KPPR: Exploiting Momentum Contrast for Point Cloud-Based Place Recognition

Louis Wiesmann Lucas Nunes Jens Behley Cyrill Stachniss

***Abstract*—Place recognition plays an important role in robot localization and SLAM. Being able to retrieve the current position in a given map allows, for instance, localizing without relying on GPS reception. In this paper, we address the problem of point cloud-based place recognition, we especially focus on reducing the often significant training time needed by learning-based approaches. We propose a novel neural network architecture that first extracts local features using a pre-trained encoder network plus a stem architecture. The local features are aggregated to a global descriptor, which allows us to compute the similarity between locations. In line with with several existing approaches, we target the generation of descriptors, which are similar for spatially near locations and dissimilar to other places. By exploiting the recent success of feature banks, we are able to bypass the computation of the negative examples, which enables faster training, bigger batch sizes, or the use of more sophisticated networks. As a key result, able to speed up the training process by a factor of 17 against the most common training procedure while increasing also the performance.**

***Index Terms*—Localization, Deep Learning Methods**

I. Introduction

GLOBAL localization is a common component in many robotic systems and autonomous driving. Place recognition allows for finding loop closures for SLAM systems [35] or to localize in a given prerecorded map without relying on GPS reception. In this work, we address the problem of point cloud-based place recognition where we want to retrieve our current location in a given map. LiDAR-based place recognition is often solved by comparing a global descriptor from the query location to the descriptors in a database [14], [20], [49]. The descriptors are usually generated by aggregating information of local features which themselves are computed by neural networks. The training of those networks is typically done by maximizing the descriptor similarity between spatially near locations and minimizing the similarity of different locations [4], [10], [14], [19], [20], [41], [46].

Finding hard cases, i.e., those observations/places that look very similar but are in fact far away from each other, is key for obtaining competitive performance. Usually, most of the compute is spend to find those hard negatives and to avoid training on trivial examples. Many approaches look for each query position up to eighteen other point clouds [24], [41], [46] and therefore sacrifice over 80% of their training time to search for hard examples. The performance usually increases with the amount of negatives used for training (as illustrated in Fig. 1 for our approach in blue). However, the training time increases disproportionally with respect to the gain in performance. Avoiding the computation of the negatives has the potential to allow for faster training, bigger batch sizes, more sophisticated networks, or simply for more efficient research cycles.

The main contribution of this paper is a novel approach for point cloud-based place recognition, called Kernel Point Place

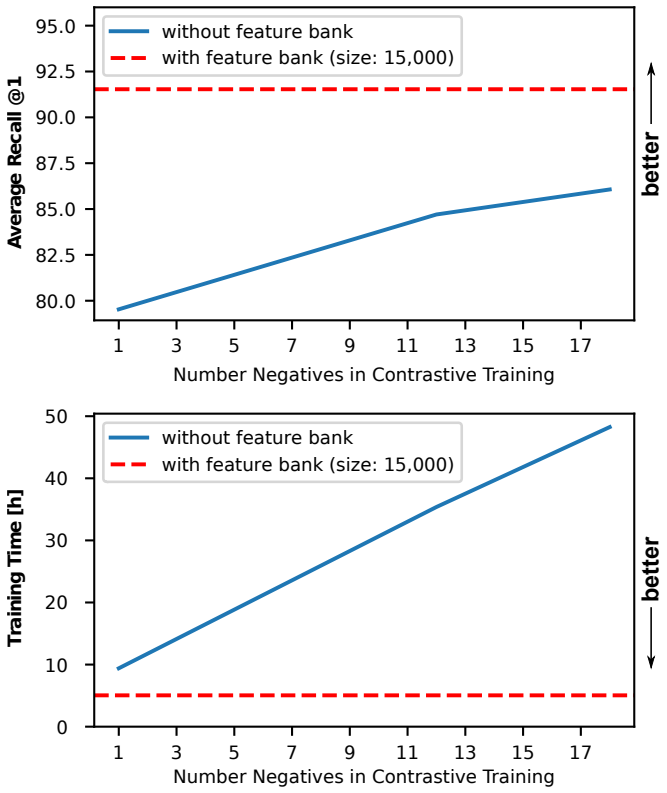


Fig. 1: The performance of learning-based place recognition systems often scales with the number of negatives used for the contrastive training. Here, we show the place recognition performance with (red dashed) and without (blue solid) the usage of a feature bank while training. The more negatives, the better the performance in terms of average recall (top), but notably the training increases disproportionally with respect to the gain in performance (bottom).

Manuscript received: Aug 24, 2022; Revised: Oct 28, 2022; Accepted: Nov 25, 2022. This paper was recommended for publication by Editor Sven Behnke upon evaluation of the Associate Editor and Reviewers' comments.

This work has partially been funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy, EXC-2070 – 390732324 – PhenoRob, by the European Union's Horizon research and innovation programme under grant agreement No 101070405 (DigiForest).

All authors are with the University of Bonn, Germany. Cyrill Stachniss is additionally with the Department of Engineering Science at the University of Oxford, UK, and with the Lamarr Institute for Machine Learning and Artificial Intelligence, Germany.

Corresponding author: louis.wiesmann@igg.uni-bonn.de

Digital Object Identifier (DOI): see top of this page.

Recognition (KPPR), which is a novel architecture for point cloud-based retrieval. We set a special focus on efficient training and inference which will be reflected in the choice of our architecture as well as in the training procedure. To the best of our knowledge, we are the first that utilize momentum contrast and feature banks in the context of point cloud-based place recognition. This allows us to achieve higher performance and faster training (as depicted in Fig. 1, red dashed line). The source code of our implementation as well as the pretrained models are available at https://github.com/PRBonn/kppr.

II. Related Work

Place recognition tackles the problem of retrieving the current position based on observations (so-called queries) in a given map (also called database). Images have often been used to represent the local surrounding of the query position and the entries in the database [27], [30], [32], [43], [44], [45]. Nowadays, more and more point cloud-based approaches emerge, which are usually less prone to appearance changes caused by illumination conditions or seasonal changes [41].

