LiDAR scans from sequence 02 to build the virtual scans and use sequence 00 and 01 for localization. As can be seen from Fig. 6, the appearance of the environment changes significantly since the dataset was collected in different seasons and in a crowded urban environment, including changes in vegetation, but also parking cars at different locations, moving people, and other objects. The link to the dataset can be found in our open-source code repository.<sup>2</sup>

Besides the open source implementation, we also provide the parameters used in the proposed method in Tables 1 and 2 for the purpose of reproducibility to the experimental results. Table 1 shows the configuration of the network architecture and Table 2 shows hyper parameters used in the proposed method. As shown in Table 2, for generating the range image following SuMa setup (Behley and Stachniss 2018), we only use points within a distance of 75 m to the sensor and generate range images with height  $h = 64$  and width  $w = 900$ . For overlap computation, see Eq. (3), we use  $\epsilon = 1$  m. We use a learning rate of  $10^{-3}$  with a decay of 0.99 every epoch and train at most 100 epochs. For the combined loss, Eq. (6), we set  $\alpha = 5$ . For the overlap loss, Eq. (7), we use  $a = 0.25$ ,  $b = 12$ , and scale factor  $s = 24$ . For the calculation of yaw angle

**Table 2** Hyper parameters of our approach

Parameter	Description	Value
$h$	Height of range image	64
$w$	Width of range image	900
$d_{\max}$	Maximum range	75 m
$\epsilon$	Threshold to count overlap pixels	1 m
$a$	Constant offset of sigmoid	0.25
$b$	Constant offset of sigmoid	12
$s$	Scaling factor of sigmoid	24
$\sigma_{\theta}$	Sigma for orientation probability	10 deg
$d_{\text{converge}}$	Threshold to success converge	5 m
Network training	Learning rate	$10^{-3}$
	Decay of every epoch	0.99
	Number of epochs	100
	Overlap loss weight $\alpha$	5



probability, Eq. (20), we set  $\sigma_\theta = 10$ . We use the a threshold of  $d_{\text{converge}} = 5\text{ m}$  to decide whether the global localization converges or not.

## 6.2 Overlap and yaw angle evaluation

In this section, we show the experiments that support the claim that our approach is able to estimate the overlap and yaw angle offset between pairs of LiDAR scans, which is well-suited for solving the general similarity estimation. We also provide an ablation study of using different input modalities and the analysis of the relationship between the overlap and yaw angle estimations.

In the case of general LiDAR scans similarity estimation, we assume that we have no prior information about the robot pose. We compare our method with the state-of-the-art learning-based methods LocNet (Yin et al. 2019), LocNet++ and OREOS (Schaupp et al. 2019), and also the traditional hand-crafted feature-based method FPFH+RANSAC. We follow the experimental setup of OREOS, where the KITTI sequence 00 is used for the evaluation. The LiDAR scans from the first 170s of sequence 00 are used to generate the database, as the vehicle starts to revisit previously traversed areas after 170s. The remaining LiDAR scans are used for place recognition queries. The query point clouds are sampled to be at least 3 m apart. Two point clouds are considered in the same place, if their ground-truth poses are within 1.5 m. The baseline results are those produced by the authors of OREOS (Schaupp et al. 2019).

### 6.2.1 Overlap estimation for Similarity measurement

In a general place recognition application, multiple candidates may be retrieved according to the similarity measurements with respect to the current query LiDAR scan. The respective place recognition candidates recall results are shown in Fig. 7. Our method outperforms all baseline methods when using a small number of candidates and attains similar performance as baseline methods for higher values of numbers of candidates. However, OREOS and LocNet++ attain a slightly higher recall if more candidates are considered.

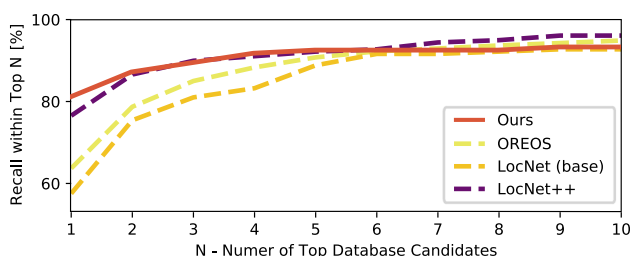


Fig. 7 Place recognition performance on KITTI sequence 00

Table 3 Yaw estimation errors

Approach	Mean (deg)	Std (deg)	Recall (%)
FPFH+RANSAC*	13.28	32.19	97
OREOS*	12.67	15.23	100
Ours	<b>1.13</b>	<b>3.34</b>	<b>100</b>

Bold numbers indicate best performance \*The results are those produced by the authors of OREOS

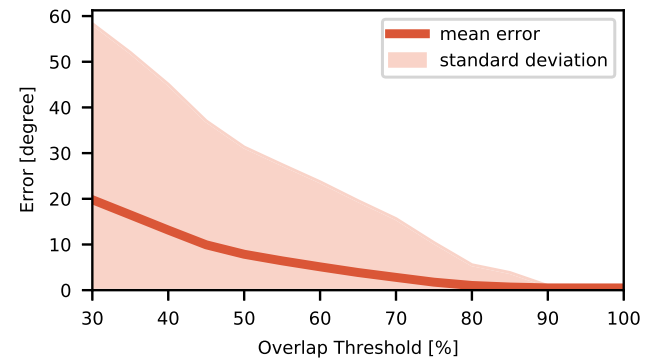


Fig. 8 Overlap and yaw estimation relationship

### 6.2.2 Yaw angle estimation

Table 3 summarizes the yaw angle errors on KITTI sequence 00. We can see that our method outperforms the other methods in terms of mean error and standard deviations. In terms of recall, OverlapNet and OREOS always provide a yaw angle estimate, since both approaches are designed to estimate the relative yaw angle for any pairs of scans in contrast to the RANSAC-based method that sometimes fails.

The superior performance can be mainly attributed to the correlation head exploiting the fact that the orientation in LiDAR scans can be well represented by the shift in the range projection. Therefore, it is easier to train the correlation head to accurately predict the relative yaw angles rather than a multilayer perceptron used in OREOS. Furthermore, there is also a strong relationship between overlap and yaw angle, which also improves the results when trained together.

### 6.2.3 Relationship between overlap and yaw estimations

Figure 8 shows the relationship between real overlap and yaw angle estimation error. As expected, the yaw angle estimate gets better with increasing overlap. Based on these plots, our method not only finds candidates but also measures the quality, i.e., when the overlap of two scans is larger than 90%, our method can accurately estimate the relative yaw angle with an average error of about only 1°.



