

MinkLoc3D: Point Cloud Based Large-Scale Place Recognition

Jacek Komorowski
Warsaw University of Technology
Warsaw, Poland

jacek.komorowski@pw.edu.pl

Abstract

The paper presents a learning-based method for computing a discriminative 3D point cloud descriptor for place recognition purposes. Existing methods, such as PointNetVLAD, are based on unordered point cloud representation. They use PointNet as the first processing step to extract local features, which are later aggregated into a global descriptor. The PointNet architecture is not well suited to capture local geometric structures. Thus, state-of-the-art methods enhance vanilla PointNet architecture by adding different mechanism to capture local contextual information, such as graph convolutional networks or using hand-crafted features. We present an alternative approach, dubbed MinkLoc3D, to compute a discriminative 3D point cloud descriptor, based on a sparse voxelized point cloud representation and sparse 3D convolutions. The proposed method has a simple and efficient architecture. Evaluation on standard benchmarks proves that MinkLoc3D outperforms current state-of-the-art. Our code is publicly available on the project website.¹

1. Introduction

Applying deep learning methods to solve 3D computer vision problems is an area of active development. A number of methods for classification [28], semantic segmentation [28, 7] and local [8] or global [1] features extraction from 3D point clouds was recently proposed. We focus our attention on finding a discriminative, low-dimensional 3D point cloud descriptor for place recognition purposes. Localization is performed by searching the database for geo-tagged point clouds with descriptors closest to the query point cloud descriptor. The idea is illustrated in Fig. 1. Place recognition methods are widely used in robotics, autonomous driving [23] and augmented reality [24].

The first learning-based place recognition method operating on 3D point clouds is PointNetVLAD [1]. It uses

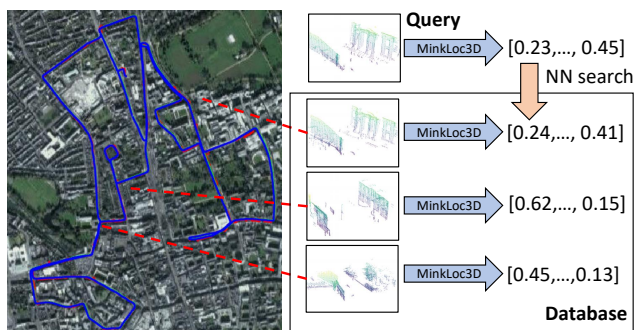


Figure 1. Point cloud-based place recognition. MinkLoc3D computes a global descriptor of a query point cloud. Localization is performed by searching the database for geo-tagged point clouds with closest descriptors.

PointNet [28] architecture to extract local features and NetVLAD [2] layer to aggregate them into a global descriptor. While PointNet proved to be successful in many applications, it was originally proposed to process point clouds representing single objects, not large and complex scenes. It is not well suited to extract informative local features. To overcome this weakness, latter 3D place recognition methods enhance vanilla PointNet architecture by adding different mechanism to capture local contextual information. PCAN [41] uses sampling and grouping operation at multiple scales. State-of-the-art LPD-Net [19] method uses rather complex architecture and combines learning-based and handcrafted local features. 3D points enhanced with pre-computed handcrafted features are processed by a PointNet module, fed to a graph neural network to aggregate neighbourhood information and further processed using PointNet architecture. Finally, a global descriptor is computed using NetVLAD [2] layer. LPD-Net surpasses previous state-of-the-art by a large margin. However, at the expense of architectural and computational complexity.

Increasing complexity of recent 3D point cloud-based place recognition methods, all based on unordered set of points representation, motivated us to investigate feasibility of using alternate approach and network architecture. We

¹<https://github.com/jac99/MinkLoc3D>

choose sparse voxelized representation and sparse convolutions, as recently it proved successful in many 3D vision tasks, including local feature extraction [8], semantic segmentation [7] and point cloud registration [6].

Our method, dubbed MinkLoc3D, has a simple, elegant and effective architecture and outperforms prior state-of-the-art. MinkLoc3D consists of two parts, local feature extraction network followed by feature aggregation layer. In order to produce local features with richer semantic content, we adapted Feature Pyramid Network (FPN) [18] architecture. The input point cloud is first quantized into a sparse voxelized representation and processed by a local feature extraction network. Different than prior methods we use a simple Generalized-Mean pooling [18] layer, instead of NetVLAD [2] layer, to aggregate local features into a discriminative global point cloud descriptor.

MinkLoc3D achieves state-of-the-art results on standard 3D place recognition benchmarks. It outperforms PointNetVLAD [1] by a large margin. It improves over the current state-of-the-art LPD-Net [19] despite having simpler architecture and being more computationally effective. Comparison with vision-based RobotCar Seasons [34] benchmark proves its robustness to challenging environmental conditions.

Our main contribution is the development of a global point cloud descriptor extraction method based on an alternate point cloud representation and network architecture than prior state-of-the-art. Our MinkLoc3D method advances state-of-the-art on the popular benchmarks. It proves the potential of using sparse voxelized representation and sparse convolutions for efficient extraction of discriminative features from 3D point clouds. We believe our work can spark further improvements in the point cloud-based place recognition field by showing promising development direction.

2. Related work

Point cloud representation for deep learning. Early deep learning methods for 3D point cloud processing use volumetrically discretized representations [22]. It's a natural extension of 2D image representation as a grid of pixels and 3D convolutions can be used to effectively process such data. However, such representation is very inefficient. The memory requirement grows cubically as spatial resolution increases, making it inappropriate for processing larger point clouds.