The similarity between positions is often computed based on the similarity of global descriptors. Those descriptors can be computed by aggregating local features through bag of words [8], [33], [34], vector-of-locally-aggregated-descriptors (VLAD) [17], or the differentiable version NetVLAD [1] for learning-based approaches [41]. The local features can be computed using classical handcrafted methods [2], [26], [31], [36], [37] or learned by neural networks [4], [6], [10], [15], [21], [49]. Most of the learning-based approaches utilize convolutional networks, which typically operate on graphs [22], [38], sparse voxel grids [4], [20], range images [6], or directly on the points [40]. The performance can be additionally improved by taking sequential information into account to resolve ambiguities [23], [28]. Networks that are pre-trained on different tasks can provide useful features, which can then be used not only for place recognition [11], [39], [50], but also for other tasks. Similar to our prior work [46], we use a compression encoder [47] for feature enhancement, which allows us to retrieve our position directly in a compressed map. In contrast to our prior work [46], we use in this work a convolutional stem architecture instead of attention [16], [42], [45].

Training such networks can be very compute intensive due to the amount of negative examples [24], [41], [46]. Komorowski et al. [20] try to avoid unnecessary computations by mining over the whole batch and dynamic batching to prevent the descriptors to collapse. Hui et al. [14] on the other hand focus on efficient inference by training a smaller student network with a bigger teacher model. We on the other hand, exploit feature banks and a momentum encoder from momentum contrast [13] which allows us to use an arbitrary number of negatives that comes at basically no cost. Other contrastive learning approaches focus on large batch size training [5], online clustering [3], or mining strategies [48] to deal with negatives. In contrast, Zbontar et al. [51] does not need any of those techniques, but rather simple introduces a loss based on the cross correlation. A different direction in the unsupervised domain is to train entirely without negatives. To optimize for only positive examples one either uses a momentum encoder [12] or stopping certain gradients [7]. Since in the unsupervised domain, the positive examples are usually only augmented views and the negatives are different images, not pushing away negatives that might actually be structural similar might be an advantage. In the case of place recognition, which is usually done supervised, we know which ones are positives and which ones are negatives, therefore the risk is smaller to optimize for false negatives.

III. Our Approach for Place Recognition

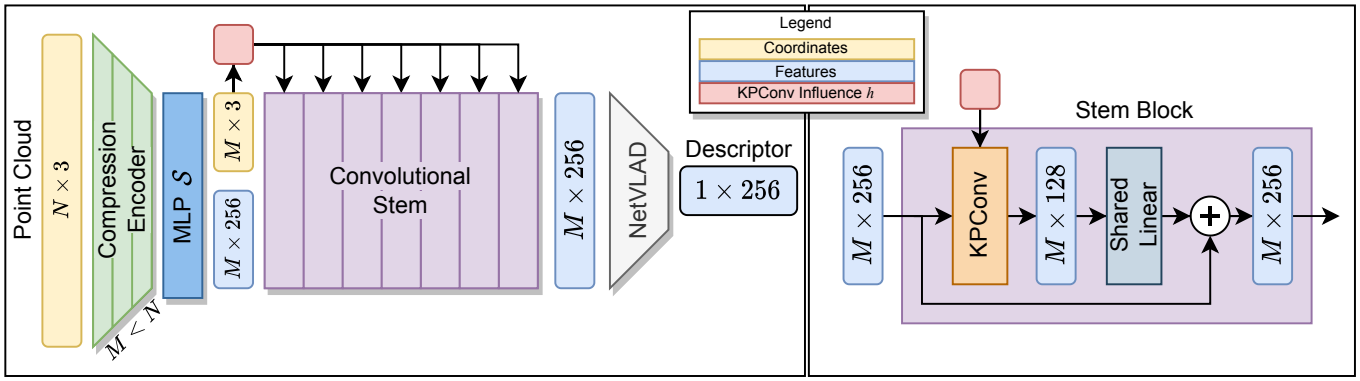
In this work, we propose a novel neural network architecture for point cloud-based place recognition with special focus on efficient training and inference. We follow the common paradigm [4], [20], [41], [46] of first computing local features, which are then aggregated into a global descriptor. A query location can in the end be retrieved by comparing its descriptor to the ones in the database. The point clouds are first fed into a compression encoder to create a compact representation. A convolutional stem enhances the features by increasing the receptive field, which are then aggregated by NetVLAD [1] into the global descriptor. For the training, we use use a contrastive loss with entropy-based regularization and utilize feature banks to increase the number of negatives we can use. The network architecture is illustrated in Fig. 2 (a) and will be explained in more detail in the following.

A. Compression Encoder

Local point clouds, as they are commonly used in the domain of autonomous vehicles and mobile robotics, can easily contain hundred thousands of points. To deal with these amount of points, we utilize a frozen pretrained convolutional encoder from a compression network [47], which provides us with multiple benefits, such as a reduced number of points with descriptive features, but also the ability to efficiently store the compressed point clouds.

The encoder E generates for a given point cloud $\mathbf{X} \in \mathbb{R}^{N\times 3}$ a sparse representation consisting of points $\mathbf{X}_c \in \mathbb{R}^{M\times 3}$ with their associated features $\mathbf{F}_c \in \mathbb{R}^{M\times D_c}$. Here, N is the number of input points, M the number of compressed points with $N < M$ and D_c is the dimensionality of the compression features. By training directly on the compressed representation, we can bypass the biggest computational part of the early convolutions, but due to the additional features we do not lose so much of the fine grained information compared to simple downsampling. The input can be reconstructed from the compressed representation $\{\mathbf{X}_c, \mathbf{F}_c\}$, which will be preserved through freezing the encoder while training.

The features F_c from the compression encoder are not necessarily well suited for our task. Therefore, instead of fine-tuning the weights of E, we use a small shared multi-layer perceptron (MLP) $S : \mathbb{R}^{M\times(3+D_c)} \rightarrow \mathbb{R}^{M\times D_o}$ to transform the compression specific features into a features space, which is better suited for place recognition. This MLP S consists of three layers with ReLu as non-linearity and layer normalization. The input to the the MLP are not only the features $\mathbf{F}_c$ but also the points $\mathbf{X}_c$ which can be seen as a learned positional encoding.



(a) Network Architecture (b) Convolutional Stem Block

Fig. 2: Our proposed architecture (a) consists of a compression encoder to create a sparse lightweight representation which will be enhanced by a convolutional stem. The resulting local features will be aggregated by NetVLAD into one global descriptor. The inputs to our networks are the coordinates of a point cloud. The descriptors can be used to compare point clouds in the descriptor space using the cosine similarity. The Convolutional stem consists of ResNet-like KPConv blocks (b) with precomputed neighborhoods and weights for faster inference.