**Table 4** Ablation study on usage of input modalities

$\mathcal{R}$	$\mathcal{N}$	$\mathcal{I}$	$\mathcal{S}$	Overlap		Yaw (degree)	
				AUC	F1	Mean	Std
✓				0.86	0.87	11.67	25.32
✓	✓			0.86	0.85	2.97	14.28
✓	✓	✓		0.87	0.87	2.53	14.56
✓	✓	✓	✓	0.87	0.88	1.13	3.34

$\mathcal{R}$ , range depth;  $\mathcal{N}$ , normal;  $\mathcal{I}$ , intensity;  $\mathcal{S}$ , semantic information. AUC, area under the curve, F1, balanced F-score

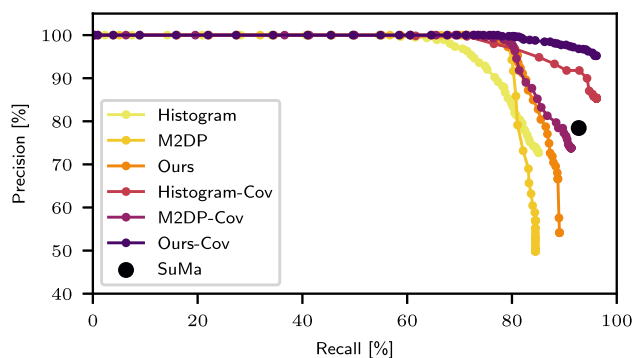
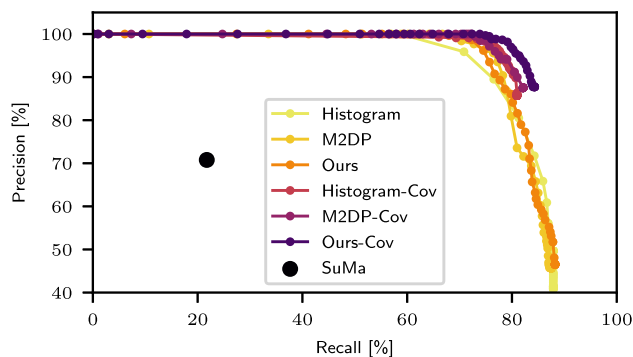
### 6.2.4 Ablation study on input modalities

An ablation study on the usage of different inputs is shown in Table 4. As can be seen, when employing more input modalities, the proposed method gets more robust. We notice that exploiting only depth information with OverlapNet can already perform reasonably in terms of overlap prediction, while it does not perform well in yaw angle estimation. When combining this with normal information, the OverlapNet can perform well on both tasks. Another interesting finding is the drastic reduction of yaw angle mean error and standard deviation when using semantic information. One reason could be that adding semantic information will make the input data more distinguishable when the car drives in symmetrical environments.

### 6.3 OverlapNet-based loop closing

In the following experiments, we investigate the loop closing performance of our approach and compare it to existing methods. Different from the general place recognition, loop closure detection typically assumes that robots revisit places during the mapping while moving with uncertain odometry. Therefore, the prior information about robot poses extracted from the pose graph is available for the loop closure detection. The following criteria are used in these experiments:

- To avoid detecting a loop closure in the most recent scans, we do not search candidates in the latest 100 scans.
- For each query scan, only the best candidate is considered throughout this evaluation.
- Most SLAM systems search for potential loop closures only within the  $3\sigma$  area around the current pose estimate. We do the same, either using the Euclidean or the Mahalanobis distance, depending on the approach.
- We aim to find more loops even in some challenging situations with low overlaps, e.g., when the car drives back to an intersection from the opposite direction as shown in Fig. 1. We use the overlap value to decide if a candidate is a true positive rather than distance. Furthermore,

**(a)** KITTI sequence 00**(b)** Ford campus sequence 00**Fig. 9** Precision–Recall curves of different approaches

ICP can find correct poses if the overlap between pairs of scans is around 30%, as illustrated in Sect. 6.3.2.

We evaluate OverlapNet on both the KITTI dataset and the Ford campus dataset to showcase the generalization capabilities of the approach.

#### 6.3.1 Quantitative analysis

Figure 9 shows the precision–recall curves of different loop closure detection methods. We compare our method, trained with two heads and all cues labeled as *Ours* (*AllChannel*, *TwoHeads*), with three state-of-the-art approaches: a histogram-based approach (*Histogram*) by Röhling et al. (2015), *M2DP* by He et al. (2016), and the original heuristic approach of SuMa by Behley and Stachniss (2018). Since SuMa always uses the nearest frame as the candidate for loop closure detection, we can only get one pair of precision and recall value resulting in a single point. We also show the result of our method using prior information, *Ours-Cov*, which uses covariance propagation (Sect. 4.2) to define the search space with the Mahalanobis distance and use the nearest in Mahalanobis distance of the top 10 predictions of OverlapNet as the loop closure candidates. For a fair comparison, we also show the results that can be obtained by enhancing the base-

lines with the proposed covariance-based method. These are labeled as *Histogram-Cov* and *M2DP-Cov* respectively.

Table 5 shows the comparison between our approach and the state of the art using the F1 score and the area under the curve (AUC) on both, KITTI and Ford campus dataset. For the KITTI dataset, our approach uses the model trained with all cues, including depth, normals, intensity, and a probability distribution over semantic classes. For the Ford campus dataset, our approach uses the model trained on KITTI with geometric information only, namely *Ours (GO)*, since other cues are not available in this dataset. We can see that our method outperforms the other methods on the KITTI dataset and attains a similar performance on the Ford campus dataset. There are two reasons to explain the worse performance on the Ford campus dataset. First, we never trained our network on the Ford campus dataset or even US roads, and secondly, there is only geometric information available on the Ford campus dataset. However, our method outperforms all baseline methods in both, KITTI and Ford campus dataset, if we integrate prior information.

