Neural Computing and Applications
https://doi.org/10.1007/s00521-023-08931-0

S.I.: MACHINE LEARNING/COMPUTATIONAL INTELLIGENCE IN UNMANNED MARINE SYSTEMS

PL-Net: towards deep learning-based localization for underwater terrain

Xin Peng[1] · Yang Zhang[2] · Zhen Xu[3] · Zhiguo Zhang[3] · Lijun Chen[3] · Cong Li[3]

Received: 18 October 2022 / Accepted: 24 July 2023

Abstract
Underwater terrain matching localization is a method for achieving precise localization by using the features of submarine topography to perform relevant matching operations. Existing underwater terrain matching algorithms suffer from issues such as limited application range, poor real-time performance, and low localization accuracy because of the variable carrier motion state, insignificant undulatory features of underwater terrain, and low underwater terrain data measurement stability. To solve the above problems, we present PL-Net, a novel learning-based underwater terrain matching localization method. First, we extract the terrain's keypoints to avoid the complexity of large-scale computation and improve the algorithm's real-time performance. Then, the pretrained deep neural network automatically extracts the high-level terrain features, avoiding the manual setting of feature extraction patterns, reducing the probability of localization failure caused by environmental and equipment factors, and improving the algorithm's robustness. Finally, we improve the localization accuracy by adding a self-attention mechanism that enables the network to adjust the weights of each component and pay more attention to the regions with significant features. We conducted a series of tests to validate the proposed method. The results indicate that our method significantly outperforms other methods, indicating that our method can achieve accurate and real-time localization in a wide range of underwater environments.

Keywords Underwater localization · Terrain matching · 3D point cloud · Deep learning

✉ Yang Zhang
zhangyang13d@nudt.edu.cn

Xin Peng
penxin@nudt.edu.cn

Zhen Xu
xuzhen@bupt.edu.cn

Zhiguo Zhang
zhangzhiguo@bupt.edu.cn

Lijun Chen
chenlijun@bupt.edu.cn

Cong Li
CongLi@bupt.edu.cn

[1] NUDT, Academy for Advanced Interdisciplinary Studies, Dongfeng Road, Changsha 410000, Hunan, China

[2] NUDT, College of Computer Science and Technology, Dongfeng Road, Changsha 410000, Hunan, China

[3] BUPT, School of Electronic Engineering, North Taipingzhuang, Beijing 100089, China

1 Introduction

Underwater terrain matching localization technology follows the carrier's movement path and measures terrain data to form real-time terrain elevation data, compares it with the known digital terrain reference data, and integrates the heading and speed information provided by other navigation equipment, such as an inertial navigation system (INS), to determine the position of the underwater submersible. Multibeam sonar bathymetry systems with high accuracy and complete coverage provide a fundamental guarantee for underwater terrain matching localization implementation, and the obtained large amount of high-density terrain data is of great significance for improving the accuracy of underwater terrain matching. With the development of underwater terrain matching localization technology, we can effectively solve the problems of navigation error accumulation and time and energy consumption caused by the absence of navigation and localization signals at large depths and in long-range missions. Therefore, this technology is being applied to various fields, including marine surveys and scientific research.

The conventional underwater terrain matching method uses an autonomous underwater vehicle (AUV) that enters the adaptation area for real-time measurement of underwater terrain by INS instructions. After obtaining a predetermined amount of data, it starts to process the terrain data into underwater images and perform underwater terrain matching with a presaved reference terrain image. The localization results are used to correct the accumulated INS error. The current terrain matching localization method is frequently used in underwater applications. However, there are still some deficiencies: (1) underwater terrain is undulatory and variable, and the characteristics of high similarity and low differentiation of terrain in different underwater areas contribute to the fact that the current underwater terrain matching accuracy is low; (2) the process of underwater measurement of real-time elevation data requires the carrier to maintain uniform linear motion in the horizontal plane, and the amount of data to be collected for a single matching is large, resulting in a poor real-time localization method; and (3) underwater terrain matching is a nonlinear problem, and it is difficult to effectively estimate the statistical parameters of random disturbances, resulting in a lack of stable localization accuracy and a relatively limited range of application. These issues restrict the advancement of underwater terrain matching localization technology and have long plagued its practical application. Thus, the above issues have become major challenges that must be resolved.

In recent years, three-dimensional (3D) data have been used in various fields, such as autonomous driving, robotics, remote sensing, and medicine. Compared with the traditional underwater terrain matching model, shown in Fig. 1, the 3D point cloud as a standard format retains the original geometric information in space without discretization and can restore any complex scene with high accuracy and convenience. Meanwhile, underwater terrain data based on a multibeam sonar system have a high resolution. Each point includes horizontal geographic location and depth information, forming elevation data represented by 3D coordinates, and the shape and processing are more similar to those of 3D point clouds. With the rapid development of localization technology in autonomous driving, mapping, and remote sensing, only a few methods have used deep neural networks in practical applications. In contrast, underwater terrain matching localization technology development is relatively slow, and the predominant methods are still focused on manual feature design and extraction. Therefore, we use deep learning to solve the underwater positioning problem by combining the features of underwater terrain data with the algorithmic process of 3D point clouds, as shown in Fig. 2.