B. Convolutional Stem

The features from the encoder contain local information but lack a broader context due to the small receptive field of the encoder. Therefore, we propose to use a convolutional stem to increase the receptive field of view. The convolutional stem consists of J ResNet-like KPConv blocks $K : \mathbb{R}^{M \times D_o} \rightarrow \mathbb{R}^{M \times D_o}$, which do not further subsample the already sparse point cloud.

In the following, we will first briefly revisit the concept of KPConv [40], to then show how we can disentangle it into a block dependent and a block-independent part. We will use the superscript to denote block-dependent variables, e.g., $\mathbf{F}^j$ are the features in the j$^{\text{th}}$ block.

For a point $\mathbf{x} \in \mathbf{X}_c$, the convolution of the features $\mathbf{F}^{j-1}$ with the convolutional kernel g is defined as

$$\mathbf{f}^j = (\mathbf{F}^{j-1} * g^j)(\mathbf{x}) = \sum_{\mathbf{x}_i \in \mathcal{N}^j(\mathbf{x})} g(\mathbf{x}_i - \mathbf{x})^j \mathbf{f}_i^{j-1}, \quad (1)$$

where $\mathcal{N}^j(\mathbf{x}) = \{\mathbf{x}_i \in \mathbf{X}^j \mid \|\mathbf{x}_i - \mathbf{x}\| < r^j\}$ are all the points in the neighborhood within the radius $r^j \in \mathbb{R}$. The kernel g is defined by a linear combination of the weights $\{\mathbf{W}_k^j \mid k < K\}$ of the K kernel points $\{\mathbf{x}_k^j \mid k < K\}$:

$$h(\mathbf{x}_i - \mathbf{x}, \mathbf{x}_k^j) = \max\left(0, 1 - \frac{\|\mathbf{x}_i - \mathbf{x} - \mathbf{x}_k^j\|}{\sigma}\right) \quad (2)$$

$$g(\mathbf{x}_i - \mathbf{x})^j = \sum_{k<K} h\big(\mathbf{x}_i - \mathbf{x}, \mathbf{x}_k^j\big) \mathbf{W}_k^j, \quad (3)$$

where its coefficients h decrease linearly with the distances from the neighbors to the kernel points.

In contrast to the original KPConv implementation, we can make the following simplifications due to the stem architecture. First, we do not further subsample the points, therefore the coordinates in the point cloud stay the same, i.e., $\mathbf{X}^j = \mathbf{X}^{j-1} = \mathbf{X}$. Second, the neighborhoods will remain the same by always using the same radius $r \in \mathbb{R}$. Third, we arrange the kernel points always in a grid structure such that the coefficients are independent of the block. Hence, $h(\mathbf{x}_i - \mathbf{x}, \mathbf{x}_k^j) = h(\mathbf{x}_i - \mathbf{x}, \mathbf{x}_k)$, which allows us to precompute the term for all J blocks. This includes the quite costly kNN search for the neighborhoods.

Consequently, the only variables that are changing in each block $j \in \{1, \dots, J\}$ are the kernel weights $\mathbf{W}_k^j$ and the features of the target point cloud $\mathbf{F}^{j-1}$. All KPConv blocks are implemented in a ResNet-styled fashion (as illustrated in Fig. 2 (b)) with ReLu activation and layer normalization, similar to Thomas et al. [40].

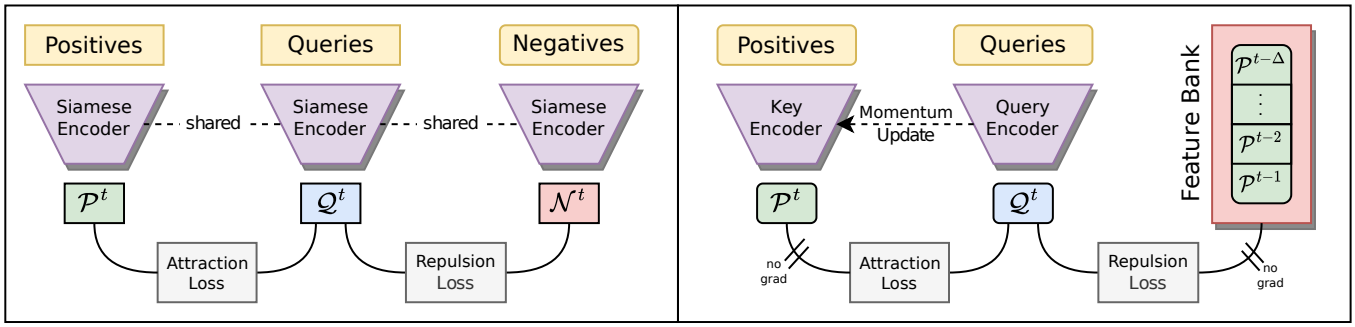
C. Feature Aggregation

The result of the convolutional stem is a point cloud $\mathbf{X}_c$ with their associated local features $\mathbf{F}_s$. In the end, we want one single global descriptor $\mathbf{d}$, that we can use to compare with the descriptors of other point clouds to search for the closest match. Therefore, we need to aggregate the local features into one descriptor vector $\mathbf{d} \in \mathbb{R}^{D_o}$. For this, we use the standard NetVLAD [1] $A : \mathbb{R}^{M \times D_o} \rightarrow \mathbb{R}^{D_o}$ layer which modifies VLAD to be fully differentiable and learnable by introducing a soft assignment between the centroids and the local features $\mathbf{F}$. Notably, our descriptors get normalized such that $\|\mathbf{d}\| = 1$. Without the normalization, we could observe the same behavior as training without the momentum encoder: the vectors diverge.