We also show the performance in comparison to variants of our method in Table 6. We compare our best model AC-

**Table 5** Comparison with state of the art

Dataset	Approach	AUC	F1 score
KITTI	SuMa	–	0.85
	Histogram	0.83	0.83
	Histogram-Cov	0.95	0.92
	M2DP	0.83	0.87
	M2DP-Cov	0.89	0.88
	Ours (AC-TH)	0.87	0.88
	Ours-Cov (AC-TH)	<b>0.96</b>	<b>0.96</b>
Ford Campus	SuMa	–	0.33
	Histogram	0.84	0.83
	Histogram-Cov	0.85	0.85
	M2DP	0.84	0.85
	M2DP-Cov	0.85	0.86
	Ours (GO)	0.85	0.84
	Ours-Cov (GO)	<b>0.85</b>	<b>0.88</b>

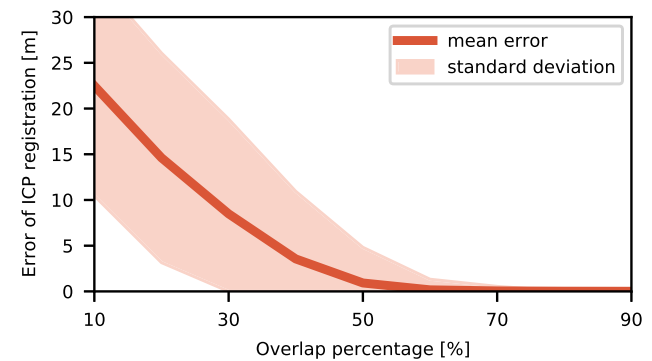
**Table 6** Comparison with our variants

Dataset	Variant	AUC	F1 score
KITTI	MLPOnly	0.58	0.65
	DeltaOnly	0.85	0.88
	Ours (AC-TH)	0.87	0.88
	Ours-Cov (AC-TH)	<b>0.96</b>	<b>0.96</b>
Ford Campus	Ours (GO)	0.85	0.84
	Ours-Cov (GO)	<b>0.85</b>	<b>0.88</b>

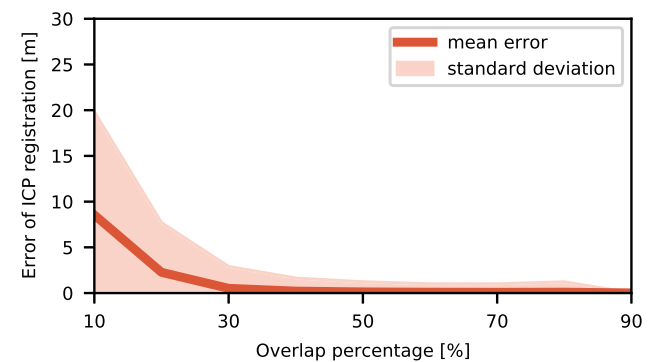
*TH* using all available cues and two heads to a variant which only uses a basic multilayer perceptron as the head named *MLPOnly* which consists of two hidden fully connected layers and a final fully connected layer with two neurons (one for overlap, one for yaw angle). The substantial difference of the AUC and F1 scores shows that such a simple network structure is not sufficient to get a good result. Training the network with only one head (only the delta head for overlap estimation, named *DeltaOnly*), has not a significant influence on the performance. A huge gain can be observed when regarding the nearest frame in Mahalanobis distance of the top 10 candidates in overlap percentage (*Ours-Cov*).

### 6.3.2 Using OverlapNet predictions as an initial guess for ICP registration

We aim at supporting the claim that our network provides good initializations for 3D LiDAR-based ICP registration in the context of autonomous driving. Figure 10 shows the relations between the overlap and ICP registration error with and without using OverlapNet predictions as initial guesses. The error of the ICP registration is here the Euclidean distance between the estimated relative translation and the ground

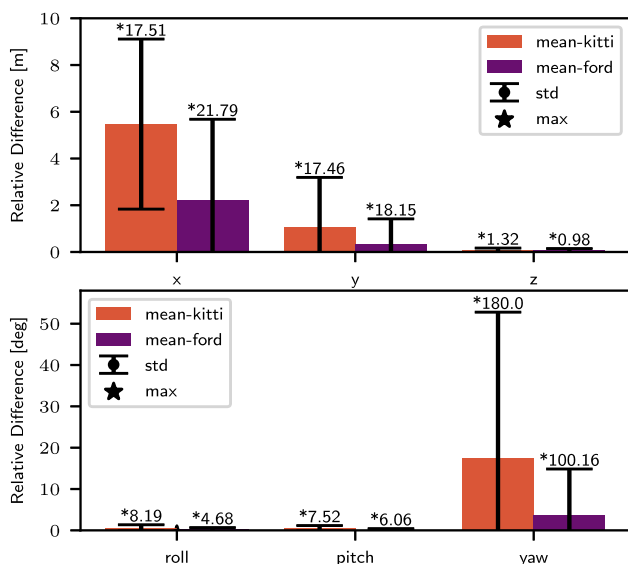


(a) With identity as initial guess



(b) With yaw angle as initial guess

**Fig. 10** ICP using OverlapNet predictions as the initial guess. The error of ICP registration here is the Euclidean distance between the estimated translation and the ground-truth translation



**Fig. 11** Statistics on relative 6 degree of freedom pose ( $x$ ,  $y$ ,  $z$ ,  $roll$ ,  $pitch$ ,  $yaw$ ) between pairs of scans with overlap larger than 30%. Results on KITTI and Ford campus datasets are shown

truth translation. As can be seen, the yaw angle prediction of the OverlapNet increases the chance to get a good result from the ICP even when two frames are relatively far away from each other, i.e., have only a low overlap. Therefore in some challenging cases, e.g., the car drives back into an intersection from a different street, our approach can still find loop closures. The results also show that the overlap estimates measure the quality of the found loop closure: larger overlap values result in better registration results of the involved ICP.

To analyze the correlation between the overlap and the 6 degree of freedom pose ( $x$ ,  $y$ ,  $z$ ,  $roll$ ,  $pitch$ ,  $yaw$ ), we show the statistics with all sequences of both KITTI and Ford Campus datasets in Fig. 11. As can be seen, with overlap values larger than 30%, the relative differences in roll, pitch are much smaller than that in yaw. This means that our method is likely to filter out such cases where there are large differences in roll and pitch corresponding to low overlaps between pairs of scans and therefore are not good to be used for loop closing. Furthermore, these experiments also show the motivation that it is more important to estimate the yaw angle rather than roll and pitch, since even when two scans have a large overlap, the yaw angle offset could be very large. For example, when the car drives in both directions on a road or a slope, the yaw angle offset is around 180 degrees. Note that, our method is not influenced by such cases, because it can estimate the yaw angle offset between pairs of scans. After rotating in yaw, the offset of other orientation angles are small and can typically be handled by the following ICP that is used in general loop closing.

