Array 28 (2025) 100569

Contents lists available at ScienceDirect

Array

journal homepage: www.elsevier.com/locate/array

MinkUNeXt: Point cloud-based large-scale place recognition using 3D sparse convolutions

Juan José Cabrera [a,*], Antonio Santo [a], Arturo Gil [a], Carlos Viegas [b], Luis Payá [a,c]

[a] Institute for Engineering Research (I3E), Miguel Hernández University, Spain
[b] Univ of Coimbra, ADAI, Department of Mechanical Engineering, Spain
[c] Valencian Graduate School and Research Network for Artificial Intelligence (valgrAI), Spain

ARTICLE INFO

Dataset link: https://juanjo-cabrera.github.io/projects-MinkUNeXt/

Keywords:
Place recognition
LiDAR
Point cloud embedding
3D sparse convolutions

ABSTRACT

This paper presents MinkUNeXt, an effective and efficient architecture for place-recognition from point clouds entirely based on the new 3D MinkNeXt Block, a residual block composed of 3D sparse convolutions that follows the philosophy established by recent Transformers but purely using simple 3D convolutions. Feature extraction is performed at different scales by a U-Net encoder–decoder network and the feature aggregation of those features into a single descriptor is carried out by a Generalized Mean Pooling (GeM). The proposed architecture demonstrates that it is possible to surpass the current state-of-the-art by only relying on conventional 3D sparse convolutions without making use of more complex and sophisticated proposals such as Transformers, Attention-Layers or Deformable Convolutions. A thorough assessment of the proposal has been carried out using the Oxford RobotCar, the In-house, the KITTI and the USyd datasets. As a result, MinkUNeXt proves to outperform other methods in the state-of-the-art. The implementation is publicly available at https://juanjo-cabrera.github.io/projects-MinkUNeXt/.

1. Introduction

In many applications, mobile robots must perform autonomous navigation in a specific environment. As it moves, the robot should be able to recognize or identify different areas of the environment. This action is equivalent to finding a correspondence between its current sensor observations and a part of the stored database. This ability is commonly denoted as place recognition. In order to speed this process, frequently, authors have concentrated on describing some parts of the environment by means of an invariant descriptor. In this way, the robot should be able to recognize a part of the environment by finding the descriptor in the database that most resembles the descriptor associated to its current observations. The concept of place recognition is of uttermost importance in tasks such as localization, mapping and navigation.

Place recognition and robot localization are two closely related concepts. Place recognition centres on the description of the current robot observations in a way that allows the robot to identify different locations in the map. Thus, place recognition focuses on the extraction and codification of relevant features found in the robot query observation in such a way that they can be compared to previously stored data (Fig. 1). Similarly, robot localization refers to the act of estimating the position and orientation of the robot within a known map. In this way, given a map of the robot, conformed by a series of submaps, a common process to carry out the global localization of the robot could consist of two phases [1]: (a) rapidly finding a submap within the global database using the feature descriptors (place recognition) and (b) performing a fine estimation of the position and orientation of the robot in that submap (robot localization). A similar technique is proposed in [2], where the descriptor is computed from Light Detection and Ranging (LiDAR) measurements. Next, a handcrafted descriptor is employed to rapidly retrieve some areas of interest in the map. The final localization step, based on the Iterative Closest Point (ICP) algorithm or other learnable methods [3] enable the computation of the position and orientation within the submap.

To date, place recognition has been performed with different types of sensors: visual cameras [4], laser [5], LiDAR [6] and Radar [7] using different types of techniques. For example, place recognition has been extensively solved by means of techniques based on the Bag of Words (BoW) algorithm using images [8,9].

During the last few years LiDAR sensors have lowered in price and weight, while increasing in resolution. Therefore, LiDAR sensors permit obtaining a large number of precise measurements from the environment that define its shape and structure. Being a self-illuminated sensor, it is insensitive to changes in natural light, making it appropriate for a wide range of outdoor robotics applications. While our

* Correspondence to: Universidad Miguel Hernández, Spain.
E-mail address: juan.cabreram@umh.es (J.J. Cabrera).

https://doi.org/10.1016/j.array.2025.100569
Received 15 July 2025; Received in revised form 30 September 2025; Accepted 29 October 2025
Available online 11 November 2025

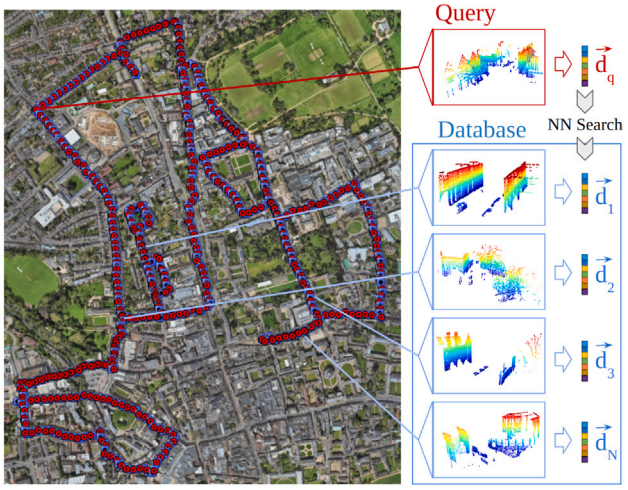


Fig. 1. Point cloud-based place recognition. Each query point cloud (red) is embedded into a global descriptor which is compared with the descriptors from the database point clouds (blue) by means of a Nearest Neighbour Search.

research focuses on place recognition, this sensor has spurred significant advancements in other critical areas like 3D object detection, where recent works have developed efficient pillar-based architectures and post-training quantization techniques to optimize performance on robotic platforms [10–12]. In consequence, across these varied applications, it is necessary to focus on methods that achieve a robust understanding of the scene. Regarding the place recognition literature, so far, we can find: (a) Classical techniques based on a handcrafted description of LiDAR data to generate rotationally invariant representations [2,13] and (b) Descriptions based on the use of Deep Neural Networks, either operating directly on the coordinates of the points [14] or on the projection of the points to image coordinates [15].

This manuscript presents a technique for the robust and invariant description of scenes captured by a LiDAR sensor. The method is based on the use of a Deep Neural Network that includes several improvements and new developments inspired from the basis of several recent architectures. In summary, the main contributions of this paper are:

- A new 3D Sparse Convolutional Neural Network for Place-Recognition, which is named MinkUNeXt (Minkowski U-Net with Next generation neural network enhancements). It is the first approach of a U-Net architecture for point cloud embedding and place-recognition. This architecture has been developed to efficiently address the place recognition problem. In addition, substantial improvements have been achieved both in terms of macro and micro design.
- The definition of a new residual block: the 3D Mink-Next Block (Min-kowski Next generation residual Block), which is entirely composed of 3D sparse convolutions and surpasses the performance of 3D ResNet Blocks. Unlike ConvNeXt Residual Block [16], which focuses on 2D convolutions and draws inspiration from transformer models to enhance performance on vision tasks, the 3D MinkNext Block extends these design philosophies into the 3D domain, utilizing sparse convolutions to efficiently process high-dimensional and irregular data such as point clouds.

As a result, the proposed topology is able to surpass significantly the current state of the art of point cloud place-recognition in terms of average recall at 1 (AR@1) and average recall at 1% (AR@1%), when compared to the most relevant methods in the literature.

The rest of the paper is organized as follows: Next, Section 2 deepens in the state of the art in relation with the use of Deep Neural Networks for the description of the structure of point clouds. After that, Section 3 defines in detail the proposed architecture to describe the point clouds. Next, Section 4 describes the datasets, experiments and results. Finally, Section 5 presents the main conclusions.

2. State of the art

This section offers a comprehensive overview of the current state-of-the-art in place recognition, specifically exploring the utilization of Deep Neural Networks with point cloud data. Many applications have emerged that concentrate on place recognition based on point clouds. In this section the methods are presented chronologically. In addition, in this manuscript a comparison of the main results achieved by the most relevant architectures is provided. In this context, the first approach to this task was tackled in [14] with PointNetVLAD, a network model based on PointNet [17] for feature extraction followed by a NetVLAD layer for feature aggregation. The point clouds taken as input by this type of architectures do not need to be sorted, as they use symmetric functions such as Multi Layer Perceptron (MLP) or Fully Connected layers. Next, a similar approach, named LPD-Net [18] improved the state of the art by incorporating a local feature extraction block at the beginning of the network and a subsequent graph-based neighbourhood aggregation.

After that, the MinkLoc3D architecture emerged [19]. It is based on a Feature Pyramid Network (FPN) with Sparse Convolutions for feature extraction [20], followed by a Generalized Mean Pooling (GeM) for the aggregation of the features into a single vector [21]. At that time, the MinkLoc3D architecture marked a significant milestone, as it significantly surpassed the existing state-of-the-art methods and also demonstrated that the use of 3D convolutional layers was a good choice for feature extraction from point clouds. Unlike previous network typologies, when using 3D convolutions, they do require a sorted point cloud as input, where the spatial relationships between points are preserved. The same situation occurs in an analogous way with images, where 2D convolutions have proven to be very efficient in feature extraction thanks to the neighbourhood relationships between pixels. In this sense, some 2D architectures have also emerged taking as input the projected point cloud into a spherical image (OverlapNet [15]). Other works, such as [1] propose creating a rotation-invariant handcrafted image: from a polar coordinate representation of the point cloud, the 2D distance between consecutive points belonging to the same elevation angle (ring) is computed and then, a histogram per ring is obtained generating a 2D handcrafted codification of the point cloud.

In addition, both monocular images and point clouds are used simultaneously by some architectures (MinkLoc++ [22], PIC-Net [23]). In this case, both architectures are formed by two branches, processing independently the image and the point cloud. Each branch results in a feature vector and both vectors are finally aggregated into a single vector by a pooling process. Alternatively, each point can be associated with a feature corresponding to the RGB value of the image [24]. This requires a precise calibration of the camera-LiDAR system. Otherwise, some authors propose to use the relative intensity returned by each LiDAR ray, referred to as MinkLoc-SI [25].

The DAGC architecture [26] was the first to introduce self-attention layers [27] for point cloud feature extraction to perform place recognition. Later, other authors continued the use of attention layers, obtaining results close to the state of the art. In this sense, NDT-Transformer was presented [28], a network model based on 3 Transformer Encoders that takes as input a modified point cloud by using a Normal Distribution Transform (NDT). This approach preserves the geometrical shape of the point cloud while decreasing the memory complexity. Building upon established sparse convolution backbones, CASSPR [29] later introduced a cross-attention mechanism on MinkLoc3D [19] to fuse voxel-based and point-based features, enhancing the description of fine-grained geometric details in sparse data.