PointNet [28] is the first deep learning architecture operating directly on raw 3D point clouds. Each point is processed in isolation by multi-layer perceptrons and point features are aggregated using a symmetric max pooling function. This makes the architecture independent from input points ordering. The drawback is that it cannot capture local geometric structures and has limited ability to recog-

nize fine-grained patterns. To alleviate this problem, PointNet++ [29] enhances PointNet with hierarchical processing.

An alternative is to use sparse voxelized representation [12]. This allows using 3D convolutions to effectively capture local structures and patterns, similarly as 2D convolutions do in 2D images. However, naive implementations are computationally inefficient. Recently, an auto-differentiation library for sparse tensors, so called Minkowski Engine ², was proposed [7]. It efficiently implements sparse convolutions by using coordinate hashing. Sparse voxelized representation proved successful and yield state-of-the-art results in many 3D vision tasks, such as local feature extraction [8] and semantic segmentation [7].

3D point cloud-based place recognition using learned global features.

PointNetVLAD [1] is the first deep network for large-scale 3D point cloud retrieval. It combines PointNet [28] architecture to extract local features and NetVLAD [2] layer to aggregate local features and produce a discriminative global descriptor. The main weakness of PointNetVlad is its reliance on PointNet [28] for local feature extraction. PointNet architecture is weak at capturing local geometric structures which adversely impacts discriminability of the resultant global descriptor. To overcome this weakness, latter methods enhance PointNetVlad, by adding different mechanism to extract local contextual information.

PCAN [41] adds an attention mechanism to predict significance of each point based on a local context. The input point cloud is first processed using PointNet architecture to compute local features. Then, sampling and grouping approach inspired by PointNet++ [29] is used to extract local contextual information at multiple scales and produce per-point attention map. Finally, NetVLAD [2] layer aggregates attention-weighted local features into a global descriptor.

DAGC [36] combines dynamic graph CNN [39] architecture with dual attention mechanism [11] to aggregate local contextual information at multiple scales. Local features are aggregated using NetVLAD [2] layer to produce a global descriptor.

LPD-Net [19] relies on handcrafted features and uses graph neural networks to extract local contextual information. First, ten handcrafted features, such as local curvature or point density, are computed for each point. Then, 3D points enhanced with handcrafted features are processed using Point Net architecture, fed to a graph neural network to aggregate neighbourhood features and further processed using Point Net-like architecture. Finally, global descriptor is computed using NetVLAD [2] layer. The method yields state-of-the-art results, surpassing previously proposed solutions by a significant margin. However, at the expense of architectural complexity and high computational cost.

²<https://github.com/NVIDIA/MinkowskiEngine>

DH3D [9] is a recent 6DoF relocation method operating on 3D point clouds. It unifies global place recognition and local 6DoF pose refinement by inferring local and global 3D descriptors in a single pass through the network. The local feature extraction module uses Flex Convolution (FlexConv) [13] and Squeeze-and-Excitation (SE) [15] blocks to fuse multi-level spatial contextual information and channel-wise feature correlations into local descriptors. NetVLAD [2] layer aggregates attention-weighted local features into a global point cloud descriptor.

Deep metric learning. Deep metric learning [20] uses deep neural networks to compute a non-linear mapping from a high dimensional data point space to a low-dimensional Euclidean space, known as a representation or embedding space. The learned mapping preserves semantic similarity between objects. This technique is widely used in many recognition tasks in computer vision domain, such as pedestrian re-identification [14] and image retrieval [17]. Early deep metric learning methods use a Siamese architecture trained with a contrastive loss [3]. Latter methods propose more complex loss functions, such as triplet [14] or quadruplet [5] loss. Significant attention is put to a selection of an effective sampling scheme to choose informative training samples, so called hard negatives mining [40]. One of the most popular schemes is *batch hard* negative mining proposed in [14], which constructs training triplets by selecting the hardest positive and negative examples within each mini-batch. In the last few years a number of more sophisticated loss function formulations and sampling schemes was proposed [40, 38, 4]. However, recent works [26, 31] suggest that their advantage over classic contrastive or triplet margin loss is moderate at best. Based on these findings we choose triplet margin loss when training our network.

3. MinkLoc3D: global point cloud descriptor for place recognition

Our goal is to compute a discriminative and generalizable global descriptor from the input point cloud given as an unordered set of 3D coordinates. This section describes the proposed architecture and training process of the network computing such descriptor.

3.1. Network architecture

Our network has a very simple architecture shown in Fig. 2, yet it proved to be more effective and efficient than state-of-the-art methods on standard benchmarks. It consists of two parts: local feature extraction network and generalized-mean (GeM) pooling [30] layer. Input point cloud $P = \{(x_i, y_i, z_i)\}$, in the form of a set of 3D point coordinates, is first quantized into a single channel sparse ten-

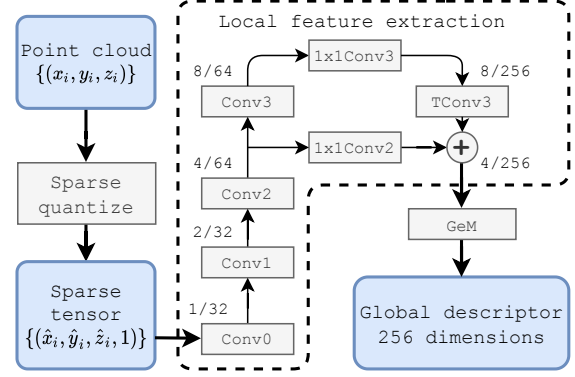


Figure 2. MinkLoc3D architecture. The input point cloud is quantized into a sparse, single channel, 3D tensor. Local features are extracted using a 3D Feature Pyramid Network [18] architecture. Generalized-mean (GeM) [30] pooling produces a global point cloud descriptor. Numbers in local feature extraction module (e.g. 1/32) denote a stride and number of channels of a feature map produced by each block.