### 6.3.3 Improving SLAM results

This experiment supports our claim that our method is able to improve the overall SLAM result. Figure 12 shows the odometry results on KITTI sequence 02. The color in Fig. 12 shows the 3D translation error (including height). The top figure shows the SuMa method and the bottom figure shows *Ours-Cov* using the proposed OverlapNet to detect loop closures. We can see that after integrating our method, the overall odometry is much more accurate since we can provide more loop closure candidates with higher accuracy in terms of overlap. The colors represent the translation error of the estimated poses with respect to the ground truth. Furthermore, after integrating the proposed OverlapNet, the SLAM system can find more loops even in some challenging situations, e.g., when the car drives back to an intersection from the opposite direction as shown in Fig. 1.

## 6.4 OverlapNet-based global localization

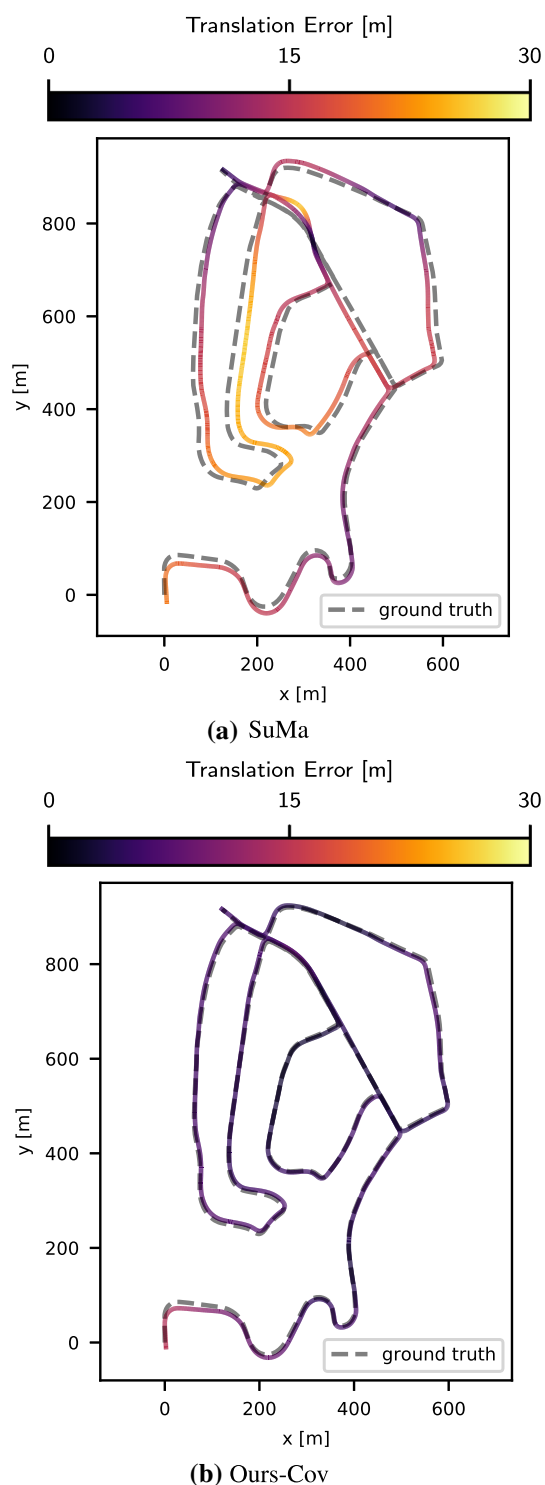
In the following experiments, we test our proposed localization method by integrating the OverlapNet based sensor model into the MCL framework. The MCL frameworks are the same for all baselines and we only exchange the observation models. Note that in general maps of the environment are built using previously recorded data. Thus the task is more difficult than loop closing because of larger environmental changes between the map and the new scans. We will show in the following experiments that our method is nevertheless able to perform global localization.

### 6.4.1 Baselines

We compare our observation model with two baseline observation models: the typical beam-end model as described by Thrun et al. (2005) and a histogram-based model derived from the work of Röhling et al. (2015).

The beam-end observation model is often used for 2D LiDAR data. For 3D LiDAR scans, it needs much more particles to make sure that it converges to the correct pose, which causes the computation time to increase substantially. In this paper, we implement the beam-end model with a down-sampled point cloud map using voxelization with a resolution of 10 cm.

Our second baseline for comparison uses the model by Röhling et al. (2015), which proposed a fast method to detect loop closures through the use of similarity measures on histograms extracted from 3D LiDAR data. The histogram contains the range information, and use it in the MCL framework as a baseline observation model. We employ the same grid map and virtual frames as used for our method with the histogram-based observation model. When updating the weights of particles, we first generate the histograms of the



**Fig. 12** Qualitative result on KITTI sequence 02 comparing SuMa to our approach

current query scan and the virtual scans of grids where the particles locate. Then, we use the same Wasserstein distance (Dobrushin 1970) to measure the similarity between them and update the weights of particles as follows:

$$p(\mathbf{z}_t | \mathbf{x}_t) \propto d(h(\mathbf{z}_t), h(\mathbf{z}_i)), \quad (21)$$

where  $d$  represents the Wasserstein distance between histograms  $h(\mathbf{z}_t)$ ,  $h(\mathbf{z}_i)$  of LiDAR scan  $\mathbf{z}_t$ ,  $\mathbf{z}_i$ .

#### 6.4.2 Localization performance

The experiment presented in this section is designed to show the performance of our approach and to support the claim that it is well suited for global localization.

First of all, we show the general localization results tested with two sequences of the IPB-Car dataset in Fig. 13. For the overlap network, we used only geometric information (range and normal images) as input. The qualitative results show that, after applying our sensor-model, the proposed method can well localize in the map with only LiDAR data collected in highly dynamic environments at different times.

For quantitative results, we calculate the success rate for different methods with different numbers of particles, as shown in Fig. 14. The x-axis represents the number of particles used during localization, while the y-axis is the success rate of different setups. The success rate for a specific setup of one method is calculated using the number of success cases divided by the total number of the tests. To decide whether one test is successful or not, we check the location error by every 100 frames after converging. If the location error is smaller than a certain threshold, we count this run as a success case.