D. Loss Function

The global descriptors that we extract from the point clouds should have the properties that descriptors from the same location (positives) should be similar while being dissimilar to descriptors from other locations (negatives). To optimize for this objective, we use the additive supervised contrastive loss with differential entropy regularization similar to El-Nouby et al. [9]. For a query descriptor, $\mathbf{q} \in \mathbb{R}^{D_o}$, a set of positive descriptors $\mathcal{P}$ and the feature bank $\mathcal{B}$ the contrastive part $\mathcal{L}_c$ is defined as

$$\mathcal{L}_c(\mathbf{q}, \mathcal{P}, \mathcal{B}) = \frac{1}{|\mathcal{P}|} \sum_{\mathbf{p} \in \mathcal{P}} \left(1 - \mathbf{q}^\top \mathbf{p}\right) + \frac{1}{\eta} \sum_{\mathbf{b} \in \mathcal{B}} \mathbb{1}_{\mathbf{b}} \mathbf{q}^\top \mathbf{b}, \quad (4)$$



(a) Training **without** a Feature Bank (b) Training **with** a Feature Bank

Fig. 3: Training procedures without (a) and with (b) using a feature bank. Both methods use an attraction loss to make descriptors from the queries $\mathcal{Q}$ to the corresponding positives $\mathcal{P}$ similar while repulsing the negatives $\mathcal{N}$ away from $\mathcal{Q}$. When using the classical training procedure (a) one computes the queries $\mathcal{Q}$, positives $\mathcal{P}$, and negatives $\mathcal{N}$ all with the same siamese network. The network is trained by backpropagation through all the descriptors. With the feature banks (b) a second network (query encoder) is introduced to compute the positives $\mathcal{P}$. This network gets updated using a momentum update rather than backpropagation. the negatives are not computed per query but are taken from the past positives. The superscript denotes the index at which time the descriptors are computed.

where $\mathbb{1}_b$ indicates whether $\mathbf{b}$ is in the negatives $\mathcal{N}$ of $\mathbf{q}$ and is within the margin β, i.e.,

$$\mathbb{1}_{\mathbf{b}} = \begin{cases} 1 & \text{, if } \mathbf{b} \in \mathcal{N} \text{ and } \mathbf{q}^\top \mathbf{b} > \beta \\ 0 & \text{, otherwise,} \end{cases} \tag{5}$$

and $\eta = \sum_{\mathbf{b} \in \mathcal{B}} \|\mathbb{1}_{\mathbf{b}}\|_1$ is the number of times, where $\mathbb{1}_{\mathbf{b}}$ is 1.

The regularization loss is an entropy-based repulsion loss $\mathcal{L}_r$ to prevent descriptors to collapse

$$\mathcal{L}_r(\mathbf{q}, \mathbf{d}^*) = -\log\left(\frac{1 - \mathbf{q}^\top \mathbf{d}^*}{2}\right), \tag{6}$$

where $\mathbf{d}^* = \{\text{argmax}(\mathbf{q}^\top \mathbf{d}) \mid \mathbf{d} \in \mathcal{P} \cup \mathcal{B}\}$ is the most similar descriptor. In contrast to El-Nouby et al. [9], we use the cosine distance instead of the Euclidean distance to reuse intermediate results of $\mathcal{L}_c$. The final loss $\mathcal{L} = \mathcal{L}_c + \alpha \mathcal{L}_r$ is the sum of both loss terms with α to weight them accordingly.

E. Feature Banks and Momentum Encoder

The recent success of momentum contrast [13] in the domain of unsupervised representation learning motivated us to apply the ideas of a feature bank and momentum encoder to the field of point cloud-based place recognition.

Traditionally, the positives $\mathcal{P}$ and negatives $\mathcal{N}$ are computed by the same network as the query $\mathbf{q}$, as visualized in Fig. 3 (a). Due to the huge amount of negatives, this requires a lot of compute time. In contrast, when using the feature banks, the negatives do not have to be computed online, but rather the former positives get recycled. The feature bank $\mathcal{B}$ is a queue of descriptors, buffering the latest Δ descriptors from the previous positive examples $\mathcal{P}$. By this, the computation of the negatives can be bypassed and therefore saves a lot of compute.

The loss $\mathcal{L}_c$ tries to maximize the similarity between the positives and minimize it for the negatives. By rapidly changing the weights of the network, the positive descriptors can simply diverge from the negatives, just learning that the current descriptors should be different from the negatives. To prevent this, He et al. [13] introduced a second encoder, the so-called key encoder, for computing the positives $\mathcal{P}$. The weights w_{key} of the key encoder are not updated by backpropagation but with a momentum update of the weights w_{query} from the query network:

$$w^t_{\text{key}} = \gamma\, w^{t-1}_{\text{key}} + (1 - \gamma)\, w^t_{\text{query}}. \tag{7}$$

The superscript denotes the index of the batch. The weights will be updated at the beginning of the forward pass of each batch. Only slowly updating the key encoder shall prevent the divergence of the positives with respect to the negatives [13]. The gradients from the feature bank are disabled, which additionally saves compute and memory. Backpropagating through tenth of thousands of descriptors would be infeasable, even for most modern accelerators. The pipeline for training with the feature banks is illustrated in Fig. 3 (b). In contrast to He et al. [13], our approach is supervised, therefore allowing us to only use descriptors of the feature bank as negatives when they are true negatives. For this, we additionally store the indices of the point clouds in the feature bank, and check for each query which descriptors belong to point clouds from negative positions, which will then be accounted for in the loss, see Eq. (5). By using the feature banks, we are able to increase the number of negatives drastically without scaling up in compute and memory. We discard the key encoder after training and we only use the query encoder for the online inference.

IV. Experimental Evaluation

The main focus of this work lies on point cloud-based place recognition. In the following, we will evaluate the performance of our approach and its great benefits for point cloud-based place recognition in terms of recognition accuracy and training efficiency. Furthermore, we show ablation studies to validate our design choices.

A. Experimental Setup

The aim of our approach is to reliably retrieve the position in a given map based on the point cloud of the local

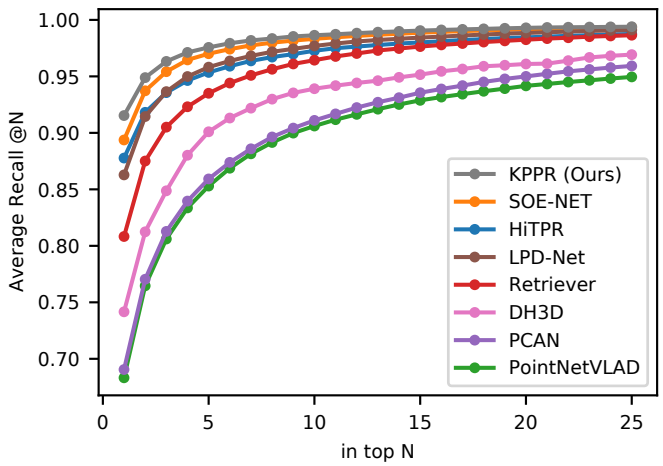


Fig. 4: Average recall @N on the Oxford Robocar dataset. Our approach is able to reliably retrieve the position of query point clouds in a given database.