sor $\hat{P} = \{(\hat{x}_i, \hat{y}_i, \hat{z}_i, 1)\}$. The values of this single channel are set to one for non-empty voxels. The sparse tensor is fed to the local feature extraction network, which produces a sparse 3D feature map $\hat{F} = \left\{ \left(\hat{x}_j, \hat{y}_j, \hat{z}_j, f_j^{(1)}, \dots, f_j^{(c)} \right) \right\}$, where c is a feature dimensionality (256 in our experiments), $\hat{x}_j, \hat{y}_j, \hat{z}_j$ quantized coordinates and $f_j^{(1)}, \dots, f_j^{(c)}$ features of j -th feature map element. The sparse 3D feature map \hat{F} is pooled using a generalized-mean (GeM) pooling [30] layer, which produces a global descriptor vector \mathbf{g} . GeM is generalization of a global max pooling and global average pooling operators and is defined as: $g^{(k)} = \left(\frac{1}{n} \sum_{j=1 \dots n} \left(f_j^{(k)} \right)^p \right)^{\frac{1}{p}}$, where $g^{(k)}$ is k -th element of the global descriptor vector \mathbf{g} , n is a size (number of non-zero elements) in the sparse local feature map \hat{F} , $f_j^{(k)}$ is k -th feature of the j -th local feature map element and p is a learnable pooling parameter.

The design of the local feature extraction network is inspired by MinkowskiNet [7] sparse convolutional network architecture, and Feature Pyramid Network [18] design pattern. Bottom-up part of the network contains four convolutional blocks producing sparse 3D feature maps with decreasing spatial resolution and increasing receptive field. The top-down part contains a transposed convolution generating an upsampled feature map. Upsampled feature map is concatenated with the skipped features from the corresponding layer of the bottom-up pass using a lateral connection. Such design is intended to produce a feature map with relatively high spatial resolution and large receptive field. Our initial experiments proved its advantage over a simple convolutional architecture without top-down processing.

Tab. 1 shows details of each convolutional block in a lo-

Block	Layers
Conv0	$\mathbb{C}_{5_k 1_s}^{32}$
Conv1	$\mathbb{C}_{2_k 2_s}^{32} \langle \mathbb{C}_{3_k 1_s}^{32} \mathbb{C}_{3_k 1_s}^{32} \rangle$
Conv2	$\mathbb{C}_{2_k 2_s}^{64} \langle \mathbb{C}_{3_k 1_s}^{64} \mathbb{C}_{3_k 1_s}^{64} \rangle$
Conv3	$\mathbb{C}_{2_k 2_s}^{64} \langle \mathbb{C}_{3_k 1_s}^{64} \mathbb{C}_{3_k 1_s}^{64} \rangle$
1x1Conv2	$\mathbb{C}_{1_k 1_s}^{256}$
1x1Conv3	$\mathbb{C}_{1_k 1_s}^{256}$
TConv3	$\mathbb{C}_{t 2_k 2_s}^{256}$

Table 1. Details of a local feature extraction part of MinkLoc3D network. All convolutions in bottom-up Conv0...3 blocks are followed by batch norm and ReLU non-linearity. $\langle \dots \rangle$ denotes a residual block.

cal feature extraction network. We use notation introduced in [35], where $\mathbb{C}_{a_k b_s}^c$ denotes a convolution with c kernels of shape $a \times a \times a$ and stride b . t decorator is used to indicate a transposed convolution. $\langle \dots \rangle$ denotes a residual block with a skip connection, defined as $\langle f \rangle(x) \doteq f(x) + x$. The first convolutional block (Conv0) has bigger $5 \times 5 \times 5$ kernels, in order to aggregate information from a larger neighbourhood. Bottom-up blocks (Conv1, Conv2 and Conv3) are made of a stride two convolution, which decreases spatial resolution by two, followed by residual block consisting of two convolutional layers with $3 \times 3 \times 3$ kernel. All convolutional layers in bottom-up blocks are followed by batch normalization [16] layer and ReLU non-linearity. Two 1x1Conv blocks have the same structure, both contain a single convolutional layer with $1 \times 1 \times 1$ kernel. The aim of these blocks is to unify the number of channels in feature maps produced by bottom-up blocks, before they are merged in the top-down pass through the network. The top-down part of the network consists of a single transposed convolution layer (TConv3) with $2 \times 2 \times 2$ kernel.

3.2. Network training

To train our network we use a deep metric learning approach [20] with a triplet margin loss [14] defined as:

$$L(a_i, p_i, n_i) = \max \{d(a_i, p_i) - d(a_i, n_i) + m, 0\},$$

where $d(x, y) = \|x - y\|_2$ is an Euclidean distance between embeddings x and y ; a_i, p_i, n_i are embeddings of an anchor, a positive and a negative elements in i -th training triplet and m is a margin hyperparameter. The loss function is minimized using a stochastic gradient descent approach with Adam optimizer.

Previous methods, such as PointNetVLAD [1] and LPD-Net[19], use rather inefficient training strategy. In order to construct informative triplets, for each anchor point cloud they sample 2 positive and 18 negative candidates. Embeddings of all candidate point clouds are calculated and only one hardest positive and negative example is taken to construct a training triplet. Thus, 21 point clouds need to be processed to construct one triplet.