Simultaneously, PPT-Net [30], a Transformer with a pyramidal distribution followed by a NetVLAD layer, emerged. Following this line of

point-based transformers, the Point-Wave method [31] was proposed, which models each point as a wave function to better capture the relationships between them. It is applied to different architectures, such as PPT-Net [30] and LPD-Net [18], obtaining the best configuration with PPT-Net [30]. Next, SOE-Net [32] extracts local features using a series of MLPs and subsequently, it applies attention layers in the aggregation of those features. In addition, the Retriever [33] network also introduces self-attention layers within an autoencoder to perform local feature aggregation. Besides, looking for efficiency and the use of these architectures in real localization systems (which must work in real time), SVT-Net, an efficient Sparse Voxel Transformer based on sparse convolutional layers for feature extraction, was presented in [34].

Furthermore, HiTPR [35] employs Farthest Point Sampling [36] to reduce the dimensionality of the input cloud while preserving its original topological information. In addition, this work introduces a Transformer block for short-range local feature extraction and an additional Transformer block for extracting global information over long distances. The mentioned Transformer-based approaches presented similar results to those found in the state of the art. However, the proposal of TransLoc3D [37] constituted a significant advance. It is a network model also based on sparse convolutions but unlike other proposals, it extracts features at different scales in parallel by means of convolutional layers with different kernel size. In addition, it also employs ECA (Efficient Channel Attention) layers [38] in order to interact local features from different channels. This type of layers are also used by MinkLoc3Dv2 [39], an architecture based on MinkLoc3D [19]. MinkLoc3Dv2 includes the use of ECAs with an increased number of planes or channels (depth of the convolution matrices). To date, this network architecture shows the best results in terms of average recall at 1 (AR@1) in the Oxford RobotCar Dataset [40], partly due to the loss function they introduce in the training process and the high batch size with which they train.

Finally, the best result in terms of average recall at 1% (AR@1%) was obtained by KPPR [41], a network model based on Flexible and Deformable Convolutions (KPConv [42]). However, Minkloc3Dv2 is still ahead in terms of average recall at 1 (AR@1), which is a more demanding metric. Additional architectures have been proposed to date, making other types of contributions such as rotation invariance E^2PN-GeM [43] and RPR-Net [44] or inference efficiency EPC-Net [45] and BPT [46]. Other works have also explored more fundamental shifts in data representation, such as ComPoint [47], which leverages complex-valued neural networks to encode richer phase and magnitude information from the point cloud.

This paper presents MinkUNeXt, an architecture based on MinkUNet [20] modified and enhanced to perform place-recognition from point clouds. It is an encoder–decoder architecture entirely based on the proposed 3D MinkNeXt Block, a residual block composed of 3D sparse convolutions that follows the philosophy proposed by ConvNeXt [16]. The feature extraction is performed by the U-Net encoder–decoder and the feature aggregation of those features into a single descriptor is carried out by a Generalized Mean Pooling (GeM) [48]. The proposed architecture demonstrates that it is possible to surpass the current state of the art by only relying on conventional 3D sparse convolutions without making use of more complex and sophisticated frameworks such as Transformers, Attention-Layers or Deformable Convolutions. In this way, this paper shows that the proposed architecture outputs results which are superior to those found in the literature while maintaining the efficiency, scalability and performance.

3. MinkUNeXt: global point cloud descriptor for place recognition

Place recognition from point clouds can be approached as an embedding task. For this purpose, it is desirable to have an architecture capable of extracting the more descriptive features of the scene and, in addition, aggregating them into a single vector that most generally describes the information present in the scene. The present work presents a pioneering solution that employs a U-Net architecture [49] in the context of place recognition. Most architectures resembling U-Net were originally designed for semantic segmentation, where the goal is to assign a category to each pixel of an input image, or in this case, to each point of the input point cloud. However, the encoder–decoder topology of a U-Net is also capable to extract and fuse relevant features from the scene as will be shown in the experimental section.

3.1. Global architecture

The proposed model is fed by a point cloud given as an unordered set of 3D coordinates $P = \{(x_i, y_i, z_i)\}$. This point cloud is quantized into a sparse tensor, which is a high-dimensional extension of a sparse matrix where non-zero elements are represented as a set of indices C (coordinates) and associated values (or features) F. Some papers [18, 28] propose to employ as feature some handcrafted attributes such as the vertical component of the normal vector, height variance, change of curvature or just the value of the coordinates. Others [19,39] prefer initializating each coordinate's feature to one, i.e., the first convolution (stem) will only take as input features 'ones' for the non-empty voxels. This idea is also taken in the present paper, where the input data $\hat{P} = \{(\hat{x}_i, \hat{y}_i, \hat{z}_i, 1)\}$ is conformed by C, a set of 3D quantized coordinates and F, a vector of 'ones' whose length is equal to the number of quantized points.

The global architecture is represented in Fig. 2. The encoder of the network consists of five 3D Sparse Convolutions (coloured in yellow). Among them, the stem is the first convolution and it preserves the input dimension of the point cloud since its stride is fixed to 1 and the kernel size is 5. While each of the following four convolutions gradually decrease the spatial dimension, the receptive field increases since successive convolutional layers capture larger and larger patterns by combining information from previous layers. Each of those convolutions downsample its input dimension by 2 since they employ both kernel size and stride of 2. After the encoder, the dimension of the input point cloud is downsampled by 32.

In a common U-Net the decoder is composed of four 3D Sparse Transpose Convolutions that upsample the spatial dimension by 2, progressively reconstructing the input cloud. However, in this architecture it is proposed to partially reconstruct the input point cloud by only applying three transpose convolutions (coloured in orange), since our purpose is point cloud embedding and not semantic segmentation. Section 4.5 will justify that features extracted with only three transpose convolutions are more robust for understanding the overall context of the scene. Furthermore, a Batch Normalization and a ReLU activation function (coloured in red) are applied after all the convolutions, which helps in stabilizing the training process. In addition, in this architecture it is proposed to employ the presented Residual MinkNeXt Block (coloured in blue) instead of the common ResNet Block after each ReLU (without taking into account the one corresponding to the stem). This kind of residual blocks provide a direct path for gradients to flow through the network, reducing overfitting and boosting the generalization capabilities on unseen data. In this architecture, it is also used to increase the number of features maps as it will further detailed in the following Section 3.2.

The U-Net architecture is characterized for having skip connections between the encoder and the decoder. On the one hand, the encoder would capture features at different spatial scales, from fine details (low-level) to more global structures (high-level) present in point clouds. On the other hand, thanks to the skip connections, the decoder would fuse the low-level and high-level features. After that, a Fully Connected Layer is added since it outputs features have been proven to perform robustly against viewpoint changes in visual place recognition [50]. Furthermore, this Fully Connected Layer is also employed to extend the feature maps up to a dimensionality of 512. Subsequently, the points descriptors that conform that feature map are aggregated into a single global descriptor by a Generalized Mean Pooling (GeM) [48].

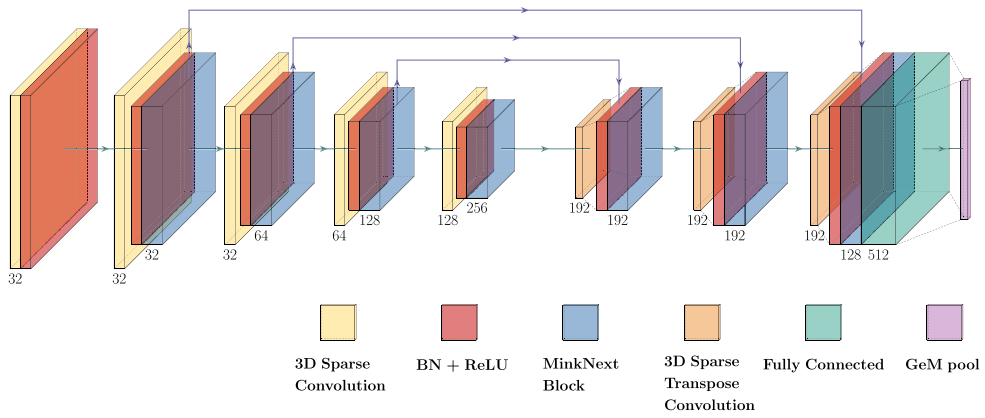


Fig. 2. This diagram shows the architecture of the proposed MinkUNeXt, which is based on a semantic segmentation network (U-Net) modified and enhanced to perform place-recognition from point clouds.

3.2. Residual block architecture

As mentioned before, in this paper both a global and a residual block architecture are proposed. In this sense, a new residual block is designed (Fig. 3) which is entirely composed of 3D Sparse Convolutions and follows the philosophy proposed by ConvNeXt [16], surpassing the performance of ResNet Blocks. We have named this block MinkNeXt, since it takes advantage of the ResNet blocks and is fully implemented in Minkowski Engine [20].

In the global architecture (Fig. 2), the proposed residual block appears in blue colour after each ReLU activation function (except for the one corresponding to the stem). Since the residual block is generally employed to increase the number of features maps, the stem of the residual block is formed by a $1 \times 1 \times 1$ convolution that widens the input dimension to the output channels size. After that, an inverted bottleneck is applied by expanding the dimension four times and then reducing it again to the output dimension through two 3D Sparse Convolutions. This inverted bottleneck was originally proposed by MobileNetV2 [51] and nowadays, it is an important design in every Transformer block. In addition, a $1 \times 1 \times 1$ Convolution in the residual connection is also applied when the input and the output dimensions differ.

The activation function employed in this block is the Gaussian Error Linear Unit (GeLU) [52], which is smoother than ReLU and is utilized in the most advanced Transformers. Finally, the normalization is carried out by LayerNorms [53] in the main stream of the block and by BatchNorms [54] in the residual connection.

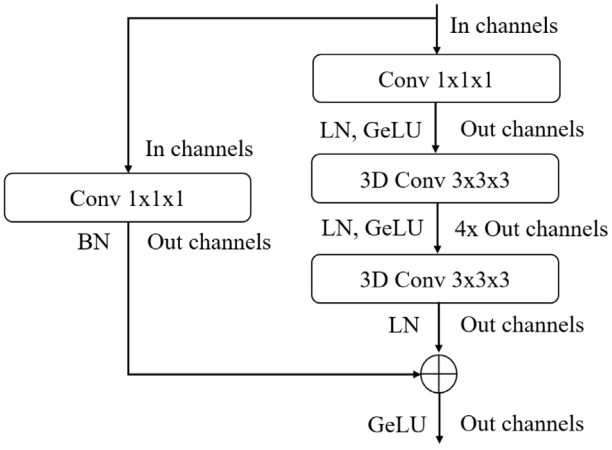


Fig. 3. This diagram shows the proposed MinkNeXt Block. This residual block is an essential part of the global network, since it increases the number of feature maps through an inverted bottleneck.

Table 1
The number of training and testing point clouds for the baseline, refined and further test protocols.

	Baseline protocol		Refined protocol		Further test protocol	
	Training	Test	Training	Test	Training	Test
Oxford	21.7k	3.0k	21.7k	3.0k	–	–
In-house	–	4.5k	6.7k	1.7k	–	–
USyd	–	–	–	–	–	8.8k
KITTI	–	–	–	–	–	0.2k

4. Experiments

This section describes the datasets (Section 4.1), the labelling (Section 4.2) and the training and evaluation of the proposed architecture (Section 4.3). Later, the implementation details are described in Section 4.4. Subsequently, in Section 4.5, we present an ablation study of the designing steps carried out to obtain the final architecture. Finally, the main results are compared with other approaches in the literature in Section 4.6.