In this paper, we present for the first time a deep neural network framework to perform matching localization of underwater terrain. First, we detect terrain keypoints with a fixed number and significant features from underwater terrain data. Then, we automatically extract the high-level terrain features by using a pretrained deep neural network. Finally, we combine a self-attention mechanism to adaptively adjust the weights and salience of different terrain regions to calculate the final position based on the output's relative offset.

To summarize, our main contributions are as follows:

- We present for the first time a learning-based underwater terrain matching localization method that is adaptable to various underwater environments and performs accurate real-time localization.
- We use terrain keypoint extraction and matching instead of wide-range terrain point-by-point matching localization, which avoids additional complexity and redundancy and improves the algorithm's real-time performance. Meanwhile, the pretrained deep neural network automatically extracts the high-level terrain features, avoids manual setting of features, and improves the algorithm's robustness.
- We address the limitations of computational resources and capacity by adding a self-attention mechanism, which enables the network to adaptively alter the weights of each component to improve the localization accuracy by focusing more on the salient regions of the features.

2 Related work

2.1 Terrain matching location methods

In recent years, the underwater terrain matching method has become an increasingly popular subject of research with the advancement of marine mapping technologies. Initial underwater terrain matching methods only used terrain matching to achieve cumulative error correction, primarily using a one-dimensional form of bathymetry data. The matching algorithm's principle was derived from the localization method used by aircraft. Furthermore, its point-linear data collection method and low data density can only meet the needs of submersible localization within 100 ms, and its real-time performance is not high. Currently, researchers have conducted several studies on terrain matching algorithms [1–3]; terrain correlation matching, the extended Kalman filter, and the direct probability criterion are several fundamental underwater

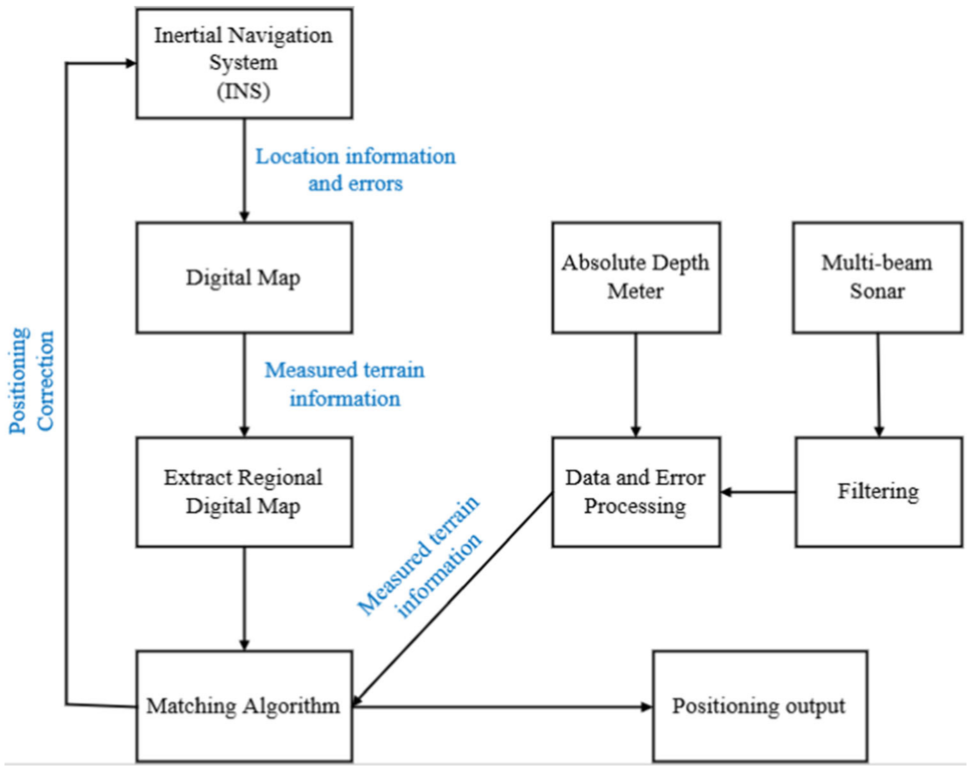

Inertial Navigation System (INS)
Location information and errors
Digital Map
Measured terrain information
Extract Regional Digital Map
Matching Algorithm
Positioning Correction
Absolute Depth Meter
Multi-beam Sonar
Data and Error Processing
Filtering
Measured terrain information
Positioning output