We developed an alternative, more efficient, training procedure based on batch hard negative mining approach [14]. At the beginning of each epoch we randomly partition a training set into batches. A batch of size n is constructed by sampling $n/2$ pairs of structurally similar elements. After a batch is constructed, we compute two $n \times n$ boolean masks, one indicating structurally similar pairs and the other structurally dissimilar. We use hash-based indexing to efficiently check if two elements are structurally similar, dissimilar or similarity is indefinite. Then, the batch is fed to the network to compute embeddings. Using similarity and dissimilarity boolean masks and computed embeddings, we mine hardest positive and hardest negative examples and construct informative training triplets. In our approach, processing one batch consisting of n elements produces n training triplets. This approach brings down network training time from days to hours.

During experiments we noticed that with larger batch sizes, the training process is prone to collapse, where all embeddings approach the same value. To overcome this problem, we use a simple yet effective dynamic batch sizing strategy. The training starts with a small batch size, say 16 examples. At the end of each epoch, the average number of active triplets (i.e. triplets producing non-zero loss) per batch is examined. If the ratio of active triplets to all triplets in a batch falls below the predefined threshold Θ , the batch size is increased by a fixed batch expansion rate α .

To increase variability of the training data and reduce overfitting, we apply on-the-fly data augmentation. It includes random jitter with a value drawn from a normal distribution $\mathcal{N}(\mu = 0, \sigma = 0.001)$; random translation by a value sampled from $0 \dots 0.01$ range; and removal of randomly chosen points, where the percentage of points to remove is uniformly sampled from $0 \dots 10\%$ range. We also adapted a random erasing augmentation [42] and randomly remove all points within a fronto-parallel cuboid with a random size and position.

4. Experimental results

In this section we describe the datasets and evaluation methodology, compare our method to the state-of-the-art and conduct ablation study. We also compare our method with image-based visual localization method on the standard visual localization benchmark.

4.1. Datasets and evaluation methodology

The network is trained and evaluated using a modified Oxford RobotCar dataset and three in-house datasets: University Sector (U.S.), Residential Area (R.A.) , Business District (B.D.) introduced in [1]. The datasets are created using a LiDAR sensor mounted on the car travelling through these four regions at different times of day and year. Oxford RobotCar dataset is build using SICK LMS-151 2D LiDAR scans and in-house dataset using Velodyne HDL-64 3D LiDAR.

All point clouds are preprocessed with the ground planes removed and downsampled to 4096 points. The point coordinates are shifted and rescaled to be zero mean and inside the $[-1, 1]$ range. See Fig. 4 for exemplary data items. Training tuples are generated using ground truth UTM coordinates. Structurally similar point clouds are at most 10m apart. Dissimilar point clouds are at least 50m apart. For point clouds between 10 and 50m apart similarity is indefinite. Each dataset is split into disjoint training and test subsets. For more information please refer to [1].

Same as in previous works, we evaluate our network in two scenarios. In *baseline* scenario, the network is trained using the training subset of Oxford dataset and evaluated on test splits of Oxford and in-house datasets. In *refined* scenario, the network is trained on the training subset of Oxford and in-house datasets; and evaluated on test splits of Oxford and in-house datasets. The number of training and test elements used in each scenario is shown in Tab. 2.

	Baseline Dataset		Refined Dataset	
	Training	Test	Training	Test
Oxford	21.7k	3.0k	21.7k	3.0k
In-house	-	4.5k	6.7k	1.7k

Table 2. Number of elements in datasets used in *baseline* and *refined* evaluation scenarios.

Evaluation metrics We follow the same evaluation protocol as in [1, 19]. A point cloud from a testing dataset is taken as a query and point clouds from different traversals that cover the same region form the database. The query point cloud is successfully localized if at least one of the top N retrieved database clouds is within $d = 25$ meters from the ground truth position of the query. $Recall@N$ is defined as the percentage of correctly localized queries. As in [1] we report Average Recall@1 (AR@1) and Average Recall@1% (AR@1%) metrics.

Implementation details. In all experiments we quantize 3D point coordinates with 0.01 quantization step. As point coordinates in the Baseline and Refined datasets are normalized to be in $[-1, 1]$ range, this produces up to 200 voxels in

each spatial direction. Other parameters of the training process are listed in Tab.3. Initial learning rate is divided by 10 at the epoch given in LR scheduler steps row. The Refined Dataset is larger and more diverse than Baseline Dataset, hence in refined scenario the network is trained twice as long. The dimensionality of the resultant global descriptor is set to 256 , same as in prior methods.

	Baseline	Refined
Initial batch size	32	16
Batch size limit	256	256
Batch expansion threshold (Θ)	0.7	0.7
Batch expansion rate (α)	1.4	1.4
Number of epochs	40	80
Initial learning rate	1e-3	1e-3
LR scheduler steps	30	60
L_2 weight decay	1e-3	1e-3
Triplet loss margin (m)	0.2	0.2

Table 3. Parameters of the training process in *baseline* and *refined* evaluation scenarios.

All experiments are performed on a server with a single nVidia RTX 2080Ti GPU, 12 core AMD Ryzen Threadripper 1920X processor, 64 GB of RAM and SSD hard drive. We use PyTorch 1.5 [27] deep learning framework, MinkowskiEngine 0.4.3 [7] auto-differentiation library for sparse tensors and PML Pytorch Metric Learning library 0.9.88 [26].