4.1. Datasets

In order to train and evaluate the proposed architecture, the protocols introduced in [14] have been used together with the Oxford RobotCar Dataset [40] and the In-house Dataset [14]. Additionally, to assess the generalization capability of the proposed method in comparison with the state-of-the-art, both the USyd Campus Dataset [55] and the KITTI Dataset [56] are considered, following the evaluation criteria established in [25]. These are common frameworks employed and respected by a large number of studies that are used to compare different proposals that address the place recognition task using both point clouds submaps and single-scans. The benchmark consists of 4 datasets and 6 different environments:

- **Oxford RobotCar Dataset** [40]. This dataset is generated using some SICK LMS-151 2D sensors mounted on a car. The dataset covers a 10 km trajectory along the city of Oxford. In total, 44 sequences of the same trajectory which are geographically divided into training (70%) and test (30%) are used. This results in 21,711 training submaps and 3,030 test submaps.
- **In-house Dataset** [14]. This dataset consists of three different environments: a University Sector (U.S.), a Residential Area (R.A.), and a Business District (B.D.). These datasets are captured using a Velodyne-64 LiDAR mounted on a motorized vehicle that covers

each of the three regions. The paths lengths are 10 km, 8 km and 5 km respectively. It is conformed by 5 different sequences from each of the U.S., R.A. and B.D. regions, which were captured at different times. In addition, each U.S. and R.A. sequence are geographically divided into train and test. While the B.D. environment is only used for testing.

- **USyd Campus Dataset** [55]. A dataset captured using a Velodyne VLP-16 LiDAR mounted on a buggy-style vehicle over the course of 50 weeks. Data was captured across a wide variety of weather conditions. As a result, it contains 40 sequences, each one including approximately 735 scans. In the experimentation, these LiDAR scans were only used for evaluation. Specifically, four 100 × 100 m regions were chosen randomly to serve as test areas. In total, the dataset contains 8,797 single LiDAR scans for testing.
- **KITTI Dataset** [56]. This is a widely used benchmark for autonomous driving research, recorded in and around the city of Karlsruhe, Germany. The data was captured with a Velodyne HDL-64E LiDAR sensor mounted on a vehicle. In this paper, this dataset is also used exclusively to evaluate the generalization capability of the models, meaning that no training is performed on this data. Following the established protocol in [25], the reference database is built using the first 170 s of Sequence *00*, while the rest of the sequence serves as queries.

In the Oxford RobotCar Dataset [40] and In-house Dataset [14], the LiDAR scans are taken at regular intervals of 12.5 m and 25 m for the training and test set, respectively. Also, both datasets are formed by a number of submaps. Each submap is constructed by capturing LiDAR scans consecutively along 20 m. Next, the scans are registered in a common frame and further processed to create a consistent submap. Each of these training and test submaps are filtered by removing the ground plane and also regularly sampled by a voxel grid filter in order to reduce its size to 4096 points. The XYZ coordinates of the points that constitute each submap are then shifted and scaled in order to obtain a point distribution with zero mean in the [−1, 1] range for each coordinate. Conversely, the USyd Campus Dataset [55] and KITTI Dataset [56] are composed of single LiDAR scans used in their raw format. This means that they undergo no further preprocessing steps like ground plane removal or point cloud normalization. Following [25], the USyd scans are uniformly filtered to maintain a 5-meter spatial interval, while the KITTI data is downsampled every 10 m.

4.2. Labelling and similarity

Each point cloud in the dataset is tagged with the UTM coordinates of its respective centroid. This constitutes the identifier of each submap (or single scan, in the case of USyd and KITTI) and is later used during the training and evaluation of the network. Next, we define the similarity between the submaps in the datasets. This concept is generally denoted as labelling in the literature and it is important because it is necessary to feed the model with structurally similar point clouds captured from the same place and structurally dissimilar scans from different places. In this sense, most of the proposed labelling protocols are based on the Euclidean distance of the UTM coordinates from which point clouds are captured (two point clouds are considered structurally similar if they are captured within a distance p and structurally different if they are taken from a distance greater than n where $p < n$). This procedure, of course, is a coarse approximation that assumes that point clouds captured from the same area will possess a similar structure. However, it is a simple but effective manner of labelling the training data. In this paper, this method is adopted with $p = 10$ m and $n = 50$ m as in the majority of the referred manuscripts. Authors, have also proposed other methods for similarity labelling in the context of place-recognition. For example, [15] proposes to use the overlap between point clouds as an alternative method for labelling similar and dissimilar point clouds. In order to compute the overlap between two point clouds a precise registration must be carried out, which hinders the application of this technique to large datasets.

Table 2
Training Parameters in Baseline and Refined Protocols.

Parameter	Baseline	Refined
Batch Size (b)	2048	2048
Number of Epochs	400	500
Initial Learning Rate	1×10^{-3}	1×10^{-3}
LR Scheduler Steps	250, 350	350, 450
L2 Weight Decay	1×10^{-4}	1×10^{-4}
Sigmoid Temperature (τ)	0.01	0.01
Positives per Query (k)	4	4
Quantization Scale (qs)	0.01	0.01

4.3. Training and evaluation

As for the training and evaluation of the proposed method, the two evaluation protocols established in [14] have been followed. Moreover, an additional assessment of the generalization capabilities of the proposed approach is performed following the testing benchmark introduced in [25]. In total, three different protocols have been used in the experiments:

- The first, baseline protocol, consists in training the model only with the Oxford training data and evaluating with the Oxford and In-house (U.S., R.A. and B.D.) test data.
- The second, refined protocol, consists in training with the Oxford and In-house (U.S., R.A.) training data and evaluating with the Oxford and In-house (U.S., R.A. and B.D.) test data.
- The third, further test protocol, consists in evaluating the architectures trained in the two previous protocols with two totally different datasets, that is, the USyd and KITTI datasets.

Table 1 summarizes the number of training and testing point clouds corresponding to each dataset and each of the protocols defined above. The assessment of the LiDAR-based place recognition descriptors is carried out by means of the recall rate at top-K candidates. Following the most common evaluation methods (as in the manuscripts cited in Section 2), the average recall at 1 (AR@1) and average recall at 1% (AR@1%) are used in order to ease the comparison with other techniques. We start with a "query submap" formed by a point cloud which is taken from the test dataset and point clouds submaps from different trajectories that cover the same region of the map. Each query submap is processed by the network and it outputs, as a result, a descriptor vector that codifies its appearance. This descriptor is referred to as the "query descriptor". Next, the query descriptor is compared to all the descriptors in the map, and an Euclidean distance is computed. The point cloud in the database that minimizes this distance in the descriptors' space is selected. Finally, a place recognition match is considered successful if the Euclidean distance between the query and the retrieved point cloud is below a specific threshold (computed from their corresponding GPS coordinates). This threshold is set to 25 m for the Oxford, In-house, and KITTI datasets, and 10 m for the USyd dataset.

4.4. Implementation details

In the present approach the proposed model is trained following the procedure established in [39]. In this regard, the Truncated Smooth-AP (TSAP) loss function is employed, which tries to maximize the ranking of the positive top-k candidates:

$$\mathcal{L}_{TSAP} = \frac{1}{b}\sum_{q=1}^{b}(1 - AP_q) \tag{1}$$

Where b is the batch size and AP_q is the smooth average precision:

$$AP_q = \frac{1}{|P|}\sum_{i\in P}\frac{1 + \sum_{j\in P, j\neq i} G(d(q,i) - d(q,j);\tau)}{1 + \sum_{j\in \Omega, j\neq i} G(d(q,i) - d(q,j);\tau)} \tag{2}$$

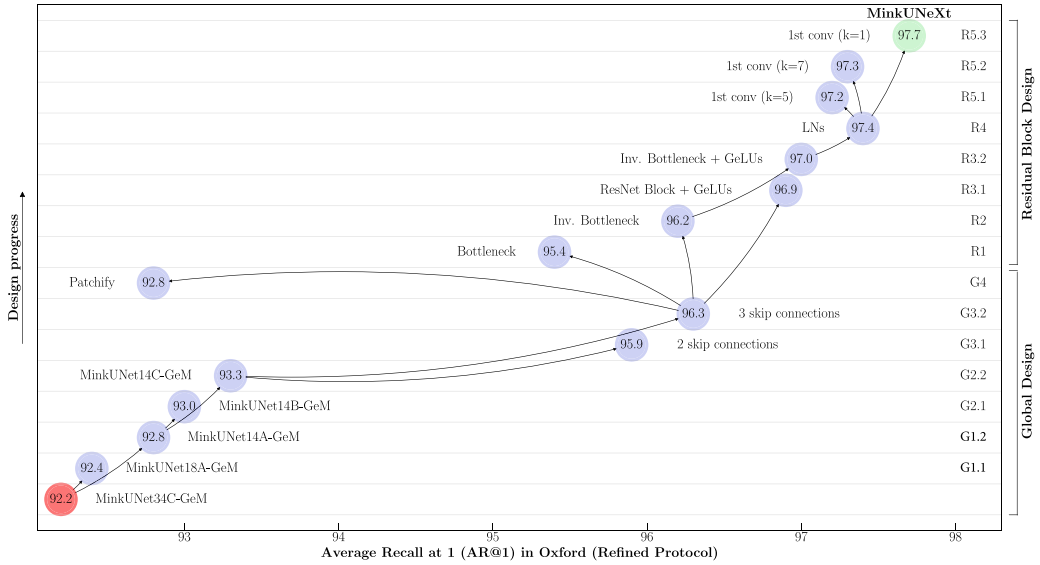


Fig. 4. This diagram illustrates the main design progress of the proposed architecture from MinkUNet up to MinkUNeXt. Additional experiments are shown in Table 3 and all the proposed modifications are summarized in Table 4.

Given a query point cloud q, the average precision AP_q is computed from the set of k closest candidates P (positives) and the set of all positives and negatives Ω. Also, the function G constitutes a Sigmoid function $G(x;\tau) = \left(1+\exp\left(-\frac{x}{\tau}\right)\right)^{-1}$ with a parameter τ that controls the sharpness. The term $d(q,i)$ represents the Euclidean distance between the descriptor of a query point cloud q and the ith point cloud. The numerator represents a soft ranking of a positive point i among the top k positives (where $k=4$), while the denominator represents a soft ranking of a positive point i among all other positives and negatives.

For the correct performance of this type of loss function, it is necessary to train with a high batch size, specifically a size of 2048 has been used with 400 and 500 training epochs for the baseline and refined protocol, respectively. The optimizer used to minimize the loss function is Adam with an Initial Learning Rate of 1e−3 and it is divided by 10 in the epochs given by the LR scheduler steps, which are epochs 250 and 350 for baseline protocol and epochs 350 and 450 for refined protocol. Table 2 summarizes all the parameter values described above.

