A Coarse to Fine 3D LiDAR Localization with Deep Local Features for Long Term Robot Navigation in Large Environments

Míriam Máximo[1] | Antonio Santo[1] | Arturo Gil[1] | Mónica Ballesta[1] | David Valiente[1]

[1]University Institute for Engineering Research, Miguel Hernández University, Avda. de la Universidad s/n, 03202 Elche, Alicante, Spain

Correspondence
Corresponding author Míriam Máximo.
Email: mmaximo@umh.es

The location of a robot is a key aspect in the field of mobile robotics. This problem is particularly complex when the initial pose of the robot is unknown. In order to find a solution, it is necessary to perform a global localization. In this paper, we propose a method that addresses this problem using a coarse-to-fine solution. The coarse localization relies on a probabilistic approach of the Monte Carlo Localization (MCL) method, with the contribution of a robust deep learning model, the MinkUNeXt neural network, to produce a robust description of point clouds of a 3D LiDAR within the observation model. The MCL method has been approached from a topological perspective, considering that the particles are initialized on the map positions where LiDAR scans have been previously captured. For fine localization, global point cloud registration has been implemented. MinkUNeXt aids this by exploiting the outputs of its intermediate layers to produce deep local features for each point in a scan. These features facilitate precise alignment between the current sensor observation (query) and one of the point clouds on the map. The proposed MCL method incorporating Deep Local Features for fine localization is termed MCL-DLF. Alternatively, a classical ICP method has been implemented for this precise localization aiming at comparison purposes. This method is termed MCL-ICP. In order to validate the performance of MCL-DLF method, it has been tested on publicly available datasets such as the NCLT dataset, which provides seasonal large-scale environments. Additionally, tests have been also performed with own data (UMH) that also includes seasonal variations on large indoor/outdoor scenarios. The results, which were compared with established state-of-the-art methodologies, demonstrate that the MCL-DLF method obtains an accurate estimate of the robot localization in dynamic environments despite changes in environmental conditions. For reproducibility purposes, the code is publicly available [†].

KEYWORDS
Global Localization, Fine Localization, Point Cloud, Light detection and ranging (LiDAR), Pose estimation, Feature descriptor, 3D deep learning

1 | INTRODUCTION

The fundamental characteristic of a mobile robot [1] lies in its capacity for autonomous navigation. This capability needs precise determination of its position and orientation, a problem commonly referred to as localization [2]. The algorithms developed to address this localization challenge can be broadly categorized as global or local, depending on the robot's initial state.

Global localization algorithms are designed to operate under the assumption that the robot's initial pose is unknown, a scenario often termed the kidnapped robot problem. These methods enable the coarse estimation of the robot's pose leveraging sensor measurements.

However, some of these sensors have limitations. For instance, Global Navigation Satellite System (GNSS)-based positioning systems do not provide reliable data. This is due to the presence of significant errors, particularly in areas with close proximity to buildings, where there is no reliable satellite signal. Furthermore, this signal is not available in indoor environments either, rendering these sensors unsuitable for such applications and necessitating alternative sensing modalities for ac-

[†]`https://github.com/miriammaximo/MCL-DLF.git`

curate localization. Other sensors that provide external environmental information include cameras, radar, and LiDAR.

The use of cameras [3] enables a localization system based on visual perception of the environment. Image features are extracted through a variety of methods, ranging from more classical approaches [4, 5] to more recent methods that integrate deep learning networks [6]. While cameras offer a rich source of environmental information and are cost-effective, they are highly susceptible to variations in lighting conditions and environmental factors such as fog or rain, limiting their applicability in dynamic environments.

In contrast, LiDAR (Light Detection and Ranging) sensors [7] exhibit superior robustness to changes in illumination and environmental conditions. By employing laser pulses to measure distances and construct 3D environmental maps, LiDAR sensors provide high accuracy in obstacle and environmental feature detection [8, 9].

Within the context of global localization, place recognition methods play a pivotal role [10, 11]. These methods aim to identify previously visited locations by comparing current sensor measurements against a database of known maps, frequently constructed using SLAM techniques [12]. In many instances, this comparison is achieved through methods that infer the similarity between two point clouds, often by embedding each point cloud into a global descriptor and subsequently comparing these descriptors using techniques such Euclidean distance or cosine distance for nearest neighbor search.

A common technique for global localization is MCL [13], a particle filter-based method. A key aspect of MCL is the utilization of environmental information to update the particle set, representing hypotheses of the robot's pose. This information can be derived from various sources, including image features [14, 15], LiDAR scan features [16], or sensor fusion [17]. Thus, incorporating point cloud similarity into the observation model allows for more robust localization compared to a simple nearest neighbor search.

Place recognition methods [18] typically yield a coarse pose estimate, necessitating the integration of local localization methods for refined pose estimation. These local localization algorithms aim to determine the robot's pose relative to its immediate surroundings. Dead reckoning, which estimates the robot's current pose based on its last known position and motion measurements from sensors like wheel encoders, is a common approach. However, it is prone to error accumulation due to factors such slip or friction of the wheels. Thus, it is necessary to combine it with other methods to achieve greater accuracy.

Point cloud registration algorithms are employed for precise local localization. This can be addressed with the position of the points, as in Iterative Closest Point (ICP) [19] and related methods [20], or by utilizing 3D point cloud features [21, 22]. In some cases this features are obtained with a deep neural network [23, 24].

All in all, to implement a complete localization method, a coarse-to-fine approach is often adopted. This strategy involves initially obtaining an approximate global pose, subsequently refined through local localization techniques. Some methods focus on extracting local features and global descriptors from point clouds using the same network [25]. However, in certain instances, it is requisite for the architecture to comprise a dedicated network branch for the derivation of the global descriptor and a separate branch for the extraction of keypoints and local descriptors, thereby necessitating additional convolution operations for these outputs.

In this paper, we explore a similar coarse-to-fine localization strategy. For coarse localization, we employ MCL with an observation model that compares point clouds. This comparison is performed by embedding the point clouds into a deep learning descriptor using the MinkUNeXt network [26], a U-Net architecture that utilizes 3D sparse convolutions.

The implementation of sparse networks facilitates the processing of three-dimensional data, which can attain excessively large dimensions. Given the inherent sparsity of 3D point clouds, these networks efficiently process only occupied points or voxels, significantly reducing computational overhead compared to dense networks. Additionally, U-Net architectures [27], with their encoder-decoder design, effectively capture spatial relationships, detailing 3D geometry analysis, focusing on relevant data while mitigating the impact of noise.

Following the coarse localization stage, a fine localization method is applied. This method analyzes correspondences between local features of LiDAR scans, comparing the current scan with a map scan from the region identified by the coarse localization. Feature extraction is performed using the same network employed in coarse localization, but in this case utilizing intermediate layer outputs. Notably, notwithstanding that this network is not specifically trained to obtain local features. This paper aims to investigate point cloud registration using sparse networks without relying on predefined keypoints.

The proposed methodology enables precise robot pose estimation without prior knowledge of the initial location. This approach has been evaluated in environments characterized by dynamic changes, including the presence of people, vehicles, and seasonal variations.

The contributions of this paper are presented as follows:

- A complete method, termed MCL-DLF, for the coarse-to-fine localization of mobile robots in large scale environments and environmental changing scenarios. This approach is suitable for both outdoor and indoor environments, achieving robust performance during handover situations, specifically transitions from outdoor to indoor settings where environmental conditions undergo significant alterations.
- An implementation of the MCL method with an observation model that allows robust coarse localization by using point clouds embedded in a deep learning descriptor. This descriptor is provided by the use of a 3D sparse convolutional neural, MinkUNeXt.
- A point cloud registration method for fine localization that reduces position and orientation errors to achieve an accurate solution to the robot's pose. This method involves the correspondence between local features of the point cloud, obtained through the intermediate layers of the MinkUNeXt network. These correspondences are then used to perform the registration with RANSAC.

2 | RELATED WORK

2.1 | Point cloud-based localization

Localization based on 3D LiDAR data has become very popular in recent years due to the fact that this type of sensor is able to capture the elements of the environment with a high degree of accuracy. Furthermore, its robustness to variations in illumination and seasonal conditions makes it preferable to other sensors such as cameras.

There are two approaches to LiDAR localization: coarse localization, when the robot's position is initially unknown; and point cloud registration, which refines the position after it is approximately estimated.

2.1.1 | Coarse localization

Several publications have focused on identifying previously visited locations, a task known as place recognition. This is accomplished using 3D data by comparing the similarity between point cloud representations to retrieve a query from a map database. Point clouds are encoded as descriptors that encapsulate the global geometric features of the scene, a process facilitated by deep learning methodologies.

One of the earliest studies related to this topic was PointNetVLAD [28]. In this work, PointNet [29] and NetVLAD [30] are combined. The extraction of local features is facilitated by PointNet, and these features are then compactly grouped into a single descriptor by NetVLAD. The training is performed by applying metric learning, so that it is capable of generating similar descriptors for similar areas. This is done by applying a loss function called "lazy triplet and quadruplet" which maximizes the differences between descriptors of distant point clouds.