surrounding. For the evaluation, we use the classic datasets Oxford Robocar [29] and the three In-House datasets [41]. We follow the common train/test splits and the evaluation metric average Recall R at a specific threshold τ as in [41]. The recall will be denoted as $R@\tau$, e.g., $R@1$ is how often a positive database descriptor is within the top-1 most similar descriptors while $R@1\%$ denotes within the top-1%. Additionally, we will provide the training time for the ablation studies, since one of our key aims is to reduce the training time. We use the following parameters unless stated differently. The output feature size of the shared MLP and the Stem blocks, as well as for the global descriptor are set to $D_o = 256$, while the intermediate kernel point convolution outputs 128-dimensional features (similar to [40]). We normalize the input coordinates to lay within -1 and 1. The radius for the convolution is set to $r = 0.05$. We use $J = 7$ stem blocks and a Feature Bank size of $|\mathcal{B}| = 15,000$. The weights w_q get updated with a momentum of $\gamma = 0.999$ and for the loss, we use $\alpha = 0.3$ as well as $\beta = 0.5$. To enable batched training, we pad the compressed point clouds to always have the same number of points. Padded points get masked out in the blocks accordingly to not affect the training. We use ADAMw [25] with a learning rate of 10^{-5}, which will be reduced to 10^{-8} in a cosine annealing schedule. When training with the feature bank, we use a batch size of 32 while only being able to use a batch size of 16 for batch negative mining and a batch size of 3 when training without the feature bank. All our experiments are trained on the same machine with an Nvidia RTX A6000 for a fair comparison.

B. Place Recognition Performance

In the first experiment, we analyze the performance of our approach with respect to the baselines. In Fig. 4, we show the results on the Oxford Robocar dataset [29]. We are able to outperform all other methods throughout the different recall rates. To evaluate the generalizability, we show the results (see Tab. I) on different datasets while the networks are only trained on Oxford Robocar [29]. We are able to outperform all the approaches on three out of four datasets. We are between 2.0 and 3.6 percent points better than the second best approaches on those datasets, while only being 0.8 percent points worse than MinkLoc3D [20] on Oxford. In Fig. 5 are our results for the B.D. dataset visualized, while in Fig. 6 we visualize some succeeding and failure cases.

Method	Oxford	U.S.	R.A.	B.D
PointNetVLAD[1] [41]	85.21	74.80	73.39	71.96
PCAN[2] [52]	83.81	79.05	71.18	66.82
Retriever[2] [46]	92.22	91.88	87.44	85.53
HiTPR[2] [46]	94.64	94.01	89.11	88.31
LPD-Net[2] [24]	94.92	96.00	90.46	89.14
SOE-Net[2] [49]	96.40	93.17	91.47	88.45
SVT-Net[3] [10]	97.80	96.50	92.70	90.70
MinkLoc3D[3] [20]	**97.90**	95.00	91.20	88.50
KPPR (Ours)	97.08	**98.01**	**95.10**	**92.09**

TABLE I: Average recall for finding the queries within the top-1% of the databases. All models have *only* been trained on the Oxford Robocar dataset to show their generalization capabilities. Our approach outperforms the baselines on three out of four datasets and is not far from the best one in the first dataset (Oxford).

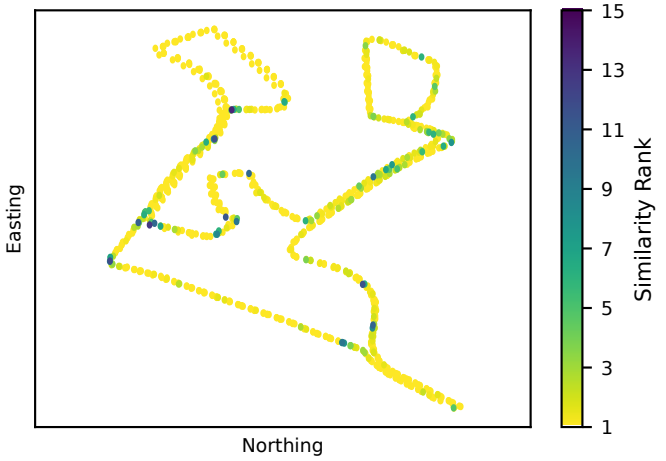


Fig. 5: Qualitative results for our approach on the B.D. dataset. The color of the points denote at which position the most similar positive was ranked within the database (the brighter the better). Most of the positions are successfully retrieved by only looking on the most similar descriptor.

C. Ablation Studies

In the following, we will show ablation studies to show the impact of the proposed design choices and provide deeper insights. The ablation studies have been evaluated using a second feature bank of size $|\mathcal{B}| = 500$ on the validation dataset on Oxford Robocar [29].

1) Negative Mining and Loss Function: In this experiment, we analyze the impact of different mining strategies. The first part focuses on the performance and how it is affected by the choice of the loss function, while the second part focuses on the impact on the training time. In Tab. II are the results for our network trained with different loss functions and under

[1]Approach has been retrained using own implementation. Due to better results, we report those numbers rather than the original ones from the paper.
[2]Numbers provided by the authors or from the original repositories.
[3]Numbers from the original papers.