Fig. 1 Architecture of conventional underwater terrain matching

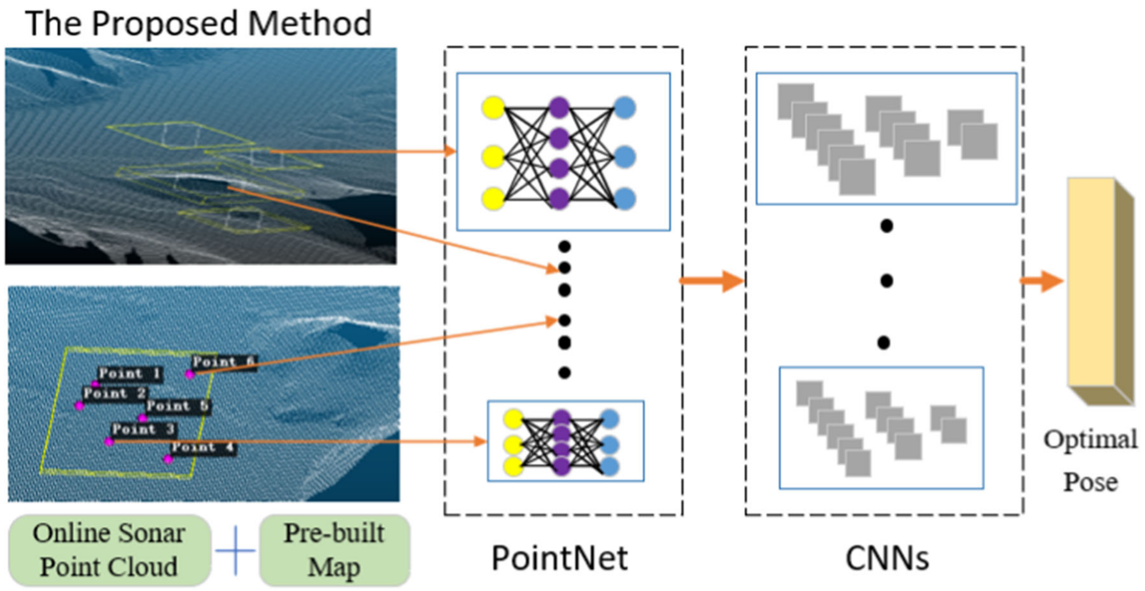


Fig. 2 Architecture of the traditional and proposed deep learning-based methods. In our method, PL-Net takes multibeam sonar scans, a prebuilt map, and a predicted pose as inputs, learns features by PointNet, constructs an offset geometry over the space, and applies CNNs to estimate the optimal pose

terrain matching methods that are widely studied and applied [4]. Among the methods based on the extended Kalman filter, Sandia Inertial Terrain-Aided Navigation (SITAN) introduced a recursive approach to terrain localization by multiple successive matching [5] and was previously applied to underwater terrain matching localization. Li et al. [6] proposed an intelligent Kalman filtering algorithm with higher accuracy to improve the terrain adaptability of navigation algorithms by introducing artificial neural networks. Anonsen et al. [8] provided an improved state space model for underwater terrain navigation based on Bayesian estimation [7]; Ingemar et al. proposed an AUV underwater terrain matching method based on Bayesian estimation and correlation comparison; Mahmoud Zadeh et al. proposed an uninterrupted path planning system for Multi-USV sampling mission in a cluttered ocean environment [9]; Wang et al. proposed a recurrent neural network for modelling crack growth of aluminium alloy [10].

In general, algorithms based on the extended Kalman filter and those based on the direct probability criterion are limited by the linearization of terrain and the acquisition of model a priori information, respectively. In contrast, algorithms based on terrain correlations are simple in principle, and the most representative of these algorithms is terrain contour matching (TERCOM) [11]. However, the TERCOM method requires the carrier to maintain uniform linear motion in the horizontal plane while measuring real-time elevation data. The amount of data collected for a single match is significant, resulting in lower real-time performance. These properties of TERCOM make it difficult for underwater applications to achieve the same localization accuracy as aircraft applications. To address this problem, researchers have proposed other terrain-dependent matching algorithms that can be used underwater in recent years. Chengxiang Liu improved the robustness of the iterative closest point (ICP) algorithm proposed by Besl and McKay for 3D shape alignment and introduced it to underwater terrain matching [12]. Behzad et al. provided the iterative closest contour point (ICCP) algorithm based on the ICP algorithm and applied it to the underwater environment [13]. Yuan et al. [14] combined the TERCOM/ICCP algorithm with Kalman filtering to perform terrain matching. These algorithms can be categorized as algorithms that use terrain correlation analysis for matching localization. However, these methods require much measured terrain data and massive reference maps and thus do not meet the criteria for real-time performance and precise localization.

2.2 3D point cloud-related methods

The extensive study and development of deep learning have led to better matching localization methods based on 3D point clouds, which are widely used as data structures to describe 3D objects and scenery. In this research, the method consists primarily of three modules: a feature extraction network, keypoint detection module, and matching localization module for underwater terrain point cloud data. According to the function of each module, the method is divided into three parts.

2.2.1 Feature extraction of point clouds

Before the advent of deep learning, point cloud feature descriptions were mainly hand-crafted. Due to the advantages of being data-driven, deep learning techniques have been used for point cloud feature learning in recent years. PointNet [15] accepts point clouds as direct input and extracts point state features through the MLP layer. Later, PointNet++ [16] expanded the feature acceptance domain by referencing convolutional neural networks (CNNs) and learning point state features from the local neighbourhood of each point. 3DMatch [17] builds point correspondence by learning local voxel descriptors. PPFNet [18] combines coordinates, normals, and point pair features (PPFs) to construct distinct and rotation-resistant local features. Graphit [19] learns point state features with graphical convolutional networks (GCNs) [20] and extracts salient points with GCNs and fully connected layers.