4.2. Results and discussion

Comparison with state-of-the-art. We compare performance our global descriptor with prior art: PointNetVLAD [1], PCAN [41], DAGC [36] and LPD-Net [19]. We also include DH3D [9] method in an evaluation. DH3D is a recent 6DOF localization method, which includes a global point cloud descriptor computation as a part of the pose estimation pipeline.

Tab. 4 compares performance of our MinkLoc3D with state-of-the-art methods trained on the Baseline Dataset. When evaluated on Oxford dataset, MinkLoc3D wins with AR@1% 3.0 p.p. higher than the runner-up, LPD-Net. When evaluated on three in-house datasets, it performs slightly worse compared than LPD-net (1.0 and 0.6 p.p. worse for U.S. and B.D. sets respectively and 0.7 p.p. better for R.A.). It must be noted that Oxford dataset and three in-house datasets were acquired using LiDARs with different characteristics. Even so, our method yields comparable results to LPD-Net which relies on hand-crafted features. MinkLoc3D discriminability and generalization capability is significantly higher than all other fully learning based methods.

Tab. 5 shows evaluation results of state-of-the-art methods trained on a larger and more diverse Refined Dataset.

	Oxford	U.S.	R.A.	B.D.
PointNetVLAD [1]	80.3	72.6	60.3	65.3
PCAN [41]	83.8	79.1	71.2	66.8
DH3D-4096 [9]	84.3	-	-	-
DAGC [36]	87.5	83.5	75.7	71.2
LPD-Net [19]	94.9	96.0	90.5	89.1
MinkLoc3D (our)	97.9	95.0	91.2	88.5

Table 4. Evaluation results (Average Recall at 1%) of place recognition methods trained on the Baseline Dataset.

For PointNetVLAD and PCAN we run the evaluation using the trained models provided by authors. LPD-Net was trained from scratch and evaluated on Refined Dataset using the open source code. Our MinkLoc3D is a clear winner. Compared to the state-of-the-art LPD-Net, the AR@1% is higher between 0.8 p.p. and 2.9 p.p. on all evaluation subsets. The advantage over other methods is even higher, between 5-18 p.p. Average Recall plots in Fig. 3 show that our method outperforms previous methods on all evaluation subsets.

Tab. 6 compares the number of trainable parameters and inference time (runtime per cloud). Our MinkLoc3D method is significantly faster compared to LPD-Net. LPD-Net requires time consuming preprocessing of the input point cloud to compute 10 handcrafted features. Even without including hand-crafted feature extraction time, LPD-Net has longer inference time compared to MinkLoc3D (26 vs 22 ms). Our model is also much lighter compared to prior methods. It has only 1.5 million trainable parameters, whereas other methods have an order of magnitude more. This can be explained by the fact, that our method produces informative local features, that can be pooled using a simple Generalized-Mean pooling [25] which has few learnable parameters. Other methods use NetVLAD [2] aggregation layer with millions of learnable parameters.

Figure 4 visualizes nearest neighbour search results using our MinkLoc3D descriptor in Oxford evaluation subset. The leftmost column shows a query point cloud and other columns show its five nearest neighbours. Figure 5 shows failure cases. More visualizations of nearest neighbour search results can be found in Supplementary Material.

Ablation study. In this section we investigate impact of the network design choices on the discriminativity and generalization capability of our method. In all experiments, the network is trained using the Baseline Dataset and evaluated on Oxford and three in-house datasets (U.S., R.A. and B.D.).

Tab. 7 shows the impact of a feature aggregation method on the performance of the global descriptor. The following methods are evaluated: global max pooling (MAC), Generalized-Mean (GeM) pooling [30], NetVLAD [2] and

NetVLAD with Context Gating [25] (NetVLAD-CG). Surprisingly, a simple GeM layer with few learnable parameters produces the most discriminative global descriptors and has the best generalization capability. More sophisticated methods, NetVLAD and NetVLAD with Context Gating, score similarly on Oxford dataset, but noticeably worse on in-house datasets. This can be attributed to two factors. First, our training datasets have a moderate size. Using NetVLAD layer, with millions of trainable parameters, increases the risk of overfitting. Second, our local feature extraction network works very well and produces informative features, that can be effectively pooled using a simple GeM layer to produce a discriminative global descriptor.

Tab. 8 shows impact of a descriptor size on the discriminability of the global descriptor. The number of channels in lateral connections (1x1Conv2, 1x1Conv3 blocks) and in a transposed convolution TConv3 block is set to the same value as the dimensionality of the final descriptor. Parameters of bottom-up convolutional blocks remain unchanged. The network performance is relatively similar with larger descriptor sizes (between 64 and 512) with AR@1% between 97.3 and 98.0% on Oxford dataset and between 90.3 and 93.2% on in-house datasets. The performance deteriorates, when the descriptor size falls to 32.

Comparison with image based methods. In this paragraph we compare performance of our MinkLoc3D with state-of-the-art image-based place recognition and visual localization methods in challenging environmental conditions. The comparison is done using RobotCar Seasons [34] dataset. It contains outdoor images captured in the city of Oxford at various periods of a year in different atmospheric conditions, e.g. snow, rain, dawn or night.