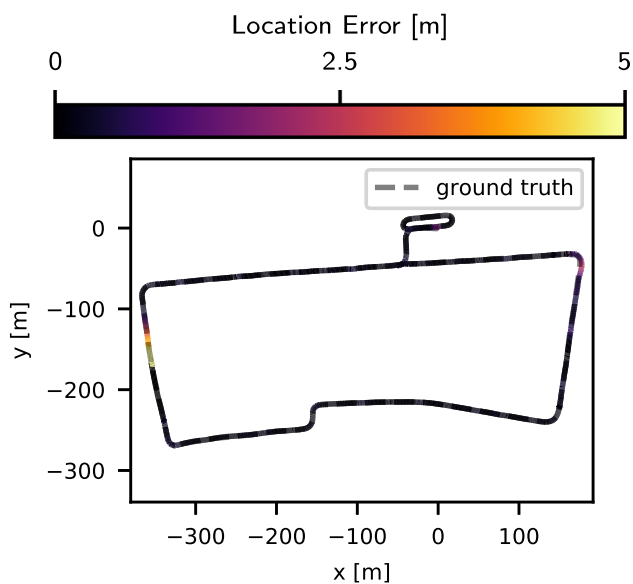
We test our method together with two baselines using five different numbers of particles  $N = \{1000; 5000; 10,000; 50,000; 100,000\}$ . For each setup, we sample 10 trajectories and perform global localization.

Quantitative results of localization accuracy are shown in Table 7. The upper part shows the location error of all methods tested with both sequences. The location error is defined as the root mean square error (RMSE) of each test in terms of the Euclidean error computed in  $(x, y)$  with respect to the ground truth poses. It shows the mean and the standard deviation of the error for each observation model. Note that the location error is only calculated for success cases.

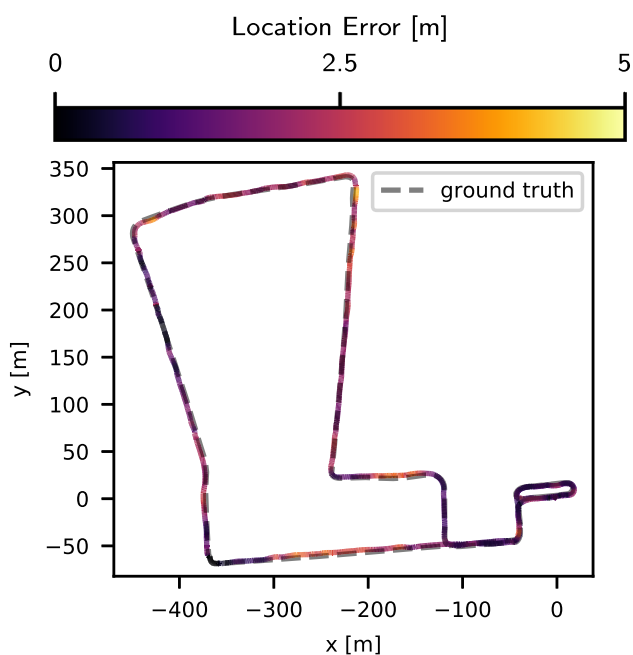
The lower part shows the yaw angle error. It is the RMSE of each test in terms of yaw angle error with respect to the ground truth poses. The table shows the mean and the standard deviation of the error for each observation model. As before, the yaw angle error is also only calculated for cases in which the global localization converged.

As can be seen from the results, our method achieves higher success rates with a smaller number of particles compared to the baseline methods, which also makes the proposed method faster than baseline methods. Furthermore, our method converges already with 100,000 particles in all cases, whereas the other observation models still need more particles to sufficiently cover the state space. Moreover, the





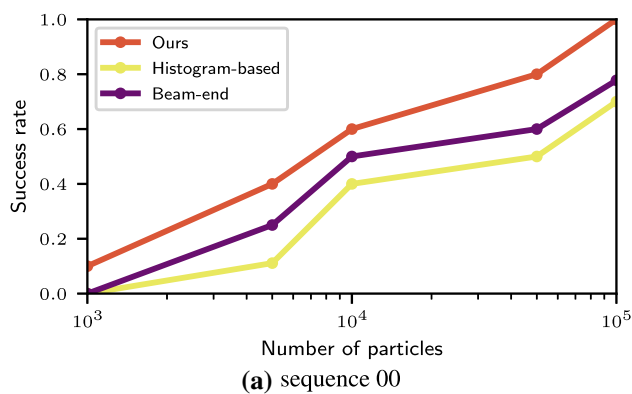
(a) Localization trajectory from sequence 00



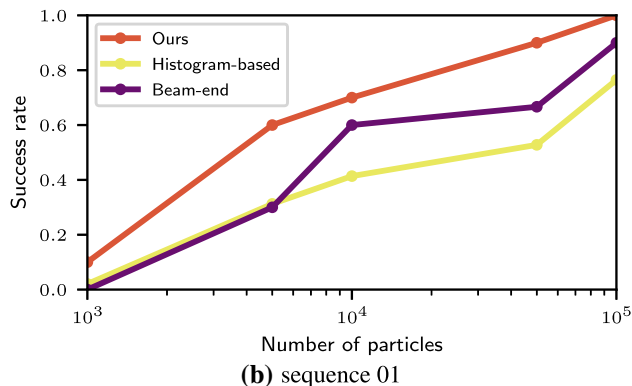
(b) Localization trajectory from sequence 01

**Fig. 13** Localization results of our method with 10,000 particles on two sequences recorded with the setup depicted in Fig. 6. Shown are the ground truth trajectory (black) and our estimated trajectory using our observation model (red) (Color figure online)

proposed method gets similar performance in location error compared to the baseline methods but it achieves better results in yaw angle estimation. This is because the proposed method decouples the location and yaw angle estimation and, therefore, can exploit more constraints in yaw angle corrections.