Additionally, when working with sparse convolutions, the input point clouds need to be quantized by a factor of qs, which is set to 0.01 for the Oxford and the In-house datasets since these submaps are already normalized to [−1, 1]. However, for the evaluation in the USyd and KITTI datasets, a quantization factor qs of 0.1 is used, as these raw point clouds are neither preprocessed nor normalized. Furthermore, to increase the number of training instances and reduce model overfitting, a data augmentation has been carried out by applying a random jitter of a value between [0, 0.001] individually to each point of the point cloud, a random transformation to the global point cloud with a value between [0, 0.01] and a random removal of 10% of the points.

All experiments are carried out on a NVIDIA GeForce RTX 3090 GPU with 24 GB. Our code is publicly available on the project website: https://juanjo-cabrera.github.io/projects-MinkUNeXt/.

4.5. Ablation study: From MinkUNet to MinkUNeXt

The design departs from the MinkUNet34C architecture [20] as a baseline. Next, the series of design decisions are described. Each design step is summarized in two main subsections: (1) global design and (2) residual block design, which are included next. For every step, both the procedure and the results are presented, starting from the MinkUNet34C until obtaining the MinkUNeXt architecture. It is important to clarify that this was not a sequential process of adding layers, but rather a holistic study involving the modification of numerous parameters across the entire network architecture. Specifically, the general evolution of the network and its performance is presented in Fig. 4, while Table 3 extends these results with intermediate experiments conducted during the design and development process of the architecture. Finally, Table 4 summarizes and describes the main design steps.

4.5.1. Global design

As mentioned above, the starting point is the MinkUnet34C [20] architecture and it is first modified by adding a GeM pool layer. This baseline is coloured in red in Fig. 4. The rest of the roadmap followed towards the final design is described next. Each of the designing steps are classified in one of the following points: evaluating the cardinality and the number of channels, changing the number of skip connections and replacing the stem to a “Patchify” configuration.

G1. Evaluating the cardinality. The cardinality is defined as the number of parallel blocks, that enables the network to learn various input representations. On the image domain, a high cardinality (as in ResNeXt [57]) can lead to more expressive representations. However, for the task of 3D place recognition, a simpler, deeper feature hierarchy may be more effective at learning abstract geometric concepts without excessive parameterization. Different experiments were conducted to check the effects on the results with different cardinalities: (2, 3, 4, 6, 2, 2, 2, 2), (2, 2, 2, 2, 2, 2, 2, 2) and (1, 1, 1, 1, 1, 1, 1, 1), corresponding to MinkUNet34, MinkUNet18 and MinkUNet14, respectively. These cardinality values represent the number of instances of each Residual Block that appear in blue colour in Fig. 2, but at this point still with ResNet Blocks. In addition, these cardinality configurations are summarized respectively in steps G1.1 and G1.2 in Fig. 4. In addition, Table 3 also shows these experiments in terms of average recall at 1 (AR@1) and the increment of performance (ΔAR@1) with respect to MinkUNet34C. As illustrated both in the diagram and in the table, reducing the cardinality to the minimum (G1.2), with no parallel blocks, empirically shows a better performance and improves the AR@1 from 92.2% to 92.8%. This result supports our hypothesis that a lean, focused architecture is better suited for place recognition with 3D point clouds. From now on, a cardinality of 1 will be used for each residual block.

Table 3
This table details every experiment conducted during the evolution from MinkUNet to MinkUNeXt. The first column shows the backbone used for the global architecture, while the second indicates the residual block architecture. In the third column, the ID in parentheses refers to the summary of modifications in Table 4. Finally, the results of each configuration are presented in terms of AR@1 and ΔAR@1 (the increment in performance with respect of the initial baseline).

Backbone	Residual block	Design modification	AR@1 (%)	ΔAR@1 (%)
Phase 1: Global Design				
MinkUNet34C	**ResNet Block (ReLU, BN)**	**Initial Baseline**	**92.2**	–
MinkUNet34A	ResNet Block (ReLU, BN)	Tune MinkUNet34C Decoder Channels (G2.3)	92.3	+0.1
MinkUNet34B	ResNet Block (ReLU, BN)	Tune MinkUNet34C Decoder Channels (G2.4)	91.0	−1.2
MinkUNet18A	ResNet Block (ReLU, BN)	Reduce MinkUNet34C Cardinality (G1.1)	92.4	+0.2
MinkUNet18B	ResNet Block (ReLU, BN)	Tune MinkUNet18A Decoder Channels (G2.2)	91.8	−0.4
MinkUNet18D	ResNet Block (ReLU, BN)	Tune MinkUNet18A Decoder Channels (G2.5)	89.2	−3.0
MinkUNet14A	**ResNet Block (ReLU, BN)**	**Reduce MinkUNet34C Cardinality (G1.2)**	**92.8**	+0.6
MinkUNet14B	ResNet Block (ReLU, BN)	Tune MinkUNet14A Decoder Channels (G2.1)	93.0	+0.8
MinkUNet14C	**ResNet Block (ReLU, BN)**	**Tune MinkUNet14A Decoder Channels (G2.2)**	**93.3**	+1.1
MinkUNet14D	ResNet Block (ReLU, BN)	Tune MinkUNet14A Decoder Channels (G2.5)	88.0	−4.2
MinkUNet14C (2-skip)	ResNet Block (ReLU, BN)	Change MinkUNet14C Skip Connections to 2 (G3.1)	95.9	+3.7
MinkUNet14C (3-skip)	**ResNet Block (ReLU, BN)**	**Change MinkUNet14C Skip Connections to 3 (G3.2)**	**96.3**	+4.1
MinkUNet14C (3-skip)	ResNet Block (ReLU, BN)	Change Stem of MinkUNet14C (3-skip) to “Patchify” (G4)	92.8	+0.6
Phase 2: Residual Block Design				
MinkUNet14C (3-skip)	Bottleneck (ReLU, BN)	Block Type: Basic → Bottleneck (R1)	95.4	+3.2
MinkUNet14C (3-skip)	Inv. Bottleneck (ReLU, BN)	Block Type: Basic → Inv. Bottleneck (R2)	96.2	+4.0
MinkUNet14C (3-skip)	ResNet Block (GeLU, BN)	Activation: ReLU → GeLU (R3.1)	96.9	+4.7
MinkUNet14C (3-skip)	**Inv. Bottleneck (GeLU, BN)**	**Block Type + Activation (R3.2)**	**97.1**	+4.9
MinkUNet14C (3-skip)	Inv. Bottleneck (BN)	Remove intermediate activations (R3.3)	95.4	+3.2
MinkUNet14C (3-skip)	**Inv. Bottleneck (GeLU, LN)**	**Normalization: BN → LN (R4.1)**	**97.4**	**+5.2**
MinkUNet14C (3-skip)	Inv. Bottleneck (GeLU, LN)	Add LN to shortcut connection (R4.2)	97.2	+5.0
MinkUNet14C (3-skip)	Inv. Bottleneck (GeLU, LN)	Kernel Size (1st Conv): 3 → 5 (R5.1)	97.2	+5.0
MinkUNet14C (3-skip)	Inv. Bottleneck (GeLU, LN)	Kernel Size (1st Conv): 3 → 7 (R5.2)	97.3	+5.1
MinkUNet14C (3-skip)	**Inv. Bottleneck (GeLU, LN)**	**Kernel Size (1st Conv): 3 → 1 (R5.3)**	**97.7**	**+5.5**
MinkUNet14C (3-skip)	Inv. Bottleneck (GeLU, LN)	Kernel Size (2nd and 3rd Conv): 3 → 1 (R5.4)	97.1	+4.9
MinkUNet14C (3-skip)	Inv. Bottleneck (GeLU, LN)	Kernel Size (2nd and 3rd Conv): 3 → 5 (R5.5)	97.2	+5.0
MinkUNet14C (3-skip)	Inv. Bottleneck (GeLU, LN)	Kernel Size (2nd and 3rd Conv): 3 → 7 (R5.6)	97.3	+5.1

G2. Evaluating the number of channels. The number of channels or planes correspond to the number of feature maps that the convolutional layer can learn. The number of filters corresponding to the convolutional layers of the encoder are fixed to (32, 64, 128, 256), while multiple decoder channel configurations across all cardinalities are evaluated. As detailed in Table 3, this exploration was crucial, as several configurations led to a severe performance degradation, specifically when drastically reducing the number of channels (MinkUNet34B) or increasing it (MinkUNet18D, MinkUNet14D). The best result is obtained with MinkUNet14C (G2.2) with an AR@1 of 93.3%. This configuration allocates higher capacity to the deeper layers of the decoder, suggesting that it is critical for fusing the most salient high-level features. Thus, the number of channels of the transposed convolutions that will be adopted in the subsequent architecture variations is (192, 192, 128, 128).

G3. Changing the number of skip connections. The original U-Net is characterized by the presence of 4 skip connections between the encoder and the decoder networks. Since it was originally designed for semantic segmentation, the predicted segmented point cloud must have the same dimension as the input point cloud. However, when it comes to place recognition, a totally different output from the model is expected, i.e., a global embedding. For this purpose, there is no need to retain the entire decoder along with its skip connections. In this sense, in the present paper the performance of the network is studied when reducing the number of skip connections and removing the transposed convolutions after the last connection. In addition, when changing the number of skip connections, the overall depth of the model also varies. Apart from the 4 skip connections already implemented in the previous configurations, 2 and 3 skip connections together with the removal of the decoder blocks attached to the last skip connection are evaluated (it corresponds to G3.1 and G3.2 in Fig. 4). By reducing the number of skip connections to 3 and removing the layers after the last connection (G3.2), the model shows, by far, the greatest improvement on the AR@1, increasing the performance from 93.3% to 96.3%. When using the full encoder–decoder architecture, low-level details from early skip connections can introduce noise and hinder the formation of a robust global descriptor. By reducing the number of skip connections and removing the decoder blocks attached to them, we force the model to rely more heavily on the semantic information from the deeper layers, which results in more discriminative global embeddings.

G4. Changing the stem to “Patchify”. The stem refers to the first layer in the network, which performs the initial processing. In this case, the first processing is carried out by a 3D Sparse Convolution with kernel size 5 and stride 1. The term “Patchify” refers to the act of splitting the input data into an independent sequence of patches. Visual Transformers [58] introduced this concept, originally inspired by NLP Transformers [27]. The Swin Transformer [59] uses as stem a non-overlapping convolution with kernel size 4 and stride 4 on images. In this sense, these parameters are adopted for the stem in G4. For sparse 3D point clouds, unlike dense 2D images, this aggressive initial downsampling empirically decreases the performance of the network from 96.3% to 92.8%.

4.5.2. Residual block design

This section describes each design step from ResNet Block to the proposed MinkNeXt Block. The roadmap of the design of this residual block is divided in the following points: creating a Bottleneck in the residual block, creating an Inverted Bottleneck in the residual block, replacing ReLUs with GeLUs, substituting BN with LN and evaluating different kernel sizes.