Further work [31] is also focused on using point clouds in place recognition, including the relative orientation between the two point clouds. In this case, the point cloud is converted to a 2D range image to be used as input to the network. The network consists of a convolutional neural network that provides two outputs: a rotation-invariant vector v and a vector w that embeds the rotation information. The vector v is compared with the vectors of the map's point clouds to perform a nearest neighbor search, and once the closest one is obtained, the relative orientation is obtained using the w vectors of each cloud.

On [32], it also focuses on the orientation between point clouds. The network input, in this case, is a 2D image obtained from the horizontal plane projection of a point cloud. Utilizing a feature extraction module, 2D images are transformed into their frequency domain representation, facilitating the creation of distinct place descriptors. Furthermore, the system incorporates a differentiable phase correlation module, which calculates the relative orientation between scan pairs through correlation analysis performed within the frequency spectrum.

Another method proposed in [33], estimates the degree of similarity between two LiDAR scans and their relative orientation. The network input is a pair of LiDAR scans, which are converted into spherical projections. The network architecture is a siamese, which has two branches with shared weights. The outputs of these branches both go to two different ones. The output is the degree of overlap between them, and the yaw angle between them.

Recently, other architectures such as MinkUNeXt [26] relies on the encoding of point clouds into global descriptors using 3D sparse convolutions, this allows the use of all the information from 3D point clouds without the need to be projected into a 2D image, so a large amount of information is captured, but in an efficient way, since sparse networks allow focusing only on the areas of the scan where there are points. Furthermore, the use of networks with U-Net architecture allows capturing the relative information between objects in the scene at different resolutions. The developed architecture has proven to be robust on large-scale datasets such as the Oxford RobotCar dataset [34], demonstrating better performance than other architectures.

2.1.2 | Fine Localization

Fine localization can be achieved by performing a global point cloud registration. This approach provides a more accurate estimate of the position and orientation of the robot. As a consequence, a LiDAR scan can be aligned with a reference map or with a previously captured scan. The result of this process is a transformation matrix, which includes information about the 6 degrees of freedom of the robot.

A classic method used for point cloud registration is ICP [19]. This method requires an initial transformation that roughly aligns the point clouds. Then, the method performs a refinement of this transform. However, when the initial transformation differs significantly from the correct alignment between point clouds, the method does not guarantee satisfactory results. For this reason alternative global registration techniques have been developed, such as [35], where performance is improved without the need of a more precise initial transform.

In other papers, the alignment between point clouds is achieved by matching local features of the point clouds. These local features are obtained through the use of neural networks. In [36], a neural network capable of describing local areas is developed to set up correspondences between 3D data.

In more recent work [37], deep neural networks are used to learn robust representations and associate corresponding points between point clouds, improving registration accuracy in the presence of misalignment and noise. These deep learning-based approaches offer a significant improvement over traditional methods, providing a more efficient and accurate solution for point cloud registration.

Other approaches [38], take into account the registration of point clouds from different modalities, i.e., captured with different sensors. This is achieved by employing feature filtering to select points with high confidence, then detecting the local areas with the most relevant information. Finally, a transformation is obtained by optimising the alignment between the scans from local to global. This approach ensures a highly accurate registration, making it particularly useful in environments where multiple types of sensors are used.

2.2 | Localization with Monte Carlo algorithm

The MCL algorithm is a widely used method in the field of mobile robotics to solve the kidnapped robot problem and to estimate the pose of a robot in a previously known environment. This method is based on particle filters. One of the best-recognized approaches was [13], where the basis for the use of particle filters was established in the field of localization.

The key aspect of this method is the use of a good observation model which determines, depending on the current environment captured by the robot, which weight is given to each of the particles that are distributed in a space. In some works, these observation models have relied on the appearance of images. In [14], visual descriptors extracted from images are used as an observation method. However, the use of such observation methods presents challenges due to the noise in the images.

In [33], the similarity of two LiDAR scans have been used as an observation method. This method uses deep neural networks to obtain the similarity between the two point clouds. It has been shown that this type of observation method works well in large and challenging environments.

A methodology for localization in large-scale environments is proposed in [39]. The proposed method utilizes a deep neural network to learn a probability distribution of the robot's pose from bird's-eye view (BEV) images of LiDAR point clouds. This learned distribution is used to initialize the particles in MCL, instead of a random uniform distribution. This allows MCL to start with considerably more accurate pose estimation, which accelerates convergence and improves overall accuracy.

In this paper, we propose an integration of MCL with an observation model predicated on the MinkUNeXt network. MinkUNeXt's encoder-decoder architecture, coupled with its sparse nature, facilitates the extraction of detailed and differentiated information from LiDAR scans, adapting to diverse environmental configurations. Consequently, by employing a pre-existing map enriched with this information, the MCL method exhibits accelerated convergence towards the optimal solution. Furthermore, the method has significant robustness, enabling the acquisition of precise localizations even in the presence of map areas with environmental similarities but distant locations.

3 | METHODOLOGY

The following section will provide a detailed description of the methodology employed in the localization process, including both coarse and fine localization. The coarse localization is supported by the MCL method, using as an observation model the descriptors extracted with the neural network MinkUNeXt from the 3D LiDAR scans. Once this first estimate for the robot's pose has been obtained, fine localization is carried out, for which two alternative methods are proposed. The first alternative is the classic ICP method. The integrated method resulting from the combination of MCL and ICP, is refered to throughout this paper as MCL-ICP. The second alternative is to use the output of one of the intermediate layers of the MinkUNeXt network. This approach extracts information from the scan points in order to establish the transformation between the two point clouds. The combination of the MCL method and this Deep Local Features (DLF) is denoted as MCL-DLF. The coarse and fine methods are illustrated in Figure 1.

3.1 | Monte Carlo Localization method

3.1.1 | Description of the method

The MCL algorithm [13] aims to estimate the position and orientation of the robot, i.e. its state $x_t = (x, y, \theta)$ at time t, utilizing data $Z_k = z_{k,i} = 1, ..., k$ of the environment and the robot's movements $u_{1:t} = \{u_1, u_2, ..., u_t\}$. The probability density function $p(x_t|z_{1:t}, u_{1:t})$ is represented by M random samples, termed particles $S_k = \{s_k^i; i = 1, ..., M\}$.

At the initialization of this method at time $t = t_0$, a random set of particles $S_0 = s_0^i$ is generated. In this paper, the MCL method has been implemented using a topological approach, so that the particles are initialized at the (x, y) positions where the point clouds that constitute the map have been captured. The initial orientation of each particle is random. Consequently, each particle is given by $s_0 = \{x_0, y_0, \theta_0\}$. Once the initial set of particles is defined, the following process is performed iteratively:

- Prediction phase: In this phase, a new set of particles, denoted by S_t and corresponding to the time t, is generated from the preceding set, denoted by S_{t-1}, and a control signal u_t. This new set is the result of shifting each of the particles in the set at the previous time according to the motion of the control signal. The set S_t represents the density $p(x_t|z_1{:}t, u_1{:}t)$. The motion model shall be defined in Section 3.1.2.
- Update phase: In this phase, the observation z_t taken by the robot is used to calculate each of the weights w_t^i of the particles in the set S_t. The process of weight assignment is described in Section 3.1.3. From this process, $s_t^i = (x_t^i, w_t^i)$ is obtained for each of the particles on which afterwards the resampling is performed. The resampling process is conducted randomly with the probability of selection given by the values of the particle weights. The resulting set of particles will have the same extent M as the input set, and will be composed of the particles with the highest weights.

Once these two procedures have been carried out, the estimated position of the MCL method will be given by Equation 1 as the mean position of the set of particles of that iteration.

$$\hat{s}_t = \frac{1}{M} \sum_{i=1}^{M} s_t^{[i]} \tag{1}$$

Subsequently, the process continues iteratively. In order to obtain a more accurate position, the position $\hat{s}_t$ obtained in each iteration is refined in accordance with the process described in Section 3.2.