(a) sequence 00



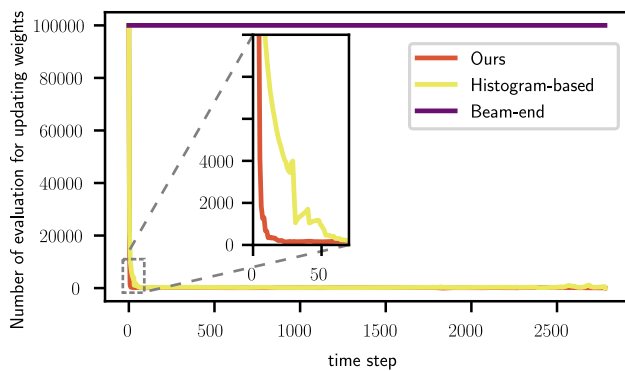
(b) sequence 01

**Fig. 14** Success rate of the different observation models for 10 globalization runs. Here, we use sequence 00 and sequence 01 to localize in the map of the IPB-Car dataset. We count runs as success if converge to the ground truth location within 5 m

**Table 7** Localization results

Sequence	Location error (meter)		
	Beam-end	Histogram	Ours
0	0.92 ± 0.27	1.85 ± 0.34	<b>0.81 ± 0.13</b>
1	<b>0.67 ± 0.11</b>	1.86 ± 0.34	0.88 ± 0.07
Sequence	Yaw angle error (degree)		
	Beam-end	Histogram	Ours
0	1.87 ± 0.47	3.10 ± 3.07	<b>1.74 ± 0.11</b>
1	2.10 ± 0.59	3.11 ± 3.08	<b>1.88 ± 0.09</b>

To sum up, for global localization the proposed method outperforms the baseline methods in terms of success rate, while getting similar results in terms of location error. Moreover, our method outperforms baseline methods in yaw angle estimation, because of the proposed de-coupled observation model. Furthermore, our method is faster than the baseline methods.



**Fig. 15** Number of observation model evaluations for updating the weights at each timestep with 100,000 particles. The beam end model needs to be evaluated for each and every particle individually. The histogram-based method is more computationally efficient, while our proposed method still needs the fewest evaluations

## 6.5 Runtime

We tested our method on a system equipped with an Intel i7-8700 with 3.2 GHz and an Nvidia GeForce GTX 1080 Ti with 11 GB memory. When we use only geometric information, our method takes on average per frame 2 ms for feature extraction and 5 ms for head comparison. It takes additional 75 ms to perform semantic segmentation using RangeNet++.

In a real SLAM implementation, most systems only search loop closure candidates inside a certain search space given by pose uncertainty using the Mahalanobis distance (see Sect. 4.2). Once we generated a feature volume for a scan, it will be stored in memory. During the search process, we need only to generate the feature volume for the current scan and compare it to the feature volumes in memory. Therefore, our method can achieve online operation in long-term tasks, since we usually have to compare only a small number of candidate poses.

For global localization, we show the *number* of observation model evaluations necessary for updating the weights at each time step in Fig. 15. This is a fairer way to compare the computational cost of different methods, since our neural network-based method uses a GPU to concurrently updating the weights of particles, while the other methods only use a CPU. As can be seen, our method needs a smaller number of observation model evaluations to update the weights for all particles. This is because we only need to perform the network inference for all particles which are localized in the same grid cell once. For an incoming frame and the virtual frame of that grid cell, the inputs of the network and thus the outputs are the same for all particles in that grid cell.

For *initializing in the large-scale map*, the worst case will take around 43 s to process one frame. After convergence, the proposed method takes only 1 s on average to process one frame with 10,000 particles.

## 7 Conclusion

This paper addressed loop closing and localization for a vehicle using 3D LiDAR data. We proposed a modified Siamese network to estimate the similarity between pairs of scans using an image overlap generalized to range images and that provides a relative yaw angle estimate. For loop closure detection, we integrated our approach into an existing SLAM system to improve its mapping and odometry results. For global localization, we proposed a novel observation model using the concept of overlap and integrated this observation model into a Monte-Carlo localization framework to achieve good localization results. The experiments suggest that our method is able to estimate the overlap and yaw angle estimates between pairs of LiDAR scans, which can be successfully used for both, loop closing and global localization. Moreover, our method generalizes well to different environments at different times.

**Funding** Open Access funding enabled and organized by Projekt DEAL.

**Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

## References

- Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., & Kudlur, M. (2016). Tensorflow: A system for large-scale machine learning. In *12th USENIX symposium on operating systems design and implementation (OSDI 16)* (pp. 265–283).
- Barsan, I., Andrei, I., Wang, S., Pokrovsky, A., & Urtasun, R. (2018). Learning to localize using a LiDAR intensity map. In *Proceedings of the second conference on robot learning (CoRL)* (pp. 605–616).
- Behley, J., & Stachniss, C. (2018). Efficient surfel-based SLAM using 3D laser range data in urban environments. In *Proceedings of robotics: science and systems (RSS)*.
- Besl, P., & McKay, N. (1992). A method for registration of 3D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *14*(2), 239–256.
- Bogoslavskyi, I., & Stachniss, C. (2016). Fast range image-based segmentation of sparse 3d laser scans for online operation. In *Proceedings of the IEEE/RSJ international conference on intelligent robots and systems (IROS)*.
- Bromley, J., Guyon, I., LeCun, Y., Säckinger, E., & Shah, R. (1993). Signature verification using a “Siamese” time delayed neural network.