2.2.2 Keypoint detection of point clouds

Hand-crafted detectors extract 3D keypoints according to specially designed rules, which usually can only detect 3D keypoints with specified geometric properties and have low detection accuracy and efficiency. Therefore, we use a learning-based detector to extract 3D keypoints, which can effectively avoid the limits of hand-crafted detectors. FAST [21] was the first machine learning method to obtain corner keypoints. 3DFeatNet [17] is a weakly supervised 3D keypoint detector that uses GPS/INS-tagged point clouds to undergo training. To improve the accuracy of keypoint detection in point clouds, these detectors must be improved. USIP [22] uses a feature suggestion network (FPN) to calculate saliency uncertainties and selects the minimal points as the 3D keypoints. The 3D3L [23] method generates reliability and repeatability score maps and multiplies these two score maps to obtain the keypoint score maps. The points with larger values in the score map are extracted as keypoints.

2.2.3 Matching localization of point clouds

The simultaneous localization and mapping (SLAM) [24] algorithm was an early solution for point cloud-based geometric alignment and position recognition of 3D objects [25–27]. It constructs a map of the environment by combining multiple sensors and estimating the position of the underwater terrain. PPF-FoldNet [18] operates on the original points by combining point pair features and global context to improve the descriptor representation. L3-Net [28] achieves centimetre-level localization by constructing a matching customer to calculate the error between the predicted and actual values. PointNetLK [29] extracts the global features of two input point clouds before utilizing the inverse combination (IC) algorithm to estimate the transformation matrix and minimize the feature difference between the two features. Huang et al. [30] further improved PointNetLK by using autoencoders and point distance loss while reducing its reliance on labels. FMR [30] is a feature-metric alignment method that modifies the alignment problem from one that minimizes point-to-point projection error to one that minimizes feature discrepancy. DH3D [31] presents a hierarchical twin network that uses a coarse-to-fine relocation technique to discover local features for accurate posture estimation. GeoTransformer [32] has recently been made robust in cases of low overlap and achieved higher accuracy by encoding pairwise distances and angle triplets.

3 Problem statement

We design a deep learning framework for localization based on 3D sonar point cloud matching using online real-time sonar point clouds and prebuilt 3D point cloud maps. A real-time online 3D point cloud is a continuous frame detected by sonar, which is represented as a set of 3D points $\{P_i \mid i = 1, \ldots, n\}$, where each point P_i is of the form (x, y, z). The preconstructed 3D point cloud map is a collection of point clouds with obtained global coordinates.

In addition to the online point cloud and the prebuilt map, our localization method takes the predicted pose generated by inertial measurements as input. Therefore, the task is to find the best offset between the actual and predicted poses by minimizing the matching offset between the online point cloud and the 3D map. To make our method more efficient, we estimate only the two-dimensional (2D) horizontal and heading offsets $(\Delta x, \Delta y, \Delta\varphi)$.

Finally, we directly define the root mean square: $\mathrm{RMS} = \frac{1}{n}\sqrt{\sum_{i=1}^{n}\left[f(x_i)^2\right]}$, $(i = 1, 2, \ldots, n)$, where $f(x_i)$ denotes the error of the distance between the predicted location offsets and the ground truth.

4 PL-Net

We constructed a novel network architecture called PL-Net. As shown in Fig. 3, it consists of three parts: (1) a key point detection module (KDM). This module uses the k-nearest neighbours (KNN) algorithm to filter a fixed number of suitable keypoints and group neighbourhood points by various aspects, such as integrated density, geometric features, and distributed point clouds. (2) A feature extraction module (FEM). This module shares the same filter to detect features and extracts meaningful feature descriptors in the neighbourhood range of each keypoint. To solve the problem of disorderly point consumption, it uses multilayer PointNet in the network architecture. (3) A matching localization module (MLM). This module constructs an adjustable volume in the solution space and calculates the corresponding elasticity between real-time point cloud frames and maps based on keypoints. Finally, we add a self-attention mechanism to address the problem of limited processing resources with optimization techniques and define localization offsets and losses.

4.1 Keypoint detection module

Due to the vast number of original point clouds, some point clouds have high feature similarities, resulting in computational redundancy and resource waste. To solve these problems, we propose a keypoint detection module that efficiently extracts a fixed number of keypoints for the clustered sonar point cloud topographic map and the online collected point cloud area. Considering the integrated density, geometric features, and distribution of point clouds, we use a *KD*-Tree based on the KNN classification algorithm. Finally, the keypoints of small pieces of the terrain map are stitched to form the keypoints of the entire terrain.