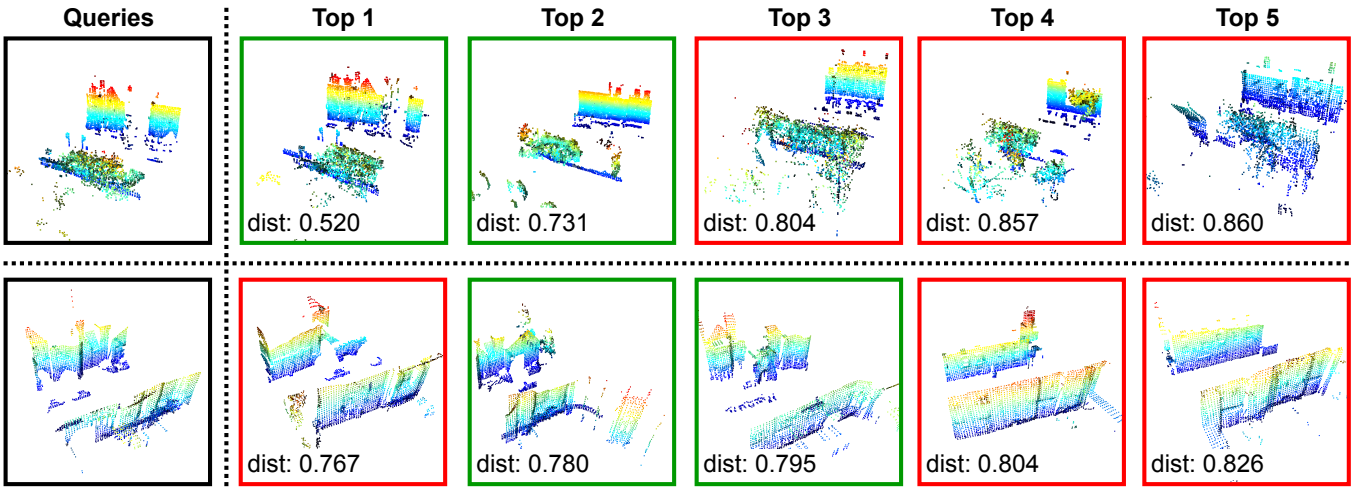


Fig. 6: Top 5 database results for the provided query point clouds. A green border denotes a correct match while a red border is from a different location. Additionally, the Euclidean distance between the descriptors are provided. In the top row our approach successfully retrieved both True positives. In the second row we have shown a failure case where the closest descriptor does not correspond to the same location. We can see, that in the succeeding case the positives have a substantially smaller distance, while in the failure case the distances are more similar.

different mining strategies. Namely, first the classical method of computing for each query 18 negatives ([A] - [C]). The second method is batch negative mining (similar to [20]) where the negatives are searched within the positives of the batch ([D] - [F]). For a batch size of 16 with two positives per query, we have 32 negatives. False negatives are masked out in the same way as for the feature banks (see Eq. (5)). Finally, using the feature banks (FB) as descriped in Sec. III-E ([G] - [I]). For the loss functions, we evaluate with the lazy Triplet [41], lazy Quadruplet[41], and additive contrastive loss [9] with (here denoted as Entropy) and without (denoted as Contrastive) entropy regularization. Notably, the Quadruplet loss cannot be trained with the feature bank since the descriptors of the negatives do not have gradients.

Independently of the mining strategy, we can see that the contrastive loss with the entropy regularization ([C], [F], [I]) outperforms the networks trained with different losses. Batch negative mining outperforms the classic method but is throughout worse than using the feature bank. From the training time perspective, we see the huge impact of using a mining strategy rather than computing the negatives classically. Additionally, the performance increases, which is likely due to the higher amount of negatives (FB: 15,000, Batch: 32, Classic: 18) as well as a bigger batch size (FB: 32, Batch: 16, Classic: 3), therefore compensating for the problem of comparing descriptors that got computed at different stages in the training. The training time using the feature banks is lower than the batch negative mining, even though the substantially larger amount of negatives. The reason for this is that the gradients are not computed for the negatives in the feature bank but are still needed for the batch negative mining. When comparing the contrastive loss with entropy regularization [I] against the training without the regularization [H], we can see that the regularization boosts the performance. Throughout our experiments, we did not see that the regularization also helps for the other losses. Training with the exponential contrastive loss [18] was more unstable and led to worse results in our experiments.

TABLE II: Ablation: Negative Mining and Loss Function

	Loss	Mining	R@1	R@1%	Time
[A]	Triplet	Classic	85.49%	93.59%	95.36h
[B]	Quadruplet	Classic	84.94%	93.74%	102.06h
[C]	Entropy	Classic	86.07%	94.63%	96.50h
[D]	Triplet	Batch	86.41%	93.96%	6.68h
[E]	Contrastive	Batch	85.69%	93.68%	6.93h
[F]	Entropy	Batch	89.04%	95.74%	7.71h
[G]	Triplet	FB	89.91%	95.71%	5.49h
[H]	Contrastive	FB	88.73%	95.51%	5.50h
[I]	Entropy	FB	**91.53%**	**97.08%**	5.05h

TABLE III: Ablation: Feature Bank Size

	#FB Size	R@1	R@1%	Time
[J]	10,000	91.05%	96.66%	4.91h
[K]	15,000	**91.53%**	**97.08%**	5.05h
[L]	20,000	90.44%	96.15%	4.91h

2) Feature Bank Size: In Tab. III are the results with respect to varying feature bank sizes. The best result can be achieved by a feature bank with 15,000 descriptors [K], which corresponds to 2/3 of the size of the training set. As we can see the training time does not vary a lot for changing feature bank sizes. Throughout the experiments, where we not use the feature banks, i.e., when computing the negatives explicitly, it scales linearly with the number of negatives (as in Fig. 1). Training with more negatives is not feasible due to limiting memory resources of the GPU.

3) Architecture: In this section, we analyze different parts of our network architecture. In Tab. IV are the results for different numbers of blocks in the convolutional stem as well as the network without using the pre-trained compression encoder. Seven convolutional blocks provide the best

TABLE IV: Ablation: Architecture

	#Blocks	Compr.	R@1	R@1%	Time
[M]	7	✗	88.99%	95.66%	10.87h
[N]	0	✓	68.00%	82.04%	0.21h
[O]	5	✓	89.83%	95.92%	3.69h
[P]	7	✓	**91.53%**	**97.08%**	5.05h
[Q]	9	✓	90.73%	96.39%	7.02h

TABLE V: Ablation: MinkLoc3D Backbone

	Loss	Mining	R@1	R@1%	Time
[R]	Triplet	Classic	71.05%	86.71%	30.07h
[S]	Triplet	Batch	73.94%	88.66%	2.72h
[T]	Triplet	FB	77.47%	89.96%	2.17h
[U]	Entropy	Classic	73.41%	89.14%	30.14h
[V]	Entropy	Batch	73.58%	88.86%	2.75h
[W]	Entropy	FB	**78.07%**	**90.97%**	2.15h

performance, additionally one can see that the training time increases with the number of blocks. Not having any further convolutional blocks [N] provides the worst results, showing that the convolutional stem is necessary to yield good results. On the other hand, it can be trained in under 13 min and already has a top 1 recall of 68%. We can see that using the compression encoder [P] provides better results and trains faster than without [M]. The slowing down of the training process can be explained by the increasing number of points in the point cloud. The decreasing performance shows that the features computed by the compression encoder are more valuable than the higher number of points. The inference of a single compressed point cloud runs at 95 Hz. Precomputing the neighborhoods $\mathcal{N}$ and h in the convolutional stem speeds-up the inference by 44.8%, i.e. being almost twice as fast.