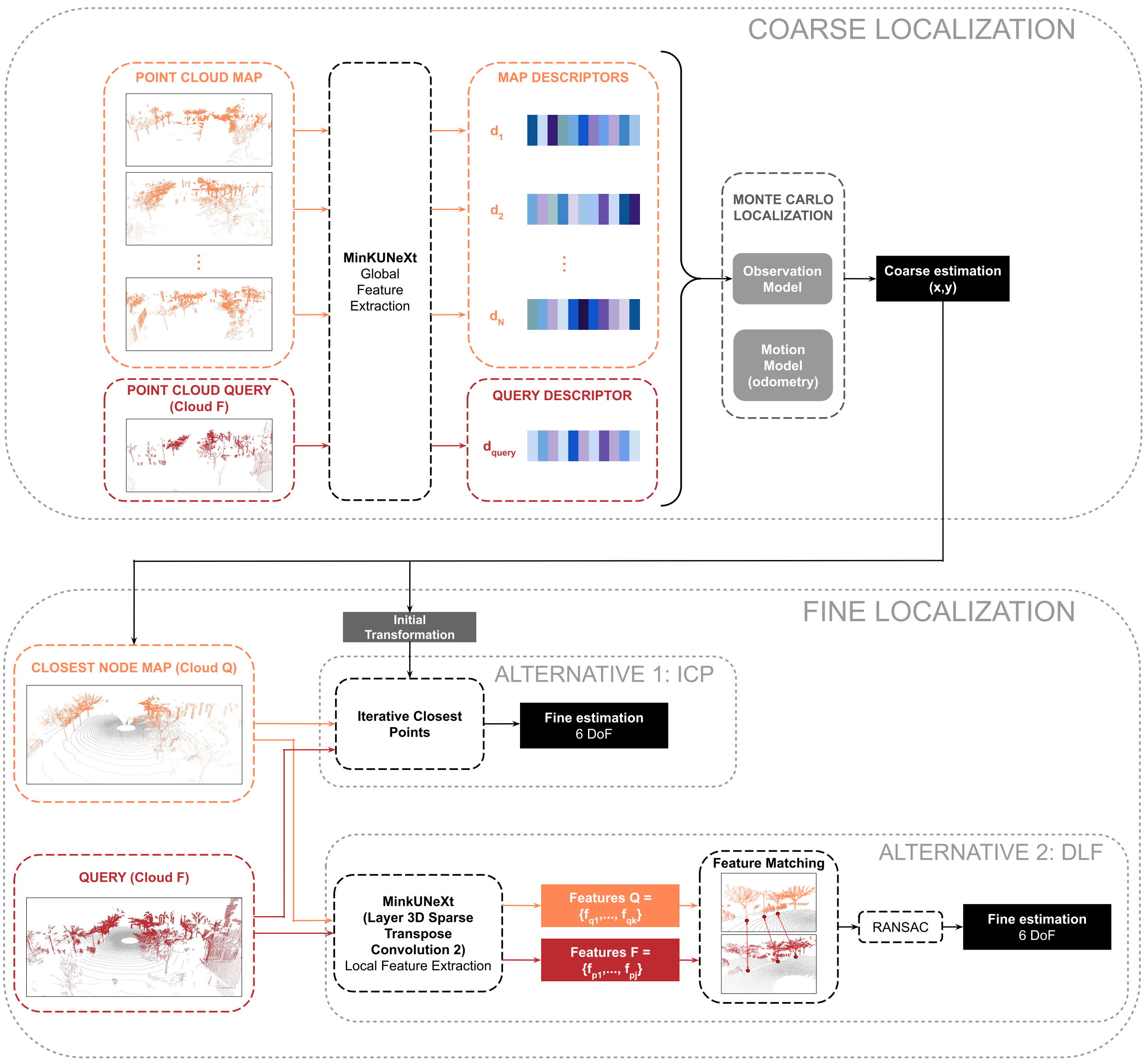


FIGURE 1 Schematic of the proposed localization method, consisting of coarse localization and fine localization. **Coarse localization process**: The inputs are the point clouds of the map and the query point cloud, which is derived from the current observation of the robot. From these points clouds the global descriptors are extracted with MinkUNeXt. These descriptors are used as an observation model in the MCL process. Finally, the output of this process is the estimated (x,y) position of the robot. **Fine localization process**: Two alternative methods are presented, both requiring as input the query point cloud and the closest point cloud of the map to the position (x,y) obtained with MCL. The first alternative method, is ICP, which also requires an initial transformation to relate the two point clouds in position and orientation. These data are obtained from the coarse localization process. The second alternative method, involves the use of an intermediate layer (3D Sparse Transpose Convolution 2) of the MinkUNeXt network to obtain the features of each point of the scans F and Q, which are used to do feature matching between points in both scans. The outputs of these two methods are the 6DoF estimated position of the robot.

3.1.2 | Motion Model

In the prediction phase of the MCL method, the particles are propagated with the odometry data. The relative odometry between the previous and the current pose is thus considered. With these relative odometry defined as $u_t = (\Delta x_t, \Delta y_t, \Delta\theta_t)$. The distance travelled by the robot on the basis of this odometry will be $d_t = \sqrt{\Delta x_t \check{s} + \Delta y_t \check{s}}$. Therefore, the new state of the particle is obtained according to Equations 2, 3 y 4.

$$x_t = x_0 + d_t * cos(\theta_0 + \Delta\theta_t) \tag{2}$$

$$y_t = y_0 + d_t * sin(\theta_0 + \Delta\theta_t) \tag{3}$$

$$\theta_t = \theta_0 + \Delta\theta_t \tag{4}$$

Once these data have been obtained and following the perspective of the topological method. Each particle will be approximated in x and y with the position of the nearest node.

3.1.3 | Observation model

The MCL method evaluates the probability of each generated particle in the set being at the current position of the robot. Therefore, this measure reflects the proximity of each particle to the actual pose of the robot. To conduct this evaluation, MCL employs an observation model. This observation model assigns a weight to each particle, based on a comparison of the features of the environment observed by the robot and those of the map. In this case, the study of the features of the environment is carried out on the basis of the point clouds obtained with a LiDAR sensor.

The point clouds have a great amount of information. Consequently, the process of comparing scans to assess their similarity would necessitate a significant computational cost. Therefore, we have opted to utilize deep learning techniques that facilitate the embedding of the point clouds, thereby ensuring the preservation of the most salient information and significantly reducing the time required for computational comparisons. In this paper, the MinkUNeXt [26] architecture is employed, a neural network capable of efficiently encoding extensive three-dimensional geometric environments.

The main objective of the MinkUNeXt architecture is to solve the problem of place recognition using point clouds, for which this neural network uses 3D sparse convolutions. Feature extraction is performed at different scales according to a U-net architecture with an encoder-decoder topology. In the encoder section, a progressive reduction of resolution is done to obtain from finer to more general features. In the decoder section, a progressive upsampling is implemented, reconstructing the point cloud until the original resolution. Also, in the decoder, skip connections are performed with the encoder to merge general and fine features. After the decoder, there is a fully connected layer to provide the descriptors of each point in the cloud with invariance to viewpoint changes. Finally, the feature map passes through a Generalized Mean Pooling layer, where it is embedded in a single descriptor of length 512. In this way, thanks to the MinkUNeXt architecture, it is possible to describe the relevant information of a point cloud in a single vector. The layers of this network employ 3D sparse convolutions, enabling the efficient capture of relevant information from the point cloud. This network is implemented using the Minkowski Engine library. [40]

This architecture is employed to calculate the map descriptors, denoted as $D_{map} = \{d_1, .., d_N\}$ and the robot descriptor d_{query}. d_{query} is derived from the point cloud captured by the robot at time t. Conversely, each descriptor in the set D_{map} is generated from a point cloud of the map. The map consists of N two-dimensional nodes $\{(n_{1,x}, n_{1,y}), ..., (n_{N,x}, n_{N,y})\}$, each containing a point cloud that has been captured at the position where the node is located.

During the robot's localization process, the current robot descriptor d_{query}, is compared against all the map descriptors in D_{map} to assess the similarity between the robot's current perception and the map's representations. Subsequently, the B map descriptors that exhibit the highest similarity to d_{query}, as determined by the shortest Euclidean distance in the descriptor space, are selected.

The weight of each particle is determined by two distances: the metric distance, v_j, and the distance in the descriptor space h_j.

The metric distance is calculated as $v_j = (n_{k,x}, n_{k,y}) - (x_i, y_i)$, where $(n_{k,x}, n_{k,y})$ is the position of one of the B nodes of the map and (x_i, y_i) is the position of the particle i.

Alternatively, the distance of the descriptors can be expressed as $h_j = |d_k - d_i|$, where d_k is the descriptor of one of the B nodes of the map and d_i is the descriptor of the closest node of the particle in the map.

The matrices $\Sigma_l = diag(\sigma_l^2, \sigma_l^2)$ and $\Sigma_m = 1/\sigma_m$ are also employed to model the adjustment of the weights. The total weight of each particle is calculated in Equation 5 and the described process can be observed in Figure 2.

$$w_t^i = \sum_{j=1}^{B} exp(-v_j \Sigma_l^{-1} v_j^T) exp(-h_j \Sigma_m^{-1} h_j^T) \quad (5)$$

Thus, the observation process allows to weight each particle in the set and perform successive iterations as the robot moves. The convergence process of the particles during MCL iterations is illustrated in Figure 3.

3.2 | Fine localization

Once the global localization has been carried out using the MCL method with the proposed observation model, we proceed to estimate the pose more precisely.