- IEEE Transactions on Pattern Analysis and Machine Intelligence*, 07(04), 669–688. <https://doi.org/10.1142/S0218001493000339>.
- Chen, X., Läbe, T., Milioto, A., Röhling, T., Vysotska, O., Haag, A., Behley, J., & Stachniss, C. (2020a) OverlapNet: Loop closing for LiDAR-based SLAM. In *Proceedings of robotics: science and systems (RSS)*.
- Chen, X., Läbe, T., Nardi, L., Behley, J., & Stachniss, C. (2020b). Learning an overlap-based observation model for 3D LiDAR localization. In *Proceedings of the IEEE/RSJ international conference on intelligent robots and systems (IROS)*.
- Chen, X., Milioto, A., Palazzolo, E., Giguère, P., Behley, J., & Stachniss, C. (2019). SuMa++: Efficient LiDAR-based semantic SLAM. In *Proceedings of the IEEE/RSJ international conference on intelligent robots and systems (IROS)*.
- Chen, X., Vizzo, I., Läbe, T., Behley, J., & Stachniss, C. (2021). Range image-based LiDAR localization for autonomous vehicles. In *Proceedings of the IEEE international conference on robotics & automation (ICRA)*.
- Cop, K., Borges, P., & Dub, R. (2018). Delight: An efficient descriptor for global localisation using lidar intensities. In *Proceedings of the IEEE international conference on robotics & automation (ICRA)*.
- Cramariuc, A., Dube, R., Sommer, H., Siegwart, R., & Gilitschenski, I., (2018). Learning 3D segment descriptors for place recognition. *arXiv preprint*.
- Dellaert, F., Fox, D., Burgard, W., & Thrun, S. (1999). Monte Carlo localization for mobile robots. In *IEEE international conference on robotics and automation (ICRA)*.
- Dobrushin, R. (1970). Prescribing a system of random variables by conditional distributions. *Theory of Probability & its Applications*, 15(3), 458–486.
- Dubé, R., Dugas, D., Stumm, E., Nieto, J., Siegwart, R., & Lerma, C. (2017). SegMatch: Segment based place recognition in 3D point clouds. In *Proceedings of the IEEE international conference on robotics & automation (ICRA)*.
- Fischler, M., & Bolles, R. (1981). Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6), 381–395.
- Fox, D., Burgard, W., Dellaert, F., & Thrun, S. (1999). Monte Carlo localization: Efficient position estimation for mobile robots. In *Proceedings of the conference on advancements of artificial intelligence (AAAI)* (pp. 343–349).
- Geiger, A., Lenz, P., & Urtasun, R. (2012). Are we ready for autonomous driving? The KITTI vision benchmark suite. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 3354–3361).
- Grisetti, G., Stachniss, C., & Burgard, W. (2007). Improved techniques for grid mapping with Rao–Blackwellized particle filters. *IEEE Transactions on Robotics*, 23(1), 34–46.
- Guo, J., Borges, P., Park, C., & Gawel, A. (2019). Local descriptor for robust place recognition using LiDAR intensity. *IEEE Robotics and Automation Letters*, 4(2), 1470–1477.
- He, L., Wang, X., & Zhang, H. (2016). M2DP: A novel 3D point cloud descriptor and its application in loop closure detection. In *Proceedings of the IEEE/RSJ international conference on intelligent robots and systems (IROS)*.
- Huber, P. J. (1981). *Robust statistics*. Wiley.
- Hussain, M., & Bethel, J. (2004). Project and mission planing. In: C. McGlone, E. Mikhail, J. Bethel, & R. Mullen (Eds.), *Manual of Photogrammetry, chapter 15.1.2.6* (pp. 1109–1111) American Society for Photogrammetry and Remote Sensing.
- Kim, G., Park, B., & Kim, A. (2019). 1-day learning, 1-year localization: Long-term LiDAR localization using scan context image. *IEEE Robotics and Automation Letters*, 4(2), 1948–1955.
- Kong, X., Yang, X., Zhai, G., Zhao, X., Zeng, X., Wang, M., Liu, Y., Li, W., & Wen, F. (2020). Semantic graph based place recognition for point clouds. In *Proceedings of the IEEE/RSJ international conference on intelligent robots and systems (IROS)*.
- Levinson, J., Montemerlo, M., & Thrun, S. (2007). Map-based precision vehicle localization in urban environments. In *Proceedings of robotics: science and systems (RSS)*.
- Ma, W., Tartavull, I., Bârsan, I.A., Wang, S., Bai, M., Mattyus, G., Homayounfar, N., Lakshmikanth, S. K., Pokrovsky, A., & Urtasun, R. (2019). Exploiting sparse semantic HD maps for self-driving vehicle localization. In *Proceedings of the IEEE/RSJ international conference on intelligent robots and systems (IROS)*.
- Merfels, C., & Stachniss, C. (2016). Pose fusion with chain pose graphs for automated driving. In *Proceedings of the IEEE/RSJ international conference on intelligent robots and systems (IROS)*.
- Milioto, A., & Stachniss, C. (2019). Bonnet: An open-source training and deployment framework for semantic segmentation in robotics using CNNs. In *Proceedings of the IEEE international conference on robotics & automation (ICRA)*.
- Milioto, A., Vizzo, I., Behley, J., & Stachniss, C. (2019). RangeNet++: Fast and accurate LiDAR semantic segmentation. In *Proceedings of the IEEE/RSJ international conference on intelligent robots and systems (IROS)*.
- Nagashima, S., Ito, K., Aoki, T., Ishii, H., & Kobayashi, K. (2007). A high-accuracy rotation estimation algorithm based on 1D phase-only correlation. In *Proceedings of the international conference on image analysis and recognition* (pp. 210–221).
- Pandey, G., McBride, J., & Eustice, R. (2011). Ford campus vision and LiDAR data set. *International Journal of Robotics Research*, 30(13), 1543–1552.
- Röhling, T., Mack, J., & Schulz, D. (2015). A fast histogram-based similarity measure for detecting loop closures in 3-D LiDAR data. In *Proceedings of the IEEE/RSJ international conference on intelligent robots and systems (IROS)* (pp. 736–741).
- Schaefer, A., Büscher, D., Vertens, J., Luft, L., & Burgard, W. (2019). Long-term urban vehicle localization using pole landmarks extracted from 3-D LiDAR scans. In *Proceedings of the European conference on mobile robotics (ECMR)* (pp. 1–7).
- Schaupp, L., Bürki, M., Dubé, R., Siegwart, R., & Cadena, C. (2019). OREOS: Oriented recognition of 3D point clouds in outdoor scenarios. *Proceedings of the IEEE/RSJ international conference on intelligent robots and systems (IROS)*.
- Silver, D., & Stentz, A. (2011). Monte Carlo localization and registration to prior data for outdoor navigation. In *Proceedings of the IEEE/RSJ international conference on intelligent robots and systems (IROS)*.
- Stachniss, C., Leonard, J., & Thrun, S. (2016). *Springer Handbook of Robotics*, chapter 46: Simultaneous Localization and Mapping (2nd ed.) Springer.
- Steder, B., Ruhnke, M., Grzonka, S., & Burgard, W. (2011). Place recognition in 3D scans using a combination of bag of words and point feature based relative pose estimation. In *Proceedings of the IEEE/RSJ international conference on intelligent robots and systems (IROS)*.
- Steder, B., Rusu, R., Konolige, K., & Burgard, W. (2010). NARF: 3D range image features for object recognition. In *Workshop on defining and solving realistic perception problems in personal robotics at the IEEE/RSJ international conference on intelligent robots and systems (IROS)*.
- Sun, L., Adolfsson, D., Magnusson, M., Andreasson, H., Posner, I., & Duckett, T. (2020). Localising faster: Efficient and precise lidar-based robot localisation in large-scale environments. In *Proceedings of the IEEE international conference on robotics & automation (ICRA)*.
- Thrun, S., Burgard, W., & Fox, D. (2005). *Probabilistic robotics*. MIT Press.
- Tinchev, G., Penate-Sanchez, A., Fallon, M. (2019) Learning to see the wood for the trees: Deep laser localization in urban and natural



environments on a CPU. *IEEE Robotics and Automation Letters (RA-L)* 4(2): 1327–1334.