R1. Creating a Bottleneck in the residual block. A Bottleneck consists in reducing the dimensionality of the hidden layer and then expanding it to its original size using 1×1 convolutions. As shown both in Fig. 4 and Table 3, this modification led to

worse results in the performance of the proposed architecture comparing to G3.2 which is the best model so far.

R2. Creating an Inverted Bottleneck in the residual block. Every Transformer block is characterized by an inverted bottleneck, which consists in expanding the dimensionality of the feature map of the hidden layer and then reducing it to its original size by 1×1 convolutions. In this case, 3D sparse convolutions with kernel size 3 and stride 1 are employed to create the inverted bottleneck with a hidden dimension four times wider than the input dimension. Fig. 4 shows that this inverted bottleneck block produces better results compared to the previous ResNet block when analysed jointly with the following modification (R3).

R3. Replacing ReLUs with GeLUs. The Rectified Linear Unit (ReLU) [60] is the most employed activation function over time due to its simplicity and efficiency. However, recent advanced Transformers such as Google's BERT [61] or OpenAI's GPT-4 [62] employ Gaussian Error Linear Units (GeLUs) [52], which is a smoother variant of ReLUs. Following the same philosophy, ReLUs are replaced with GeLUs in both the ResNet Block and the inverted bottleneck block, steps R3.1 and R3.2 in Fig. 4, respectively. In both cases, the performance of the architecture improves, but better results are obtained with the proposed inverted bottleneck block (R3.2), achieving an AR@1 of 97.0%. Additionally, removing intermediate activation functions in the Inverted Bottleneck Block (R3.3) does not yield any improvement comparing to R3.2 (see Table 3). In consequence, an inverted bottleneck with GeLUs will be used as residual block.

R4. Substituting BN with LN. Batch Normalization (BN) [54] plays a critical role in convolutional networks by enhancing convergence and mitigating overfitting. However, BN may introduce complexities that may negatively impact the model's performance. Recently, the simpler Layer Normalization [53] (LN) has been successfully implemented in Transformers. Thus, BN is replaced with LN in the main stream (R4.1) of the residual block, obtaining an improvement of the model performance up to 97.4%. However, we also found that over-normalization, such as adding LN to the shortcut connection (R4.2), was detrimental. As a result, Layer Normalization will be employed instead of Batch Normalization in the main stream of the residual block.

R5. Evaluating different kernel sizes. Modern architectures like ConvNeXt [16] often favour large kernel sizes to increase the receptive field. However, their effectiveness on sparse 3D data is not guaranteed. While sparse convolutions only operate on active voxels, a large kernel may still connect points that are structurally unrelated in the real world, potentially degrading the quality of learned local patterns. As shown in Fig. 4 and Table 3 (R5), the usage of smaller kernel sizes is beneficial in the present place recognition task. We find the best parameter configuration with a kernel size of 1 in the first convolution and kernel sizes of 3 in the hidden and last convolutions (R5.3). This result suggests that for sparse data, efficient channel-wise feature mixing, as performed by 1×1 convolutions, is more effective than expanding the spatial receptive field within the residual block. This leads to the final model and residual block architectures, which we have named MinkUNeXt and MinkNeXt block, respectively.

The final MinkUNeXt architecture balances depth and compactness: after removing redundant skip connections and reducing cardinality, the resulting 7 residual blocks constitute the minimum depth that still preserves a hierarchical abstraction of geometric features while avoiding over-parameterization. Further reduction degraded performance, whereas deeper configurations did not provide additional gains (see Table 3).

Table 4
This table summarizes all modifications proposed in the architecture design progress from MinkUNet up to MinkUNeXt.

ID	Design modifications
G1.1	Cardinality: (2, 3, 4, 6, 2, 2, 2, 2) → (2, 2, 2, 2, 2, 2, 2, 2)
G1.2	Cardinality: (2, 2, 2, 2, 2, 2, 2, 2) → (1, 1, 1, 1, 1, 1, 1, 1)
G2.1	Decoder channels: (128, 128, 96, 96) → (128, 128, 128, 128)
G2.2	Decoder channels: (128, 128, 96, 96) → (192, 192, 128, 128)
G2.3	Decoder channels: (256, 128, 96, 96) → (256, 128, 64, 64)
G2.4	Decoder channels: (256, 128, 96, 96) → (256, 128, 64, 32)
G2.5	Decoder channels: (256, 128, 96, 96) → (384, 384, 384, 384)
G3.1	4 skip connections → 2 skip connections
G3.2	4 skip connections → 3 skip connections
G4	Stem (k=5, s=1 → k=4, s=4)
R1	ResNet Block → Bottleneck
R2	ResNet Block → Inv. Bottleneck
R3.1	ResNet Block with ReLUs → ResNet Block with GeLUs
R3.2	Inv. Bottleneck with ReLUs → Inv. Bottleneck with GeLUs
R3.3	Inv. Bottleneck without activation functions
R4.1	BNs → LNs in the main stream of the Inv. Bottleneck
R4.2	BNs → LNs in the whole Inv. Bottleneck
R5.1	Inv. Bottleneck 1st convolution (k=3 → k=5)
R5.2	Inv. Bottleneck 1st convolution (k=3 → k=7)
R5.3	Inv. Bottleneck 1st convolution (k=3 → k=1)
R5.4	Inv. Bottleneck 2nd and 3rd convolutions (k=3 → k=5)
R5.5	Inv. Bottleneck 2nd and 3rd convolutions (k=3 → k=7)
R5.6	Inv. Bottleneck 2nd and 3rd convolutions (k=3 → k=1)

4.6. Comparison with the state of the art

As defined in Section 4.3, the two training and evaluation protocols established in [14] have been followed for place recognition with the Oxford RobotCar and In-house datasets. The baseline protocol consists in training the model only with the Oxford training data and evaluating with the Oxford and In-house (U.S., R.A. and B.D.) test data. In contrast, the refined protocol consists in training with Oxford and In-house (U.S., R.A.) training data and evaluating with the Oxford and In-house (U.S., R.A. and B.D.) test data. These protocols are widely used in the literature, so that the comparison is performed on the same terms and conditions. Additionally, the comparative results shown here have been obtained from the same works that are referenced.

Tables 5 and 6 present an overview of the results with different techniques proposed in the state of the art compared to the one proposed in this paper under the same training and evaluation protocols (baseline and refined), in terms of average recall at 1 (AR@1) and average recall at 1% (AR@1%). Each column presents the results obtained with each of the datasets, whereas the last two columns present the mean results.

4.6.1. Results with the baseline protocol

Table 5 presents the results of several methods in terms of average recall at 1 (AR@1) and average recall at 1% (AR@1%). It can be observed that, PointNetVLAD established the starting point for place-recognition from point clouds with the Oxford Robotcar and the In-house dataset. PCAN slightly outperforms PointNetVLAD on most datasets. BPT stands out with really competitive results, especially in Oxford and U.S. RPR-Net outperforms BPT in U.S, R.A and B.D., showing better generalization capabilities. Some works, such as DAGC and Retriever, do not provide AR@1 results for all datasets. However, they presented AR@1% results which show a performance better than PCAN, but worse than BPT. Futhermore, LPD-Net, HiTPR, EPC-Net and E^{2}PN-GeM show similar, but good results across multiple scenarios. SOE-Net, only provides AR@1% results which are really promising as they are close to MinkLoc3D, the first architecture to exceed 90% in AR@1 with the Oxford dataset. Moreover, HiBi-Net, PPT-Net and SVT-Net show slightly higher performance, specifically for the In-house

Table 5
Evaluation results in terms of average recall at 1 (AR@1) and at 1% (AR@1%) of place recognition methods trained using the baseline protocol.

Method	Oxford		U.S.		R.A.		B.D.		Mean	
	AR@1	AR@1%	AR@1	AR@1%	AR@1	AR@1%	AR@1	AR@1%	AR@1	AR@1%
PointNetVLAD [14]	62.8	80.3	63.2	72.6	56.1	60.3	57.2	65.3	59.8	69.6
PCAN [63]	69.1	83.8	62.4	79.1	56.9	71.2	58.1	66.8	61.6	75.2
DAGC [26]	–	87.5	–	83.5	–	75.7	–	71.2	–	79.5
BPT [46]	85.7	93.3	80.5	89.3	77.4	86.6	74.1	78.5	79.4	86.9
Retriever [33]	–	91.9	–	91.9	–	87.4	–	85.5	–	89.2
RPR-Net [44]	81.0	92.2	83.2	94.5	83.3	91.3	80.4	86.4	82.0	91.1
LPD-Net [18]	86.3	94.9	87.0	96.0	83.1	90.5	82.5	89.1	84.7	92.6
HiTPR [35]	87.8	94.6	86.0	94.0	81.3	89.1	81.8	88.3	84.2	91.5
EPC-Net [45]	86.2	94.7	–	96.5	–	88.6	–	84.9	–	91.2
E^2PN-GeM [43]	84.8	93.2	88.1	95.3	83.7	90.5	83.3	87.7	85.0	91.7
SOE-Net [32]	–	96.4	–	93.2	–	91.5	–	88.5	–	92.4
MinkLoc3D [19]	93.0	97.9	86.7	95.0	80.4	91.2	81.5	88.5	85.4	93.2
HiBi-Net [64]	87.5	95.1	87.8	–	85.8	–	83.0	–	86.0	–
NDT-Transformer [28]	93.8	97.7	–	–	–	–	–	–	–	–
PPT-Net [30]	93.5	98.1	90.1	97.5	84.1	93.3	84.6	90.0	88.1	94.7
SVT-Net [34]	93.7	97.8	90.1	96.5	84.3	92.7	85.5	90.7	88.4	94.4
TransLoc3D [37]	95.0	98.5	–	94.9	–	91.5	–	88.4	–	93.3
MinkLoc3Dv2 [39]	96.3	98.9	90.9	96.7	86.5	93.8	86.3	91.2	90.0	95.1
KPPR [41]	91.5	97.1	–	98.0	–	**95.1**	–	**92.1**	–	95.6
ComPoint [47]	69.3	83.7	67.3	80.6	58.1	72.2	62.6	69.2	64.3	76.4
CASSPR [29]	95.6	98.5	**92.9**	97.9	**89.5**	94.8	**87.9**	**92.1**	**91.5**	**95.8**
Point-Wave [31]	92.4	97.5	92.8	**98.6**	86.2	94.5	85.5	90.8	89.2	95.3
MinkUNeXt (ours)	**95.8**	**98.6**	89.9	96.5	87.4	93.3	86.6	91.3	89.9	95.0

dataset. TransLoc3D takes a step forward with the best result so far in Oxford and solid performance in the other scenarios, and its improved version MinkLoc3Dv2 outperforms the previous approaches. In addition, KPPR also shows a remarkable performance, but only presented average recall at 1% results in the case of U.S., R.A., B.D. ComPoint, while introducing a novel approach with complex-valued networks, achieves a similar performance to PCAN with this protocol. In contrast, Point-Wave demonstrates positive results, particularly excelling in the U.S. dataset with the highest AR@1%. Most notably, CASSPR sets a new state-of-the-art in the baseline protocol. By effectively fusing MinkLoc3D with cross-attention layers, it achieves the highest AR@1 in all three in-house datasets (U.S., R.A., B.D.).