4) Backbone: In this section, we investigage if we are able to apply the proposed methodology to a different backbone. For this experiment, we exchange the compression encoder plus the stem architecture with the sparse convolutional feature pyramid network from MinkLoc3D [20]. For the network architecture, we use the same hyperparameters as used in the original work [20] and also train for the suggested 50 epochs. Based on a parameter sweep, we use a learning rate of 10^{-4} and a batch size of 32 for these experiments. Additionally, we report the results for different negative mining techniques as in Sec. IV-C1. The results are depicted in Tab. V. First of all, we can see the same behaviors as for our network, i.e., using the feature banks ([T], [W]) improves both training time and performance. The entropy loss outperforms the triplet loss (as used in the original work [20]) also for this network architecture. When comparing the performance of our network Tab. II [I] with the feature pyramid network Tab. V [W], we see that our network provides substantially better results at the cost of training time. Our proposed network has roughly twice the amount of parameters, explaining the difference in runtime and performance. The results suggest that the proposed methodology might also increase the performance of other networks. Notably, the results with the feature pyramid network do not perform as well as in Komorowski et al. [20]. Likely due to the lack of data augmentation and using a different feature aggregation head, which we did not apply for comparability to isolate the impact of the backbone.

V. Future Work and Expected Broader Impact

In this work, we have utilized the ideas of feature banks in the context of place recognition. For us, this led to a significant reduction of the training time by a factor of up to 17 times, which enabled us to run more experiments and have faster research cycles. The presented experiments with the feature banks took around 47 h of training. If we would have done those experiments with computing the classical 18 negatives for each query as in [24], [41], [46], it would have taken us around 800 h. Our network architecture is designed for efficient inference and training but did not need to specially account for using the feature banks. Our experiments suggest that the advantage of the feature banks (more negatives with less compute) can be applied also for other kinds of place recognition architectures and approaches.

VI. Conclusion

In this paper, we presented a novel architecture for point cloud-based place recognition. Due to a pre-trained compression encoder, we not only save computational time but also create an intermediate memory efficient representation that can later be used for other downstream tasks such as decompression. Additionally, this enables us to perform place recognition directly on the compressed representation. For the feature enhancement, we propose to use a convolutional stem architecture. We disentangled some of the formulations in KPConv for our stem architecture to bypass redundant computations. Our method exploits feature banks and momentum contrast for efficient training and to improve performance. The usage of feature banks by recycling previously computed positives led to a speed up of up to 17 times compared to recomputing descriptors and hard negative mining. This allows us to successfully retrieve point clouds in a given database. We implemented and evaluated our approach on different datasets and provided comparisons to other existing techniques and supported all claims made in this paper.

References

[1] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, and J. Sivic. NetVLAD: CNN Architecture for Weakly Supervised Place Recognition. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[2] H. Bay, A. Ess, T. Tuytelaars, and L.V. Gool. Speeded-up robust features (SURF). *Journal of Computer Vision and Image Understanding (CVIU)*, 110(3):346–359, 2008.

[3] M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin. Unsupervised Learning of Visual Features by Contrasting Cluster Assignments. In *Proc. of the Conf. on Neural Information Processing Systems (NeurIPS)*, volume 33, pages 9912–9924, 2020.

[4] M. Chang, S. Yeon, S. Ryu, and D. Lee. SpoxelNet Spherical Voxel-Based Deep Place Recognition for 3D Point Clouds of Crowded Indoor Spaces. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2020.

[5] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A Simple Framework for Contrastive Learning of Visual Representations. In *Proc. of the Int. Conf. on Machine Learning (ICML)*, 2020.

[6] X. Chen, T. Läbe, A. Milioto, T. Röhling, O. Vysotska, A. Haag, J. Behley, and C. Stachniss. OverlapNet: Loop Closing for LiDAR-based SLAM. In *Proc. of Robotics: Science and Systems (RSS)*, 2020.

[7] X. Chen and K. He. Exploring Simple Siamese Representation Learning. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[8] L. Di Giammarino, I. Aloise, C. Stachniss, and G. Grisetti. Visual Place Recognition using LiDAR Intensity Information. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2021.

[9] A. El-Nouby, N. Neverova, I. Laptev, and H. Jégou. Training Vision Transformers for Image Retrieval. *arXiv preprint arXiv:2102.05644*, 2021.

[10] Z. Fan, Z. Song, H. Liu, Z. Lu, J. He, and X. Du. SVT-Net: Super Light-Weight Sparse Voxel Transformer for Large Scale Place Recognition. In *Proc. of the Conf. on Advancements of Artificial Intelligence (AAAI)*, 2022.

[11] S. Garg, N. Snderhauf, and M. Milford. Don't look back: Robustifying place categorization for viewpoint and condition-invariant place recognition. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2018.

[12] J.B. Grill, F. Strub, F. Altché, C. Tallec, P. Richemond, E. Buchatskaya, C. Doersch, B. Avila Pires, Z. Guo, M. Gheshlaghi Azar, et al. Bootstrap Your Own Latent - A New Approach to Self-Supervised Learning. In *Proc. of the Conf. on Neural Information Processing Systems (NeurIPS)*, 2020.

[13] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick. Momentum Contrast for Unsupervised Visual Representation Learning. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[14] L. Hui, M. Cheng, J. Xie, J. Yang, and M.M. Cheng. Efficient 3D Point Cloud Feature learning for Large-Scale Place Recognition. *IEEE Trans. on Image Processing*, 31:1258–1270, 2022.

[15] L. Hui, H. Yang, M. Cheng, J. Xie, and J. Yang. Pyramid Point Cloud Transformer for Large-Scale Place Recognition. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*, pages 6098–6107, 2021.

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

[17] H. Jégou, M. Douze, C. Schmid, and P. Pérez. Aggregating Local Descriptors into a Compact Image Representation. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[18] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, and D. Krishnan. Supervised Contrastive Learning. In *Proc. of the Conf. on Neural Information Processing Systems (NeurIPS)*, 2020.