We use the *KD*-Tree algorithm to divide the entire point cloud into point cloud blocks. Each block contains almost the same number of point clouds, achieving uniform partitioning and filtering out a certain number of candidate key points. First, we take all point clouds as input and begin to build the *KD*-Tree. Meanwhile, we select its root node and specify its division dimension. Then, we sort the one-dimensional data, select the median as the tree's root, and divide the rest of the data into left and right subtrees. Next, the data in the left and right subtrees repeat the above process until all the data are divided axially, and a balanced *KD*-tree is constructed. Finally, we traverse all the points and find the candidate keypoints with sufficient point densities in their neighbourhood.

For the KNN algorithm, three essential elements must be considered. They are the choice of k-value, classification decision procedure, and distance metric selection. If k is too small, the classification result is vulnerable to noise. Conversely, if k is too large, the nearest neighbours may include excessive points from other categories. We use cross-validation to calculate the k-value and the majority voting classification rule to classify categories.

The greater the metric distance is, the greater the disparity and the smaller the similarity. Therefore, we choose the Euclidean distance to measure the dissimilarity between two points, assuming that a and b are represented by two n-dimensional vectors $x_a(x_{a_1}, x_{a_2}, \ldots, x_{a_n})$ and

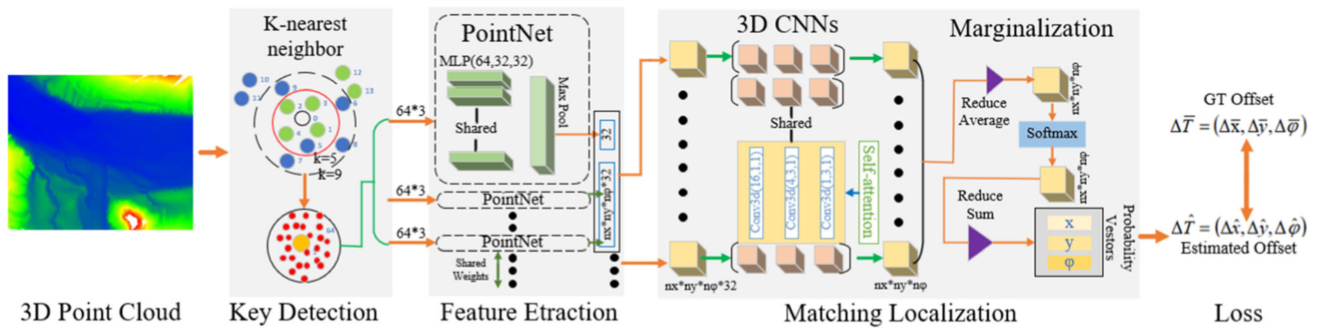


Fig. 3 Architecture of the proposed deep learning-based matching localization network, PL-Net. During the first training stage, which includes keypoint selection, PointNet feature extraction is performed. The regularization based on the 3D CNNs occurs in the second training stage, where the position is determined

$x_b(x_{b_1}, x_{b_2}, \ldots, x_{b_n})$. The Euclidean distance between a and b is defined as follows:

$$d(a,b) = \sqrt{(x_{a1} - x_{b1})^2 + (x_{a2} - x_{b2})^2 + \cdots + (x_{cn} - x_{bn})^2} \tag{1}$$

We use the well-known 3D structure tensor to evaluate the linearity and scattering of each candidate keypoint, rank the candidate keypoints according to their combined geometric features of linearity and scattering, and employ the KNN algorithm to perform the selection of keypoints. First, we use the metric function to calculate the distance between the sample classified as x training samples closest to the sample as the k-nearest neighbours of x. If the samples in one of the categories make up the majority, the classified sample x is classified into that category.

Two distinct types of sample data are represented by blue and green dots, as shown in Fig. 4. The white dot in the centre represents the classification centre, and the labelled data are the data to be classified. The data are classified according to KNN. If $k = 5$, the nearest neighbours of the centre point are four green dots and one blue dot. As the minority is subordinate to the majority, this point is classified as belonging to the green category. If $k = 9$, the nearest points to the centroid are four green and five blue dots; using the same procedure as previously described, the point is classified as belonging to the blue sample category. For each filtered keypoint, we collect 64 neighbouring points. Each neighbourhood point consists of x, y, and z coordinates used for feature extraction.

4.2 Feature extraction module

The filtered keypoints are large in dimension, including a great deal of redundant information, or are highly sparse, which makes computation difficult and direct training ineffective. To solve these problems, we present the feature extraction module. It is used to select multiple neighbouring points of each keypoint and perform feature extraction on the neighbouring points separately for the prestored sonar point cloud topographic maps and the online collected sonar point cloud topographic maps, as well as the neighbouring range of each keypoint.