Finally, the proposed architecture, MinkUNeXt, demonstrates superior performance in terms of AR@1 and AR@1% in Oxford. It outperforms all of the existing methods with a 95.8% in AR@1 and 98.6% in AR@1%. This indicates a powerful capacity for in-domain feature learning. Meanwhile, its generalization to unseen datasets is lower than CASSPR in the case of the in-house dataset, where the scenarios are considerably more open than Oxford, with fewer obstructions and a more dispersed arrangement of urban elements.

4.6.2. Results with the refined protocol

As for the performances of the models when training with the refined protocol (Table 6), PointNetVLAD introduced the starting reference point as well, surprisingly achieving a good performance in U.S. R.A. and B.D. despite the simplicity of the network architecture. PCAN and DAGC presented similar results to PointNetVLAD for the In-house dataset, but especially better in Oxford. In contrast, LPD-Net and SOE-Net show substantially better performance in all metrics and datasets. MinkLoc3D also manages to exceed 90% on average recall at 1 (AR@1) in Oxford and generally performs well in all metrics and sets. PPT-Net does not provide values for average recall at 1 (AR@1), but shows a promising performance on average recall at 1% (AR@1%). Furthermore, SVT-Net stands out especially in U.S., R.A. and B.D. In addition, TransLoc3D achieves good results in all metrics, being one of the best methods overall. MinkLoc3Dv2 boasted the best results in the state of the art so far, showing improvements over MinkLoc3D. In addition, ComPoint achieves again a similar performance to PCAN and DAGC. Regarding CASSPR, which excelled at generalization with limited data variety, it does not scale as effectively, with a mean AR@1 of 96.0%.

Finally, the proposed MinkUNeXt model shows considerable improvements in average recall at 1 (AR@1) and average recall at 1% (AR@1%) for all the scenarios obtaining the best results of the state of the art so far. The average recall at 1 (AR@1) metric in Oxford dataset is 97.7% and outperforms the runner-up MinkLoc3Dv2 by 0.8 p.p. In the R.A., B.D. scenarios it surpasses MinkLoc3Dv2 by 0.1 to 1.1 p.p. Nevertheless, slightly worse results (0.3 p.p.) are obtained with this metric in the U.S. dataset. Regarding the results in terms of AR@1% for the refined protocol, there was little room for improvement. However, the results in Oxford are improved by 0.2 p.p. to reach 99.3%, on R.A. by 0.5 p.p. to reach 99.9% and on B.D. by 0.1 p.p. to reach 97.7%. In addition, although the model previously output slightly worse results for U.S. in terms of AR@1, the performance of the network in the AR@1% metric is equal to the best previous result in the state-of-the-art with a value of 99.9%. The mean AR@1 and AR@1% over all 4 datasets improves by 0.4% and 0.2%, respectively. To conclude, training the MinkUNeXt with the refined protocol overcomes the generalization difficulties presented when training with the baseline protocol, since the model adapts to both LiDAR characteristics.

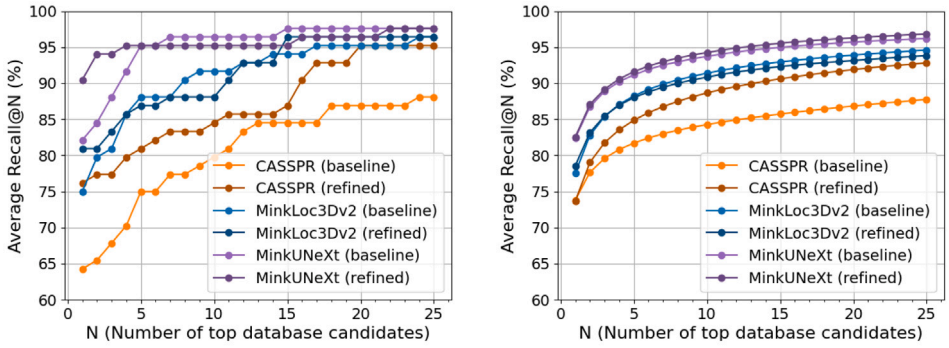
4.6.3. Results with the further test protocol

Given the potential limitations of the baseline and refined protocols, a further test protocol is employed to conduct an exhaustive assessment of the proposal. This evaluation is extended to the KITTI [56] and USyd [55] datasets to assess the generalization capability of the proposed model, MinkUNeXt, compared to other methods such as MinkLoc3Dv2 [39] and CASSPR [29].

The results are presented in Table 7. The data indicates that the proposed MinkUNeXt architecture consistently yields higher performance across both datasets and training protocols. Specifically, under the refined protocol, MinkUNeXt achieves an AR@1 of 90.5% in the KITTI dataset and 82.6% in the USyd dataset. These figures represent a notable improvement over the other evaluated methods. The enhanced performance is also reflected in the AR@1%, where MinkUNeXt outputs scores of 94.1% and 93.2% for KITTI and USyd, respectively.

Table 6
Evaluation results in terms of average recall at 1 (AR@1) and at 1% (AR@1%) of place recognition methods trained using the refined protocol.

Method	Oxford		U.S.		R.A.		B.D.		Mean	
	AR@1	AR@1%	AR@1	AR@1%	AR@1	AR@1%	AR@1	AR@1%	AR@1	AR@1%
PointNetVLAD [14]	63.3	80.1	86.1	94.5	82.7	93.1	80.1	86.5	78.0	88.6
PCAN [63]	70.7	86.4	83.7	94.1	82.5	92.5	80.3	87.0	79.3	90.0
DAGC [26]	71.5	87.8	86.3	94.3	82.8	93.4	81.3	88.5	80.5	91.0
LPD-Net [18]	86.6	94.9	94.4	98.9	90.8	96.4	90.8	94.4	90.7	96.2
SOE-Net [32]	89.3	96.4	91.8	97.7	90.2	95.9	89.0	92.6	90.1	95.7
MinkLoc3D [19]	94.8	98.5	97.2	99.7	96.7	99.3	94.0	96.7	95.7	98.6
PPT-Net [30]	–	98.4	–	99.7	–	99.5	–	95.3	–	98.2
SVT-Net [34]	94.7	98.4	97.0	**99.9**	95.2	99.5	94.4	97.2	95.3	98.8
TransLoc3D [37]	95.0	98.5	97.5	99.8	97.3	99.7	94.8	97.4	96.2	98.9
MinkLoc3Dv2 [39]	96.9	99.1	**99.0**	99.7	98.3	99.4	97.6	**99.1**	97.9	99.3
ComPoint [47]	69.3	84.7	87.2	95.8	85.6	92.5	82.6	87.6	81.2	90.2
CASSPR [29]	95.6	98.8	98.3	99.9	96.6	98.5	93.6	96.9	96.0	98.5
MinkUNeXt (ours)	**97.7**	**99.3**	98.7	**99.9**	**99.4**	**99.9**	**97.7**	99.0	**98.3**	**99.5**



(a) KITTI dataset (b) USyd dataset

Fig. 5. Recall@N results for N from 1 to 25 in the KITTI and USyd datasets, comparing our MinkUNeXt with state-of-the-art methods.

Table 7
Evaluation results in terms of average recall at 1 (AR@1) and at 1% (AR@1%) in KITTI and USyd when training the models both in the baseline and refined protocols, in which only Oxford or Oxford and In-House datasets are used to train.

Method	Trained with	KITTI		USyd	
		AR@1	AR@1%	AR@1	AR@1%
MinkLoc3Dv2 [39]	Baseline protocol	75.0	79.8	77.5	90.1
MinkLoc3Dv2 [39]	Refined protocol	81.0	81.0	78.6	89.7
CASSPR [29]	Baseline protocol	64.3	65.5	73.8	83.2
CASSPR [29]	Refined protocol	76.2	77.4	73.9	87.0
MinkUNeXt (ours)	**Baseline protocol**	**82.1**	**84.5**	**82.4**	**92.7**
MinkUNeXt (ours)	**Refined protocol**	**90.5**	**94.1**	**82.6**	**93.2**

To provide a more detailed analysis, Fig. 5 illustrates the Recall@N performance for N values ranging from 1 to 25. The plots for both the KITTI (left) and USyd (right) datasets show that the proposed method, MinkUNeXt, consistently outperforms MinkLoc3Dv2 [39] and CASSPR [29] across the entire range of N. This indicates that the global descriptors generated by MinkUNeXt are not only more accurate in identifying the top-1 candidate but are also more effective at retrieving the correct place descriptor within a larger set of top candidates.

These findings suggest the robustness and effectiveness of the MinkUNeXt architecture. The design of the model appears to facilitate the learning of more discriminative features for the place recognition task. The consistent results on different datasets and protocols support the viability of the proposed approach.

4.6.4. Computational cost analysis

In addition to the performance in place recognition, the computational efficiency of the proposed method is another key aspect to consider, especially for real-world applications. Table 8 presents a comparative analysis of the number of parameters and the inference time for the MinkUNeXt model compared to other state-of-the-art methods. The results indicate that MinkUNeXt has a significantly larger number of parameters (43.5 M) compared to the other architectures. This increased model complexity is a direct consequence of its deep, U-Net-based architecture, which is fundamental to its high descriptive performance and accuracy, as demonstrated in the previous sections. However, it is important to note that this increase in size does not translate into a prohibitive inference time, which is 10.8 ms. Therefore, our model remains competitive and operates in real-time, being faster than other recent methods such as CASSPR [29] (20.5 ms) and comparable to MinkLoc3Dv2 [39] (9.6 ms). The efficiency of MinkUNeXt comes from the use of sparse convolutions, which only perform computations on occupied voxels, effectively ignoring the vast empty space.

This analysis highlights a trade-off between model complexity and performance, where MinkUNeXt prioritizes achieving the highest possible accuracy while maintaining a viable inference speed for practical deployments.

4.6.5. Qualitative results

In this section, visual examples of the results obtained with MinkLoc3Dv2 [39], CASSPR [29] and the proposed MinkUNeXt, which were trained in the refined protocol, are presented for both the Oxford RobotCar and the In-house datasets (Fig. 6). These examples have been selected from the test sequences of each dataset to illustrate qualitatively the model's ability to recognize places in different environments and conditions.