We compare MinkLoc3D performance against place recognition methods based on a global image descriptor: DenseVLAD [37] and NetVLAD [2]; and against full 6DoF (6 degree-of-freedom) relocalization methods: NetVLAD+SP [32], DenseVLAD+D2-Net [10] and NetVLAD+SP+SG [33]. For each image in RobotCar Seasons dataset, we find LiDAR readings with corresponding timestamps in the in original RobotCar dataset [21] and construct the point cloud. Then, we use MinkLoc3D network to compute a global point cloud descriptors and link this descriptor with a corresponding image. To approximate a 6DoF pose of a query image, we search for a database image which descriptor (computed from its corresponding point cloud), is closest to the descriptor of a query image (computed from a corresponding point cloud). Then, we return the known pose of a database image as an approximation of the query image pose. Retrieved poses are evaluated using the online evaluation service at *Long-term visual localization* site.³

³<https://www.visuallocalization.net/>

	Oxford		U.S.		R.A.		B.D.	
	AR@1%	AR@1	AR@1%	AR@1	AR@1%	AR@1	AR@1%	AR@1
PointNetVLAD [1]	80.1	63.3	94.5	86.1	93.1	82.7	86.5	80.1
PCAN [41]	86.4	70.7	94.1	83.7	92.3	82.3	87.0	80.3
DAGC [36]	87.8	71.5	94.3	86.3	93.4	82.8	88.5	81.3
LPD-Net [19]	94.9	86.6	98.9	94.4	96.4	90.8	94.4	90.8
MinkLoc3D (our)	98.5	94.8	99.7	97.2	99.3	96.7	96.7	94.0

Table 5. Evaluation results (Average Recall at 1% and at 1) of place recognition methods trained on the Refined Dataset.

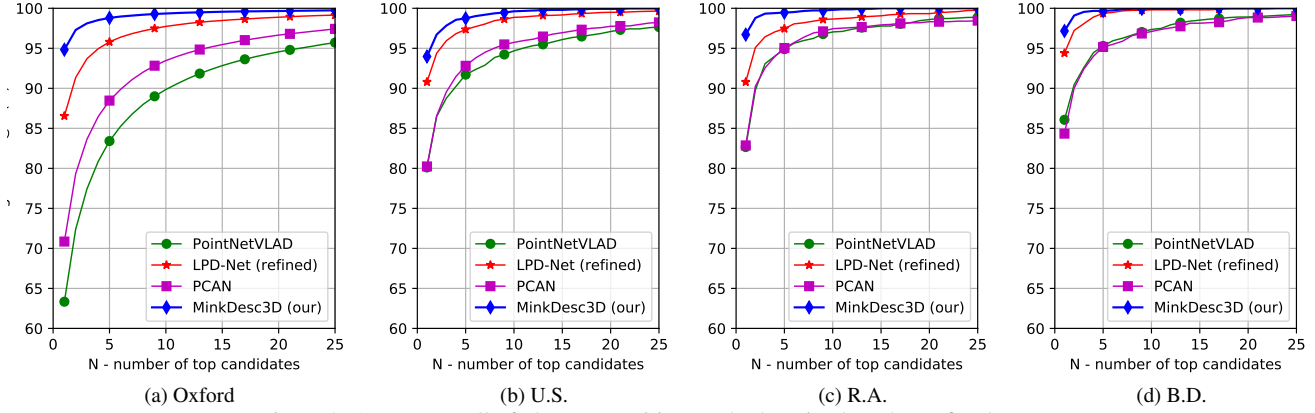


Figure 3. Average recall of place recognition methods trained on the Refined Dataset.

	Parameters	Runtime per cloud
PointNetVLAD [1]	19.8M	15 ms
PCAN [41]	20.4M	55 ms
LPD-Net [19]	19.8M	26 ms
LPD-Net [19] with f.e.	19.8M	917 ms
MinkLoc3D (our)	1.1M	21 ms

Table 6. Computation time required by different methods. *LPD-Net with f.e.* includes hand-crafted features extraction time.

Descriptor size	Oxford AR@1%	In-house AR@1%
512	98.0	92.6
* 256	97.9	93.2
128	97.5	91.5
64	97.3	90.3
32	95.8	86.4

Table 8. Impact of a descriptor size on the discriminability of the global descriptor. The network is trained on the Baseline Dataset.

Architecture	Oxford AR@1%	In-house AR@1%
MinkFPN+MAC	97.3	92.4
* MinkFPN+GeM	97.9	93.2
MinkFPN+NetVLAD	97.0	91.1
MinkFPN+NetVLAD-CG	97.2	84.7

Table 7. Impact of a feature aggregation method on the discriminability of the global descriptor. The network is trained on the Baseline Dataset. * indicates MinkLoc3D architecture.

Results are shown in Tab. 9. In day conditions image-based methods perform generally better, with up to 10 p.p. more correctly localized queries. However, both LiDAR based methods (ours and LPD-Net) operate on rel-

ative small, downsampled point clouds with 4096 points. Even with this small number of points both methods perform reasonably well. Also both LiDAR based methods approximate 6DoF pose by taking the pose of the closest nearest neighbour found. Full 6DoF localization methods, NetVLAD+SP, DenseVLAD+D2 Net and NetVLAD+SP+SG, employ much more sophisticated approach, where candidate matches found using a global descriptor are filtered by matching local features with geometric consistency criteria. In night conditions point cloud-based method show their potential. Our method surpass all image-based methods by a large margin with the exception of the latest NetVLAD+SP+SG, which has slightly higher performance.