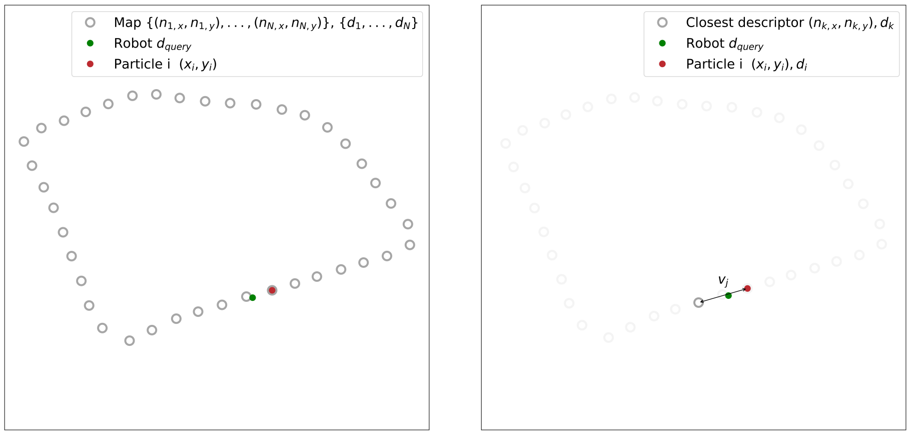
The initial location obtained through the coarse method will be utilized for the subsequent procedure. The point cloud of the map that is closest to this initial location will be employed as the reference point for the subsequent analysis. With this point cloud and the one captured by the robot at the current time, pairwise registration will be performed. In this paper, two alternative methods have been evaluated: ICP and deep local features from the intermediate layers of the MinkUNeXt network.

3.2.1 | Iterative Closest Point (ICP)

In this case, the coarse localization phase is first addressed through the MCL process described in the preceding section, and subsequently, the ICP method is employed for fine localization; this combined global method is termed MCL-ICP (Monte Carlo Localization - Iterative Closest Point).

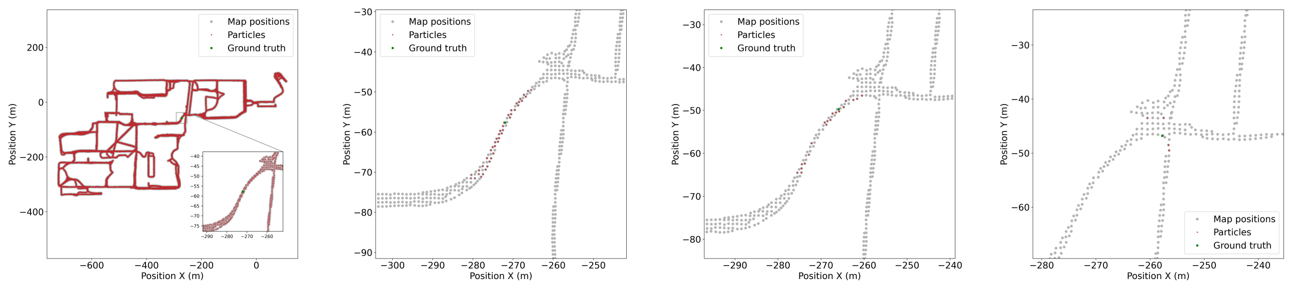
ICP is a classic registration algorithm. To start with, it needs a homogeneous transformation matrix T that roughly relates the two point clouds: target (Q) and source (F). This method allows the matrix to be refined to improve the alignment between these two point clouds.

The ICP method consists in finding correspondences $K = \{(p, q)\}$, where p and q are points of the point clouds P and Q respectively.


Map $\{(n_{1,x}, n_{1,y}), \ldots, (n_{N,x}, n_{N,y})\}$, $\{d_1, \ldots, d_N\}$
Robot d_{query}
Particle i (x_i, y_i)
Closest descriptor $(n_{k,x}, n_{k,y}), d_k$
Robot d_{query}
Particle i $(x_i, y_i), d_i$
v_j

(a) Initial conditions (b) Weight process

FIGURE 2 Weighting process of particle i. **(a)** the map nodes, located at positions $\{(n_{1,x}, n_{1,y}), ..., (n_{N,x}, n_{N,y})\}$, are visualized as grey circles. Particle i, at coordinates (x_i, y_i), is shown in red, and the robot, for which we only know the descriptor d_{query} from the LiDAR observation, is shown in green. This descriptor is used to compare with all map descriptors $\{d_1, .., d_N\}$. **(b)** the map node with the closest descriptor is shown as a grey circle. Then, v_j is calculated as the difference between the node's position $(n_{j,x}, n_{j,y})$ and the particle's position (x_i, y_i). Finally, h_j is calculated as the difference between d_k and d_i, where d_i is the descriptor of the map node where the particle i is located.



(a) Iteration 0 (b) Iteration 1 (c) Iteration 10 (d) Iteration 20

FIGURE 3 Particle locations, in red, relative to the map positions, in grey, and the robot's current location, in green. **(a)** the particles are shown distributed across all map nodes. **(b)** and **(c)** the particles converge to a smaller area of the map. **(d)** the particles have converged to very close areas near the robot's actual location.

Based on these correspondences, we then update the transformation T by minimizing a cost function $E(T)$, which is specific to the chosen variant of the ICP algorithm and its implementation. In this case, Point-to-plane ICP has been used, where E is defined according to Equation 6 (where n_p is the normal of point p).

$$E(T) = \sum_{(p,q)\epsilon K} ((p - T_q)n_p)\check{s} \tag{6}$$

3.2.2 | Deep Local Features (MinkUNeXt)

In this method, coarse localization is also achieved using MCL, and subsequently, fine localization is performed utilizing deep local features; this complete method is termed MCL-DLF (Monte Carlo Localization - Deep Local Features).

This fine localization is achieved by extracting the information from the intermediate layers of the MinkUNeXt network. The study of the output provided by a sparse encoder-decoder network, such as MinkUNeXt, for representing 3D geometric spaces has been conducted, achieving high accuracy through the use of specific intermediate layers, without the necessity of training for this specific task.

The information from these intermediate layers shall consist of descriptors for each point in the point cloud. The point cloud's size and the length of its descriptors rely on the output of the layer being used.

Once we have obtained the descriptors of the two point clouds, we will proceed to calculate the correspondences between the points of both point clouds. Considering the feature points of two scans, F and Q, represented by the sets $F_F = \{f_{p1}, f_{p2}, ..., f_{pj}\}$ (where j denotes the total number of points in F) and $F_Q = \{f_{q1}, f_{q2}, ..., f_{qk}\}$ (where k denotes the total number of points in Q), we generate correspondences $K = \{(p, q)\}$ based on their Euclidean distances. These correspondences, along with the original point clouds, are then used by the RANSAC [41] method to estimate the transformation between the scans.

4 | EXPERIMENTS

4.1 | Datasets

In order to evaluate the proposed method, data from the University of Michigan North Campus Long-Term Vision and LiDAR Dataset (NCLT) [42] dataset have been utilized. Furthermore, experiments have been carried using own data captured at the Miguel Hernández University of Elche. These datasets have been selected due to they were captured in challenging environments. They encompass large-scale settings exhibiting significant variability, particularly seasonal changes. Furthermore, they include handover scenarios involving transitions between outdoor and indoor environments.

4.1.1 | NCLT

The NCLT dataset [42] consists of data from different sensors integrated in the Segway robotic platform. These sensors include: omnidirectional imagery, 3D LiDAR, planar LiDAR, GPS, proprioceptive sensors and odometry. Specifically, the 3D LiDAR sensor is the Velodyne HDL-32E LiDAR.

The odometry is estimated with an Extended Kalman Filter (EKF) that fuses data from the robot's wheel encoder, a single-axis Fibre Optic Gyro (FOG) and an Inertial Measurement Unit (IMU). In addition, this dataset also provides ground-truth pose data for all sessions generated via SLAM.

The data have been captured in indoor and outdoor areas of the University of Michigan's North Campus. It contains 27 mapping sessions, captured on different days and at different times over a period of 15 months.

This variety of routes has enabled the acquisition of scenarios with major changes, such as alterations in lighting, variations in tree foliage, and the presence or absence of snow on the streets. Additionally, there are other elements that are subject to change, including obstacles such as people or bicycles.

4.1.2 | University Miguel Hernandez of Elche (UMH)

The second dataset consists of information related to routes recorded on the Miguel Hernandez University of Elche campus [‡]. These routes have been acquired with a Husky A200 robot. The main purpose of recording these routes is to test navigation, mapping and localization algorithms for mobile robots. The true positions of the routes have been obtained, these will be utilized to calculate the error when compared to the poses obtained through our localization method. The true positions have been obtained through the use of a SLAM process. The acquisition of point clouds has been enabled by the integration of the Ouster OS1-128 LiDAR within the robotic platform.

The routes in this environment comprise both indoor and outdoor areas as illustrated in Figure 4. The outdoor areas refer to the green spaces and structures present within the campus environment. Meanwhile, the indoor areas correspond to the indoor spaces within university buildings.

These data were collected over several months in different seasons, allowing the observation of changes in the environment, including variations in the foliage of the trees. Additionally, other dynamic changes, such as the presence of people, bicycles or cars can be discerned. The point clouds of one captured session are illustrated in Figure 5.

4.2 | Implementation Details

In MinkUNeXt [26], the training process was performed using the Oxford RobotCar dataset and a dataset proposed in [28]. The data from these databases, used for training in [26], contain scans with only 4096 points, unlike the NCLT and UMH databases, which have higher point count. To generalize to these types of scans, the weights obtained from [26] were used to perform transfer learning with the NCLT and UMH databases. The trajectories that have been selected for training the network are shown in Tables 1 and 2.