Each row in the figure displays a single query point cloud captured by the LiDAR sensor, the nearest (in the metric space) point cloud from the database (i.e. the ground truth), and the top-1 prediction

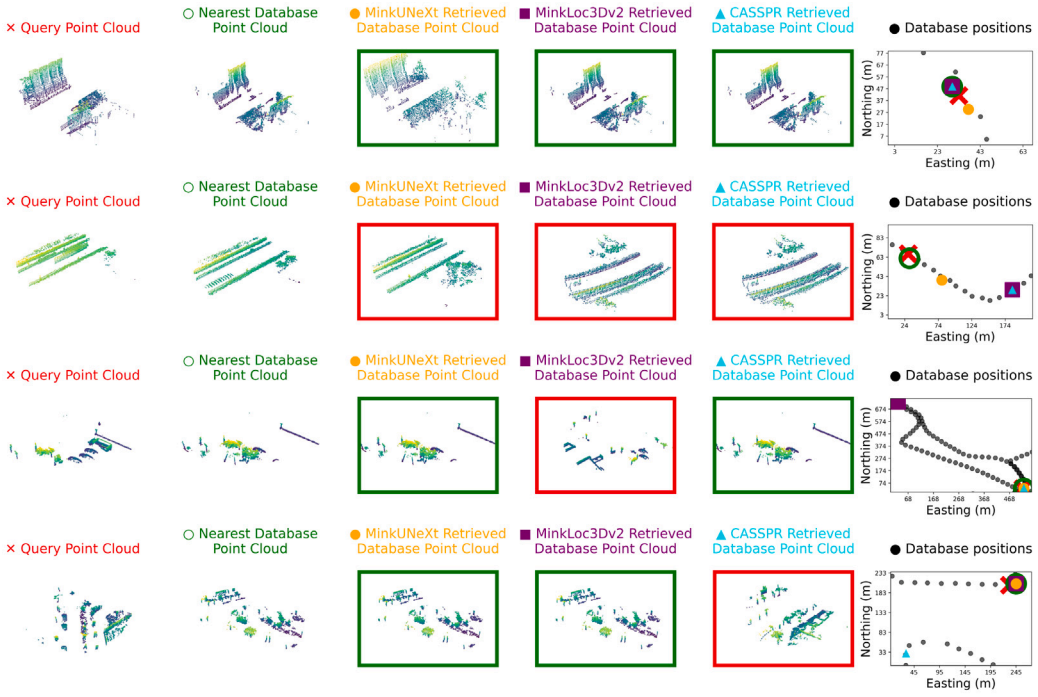


Fig. 6. Qualitative comparison in the Oxford RobotCar and In-house datasets. From top to bottom, the rows illustrate different scenarios: (1) a successful retrieval by all methods in Oxford; (2) a challenging case with perceptual aliasing from the University Sector scenario; (3) a scene with dynamic objects from the Residential Area environment; and (4) a query with significant rotation from one of the Business District sequences. A green border indicates a correct match, while red indicates a failure.

Table 8
Comparison of the number of parameters and inference time for different place recognition methods.

Method	Number of parameters (M)	Inference time (ms)
PointNetVLAD [14]	19.8	3.9
LPD-Net [18]	19.8	5.5
MinkLoc3D [19]	1.1	3.3
TransLoc3D [37]	11.0	7.3
MinkLoc3Dv2 [39]	2.7	9.6
CASSPR [29]	3.8	20.5
MinkUNeXt (ours)	43.5	10.8

retrieved from the database (using the descriptor space) by MinkUNeXt, MinkLoc3Dv2, and CASSPR. The map on the right visualizes the spatial metric context of these results. Specifically, the database positions are represented by black dots for the entire database, the red cross is the current query position, and the green ring is the nearest position from the database. The positions of the top-1 retrieved point clouds are marked with an orange circle for our MinkUNeXt, a purple square for MinkLoc3Dv2, and a cyan triangle for CASSPR. Additionally, in the point cloud visualizations, a green border indicates a correct retrieval, while a red border means a failure. Note that for both the Oxford and the In-house datasets, a retrieval is considered correct if the Euclidean distance between the query and the retrieved point cloud positions is below 25 m.

The top row, an example from the Oxford RobotCar dataset, shows a scenario where the three methods correctly retrieve a database point cloud within 25 m (a true positive). However, while CASSPR and MinkLoc3Dv2 retrieve the closest possible database location, MinkUNeXt retrieves the second nearest. In contrast, the second row highlights a more challenging case from the University Sector scenario of the in-house dataset, where all methods fail due to perceptual aliasing. In this instance, although MinkUNeXt also fails, its prediction is the closest one to the ground truth. Next, the third row, from the Residential Area of the In-house dataset, displays a query point cloud with dynamic objects (three cars and a van) that confuse MinkLoc3Dv2, whereas MinkUNeXt and CASSPR perform the task correctly. Finally, the last row, taken from the Business District of the In-house dataset, shows an example where the query point cloud is rotated with respect to the database. In this case, both MinkUNeXt and MinkLoc3Dv2 exhibit greater robustness to rotation than CASSPR.

5. Conclusion

This paper presents MinkUNeXt, an architecture based on MinkUNet [20] exhaustively modified and enhanced to perform place-recognition based on point clouds. It is an encoder–decoder architecture entirely based on the proposed 3D MinkNeXt Block: a residual block composed of 3D sparse convolutions that follows the philosophy proposed by ConvNeXt [16]. The feature extraction step is performed by a U-Net encoder–decoder. The feature aggregation of those features into a single descriptor is carried out by a Generalized Mean Pooling (GeM) [48]. The designed architecture demonstrates that it is possible to surpass the current state of the art by only relying on conventional 3D sparse convolutions without making use of more complex and sophisticated proposals such as Transformers, Attention-Layers or Deformable Convolutions.

The proposed network shows that the usage of a U-Net architecture for point cloud-based place recognition is beneficial, since it is able to capture both detailed and contextual information of the three-dimensional environment. The fusion of features from multiple spatial scales improves the robustness of the place recognition model, allowing it to adapt to variations in point cloud geometry and density, as well as to different scenarios.

It should be also noted that the proposed method outputs results outperforming an already saturated state-of-the-art. In particular, the network achieved an AR@1 of 97.5% and an AR@1% of 99.3% when trained with the refined protocol. Furthermore, the generalization capability of MinkUNeXt was validated through a further test protocol in the KITTI and USyd datasets, where it consistently outperformed state-of-the-art methods like MinkLoc3Dv2 and CASSPR. The computational

analysis revealed that despite a larger number of parameters, the inference time of our model remains competitive, highlighting that model complexity is not directly proportional to latency when leveraging efficient sparse convolutions. Qualitative results further substantiated these findings, visually demonstrating the model's superior robustness in challenging real-world scenarios, including those with significant perceptual aliasing, dynamic objects and viewpoint rotations.

Future work will consider the inclusion of visual information into the place recognition system. In this sense, we consider that it would result in a richer representation of the environment compared to the use of LiDAR with pure distance data. However, visual information is hindered by changing lighting conditions, weather and seasonal changes, which pose a great challenge.

CRediT authorship contribution statement

Juan José Cabrera: Writing – original draft, Software, Methodology, Investigation, Data curation. **Antonio Santo:** Software, Methodology, Investigation, Data curation. **Arturo Gil:** Writing – review & editing, Validation, Supervision, Software, Methodology, Conceptualization. **Carlos Viegas:** Validation, Resources, Project administration, Conceptualization. **Luis Payá:** Writing – review & editing, Validation, Supervision, Project administration, Conceptualization.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Juan Jose Cabrera reports financial support was provided by Spain Ministry of Science and Innovation. Luis Paya reports was provided by Spain Ministry of Science and Innovation. Carlos Viegas reports financial support was provided by Portuguese Foundation for Science and Technology. Arturo Gil reports financial support was provided by Spain Ministry of Science and Innovation. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The Ministry of Science, Innovation and Universities (Spain) has supported this work through FPU21/04969 (J.J. Cabrera). This research work is part of the project TED2021-130901B-I00 funded by MICIU/AEI/10.13039/501100011033 and by the European Union NextGenerationEU/PRTR. It is also part of the project PID2023-149575 OB-I00 funded by MICIU/AEI/10.13039/501100011033 and by FEDER, UE. Furthermore, it is part of the project CIPROM/2024/8, funded by Generalitat Valenciana, Conselleria de Educación, Cultura, Universidades y Empleo (program PROMETEO 2025). In addition, this research was sponsored by national funds (Portugal) through FCT - Fundação para a Ciência e a Tecnologia, under project LA/P/0079/2020, DOI: 10.54499/LA/P/0079/20-20. This work was further funded in the scope of the projects E-Forest–Multi-agent Autonomous Electric Robotic Forest Management Framework, ref. POCI-01-0247-FEDER-047104 and F4F–Forest for Future, ref. CENTRO-08-5864-FSE-000031, co-financed by European Funds through the programs Compete 2020 and Portugal 2020.

Data availability

Data is publicly available on the project website: https://juanjo-cabrera.github.io/projects-MinkUNeXt/.