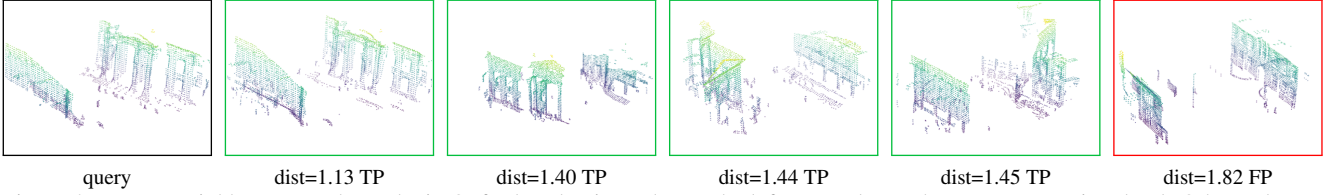


Figure 4. Nearest neighbours search results in Oxford evaluation subset. The leftmost column shows a query point cloud. Other columns show its five nearest neighbours. *dist* is an Euclidean distance in the descriptor space. TP indicates true positive and FP false positive.

	day conditions							night conditions	
	dawn	dusk	overcast summer	overcast winter	rain	snow	sun	night	night-rain
DenseVLAD [37]	92.5	94.2	92.0	93.3	96.9	90.2	80.2	19.9	25.5
NetVLAD [2]	82.6	92.9	95.2	92.6	96.0	91.8	86.7	15.5	16.4
NetVLAD+SP [32]	90.3	96.7	98.1	96.2	97.6	95.9	94.1	35.4	33.4
DenseVLAD+D2-Net [10]	94.4	95.9	98.3	96.2	96.9	94.9	91.1	53.9	56.1
NetVLAD+SP+SG [33]	97.3	97.2	99.8	96.7	98.1	97.8	96.1	91.9	92.0
LPD-Net [19]	79.7	79.9	79.7	73.8	-	-	82.3	77.3	32.8
MinkLoc3D (our)	89.2	88.3	90.3	83.1	66.3	86.3	87.4	86.1	58.0

Table 9. Comparisons with 6DoF visual localization methods on RobotCar Seasons dataset. We report percentage of queries correctly localized within 5 meter and 10° threshold. Five top rows show performance of image-based methods and two bottom rows LiDAR scan-based methods.

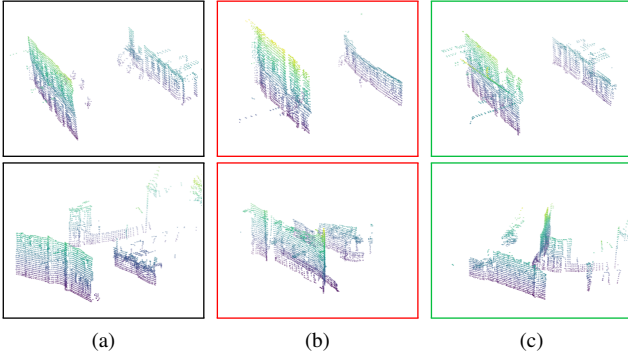


Figure 5. Failure cases. Examples of unsuccessful retrieval results using our network. (a) is the query point cloud, (b) incorrect match to the query and (c) the closest true match.

5. Conclusion

In this paper we present MinkLoc3D, a novel 3D point cloud descriptor, based on a sparse voxelized point cloud representation and 3D FPN [18] architecture. Extensive experimental evaluation proves that it outperforms prior cloud-based place recognition methods. The success of our method can be attributed to two factors. First, sparse convolutional architecture can produce informative local features, that can be used to construct a discriminative global point cloud descriptor. Second, improvements in the training process allows efficient and effective training with larger batch size, which positively affects discriminabil-

ity and generalization capability of the resultant descriptor. The natural next step is to enhance the proposed method to build full 6DoF localization solution.

It should be also noted, that achieved results (AR@1% between 96.7% and 99.4% when trained on Refined Dataset) show that standard benchmarks used to train and evaluate point cloud-based place recognition methods are close to being saturated and there's a little room for improvement. Larger and more diverse datasets would be needed to instigate further progress.

Acknowledgements The project was funded by POB Research Centre for Artificial Intelligence and Robotics of Warsaw University of Technology within the Excellence Initiative Program - Research University (ID-UB).

References

- [1] Mikaela Angelina Uy and Gim Hee Lee. Pointnetvlad: Deep point cloud based retrieval for large-scale place recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4470–4479, 2018.
- [2] Relja Arandjelovic, Petr Gronat, Akihiko Torii, Tomas Pa-jdla, and Josef Sivic. Netvlad: Cnn architecture for weakly supervised place recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5297–5307, 2016.
- [3] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature verification using a"