The clouds obtained with the LiDAR sensors have been processed to make the inferences for description extraction. During this preprocessing stage, points located at a distance exceeding 50 meters from the LiDAR origin are removed. The justification for this is that beyond that range, the point density becomes insufficient due to the LiDAR's operational limitations. After this, a normalization step is performed to center the point cloud at the LiDAR sensor's origin. Subsequently, the points are spatially scaled using an optimal scale factor of 50. This scaling ensures that all points are mapped to the range [-1, 1], given that no points are located beyond 50 meters from the origin. In addition, a downsampling process is implemented, in which the points within the point cloud are significantly reduced. In the next step, the points belonging to the ground plane are removed, as they do not provide valuable information.

A similar procedure is employed to process the point clouds to perform the point cloud registration method. The normalization step is not carried out in this case, because this would modify the dimensionality of the point cloud.

A reference map has been used in the experiments. In each case, the formation of the map has been determined by the same trajectories employed during the retraining of the network. Therefore, for each dataset, the map is given by the trajectories indicated in Tables 1 and 2. The formation of the map comprises the point clouds and the positions of each of them. Another fact is that the nodes of these maps are spaced every 1 meter, thus reducing the computational expense in the MCL method. Figure 6 illustrates these maps, showing the points that form the map from a satellite view.

[‡] Data is available at `https://arvc.umh.es/db/databases/`

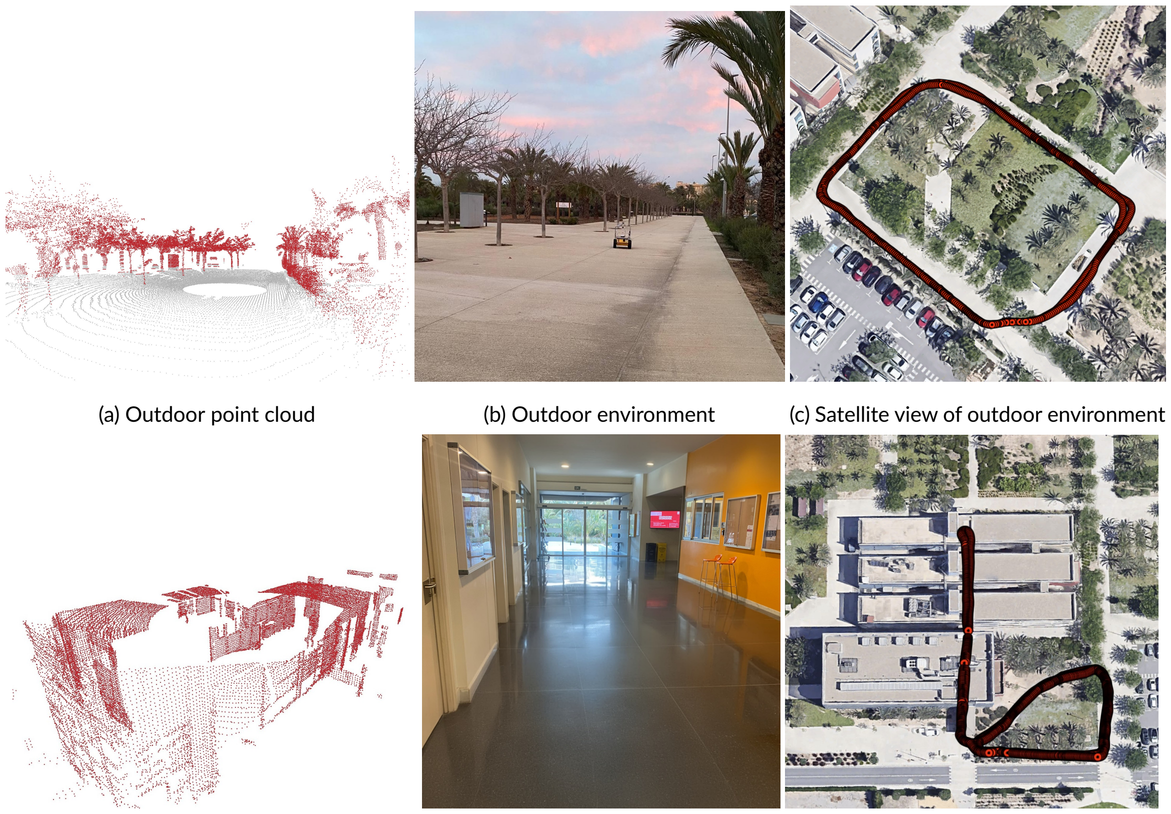


FIGURE 4 Appearance of the dataset environments captured at UMH. **(a) (b) (c)** outdoor environments and **(d) (e) (f)** indoor environments. **(c)** a trajectory captured entirely in an outdoor environment is shown. **(f)** a trajectory captured partly outdoors and partly indoors is presented.

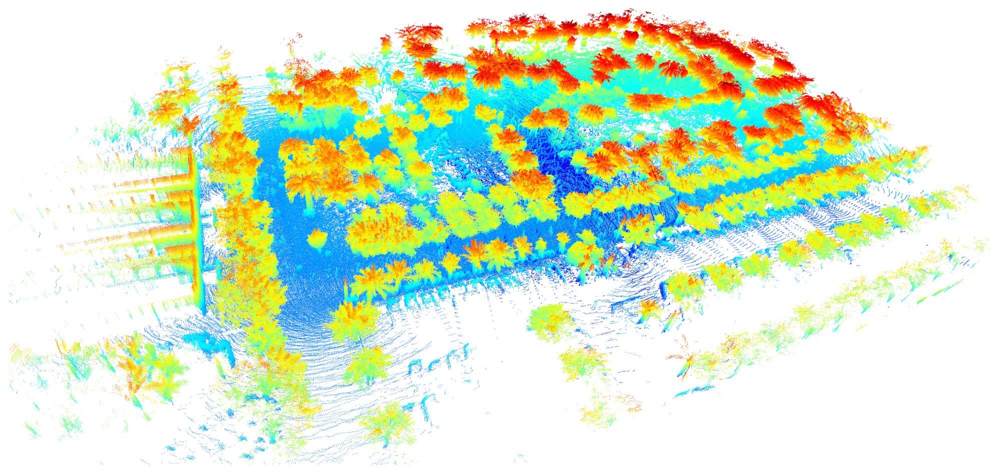
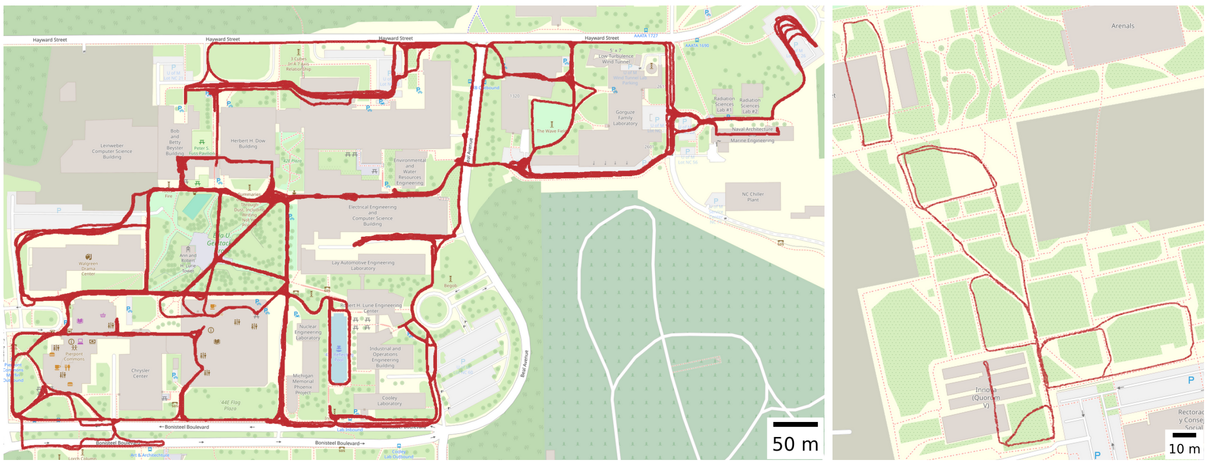


FIGURE 5 Aggregated point clouds of a robot trajectory in the campus of Miguel Hernandez University of Elche.

In order to evaluate the proposed method, experiments have been carried out with different routes. These differ from the routes that constitute the map. These routes are presented in detail in Tables 1 and 2.

The following section details the results of experiments conducted using the proposed method. In these experiments, the MCL method is employed so that the particle update is performed each time the robot has travelled one meter. The method is initialized after 20 iterations. The position is considered to be non-knowledgeable, so the particles are re-initialized at all map nodes.



(a) NCLT (b) UMH

FIGURE 6 Satellite view in OpenStreetMap of the data included in the maps of the **(a)** NCLT and **(b)** UMH datasets.

TABLE 1 Data sessions from University of Michigan North Campus Long-Term vision and LiDAR Dataset used for network training, mapping and experiments.