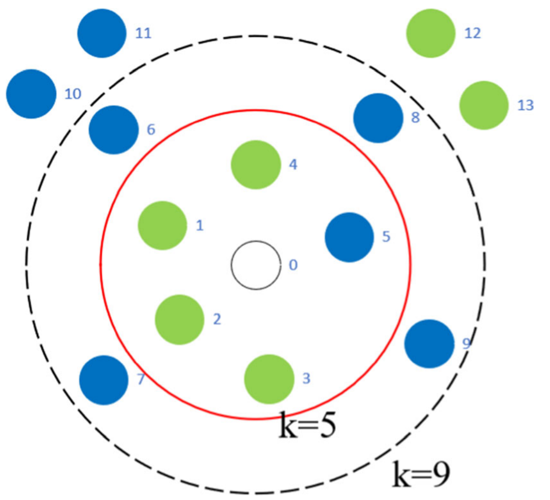


Fig. 4 Selection of keypoints

After the keypoint detection module has filtered out all eligible keypoints, we extract their meaningful feature descriptors. Using the features learned by the neural network, superficial geometric or statistical characteristics are employed to describe the similarity between point clouds. According to the conventional method, each output image cell only relates to a piece of the input image. In a conventional neural network, any output cell is influenced by all input cells. This invariably makes the extraction effect much less effective, so we want each region to have its unique features and be unaffected by other regions.

In this paper, we address the issue of disordered point consumption in the network structure by applying deep learning methods to extract features. For different regions, we share the same filter to detect features, and consequently, we share the same set of parameters, drastically reducing the number of network parameters. In addition, we can use fewer parameters to achieve better results during model training, so overfitting can be effectively avoided and the model's robustness can be enhanced.

We use the 64 neighbourhood points near each keypoint as input to extract features. As shown in Fig. 3, the original PointNet serves as the basis for a completely new network architecture consisting of a multilayer perceptron (MLP), three fully linked layers, and a max-pooling layer.

Among them, the pooling layer extracts the main features in specific regions and reduces the number of parameters to prevent model overfitting. Meanwhile, it expands the perceptual field to reduce redundancy. We execute a convolution operation to obtain local features, and the weight matrix is used to reassemble the previously local features into exclusive features by adding a fully connected layer. This network's input is a 64 $\times$ 3 vector, and its output is a 32-dimensional feature descriptor. The feature vector is extracted in the neighbourhood range of each keypoint for the prestored sonar point cloud topography and the online collected sonar point cloud topography.

4.3 Matching localization module

We present a matching localization module to compare the position information of the real-time point cloud frames with that of the original map to improve localization accuracy. A tiny volume in our defined space (x, y, φ) is constructed and regularized with 3D CNNs. Then, we accurately infer the localization offsets $(\Delta x, \Delta y, \Delta\varphi)$. Based on the keypoints, we compute the offset geometry between the real-time point cloud frame and the map.

First, we divide the space into x, y, and φ dimensions and denote the size of each dimension by nx, ny, and $n\varphi$. Then, we denote $\{f_1, \ldots, f_N\}$ as the keypoint feature descriptors of the online point cloud. Thus, the total number of offset geometries is $N \times n_x \times n_y \times n_\varphi$. Each cell represents the matching offset between the corresponding keypoint and the 3D-mapped point with the specified offset.

Given the predicted pose, we divide the neighbourhood of the predicted pose into x, y, and yaw dimensions, denoted as $\{(\Delta x_i, \Delta y_j, \Delta\varphi_k) | 1 \le i \le n_x, 1 \le j \le n_y, 1 \le k \le n_\varphi\}$, and all local keypoints in the online point cloud are transformed into global coordinates. For the corresponding coordinates in the 3D map, we use a 2 $\times$ 2 rotation matrix and a 2D translational vector transformation to compute:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \sin\Delta\varphi_k & -\cos\Delta\varphi_k \\ \cos\Delta\varphi_k & \sin\Delta\varphi_k \end{pmatrix} \times \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} \Delta x_i \\ \Delta y_j \end{pmatrix} \tag{2}$$

Then, for the nearest neighbour points with corresponding coordinates computed in the 3D map, their feature descriptors are extracted by PointNet. Each cell in the offset geometry is associated with an original keypoint and feature descriptor in the online point cloud, a transformation, and the corresponding feature descriptor in the map. Local neighbourhoods are selected in the prestored map with all offset positions as the centres, and features are extracted based on the neighbourhood range of each keypoint to form $N \times n_x \times n_y \times n_\varphi$ 32-dimensional feature vectors. The distance of each keypoint is computed from its surrounding $n_x \times n_y \times n_\varphi$ offset positions, forming an offset geometry of size $n_x \times n_y \times n_\varphi \times 32$.

Next, we construct the regularization network, which consists of two parts. One part consists of two 3D convolutional layers normalized using ReLU units and batches, one of which includes all the keypoints of a single frame. The other part consists of a single convolutional layer whose output is sent straight without normalization or activation. The more parameters a neural network model has, the more expressive it is and the more information it stores, but too many parameters can lead to information overload problems.

Finally, we add a self-attention mechanism between these two parts to focus on the most critical information for the current task among the input information and reduce the attention given to other information, or even filter out irrelevant information, to solve the problem of information overload and to improve the efficiency and accuracy of task processing. We input N $n_x \times n_y \times n_\varphi \times 32$-sized feature vectors into the network separately to obtain N offset geometries.