- siamese” time delay neural network. In *Advances in neural information processing systems*, pages 737–744, 1994.
- [4] Fatih Cakir, Kun He, Xide Xia, Brian Kulis, and Stan Sclaroff. Deep metric learning to rank. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1861–1870, 2019.
 - [5] Weihua Chen, Xiaotang Chen, Jianguo Zhang, and Kaiqi Huang. Beyond triplet loss: a deep quadruplet network for person re-identification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 403–412, 2017.
 - [6] Christopher Choy, Wei Dong, and Vladlen Koltun. Deep global registration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2514–2523, 2020.
 - [7] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3075–3084, 2019.
 - [8] Christopher Choy, Jaesik Park, and Vladlen Koltun. Fully convolutional geometric features. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 8958–8966, 2019.
 - [9] Juan Du, Rui Wang, and Daniel Cremers. Dh3d: Deep hierarchical 3d descriptors for robust large-scale 6dof relocalization. In *European Conference on Computer Vision (ECCV)*, 2020.
 - [10] Mihai Dusmanu, Ignacio Rocco, Tomas Pajdla, Marc Pollefeys, Josef Sivic, Akihiko Torii, and Torsten Sattler. D2-net: A trainable cnn for joint description and detection of local features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8092–8101, 2019.
 - [11] Jun Fu, Jing Liu, Haijie Tian, Yong Li, Yongjun Bao, Zhiwei Fang, and Hanqing Lu. Dual attention network for scene segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3146–3154, 2019.
 - [12] Ben Graham. Sparse 3d convolutional neural networks. In *Proceedings of the British Machine Vision Conference (BMVC)*, pages 150.1–150.9. BMVA Press, September 2015.
 - [13] Fabian Groh, Patrick Wieschollek, and Hendrik PA Lensch. Flex-convolution. In *Asian Conference on Computer Vision*, pages 105–122. Springer, 2018.
 - [14] Alexander Hermans, Lucas Beyer, and Bastian Leibe. In defense of the triplet loss for person re-identification. *arXiv preprint arXiv:1703.07737*, 2017.
 - [15] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018.
 - [16] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. volume 37 of *Proceedings of Machine Learning Research*, pages 448–456, Lille, France, 07–09 Jul 2015. PMLR.
 - [17] Jung-Eun Lee, Rong Jin, and Anil K Jain. Rank-based distance metric learning: An application to image retrieval. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008.
 - [18] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.
 - [19] Zhe Liu, Shunbo Zhou, Chuanzhe Suo, Peng Yin, Wen Chen, Hesheng Wang, Haoang Li, and Yun-Hui Liu. Lpd-net: 3d point cloud learning for large-scale place recognition and environment analysis. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2831–2840, 2019.
 - [20] Jiwen Lu, Junlin Hu, and Jie Zhou. Deep metric learning for visual understanding: An overview of recent advances. *IEEE Signal Processing Magazine*, 34(6):76–84, 2017.
 - [21] Will Maddern, Geoff Pascoe, Chris Linegar, and Paul Newman. 1 Year, 1000km: The Oxford RobotCar Dataset. *The International Journal of Robotics Research (IJRR)*, 36(1):3–15, 2017.
 - [22] Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 922–928. IEEE, 2015.
 - [23] Colin McManus, Winston Churchill, Will Maddern, Alexander D Stewart, and Paul Newman. Shady dealings: Robust, long-term visual localisation using illumination invariance. In *2014 IEEE international conference on robotics and automation (ICRA)*, pages 901–906. IEEE, 2014.
 - [24] Sven Middelberg, Torsten Sattler, Ole Untzelmann, and Leif Kobbelt. Scalable 6-dof localization on mobile devices. In *European conference on computer vision*, pages 268–283. Springer, 2014.
 - [25] Antoine Miech, Ivan Laptev, and Josef Sivic. Learnable pooling with context gating for video classification. *arXiv:1706.06905*, 2017.
 - [26] Kevin Musgrave, Serge Belongie, and Ser-Nam Lim. A metric learning reality check. *arXiv preprint arXiv:2003.08505*, 2020.
 - [27] Adam Paszke et al. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach et al., editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.
 - [28] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.
 - [29] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in neural information processing systems*, pages 5099–5108, 2017.
 - [30] Filip Radenović, Giorgos Tolias, and Ondřej Chum. Fine-tuning cnn image retrieval with no human annotation. *IEEE transactions on pattern analysis and machine intelligence*, 41(7):1655–1668, 2018.

- [31] Karsten Roth, Timo Milbich, Samarth Sinha, Prateek Gupta, Björn Ommer, and Joseph Paul Cohen. Revisiting training strategies and generalization performance in deep metric learning, 2020.
- [32] Paul-Edouard Sarlin, Cesar Cadena, Roland Siegwart, and Marcin Dymczyk. From coarse to fine: Robust hierarchical localization at large scale. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12716–12725, 2019.
- [33] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superglue: Learning feature matching with graph neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4938–4947, 2020.
- [34] Torsten Sattler, Will Maddern, Carl Toft, Akihiko Torii, Lars Hammarstrand, Erik Stenborg, Daniel Safari, Masatoshi Okutomi, Marc Pollefeys, Josef Sivic, et al. Benchmarking 6dof outdoor visual localization in changing conditions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8601–8610, 2018.
- [35] Władysław Skarbek. Symbolic tensor neural networks for digital media—from tensor processing via bnf graph rules to creams applications. *Fundamenta Informaticae*, 168(2-4):89–184, 2019.
- [36] Qi Sun, Hongyan Liu, Jun He, Zhaoxin Fan, and Xiaoyong Du. Dage: Employing dual attention and graph convolution for point cloud based place recognition. In *Proceedings of the 2020 International Conference on Multimedia Retrieval*, pages 224–232, 2020.
- [37] Akihiko Torii, Relja Arandjelovic, Josef Sivic, Masatoshi Okutomi, and Tomas Pajdla. 24/7 place recognition by view synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1808–1817, 2015.
- [38] Xun Wang, Xintong Han, Weilin Huang, Dengke Dong, and Matthew R Scott. Multi-similarity loss with general pair weighting for deep metric learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5022–5030, 2019.
- [39] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog)*, 38(5):1–12, 2019.
- [40] Chao-Yuan Wu, R Manmatha, Alexander J Smola, and Philipp Krahenbuhl. Sampling matters in deep embedding learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2840–2848, 2017.
- [41] Wenxiao Zhang and Chunxia Xiao. Pcan: 3d attention map learning using contextual information for point cloud based retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12436–12445, 2019.
- [42] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. *arXiv preprint arXiv:1708.04896*, 2017.