Date [YYYY-MM-DD]	Length [km]	Usage
2012-01-08	6.4	Train and map
2012-01-15	7.5	Train and map
2012-01-22	6.1	Train and map
2012-02-02	6.2	Train and map
2012-02-04	5.5	Train and map
2012-03-31	6.0	Train and map
2012-02-18	6.2	Experiment February
2012-04-29	3.1	Experiment April
2012-05-11	6.0	Experiment May
2012-06-15	4.1	Experiment June
2012-08-04	5.5	Experiment August
2012-10-28	5.6	Experiment October
2012-11-04	4.8	Experiment November
2012-12-01	5.0	Experiment December

TABLE 2 Data Sessions from University of Miguel Hernandez University data used for network training, mapping and experiments.

Date [YYYY-MM-DD]	Length [m]	Usage
2024-04-24	481.40	Train and map
2024-05-03	472.30	Train and map
2024-05-07	299.70	Train and map
2024-05-14	473.00	Train and map
2024-04-24	346.40	Experiment April
2024-06-20	303.68	Experiment June
2025-01-21	171.19	Experiment January

Furthermore, regarding fine localization estimate, ICP and registration with deep local features is tested in these experiments. On the other hand, deep local features method does not require a prior initial transformation, whereas ICP does. In order to form the transformation, it is necessary to have x, y, z position, as well as $roll$, $pitch$ and yaw orientation. We assume that the robot is above and parallel to the ground, so we consider z, $roll$ and $pitch$ as zero. The values of x and y will be determined by the results of MCL in that particular iteration. The yaw angle is calculated using trigonometric methods, with the MCL positions estimated in the current and previous iterations.

4.3 | Results and Discussion

In order to evaluate the proposed method, a series of experiments have been carried out using the datasets presented in Section 4.1. Firstly, the performance of the different intermediate layers of the MinkUNeXt network has been evaluated for fine localization. A comparison has also been made between the results obtained with the present method and those obtained in other works of the state-of-the-art in large-scale environments. In addition, results have been included differentiating between indoor and outdoor environments. Finally, to verify the applicability of this method to other datasets, the effectiveness was studied with our own data.

4.3.1 | Evaluation of fine localization with MinkUNeXt layers outputs

A study has been carried out to analyze the fine localization result with the descriptors obtained with all the layers of the MinkUNeXt network. The output of each layer is a point cloud, wherein each point possesses an associated descriptor. It is important to note that this output point cloud does not maintain the same dimensionality as the input point cloud. The reason for this is that MinkUNeXt is a network that possesses

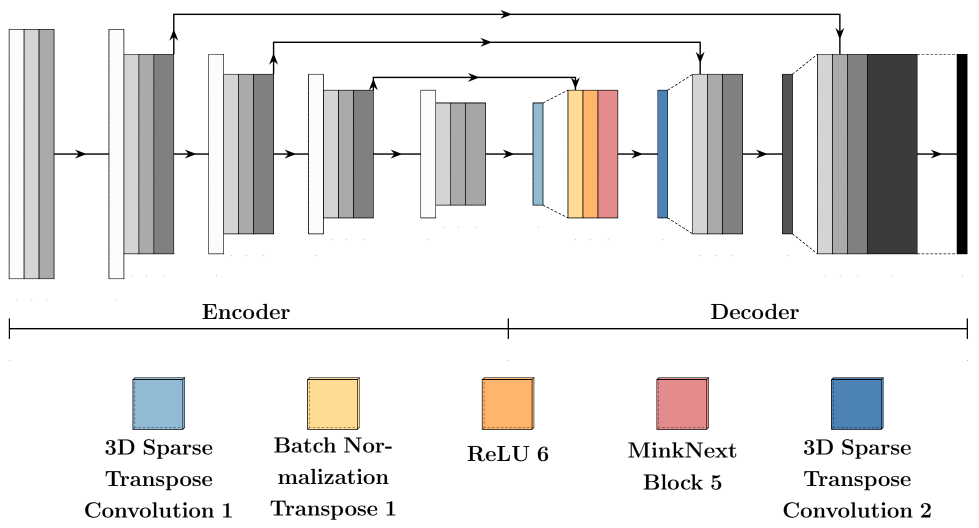

Encoder
Decoder
3D Sparse Transpose Convolution 1
Batch Normalization Transpose 1
ReLU 6
MinkNext Block 5
3D Sparse Transpose Convolution 2

FIGURE 7 MinkUNeXt network architecture. The layers whose output has demonstrated superior performance for the purpose of fine localization are highlighted in colours. The results of these layers are shown in Table 3

.

TABLE 3 Mean and median error of the position and orientation obtained with different layers from the MinkUNeXt network in the NCLT dataset. The best results are shown in bold.

		Position		Orientation	
LAYER	Length descriptor	Mean error [m]	Median error [m]	Mean error [deg]	Median error [deg]
3D Sparse Transpose Convolution 1	192	1.72	0.97	5.07	**1.83**
Batch Normalization Transpose 1	192	1.79	**0.69**	4.44	2.26
ReLU 6	192	1.71	0.96	9.42	3.20
MinkNext 5	192	3.65	1.02	11.34	3.23
3D Sparse Transpose Convolution 2	192	**1.37**	0.86	**3.59**	2.56

a U-Net architecture, wherein the point cloud undergoes a reduction in dimensionality within the layers belonging to the encoder, followed by a subsequent increase in dimensionality within the layers of the decoder.

The results obtained for all the layer outputs were analyzed, with the best results found for the layers located at the beginning of the decoder section of the network. The layers that provide better results are shown in colours in Figure 7. These layers are located immediately after the encoder, at the beginning of the decoder part of the network.

The results obtained in these layers are shown in Table 3. This table shows that the average error, both in position and orientation, is lower in the 3D Sparse Transpose Convolutional 2. Consequently, this layer will be selected in the method developed for fine localization.

4.3.2 | Evaluation in large scale environments

In this section, the performance evaluation of the global localization method is carried out. This evaluation is conducted by comparing the results obtained with MCL-DLF, which employs deep learning, against state-of-the-art outcomes and, concurrently, with MCL-ICP, a classical method based on ICP.

The dataset selected for this evaluation is the NCLT dataset. This dataset has been selected to demonstrate the robustness of the method in large-scale environments with many dynamic elements. This will allow us to test whether the method can be generalized to year-round localization using maps taken in the months of January to March.

In order to compare the results obtained with this dataset, a review of the state-of-the-art was carried out, and the Localising Faster method presented in [39] was selected to compare the error results in Tables 4 and 5. This method has been selected due to its relevance as a state-of-the-art approach for global localization, incorporating deep learning techniques and utilizing a challenging dataset with seasonal variations. This tables show the error values obtained in both position and orientation using both methods. With regard to the aforementioned tables, it should be noted that the state-of-the-art results for each month are obtained with 14,000 to 33,000 tests, whereas in our case they are obtained with 1,800 to 5,000.

TABLE 4 Mean and median error in meters of the position obtained with the global localization method in the NCLT dataset. Including fine localization results obtained with local features and with ICP. The results presented in the State-of-the-Art (SOTA) are also included. The best results are shown in bold.

	SOTA [39]		**MCL-DLF (ours)**		**MCL-ICP**	
MONTHS	median error [m]	mean error [m]	median error [m]	mean error [m]	median error [m]	mean error [m]
February	1.74	8.77	0.49	**0.85**	**0.19**	1.01
April	1.69	2.88	0.55	**0.73**	**0.22**	0.82
May	2.02	15.3	0.66	**3.93**	**0.26**	4.66
June	1.99	11.57	0.72	**4.69**	**0.29**	4.86
August	2.13	14.06	0.57	**0.75**	**0.23**	1.27
October	2.14	17.33	0.52	**0.77**	**0.22**	1.31
November	3.98	17.33	0.76	**2.23**	**0.28**	2.32
December	3.59	32.08	0.64	**2.32**	**0.25**	2.40
Overall	2.18	16.55	0.58	**1.82**	**0.23**	2.00

TABLE 5 Mean and median error in degrees of the orientation obtained with the global localization method in the NCLT dataset. Including fine localization results obtained with local features and with ICP. The results presented in the State-of-the-Art (SOTA) are also included. The best results are shown in bold.