4.4 Self-attention mechanism

Currently, computational power is still a bottleneck limiting the development of neural networks, and the models become more complex as the input data increase in size. Although optimizations such as local connectivity, weight sharing, and pooling can simplify neural networks and effectively alleviate the conflict between model complexity and expressiveness, there is a long sequence of inputs in recurrent neural networks (RNNs). For example, they lack a high "memory" capacity for information. In addition, RNNs are not computed in parallel, and each unit needs to wait until the information of the other input units has been computed before operating. As a result, we use a self-attention mechanism that can reduce the amount of data processed and the computational resources needed.

We first define three basic concepts: q, k, and v. q is used to match other units, k is used to match other units, and v is the information to be retrieved. The output of each unit is multiplied by different matrices to obtain:

$$q^i = W^q a^i \tag{3}$$

$$k^i = W^k a^i \tag{4}$$

$$v^i = W^v a^i \tag{5}$$

where the W^q, W^k, and W^v matrices represent the parameters to be learned. For each cell, we take its q and add the dot product of k to all cells including itself to obtain α as follows:

$$\alpha_{n,i} = q^n \cdot k^i / \sqrt{d} \tag{6}$$

where n denotes the number of input vectors, d depends on the values of q and k, and $\cdot$ product represents the similarity, increasing with the dimensionality. Afterwards, we apply the softmax function to $\hat{\alpha}$ and normalize:

$$\hat{\alpha}_{n,i} = \exp(\alpha_{n,i}) / \sum_j \exp(\alpha_{n,j}) \tag{7}$$

Finally, we calculate the combined information of each cell after attention:

$$b^n = \sum_i \hat{\alpha}_{n,i} v^i \tag{8}$$

The traditional sequence model contains an encoding and decoding part, which has two faults. First, the encoder must compress all the input information into a fixed-length vector. Second, it cannot model the relationships between the input and output sequences. To address these issues, we use a self-attention mechanism with a decoder with the ability to access all of the encoder's outputs to overcome these two major drawbacks of the conventional structure.

As shown in Fig. 5, we input 32-dimensional feature descriptors into CNNs and combine the attention mechanism between the convolutional layer and batch processing of this network to accelerate the training and convergence of the network while preventing gradient overfitting and controlling gradient explosion.

We designed the self-attention mechanism, which consists of three convolutional layers and a max-pooling layer, to fit the entire network. We embed its starting part between the 3D convolutional layer and the batch normalization layer. As for the end part of the self-attention mechanism, we embed it after the convolutional layer in forwards propagation to optimize the trained model.

4.5 Offsets and losses

We compute the offset geometry of all matching offsets $\{(\Delta x_i, \Delta y_j, \Delta\varphi_k)\}$ for each keypoint. A probability offset can be introduced to represent the standard features of all keypoints in the offset space, which has size $n_x \times n_y \times n_\varphi$. The space represents the matching offsets between the online point cloud and the 3D map. In this section, we follow the matching offsets to calculate the localization offsets between the sonar point cloud frame and the map underprediction in real time.

First, we assume that the offset is $\Delta T = (\Delta x_i, \Delta y_j, \Delta\varphi_k)$. Then, we calculate the matching probability of the online point cloud frame and map, denoted as $\prod_{i=1}^{N} P_i(\Delta T)$. We take the matching probability logarithmically to obtain $C(\Delta T) \propto \log\left(\prod_{i=1}^{N} P_i(\Delta T)\right) = \sum_{i=1}^{N}(\log(P_i(\Delta T)))$. $P_i(\Delta T)$ denotes the matching probability of the ith keypoint at offset ΔT, and $C(\Delta T)$ denotes the total matching offset between the online point cloud frame and the map at offset ΔT.

Then, the matching probabilities are normalized using the softmax function, and the offset probabilities in the x-direction, y-direction, and yaw-direction are extracted: $P_i(\Delta x_i) = \sum_{y,\varphi} P(\Delta T)$, $P_j(\Delta y_j) = \sum_{x,\varphi} P(\Delta T)$, and $P_k(\Delta\varphi_k) = \sum_{x,y} P(\Delta T)$, respectively.

We define the predicted offset as $\Delta\hat{T} = (\Delta\hat{x}, \Delta\hat{y}, \Delta\hat{\varphi})$ and the ground truth as $\Delta\bar{T} = (\Delta\bar{x}, \Delta\bar{y}, \Delta\bar{\varphi})$ and the predicted offset is given by:

$$\Delta\hat{T} = \left(\sum_{i=1}^{n_x} P_i(\Delta x_i) \cdot \Delta x_i, \sum_{j=1}^{n_y} P_j(\Delta y_j) \cdot \Delta y_j, \sum_{k=1}^{n_\varphi} P_k(\Delta\varphi_k) \cdot \Delta\varphi_k\right) \tag{9}$$

Finally, we directly define the loss and calculate the distance based on the squared Euclidean distance between the predicted offset and the ground truth:

$$\text{Loss} = \alpha \cdot (\|\Delta\hat{x} - \Delta\bar{x}\|^2 + \|\Delta\hat{y} - \Delta\bar{y}\|^2) + \|\Delta\hat{\varphi} - \Delta\bar{\varphi}\|^2 \tag{10}$$