- Uy, A., & Lee, G. (2018). PointNetVLAD: Deep point cloud based retrieval for large-scale place recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 4470–4479).
- Vizzo, I., Chen, X., Chebrolu, N., Behley, J., & Stachniss, C. (2021). Poisson surface reconstruction for LiDAR odometry and mapping. In *Proceedings of the IEEE international conference on robotics & automation (ICRA)*.
- Vysotska, O., & Stachniss, C. (2017). Improving slam by exploiting building information from publicly available maps and localization priors. *Photogrammetrie - Fernerkundung - Geoinformation (PFG)*, 85(1), 53–65.
- Wei, X., Bårnsan, I.A., Wang, S., Martinez, J., & Urtasun, R. (2019). Learning to localize through compressed binary maps. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)* (pp. 10316–10324).
- Wiesmann, L., Milioto, A., Chen, X., Stachniss, C., & Behley, J. (2021). Deep compression for dense point cloud maps. *IEEE Robotics and Automation Letters*, 6, 2060–2067. <https://doi.org/10.1109/LRA.2021.3059633>.
- Wilbers, D., Merfels, C., & Stachniss, C. (2019). Localization with sliding window factor graphs on third-party maps for automated driving. In *Proceedings of the IEEE international conference on robotics & automation (ICRA)*.
- Wolcott, R., & Eustice, R. (2015). Fast lidar localization using multiresolution gaussian mixture maps. In *Proceedings of the IEEE international conference on robotics & automation (ICRA)* (pp. 2814–2821).
- Yan, F., Vysotska, O., & Stachniss, C. (2019). Global localization on OpenStreetMap using 4-bit semantic descriptors. In *Proceedings of the European conference on mobile robotics (ECMR)*.
- Yin, H., Wang, Y., Ding, X., Tang, L., Huang, S., & Xiong, R. (2019). 3D LiDAR-based global localization using Siamese neural network. *IEEE Transactions on Intelligent Transportation Systems*. <https://doi.org/10.1109/TITS.2019.2905046>.
- Zaganidis, A., Zerntev, A., Duckett, T., & Cielniak, G. (2019). Semantically assisted loop closure in SLAM using NDT histograms. In *Proceedings of the IEEE/RSJ international conference on intelligent robots and systems (IROS)*.
- Zhang, C., Ang, M.H., & Rus, D. (2018). Robust LiDAR localization for autonomous driving in rain. In *Proceedings of the IEEE/RSJ international conference on intelligent robots and systems (IROS)* (pp. 3409–3415).

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.



**Xieyuanli Chen** is a Ph.D. student at Cyrill Stachniss' Photogrammetry and Robotics Laboratory, the University of Bonn. He is now also a member of the Technical Committee of RoboCup Rescue Robot League (RRL). He received his Master degree in Robotics in 2017 at the National University of Defense Technology, China. During that time, he was a member of the Organizing Committee of RoboCup Rescue Robot League. He received his Bachelor degree in Electrical Engineering and Automation in 2015 at Hunan University, China.



**Thomas Läbe** is a technical associate at the professorship of Photogrammetry of the Institute of Geodesy and Geoinformation. He finished his diploma in computer science ("Dipl.-Inform. (FH)") in 1994 at the Cologne University of Applied Sciences. From 1994 to 2000 he was employed at the Institute of Photogrammetry in the joint project "Passunktmodell Datenbank für die automatische Orientierung von Luftbildern" (automatic orientation of images) which was a cooperation with the land survey department ("Landesvermessungsamt") of North-Rhine-Westphalia. Since 2001 he is a software developer and administrator at the professorship of Photogrammetry.



**Andres Milioto** is a Senior Computer Vision Engineer at Scythe Robotics, in Boulder, Colorado, USA. Before this, he was a doctoral student and research assistant at Cyrill Stachniss' Photogrammetry and Robotics Laboratory, in Bonn, Germany. His main topics of research are semantic scene understanding for camera images and LiDAR point clouds, and efficiency techniques to make deep learning run online on embedded devices. Before his Ph.D., he got a Masters Degree in Electronics Engineering from Universidad Nacional de Rosario, in Rosario, Argentina, where he graduated top of his class in 2016.



**Timo Röhlings** got his Diploma degree in computer science from the University of Bonn and is currently working towards his Ph.D. He is employed at the Fraunhofer Institute for Communication, Information Processing and Ergonomics (FKIE). His research focuses on SLAM related problems and LiDAR sensor data processing with GPU acceleration.





**Jens Behley** is a postdoctoral researcher at the Department for Photogrammetry since February 2016. From September 2008 to July 2015, Jens worked at the Department for Computer Science III, University of Bonn and he successfully defended his Ph.D. thesis on “Three-dimensional Laser-based Classification in Outdoor Environments” supervised by Prof. Dr. Armin B. Cremers in January 2014.

self-driving cars. Cyrill Stachniss has coauthored over 220 scientific publications, he was senior editor of the IEEE Robotics and Automation Letters as well as associate editor of the IEEE Transactions on Robotics. He is a Microsoft Research Faculty Fellow and received the IEEE RAS Early Career Award in 2013. He is actively participating in German and European research projects. He is a spokesperson of the DFG Cluster of Excellence “PhenoRob - Robotics and Phenotyping for Sustainable Crop Production” at the University of Bonn. Furthermore, he acted as coordinator of the EC-funded project “ROVINA” for digitizing the Roman catacombs with robots and as the spokesperson of the DFG research unit “Mapping on Demand”.



**Cyrill Stachniss** is a full professor at the University of Bonn and heads the Photogrammetry and Robotics Lab since 2014. Before his appointment in Bonn, he was a lecturer at the University of Freiburg and the Swiss Federal Institute of Technology. The research activities of the Photogrammetry and Robotics Lab focus on probabilistic techniques in the context of mobile robotics, navigation problems, and perception. The lab has made several contributions in the areas of SLAM, localization,

place recognition, semantic scene interpretation, autonomous exploration, and planning. It covers ground vehicles including mobile robots and autonomous cars as well as UAVs. The main application areas of the research are autonomous service robots, agricultural robotics, and