	SOTA [39]		**MCL-DLF (ours)**		**MCL-ICP**	
MONTHS	median error [deg]	mean error [deg]	median error [deg]	mean error [deg]	median error [deg]	mean error [deg]
February	3.25	6.19	1.62	**2.42**	**1.23**	4.09
April	3.36	4.43	1.81	**2.77**	**1.46**	5.14
May	3.34	9.50	2.35	**4.74**	**1.61**	7.91
June	3.17	7.96	2.32	**4.78**	**1.58**	8.89
August	3.66	8.52	2.06	**2.92**	**1.47**	4.92
October	3.67	11.58	1.72	**2.80**	**1.29**	4.55
November	5.12	17.98	2.60	**5.14**	**1.66**	7.62
December	4.72	14.51	2.72	**4.13**	**1.43**	6.45
Overall	3.65	4.99	1.99	**3.39**	**1.42**	5.61

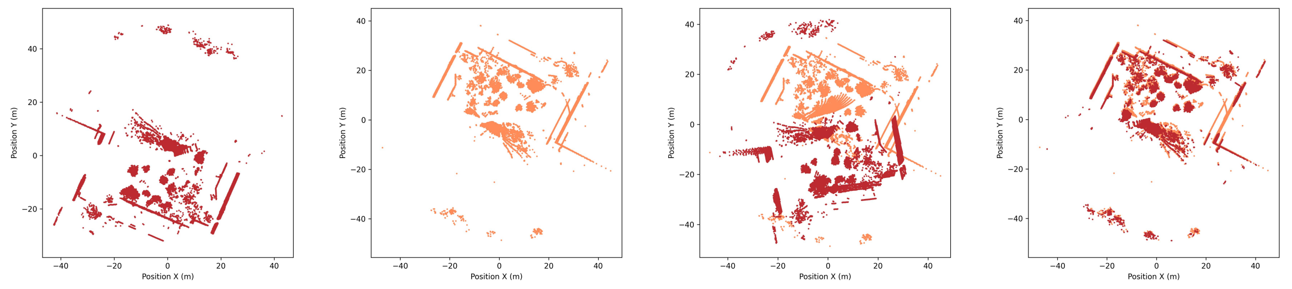
Table 4 shows the estimation position results. It can be observed that the position obtained with SOTA has higher values of mean error with respect to its median. This observation suggests the presence of considerable localization errors in certain areas. The use of MCL-DLF reduces these median and mean errors, thereby increasing precision. The use of MCL-ICP has resulted in an improvement to the median error, whereas the mean error has increased slightly compared to the deep local features method. This may be due to ICP being less robust and having more outliers than deep local features approach.

Table 5 shows orientation errors, where the SOTA presents higher median and mean error values than the other two proposed methods. Moreover, similar to the results obtained for the position estimate in Table 4, MCL-DLF provides lower mean errors than MCL-ICP. Therefore, the error values obtained using MCL-DLF and MCL-ICP show that there has been an improvement in both position and orientation accuracy compared to SOTA.

Furthermore, Tables 4 and 5 show that the SOTA method presents certain variability in the mean errors across different months, while MCL-LDF and MCL-ICP, maintains a relatively small range of variability in the mean error over the different months. This verifies that the proposed method is invariant to seasonal changes.

As mentioned above, the results obtained with MCL-ICP for both position and orientation are better than those obtained with MCL-DLF in terms of median error, but not in terms of mean error. This is because ICP is sensitive to the initial pose of the point clouds; if the initial misalignment is too large, ICP may fail to converge to the correct alignment or it might get stuck in a local minimum. This is why, despite obtaining a higher median accuracy, in many cases where the initial alignment is very inaccurate, it does not operate adequately.

On the other hand, the alignment based on deep local features is robust to these problems, as they do not require a prior initial transformation, and rely on the local features of the point clouds to perform the alignment. Figure 8 shows an example in which the query cloud captured by the robot and the nearest cloud on the map are initially completely misaligned by an angle of approximately 180 degrees. The ICP result demonstrates a significant difference from the correct alignment. By contrast, with the deep local features, a good point cloud registration result is achieved.



(a) Query point cloud (b) Closest point cloud map (c) ICP result (d) Deep Local Features result

FIGURE 8 Results obtained for the fine localization. Point cloud alignment achieved with the two localization methods evaluated, **(c)** ICP and **(d)** Deep Local Features.

The localization errors in orientation have been compared with other methods that also obtain the robot's orientation, and which have been tested with the NCLT dataset. This comparison has been made with the OREOS [31] , OverlapNet [33] and DiSCO [32] papers described in Section 2. Table 6 shows how MCL-DLF outperforms other methods in terms of mean orientation error. It should be noted that the results in Table 6 for the other methods do not present point cloud registration. Therefore, it is concluded that the use of fine localization after coarse localization improves the results. Performing values similar to those obtained with DiSCO.

TABLE 6 Mean and standard deviation errors in degrees of the orientation estimation in NCLT dataset. The presented results from OREOS, OverlapNet and DiSCO are extracted from [32] .

Approach in NCLT	**Mean [deg]**	**Std [deg]**
OREOS [31]	15.95	21.31
OverlapNet [33]	11.59	24.10
DiSCO [32]	2.81	4.01
MCL+DLF (ours)	3.39	5.40
MCL+ICP	5.61	12.69

4.3.3 | Evaluation in indoor and outdoor environments

To test the robustness of the method in indoor and outdoor environments, the results of one of the months were taken and separated into indoor and outdoor environments. Tables 7 and 8 show the indoor, outdoor and overall results for the February session.

It can be observed that the errors obtained in both cases are not very different from indoor to outdoor areas. This error variation is greater for MCL-ICP. However, for MCL-DLF, the error is very similar indoors and outdoors, both in position and orientation. Consequently, the MCL-DLF method achieves precise localization regardless of whether the robot is situated in outdoor or indoor environments, demonstrating robustness without dependence on a GPS sensor.

TABLE 7 Mean and median error in meters of the position obtained with the global localization method in the NCLT dataset, only in February. The results are divided into indoor and outdoor parts of the trajectory, and the results for the entire month are also included.

	MCL-DLF (ours)		**MCL-ICP**	
	median [m]	mean [m]	median [m]	mean [m]
February Indoor	0.46	0.48	0.14	0.15
February Outdoor	0.48	0.87	0.19	1.03
February Overall	0.48	0.86	0.19	1.74

TABLE 8 Mean and median error in degrees of the orientation obtained with the global localization method in the NCLT dataset. The results are divided in the indoor and outdoor parts of the trajectory.

	MCL-DLF (ours)		**MCL-ICP**	
	median [deg]	mean [deg]	median [deg]	mean [deg]
February Indoor	1.48	1.66	0.99	1.88
February Outdoor	1.63	2.43	1.24	4.14
February Overall	1.62	2.42	1.23	4.09

4.3.4 | Evaluation with in-house data

In addition, studies have been carried out with our own dataset, collected on the campus of the Miguel Hernandez University, to observe the results for other dataset and to be able to generalize the method presented in other environments.

The peculiarity of this dataset is that it contains many similar environments, as shown in Figure 5, so the performance is complex in this case. Thus, the deep local features of the map can be very similar in very distant places. Therefore, an initial coarse localization permits focusing on a single area of the map and hence avoiding mislocalization.

The results of this evaluation are shown in Tables 9 and 10 where similar accuracy is obtained with respect to the previous results in the NCLT dataset.

In the experiments carried out, it can be seen that the orientation errors obtained with MCL-ICP are much higher than the errors obtained with MCL-DLF. Again, this is because the first method is very sensitive to errors in the initial transformation provided, so if the initial orientation is very different from the correct orientation, the method will converge on an incorrect solution.

In contrast, MCL-DLF is robust to large changes in orientation, as shown in Figure 8, since MCL-DLF does not rely on an initial transformation to generate the registration, and therefore, is unaffected by a poor initial orientation.

TABLE 9 Mean and median error in meters of the position with the global localization method in the UMH dataset. Including fine localization results obtained with local features and with ICP. The best results are shown in bold.

MONTH	MCL-DLF (ours)		MCL-ICP	
	median [m]	mean [m]	median [m]	mean [m]
April	1.62	1.67	**1.40**	**1.62**
June	**1.49**	1.74	1.64	**1.70**
January	5.93	6.76	**5.80**	**6.41**

TABLE 10 Mean and median error in degrees of the orientation with the global localization method in the UMH dataset. Including fine localization results obtained with local features and with ICP. The best results are shown in bold.

MONTH	MCL-DLF (ours)		MCL-ICP	
	median [deg]	mean [deg]	median [deg]	mean [deg]
April	**5.76**	**9.37**	24.85	20.17
June	**10.00**	**12.81**	24.36	22.17
January	**8.34**	**11.98**	59.59	73.45

5 | CONCLUSIONS

A complete method for accurate global localization from coarse-to-fine has been presented. This approach, denoted as MCL-DLF, involves an initial localization step that roughly estimates the pose, followed by a more accurate refinement step.

The primary benefit of this strategy is improved accuracy. By first narrowing down the possible locations of the robot in the coarse stage, the subsequent fine localization step can focus its computational resources on a smaller region of the environment.

Deep learning techniques are employed for both stages of the process, enabling the extraction of features from point clouds. This information is then used to compare with features from a pre-existing map, allowing the localization of the robot.

The coarse localization is based on the use of MCL, employing the global point cloud descriptor information as an observation model. This descriptor is extracted from a point cloud using MinkUNeXt. This network employs 3D sparse convolutions, thereby enabling the efficient encoding of extensive three-dimensional geometric environments. The fine localization utilizes point-wise descriptors generated from the intermediate layers of the MinkUNeXt neural network. A comparison has been done between the results provided by the MinkUNeXt-based point cloud registration approach and the classical ICP algorithm. Furthermore, a relevant state-of-the-art method has been compared against the NCLT dataset, as well as other state-of-the-art proposals.

The evaluation of this method on the challenging NCLT dataset and an in-house UMH dataset, shows high accuracy in dynamic environments and under varying environmental conditions. It has been proven that by using a map with data from specific months, precise localization can be achieved in other months with different environmental appearances, without significant variations in the resulting error. Furthermore, the method ensures accurate localization, even in large environments. In addition, the localization method performs well in indoor and outdoor environments. In conclusion, MCL-DLF is a viable option delivering high accuracy in a variety of conditions, including indoor-outdoor scenarios, environmental changes, and the presence of dynamic entities.