References

[1] Yin H, Tang L, Ding X, Wang Y, Xiong R. LocNet: Global localization in 3D point clouds for mobile vehicles. In: 2018 IEEE intelligent vehicles symposium. IEEE; 2018, p. 728–33.
[2] Kim G, Kim A. Scan context: Egocentric spatial descriptor for place recognition within 3D point cloud map. In: 2018 IEEE/RSJ international conference on intelligent robots and systems. IEEE; 2018, p. 4802–9.
[3] Han J, Shin M, Paik J. Robust point cloud registration using hough voting-based correspondence outlier rejection. Eng Appl Artif Intell 2024;133:107985. http://dx.doi.org/10.1016/j.engappai.2024.107985.
[4] Moskalenko I, Kornilova A, Ferrer G. Visual place recognition for aerial imagery: A survey. Robot Auton Syst 2025;183:104837. http://dx.doi.org/10.1016/j.robot.2024.104837.
[5] Himstedt M, Frost J, Hellbach S, Böhme H-J, Maehle E. Large scale place recognition in 2D LiDAR scans using geometrical landmark relations. In: 2014 IEEE/RSJ international conference on intelligent robots and systems. IEEE; 2014, p. 5030–5.
[6] Dong H, Chen X, Särkkä S, Stachniss C. Online pole segmentation on range images for long-term lidar localization in urban environments. Robot Auton Syst 2023;159:104283. http://dx.doi.org/10.1016/j.robot.2022.104283.
[7] Komorowski J, Wysoczanska M, Trzcinski T. Large-scale topological radar localization using learned descriptors. In: Neural information processing: 28th international conference, ICONIP 2021, sanur, bali, Indonesia, December 8–12, 2021, proceedings, part II 28. Springer; 2021, p. 451–62.
[8] Tang X, Fu W, Jiang M, Peng G, Wu Z, Yue Y, Wang D. Place recognition using line-junction-lines in urban environments. In: 2019 IEEE international conference on cybernetics and intelligent systems (CIS) and IEEE conference on robotics, automation and mechatronics. 2019, p. 530–5. http://dx.doi.org/10.1109/CIS-RAM47153.2019.9095776.
[9] Arshad S, Kim G-W. A robust feature matching strategy for fast and effective visual place recognition in challenging environmental conditions. Int J Control Autom Syst 2023;21(3):948–62.
[10] Zhou S, Tian Z, Chu X, Zhang X, Zhang B, Lu X, et al. Fastpillars: A deployment-friendly pillar-based 3d detector. 2023, arXiv preprint arXiv:2302.02367.
[11] Zhou S, Li L, Zhang X, Zhang B, Bai S, Sun M, et al. Lidar-ptq: Post-training quantization for point cloud 3d object detection. 2024, arXiv preprint arXiv:2401.15865.
[12] Zhou S, Yuan Z, Yang D, Hu X, Qian J, Zhao Z. Pillarhist: A quantization-aware pillar feature encoder based on height-aware histogram. In: Proceedings of the computer vision and pattern recognition conference. 2025, p. 27336–45.
[13] Wang Y, Sun Z, Xu C-Z, Sarma SE, Yang J, Kong H. LiDAR iris for loop-closure detection. In: 2020 IEEE/RSJ international conference on intelligent robots and systems. IEEE; 2020, p. 5769–75.
[14] Uy MA, Lee GH. PointNetVLAD: Deep point cloud based retrieval for large-scale place recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2018, p. 4470–9.
[15] Chen X, Läbe T, Milioto A, Röhling T, Behley J, Stachniss C. OverlapNet: A siamese network for computing LiDAR scan similarity with applications to loop closing and localization. Auton Robots 2022;1–21.
[16] Liu Z, Mao H, Wu C-Y, Feichtenhofer C, Darrell T, Xie S. A convnet for the 2020s. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2022, p. 11976–86.
[17] Qi CR, Su H, Mo K, Guibas LJ. Pointnet: Deep learning on point sets for 3D classification and segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2017, p. 652–60.
[18] Liu Z, Zhou S, Suo C, Yin P, Chen W, Wang H, et al. LPD-Net: 3D point cloud learning for large-scale place recognition and environment analysis. In: Proceedings of the IEEE/CVF international conference on computer vision. 2019, p. 2831–40.
[19] Komorowski J. Minkloc3D: Point cloud based large-scale place recognition. In: Proceedings of the IEEE/CVF winter conference on applications of computer vision. 2021, p. 1790–9.
[20] Choy C, Gwak J, Savarese S. 4D Spatio-temporal ConvNets: Minkowski convolutional neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2019, p. 3075–84.
[21] Radenović F, Tolias G, Chum O. Fine-tuning CNN image retrieval with no human annotation. IEEE Trans Pattern Anal Mach Intell 2018;41(7):1655–68.
[22] Komorowski J, Wysoczańska M, Trzcinski T. MinkLoc++: LiDAR and monocular image fusion for place recognition. In: 2021 international joint conference on neural networks. IEEE; 2021, p. 1–8.
[23] Lu Y, Yang F, Chen F, Xie D. Pic-net: Point cloud and image collaboration network for large-scale place recognition. 2020, arXiv preprint arXiv:2008.00658.
[24] Song H, Choi W, Kim H. Robust vision-based relative-localization approach using an RGB-depth camera and LiDAR sensor fusion. IEEE Trans Ind Electron 2016;63(6):3725–36. http://dx.doi.org/10.1109/TIE.2016.2521346.
[25] Żywanowski K, Banaszczyk A, Nowicki MR, Komorowski J. MinkLoc3D-SI: 3D LiDAR place recognition with sparse convolutions, spherical coordinates, and intensity. IEEE Robot Autom Lett 2021;7(2):1079–86.

[26] Sun Q, Liu H, He J, Fan Z, Du X. Dagc: Employing dual attention and graph convolution for point cloud based place recognition. In: Proceedings of the 2020 international conference on multimedia retrieval. 2020, p. 224–32.

[27] Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, Polosukhin I. Attention is all you need. Adv Neural Inf Process Syst 2017;30.

[28] Zhou Z, Zhao C, Adolfsson D, Su S, Gao Y, Duckett T, et al. NDT-transformer: Large-scale 3D point cloud localisation using the normal distribution transform representation. In: 2021 IEEE international conference on robotics and automation. IEEE; 2021, p. 5654–60.

[29] Xia Y, Gladkova M, Wang R, Li Q, Stilla U, Henriques JF, Cremers D. Casspr: Cross attention single scan place recognition. In: Proceedings of the IEEE/CVF international conference on computer vision. 2023, p. 8461–72.

[30] Hui L, Yang H, Cheng M, Xie J, Yang J. Pyramid point cloud transformer for large-scale place recognition. In: Proceedings of the IEEE/CVF international conference on computer vision. 2021, p. 6098–107.

[31] Li G, Zhang R. A point is a wave: point-wave network for place recognition. In: ICASSP 2023-2023 IEEE international conference on acoustics, speech and signal processing. IEEE; 2023, p. 1–5.

[32] Xia Y, Xu Y, Li S, Wang R, Du J, Cremers D, et al. SOE-Net: A self-attention and orientation encoding network for point cloud based place recognition. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2021, p. 11348–57.

[33] Wiesmann L, Marcuzzi R, Stachniss C, Behley J. Retriever: Point cloud retrieval in compressed 3D maps. In: 2022 international conference on robotics and automation. IEEE; 2022, p. 10925–32.

[34] Fan Z, Song Z, Liu H, Lu Z, He J, Du X. SVT-Net: Super light-weight sparse voxel transformer for large scale place recognition. In: Proceedings of the AAAI conference on artificial intelligence. 36, (1):2022, p. 551–60.

[35] Hou Z, Yan Y, Xu C, Kong H. 2022 international conference on robotics and automation. IEEE; 2022, p. 2612–8.

[36] Qi CR, Yi L, Su H, Guibas LJ. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. Adv Neural Inf Process Syst 2017;30.

[37] Xu T-X, Guo Y-C, Li Z, Yu G, Lai Y-K, Zhang S-H. TransLoc3D: Point cloud based large-scale place recognition using adaptive receptive fields. 2021, arXiv preprint arXiv:2105.11605.

[38] Wang Q, Wu B, Zhu P, Li P, Zuo W, Hu Q. ECA-Net: Efficient channel attention for deep convolutional neural networks. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2020, p. 11534–42.

[39] Komorowski J. Improving point cloud based place recognition with ranking-based loss and large batch training. In: 2022 26th international conference on pattern recognition. IEEE; 2022, p. 3699–705.

[40] Maddern W, Pascoe G, Linegar C, Newman P. 1 year, 1000 km: The Oxford robotcar dataset. Int J Robot Res 2017;36(1):3–15.

[41] Wiesmann L, Nunes L, Behley J, Stachniss C. KPPR: Exploiting momentum contrast for point cloud-based place recognition. IEEE Robot Autom Lett 2022;8(2):592–9.

[42] Thomas H, Qi CR, Deschaud J-E, Marcotegui B, Goulette F, Guibas LJ. Kpconv: Flexible and deformable convolution for point clouds. In: Proceedings of the IEEE/CVF international conference on computer vision. 2019, p. 6411–20.

[43] Lin CE, Song J, Zhang R, Zhu M, Ghaffari M. Se (3)-equivariant point cloud-based place recognition. In: Conference on robot learning. PMLR; 2023, p. 1520–30.

[44] Fan Z, Song Z, Zhang W, Liu H, He J, Du X. RPR-Net: A point cloud-based rotation-aware large scale place recognition network. In: European conference on computer vision. Springer; 2022, p. 709–25.

[45] Hui L, Cheng M, Xie J, Yang J, Cheng M-M. Efficient 3D point cloud feature learning for large-scale place recognition. IEEE Trans Image Process 2022;31:1258–70.

[46] Hou Z, Shang Y, Gao T, Yan Y. BPT: binary point cloud transformer for place recognition. 2023, arXiv preprint arXiv:2303.01166.

[47] Zhang R, Li G, Gao W, Li TH. ComPoint: Can complex-valued representation benefit point cloud place recognition? IEEE Trans Intell Transp Syst 2024;25(7):7494–507.

[48] Radenović F, Tolias G, Chum O. Fine-tuning CNN image retrieval with no human annotation. IEEE Trans Pattern Anal Mach Intell 2018;41(7):1655–68.

[49] Ronneberger O, Fischer P, Brox T. U-net: Convolutional networks for biomedical image segmentation. In: Medical image computing and computer-assisted intervention–mICCAI 2015: 18th international conference, munich, Germany, October 5-9, 2015, proceedings, part III 18. Springer; 2015, p. 234–41.

[50] Sünderhauf N, Dayoub F, McMahon S, Talbot B, Schulz R, Corke P, Wyeth G, Upcroft B, Milford M. Place categorization and semantic mapping on a mobile robot. In: 2016 IEEE international conference on robotics and automation. IEEE; 2016, p. 5729–36.

[51] Sandler M, Howard A, Zhu M, Zhmoginov A, Chen L-C. Mobilenetv2: Inverted residuals and linear bottlenecks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2018, p. 4510–20.

[52] Hendrycks D, Gimpel K. Gaussian error linear units (gelus). 2016, arXiv preprint arXiv:1606.08415.

[53] Ba JL, Kiros JR, Hinton GE. Layer normalization. 2016, arXiv preprint arXiv:1607.06450.

[54] Ioffe S. Batch renormalization: Towards reducing minibatch dependence in batch-normalized models. Adv Neural Inf Process Syst 2017;30.

[55] Zhou W, Perez JSB, De Alvis C, Shan M, Worrall S, Ward J, et al. The USyd campus dataset. 2019, http://dx.doi.org/10.21227/sk74-7419.

[56] Geiger A, Lenz P, Stiller C, Urtasun R. Vision meets robotics: The kitti dataset. Int J Robot Res 2013;32(11):1231–7.

[57] Xie S, Girshick R, Dollár P, Tu Z, He K. Aggregated residual transformations for deep neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2017, p. 1492–500.

[58] Dosovitskiy A, Beyer L, Kolesnikov A, Weissenborn D, Zhai X, Unterthiner T, et al. An image is worth 16x16 words: Transformers for image recognition at scale. 2020, arXiv preprint arXiv:2010.11929.

[59] Liu Z, Lin Y, Cao Y, Hu H, Wei Y, Zhang Z, et al. Swin transformer: Hierarchical vision transformer using shifted windows. In: Proceedings of the IEEE/CVF international conference on computer vision. 2021, p. 10012–22.

[60] Nair V, Hinton GE. Rectified linear units improve restricted boltzmann machines. In: Proceedings of the 27th international conference on machine learning (ICML-10). 2010, p. 807–14.

[61] Devlin J, Chang M-W, Lee K, Toutanova K. Bert: Pre-training of deep bidirectional transformers for language understanding. 2018, arXiv preprint arXiv:1810.04805.

[62] Achiam J, Adler S, Agarwal S, Ahmad L, Akkaya I, Aleman FL, et al. Gpt-4 technical report. 2023, arXiv preprint arXiv:2303.08774.

[63] Zhang W, Xiao C. PCAN: 3D attention map learning using contextual information for point cloud based retrieval. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2019, p. 12436–45.

[64] Shu DW, Kwon J. Hierarchical bidirected graph convolutions for large-scale 3D point cloud place recognition. IEEE Trans Neural Networks Learn Syst 2023.