[19] J. Knights, P. Moghadam, M. Ramezani, S. Sridharan, and C. Fookes. Incloud: Incremental learning for point cloud place recognition. *arXiv preprint arXiv:2203.00807*, 2022.

[20] J. Komorowski. Minkloc3d: Point cloud based large-scale place recognition. In *Proc. of the IEEE Winter Conf. on Applications of Computer Vision (WACV)*, 2021.

[21] X. Kong, X. Yang, G. Zhai, X. Zhao, X. Zeng, M. Wang, Y. Liu, W. Li, and F. Wen. Semantic Graph based Place Recognition for Point Clouds. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2020.

[22] L. Landrieu and M. Simonovsky. Large-scale Point Cloud Semantic Segmentation with Superpoint Graphs. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[23] C. Li, F. Yan, and Y. Zhuang. Sequence matching enhanced 3D place recognition using candidate rearrangement. *IET Cyber-Systems and Robotics*, 4(3), 2022.

[24] Z. Liu, S. Zhou, C. Suo, P. Yin, W. Chen, H. Wang, H. Li, and Y.H. Liu. LPD-Net: 3D Point Cloud Learning for Large-Scale Place Recognition and Environment Analysis. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*, 2019.

[25] I. Loshchilov and F. Hutter. Decoupled Weight Decay Regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[26] D. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *Intl. Journal of Computer Vision (IJCV)*, 60(2):91–110, 2004.

[27] S. Lowry and H. Andreasson. Lightweight, Viewpoint-Invariant Visual Place Recognition in Changing Environments. *IEEE Robotics and Automation Letters (RA-L)*, 3:957–964, 2018.

[28] J. Ma, X. Chen, J. Xu, and G. Xiong. SeqOT: A Spatial-Temporal Transformer Network for Place Recognition Using Sequential LiDAR Data. *arXiv preprint arXiv:2209.07951*, 2022.

[29] W. Maddern, G. Pascoe, C. Linegar, and P. Newman. 1 year, 1000 km: The oxford robotcar dataset. *Intl. Journal of Robotics Research (IJRR)*, 36(1):3–15, 2017.

[30] T. Naseer, W. Burgard, and C. Stachniss. Robust Visual Localization Across Seasons. *IEEE Trans. on Robotics (TRO)*, 34(2):289–302, 2018.

[31] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: an efficient alternative to sift or surf. In *Proc. of the IEEE Intl. Conf. on Computer Vision (ICCV)*, 2011.

[32] M. Shakeri and H. Zhang. Illumination Invariant Representation of Natural Images for Visual Place Recognition. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2016.

[33] T. Shan, B. Englot, F. Duarte, C. Ratti, and D. Rus. Robust Place Recognition using an Imaging Lidar. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2021.

[34] J. Sivic and A. Zisserman. Video Google: A Text Retrieval Approach to Object Matching in Videos. In *Proc. of the IEEE Intl. Conf. on Computer Vision (ICCV)*, 2003.

[35] C. Stachniss, J. Leonard, and S. Thrun. *Springer Handbook of Robotics, 2nd edition*, chapter Chapt. 46: Simultaneous Localization and Mapping. Springer Verlag, 2016.

[36] B. Steder, G. Grisetti, and W. Burgard. Robust Place Recognition for 3D Range Data Based on Point Features. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2010.

[37] B. Steder, M. Ruhnke, S. Grzonka, and W. Burgard. Place Recognition in 3D Scans Using a Combination of Bag of Words and Point Feature Based Relative Pose Estimation. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2011.

[38] H. Su, V. Jampani, D. Sun, S. Maji, E. Kalogerakis, M.H. Yang, and J. Kautz. SPLATNet: Sparse Lattice Networks for Point Cloud Processing. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[39] N. Sünderhauf, S. Shirazi, F. Dayoub, B. Upcroft, and M. Milford. On the Performance of ConvNet Features for Place Recognition. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2015.

[40] H. Thomas, C. Qi, J. Deschaud, B. Marcotegui, F. Goulette, and L. Guibas. KPConv: Flexible and Deformable Convolution for Point Clouds. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*, 2019.

[41] A. Uy and G. Lee. PointNetVLAD: Deep point cloud based retrieval for large-scale place recognition. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 4470–4479, 2018.

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

[43] O. Vysotska and C. Stachniss. Lazy Data Association For Image Sequences Matching Under Substantial Appearance Changes. *IEEE Robotics and Automation Letters (RA-L)*, 1(1):213–220, 2016.

[44] O. Vysotska and C. Stachniss. Effective Visual Place Recognition Using Multi-Sequence Maps. *IEEE Robotics and Automation Letters (RA-L)*, 4:1730–1736, 2019.

[45] R. Wang, Y. Shen, W. Zuo, S. Zhou, and N. Zheng. TransVPR: Transformer-Based Place Recognition with Multi-Level Attention Aggregation. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 13648–13657, 2022.

[46] L. Wiesmann, R. Marcuzzi, C. Stachniss, and J. Behley. Retriever: Point Cloud Retrieval in Compressed 3D Maps. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2022.

[47] L. Wiesmann, A. Milioto, X. Chen, C. Stachniss, and J. Behley. Deep Compression for Dense Point Cloud Maps. *IEEE Robotics and Automation Letters (RA-L)*, 6:2060–2067, 2021.

[48] C.Y. Wu, R. Manmatha, A.J. Smola, and P. Krahenbuhl. Sampling Matters in Deep Embedding Learning. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*, pages 2840–2848, 2017.

[49] Y. Xia, Y. Xu, S. Li, R. Wang, J. Du, D. Cremers, and U. Stilla. SOE-Net: A Self-Attention and Orientation Encoding Network for Point Cloud Based Place Recognition. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[50] J. Yue-Hei Ng, F. Yang, and L.S. Davis. Exploiting Local Features from Deep Networks for Image Retrieval. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[51] J. Zbontar, L. Jing, I. Misra, Y. LeCun, and S. Deny. Barlow Twins: Self-Supervised Learning via Redundancy Reduction. In *Proc. of the Int. Conf. on Machine Learning (ICML)*, pages 12310–12320, 2021.

[52] W. Zhang and C. Xiao. PCAN: 3D Attention Map Learning using Contextual Information for Point Cloud based Retrieval. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.