AUTHOR CONTRIBUTIONS

Conceptualization, M.M., M.B. and D.V.; methodology, M.M.; software, M.M. and A.S.; validation, M.M., M.B., and D.V.; formal analysis and investigation, M.M.; writing—original draft preparation, M.M.; writing—review and editing, M.M., A.S., A.G., M.B. and D.V.; supervision, M.B. and D.V.; project administration, A.G, M.B. and D.V.; funding acquisition, A.G. All authors have read and agreed to the published version of the manuscript.

ACKNOWLEDGMENTS

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 PID2023-149575OB-I00 funded by MICIU/AEI/10.13039/501100011033 and by FEDER, UE.

CONFLICT OF INTEREST

The authors declare no conflicts of interest.

REFERENCES

[1] Ullah I, Adhikari D, Khan H, Anwar MS, Ahmad S, Bai X. Mobile robot localization: Current challenges and future prospective. Computer Science Review 2024;53:100651.

[2] Kramer J, Kandel A. On Accurate Localization and Uncertain Sensors. International Journal of Intelligent Systems 2012;27(5):429–456.

[3] Heredia-Aguado E, Cabrera JJ, Jiménez LM, Valiente D, Gil A. Static Early Fusion Techniques for Visible and Thermal Images to Enhance Convolutional Neural Network Detection: A Performance Analysis. Remote Sensing 2025;17(6):1060.

[4] Dalal N, Triggs B. Histograms of Oriented Gradients for Human Detection. In: 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), vol. 1 IEEE; 2005. p. 886–893.

[5] Rublee E, Rabaud V, Konolige K, Bradski G. ORB: An efficient alternative to SIFT or SURF. In: 2011 International Conference on Computer Vision IEEE; 2011. p. 2564–2571.

[6] DeTone D, Malisiewicz T, Rabinovich A. SuperPoint: Self-Supervised Interest Point Detection and Description. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops; 2018. p. 224–236.

[7] Yin H, Xu X, Lu S, Chen X, Xiong R, Shen S, et al. A Survey on Global LiDAR Localization: Challenges, Advances and Open Problems. International Journal of Computer Vision 2024;132(8):3139–3171.

[8] Wang X, Zhang X, Zu C, Yang Z, Bian G, Zhang Y, et al. An accurate cloud-based indoor localization system with low latency. International Journal of Intelligent Systems 2022;37(8):4794–4809.

[9] Santo A, Heredia E, Viegas C, Valiente D, Gil A. Ground Segmentation for LiDAR Point Clouds in Structured and Unstructured Environments Using a Hybrid Neural–Geometric Approach. Technologies 2025;13(4).

[10] 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, vol. 36; 2022. p. 551–560.

[11] Luo L, Zheng S, Li Y, Fan Y, Yu B, Cao SY, et al. BEVPlace: Learning LiDAR-based Place Recognition using Bird's Eye View Images. In: 2023 IEEE/CVF International Conference on Computer Vision (ICCV); 2023. p. 8666–8675.

[12] Zhang Y, Shi P, Li J. 3D LiDAR SLAM: A survey. The Photogrammetric Record 2024;39(186):457–517.

[13] Dellaert F, Fox D, Burgard W, Thrun S. Monte Carlo Localization for Mobile Robots. In: Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No. 99CH36288C), vol. 2 IEEE; 1999. p. 1322–1328.

[14] Fernández L, Payá L, Valiente D, Gil A, Reinoso O. Monte Carlo Localization using the Global Appearance of Omnidirectional Images. In: Proc. of the Int. Conf. on Informatics in Control, Automation and Robotics (ICINCO); 2012. p. 439–442.

[15] Xu S, Chou W, Dong H. A Robust Indoor Localization System Integrating Visual Localization Aided by CNN-Based Image Retrieval with Monte Carlo Localization. Sensors 2019;19(2):249.

[16] Zeng Q, Tao X, Yu H, Ji X, Chang T, Hu Y. An Indoor 2-D LiDAR SLAM and Localization Method Based on Artificial Landmark Assistance. IEEE Sensors Journal 2023;24(3):3681–3692.

[17] Carrasco P, Cuesta F, Caballero R, Pérez-Grau FJ, Viguria A. Monte-Carlo Localization for Aerial Robots using 3D LiDAR and UWB sensing. In: 2021 International Conference on Unmanned Aircraft Systems (ICUAS); 2021. p. 354–360.

[18] Lowry S, Sünderhauf N, Newman P, Leonard JJ, Cox D, Corke P, et al. Visual Place Recognition: A Survey. IEEE Transactions on Robotics 2016;32(1):1–19.

[19] Besl PJ, McKay ND. Method for Registration of 3-D Shapes. In: Sensor fusion IV: control paradigms and data structures, vol. 1611 Spie; 1992. p. 586–606.

[20] Wu Y, Ding H, Gong M, Qin AK, Ma W, Miao Q, et al. Evolutionary Multiform Optimization With Two-Stage Bidirectional Knowledge Transfer Strategy for Point Cloud Registration. IEEE Transactions on Evolutionary Computation 2024;28(1):62–76.

[21] Rusu RB, Blodow N, Beetz M. Fast Point Feature Histograms (FPFH) for 3D registration. In: 2009 IEEE International Conference on Robotics and Automation IEEE; 2009. p. 3212–3217.

[22] Salti S, Tombari F, Di Stefano L. SHOT: Unique signatures of histograms for surface and texture description. Computer Vision and Image Understanding 2014;125:251–264.

[23] Wang Y, Solomon JM. Deep Closest Point: Learning Representations for Point Cloud Registration. In: Proceedings of the IEEE/CVF International Conference on Computer Vision; 2019. p. 3523–3532.

[24] Yew ZJ, Lee GH. RPM-NET: Robust Point Matching using Learned Features. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition; 2020. p. 11824–11833.

[25] Komorowski J, Wysoczanska M, Trzcinski T. EgoNN: Egocentric Neural Network for Point Cloud Based 6DoF Relocalization at the City Scale. IEEE Robotics and Automation Letters 2022;7(2):722–729.

[26] Cabrera JJ, Santo A, Gil A, Viegas C, Payá L. MinkUNeXt: Point Cloud-based Large-scale Place Recognition using 3D Sparse Convolutions. arXiv preprint arXiv:240307593 2024;.

[27] 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. 234–241.

[28] Uy MA, Lee GH. PointNetVLAD: Deep Point Cloud Based Retrieval for Large-Scale Place Recognition. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR); 2018. .

[29] 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. 652–660.

[30] Arandjelovic R, Gronat P, Torii A, Pajdla T, Sivic J. NetVLAD: CNN Architecture for Weakly Supervised Place Recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition; 2016. p. 5297–5307.

[31] Schaupp L, Bürki M, Dubé R, Siegwart R, Cadena C. OREOS: Oriented Recognition of 3D Point Clouds in Outdoor Scenarios. In: 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) IEEE; 2019. p. 3255–3261.

[32] Xu X, Yin H, Chen Z, Li Y, Wang Y, Xiong R. DiSCO: Differentiable Scan Context With Orientation. IEEE Robotics and Automation Letters 2021;6(2):2791–2798.

[33] 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. Autonomous Robots 2022;p. 1–21.

[34] Maddern W, Pascoe G, Linegar C, Newman P. 1 year, 1000 km: The Oxford RobotCar dataset. The International Journal of Robotics Research 2017;36(1):3–15.

[35] Zhou QY, Park J, Koltun V. Fast Global Registration. In: Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II 14 Springer; 2016. p. 766–782.

[36] Zeng A, Song S, Nießner M, Fisher M, Xiao J, Funkhouser T. 3DMatch: Learning Local Geometric Descriptors from RGB-D Reconstructions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition; 2017. p. 1802–1811.

[37] Choy C, Dong W, Koltun V. Deep Global Registration. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition; 2020. p. 2514–2523.

[38] Ma N, Wang M, Han Y, Liu YJ. FF-LOGO: Cross-Modality Point Cloud Registration with Feature Filtering and Local to Global Optimization. In: 2024 IEEE International Conference on Robotics and Automation (ICRA) IEEE; 2024. p. 744–750.

[39] Sun L, Adolfsson D, Magnusson M, Andreasson H, Posner I, Duckett T. Localising Faster: Efficient and precise lidar-based robot localisation in large-scale environments. In: 2020 IEEE International Conference on Robotics and Automation (ICRA) IEEE; 2020. p. 4386–4392.

[40] Choy C, Gwak J, Savarese S. 4D Spatio-Temporal ConvNets: Minkowski Convolutional Neural Networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition; 2019. p. 3075–3084.

[41] Fischler MA, Bolles RC. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. Communications of the ACM 1981;24(6):381–395.

[42] Carlevaris-Bianco N, Ushani AK, Eustice RM. University of Michigan North Campus Long-Term Vision and Lidar Dataset. The International Journal of Robotics Research 2016;35(9):1023–1035.