5 Experimental results

5.1 Datasets description

It is common knowledge that underwater topography is complicated by insignificant undulatory features and low stability of underwater topographic measurements, which also significantly impact other underwater carriers. As shown in Fig. 6, we use a multibeam bathymetry system to

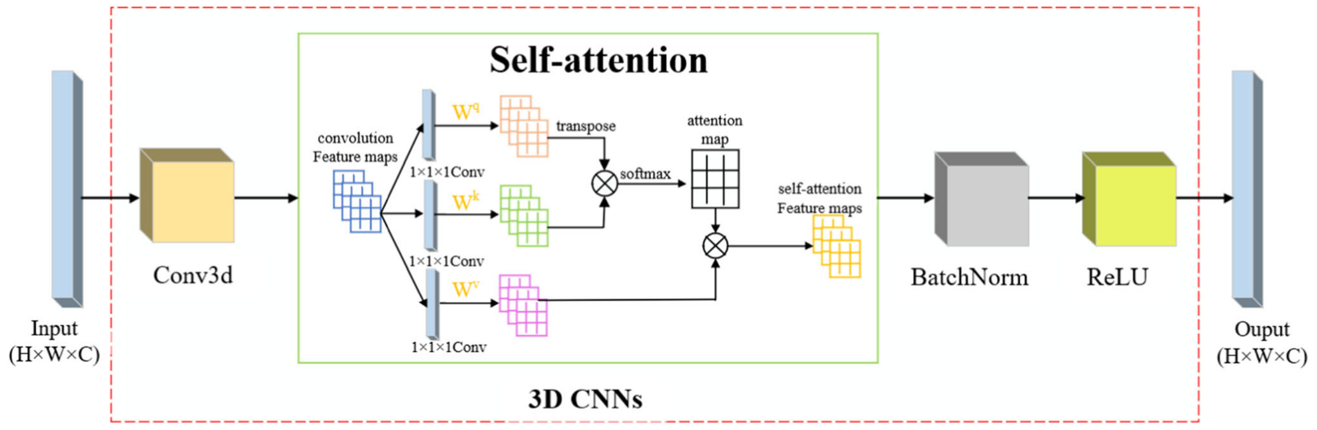


Fig. 5 Architecture of the proposed learning-based matching localization network, PL-Net. During the first training stage, which includes keypoint selection, PointNet feature extraction is performed. The regularization based on 3D CNNs takes place in the second training stage, where the position is determined

collect and acquire data on underwater topographic features and construct underwater topographic images.

The multibeam bathymetric system is a complex bathymetric system with a multisensor combination that can obtain tens, hundreds, or even thousands of bathymetric data in the vertical and trajectory directions with each acoustic wave emission and obtain a high-precision underwater 3D topographic whole picture by many high-precision bathymetric results. The process is shown in Fig. 7, which primarily contains the following core hardware and software: (1) multibeam bathymetry; (2) global navigation satellite system (GNSS); (3) surface acoustic velocimeter; (4) acoustic velocity profiler; and (5) attitude meter.

The working principle of a multibeam bathymetry system is to use the transmitting transducer array to transmit acoustic waves with wide sector coverage to the seafloor. It uses the receiving transducer array to receive a narrow beam acoustic wave and forms the irradiated footprints of the seafloor topography through the orthogonality of the transmitting and receiving sector pointing. Afterwards, it processes these footprints appropriately and obtains the water depth of hundreds or more measured points on the seafloor in the vertical plane perpendicular to the heading in one detection. It enables accurate and rapid measurement of underwater targets' size, shape, and height changes within a certain width along the route. At the same time, it more reliably depicts the 3D properties of the seafloor topography, as shown in Fig. 8.

Multibeam sonar bathymetry is a measurement technique that can scan underwater targets. It uses multiple acoustic beams to obtain information on the target location, morphology, and reflectivity. As shown in Fig. 9, we first install multibeam sonar equipment in the water where data need to be collected. Next, the multibeam sonar equipment generates an acoustic beam through the sound source, which propagates underwater and is reflected when it encounters an object. It is received by the sonar array and transmitted to the receiver for signal processing. After receiving the signal, the data acquisition system stores and processes the data.

We used CloudCompare software to load the data, as shown in Fig. 10a and b, convert them to point cloud format, save them as text, and plot the underwater topographic elevation. We collected underwater topographic data for an underwater topographic area with a resolution of 5 m. After reading the point cloud files with Point Cloud Library (PCL) software, the point clouds were processed using the Euclidean clustering algorithm.

First, we initialize two empty sets, one as the point set of alternative points and the other as the clustering result set. Then, a random point is selected from the point cloud to serve as the seed, and the closest point is searched according to the given radius. Next, a threshold is set. If the nearest point is less than the threshold value of the given Euclidean distance, the point is included in the alternative points and the corresponding clustering result set; otherwise, the search for the current point is completed. The nearest points in the remaining unclustered point cloud are compared with the threshold value one by one until the alternative point set is empty. Finally, the points in the clustering result set are removed from the original point cloud until all points in the point cloud are removed.

To achieve uniform segmentation, we add the maximum and minimum point limits and set the sampling frequency to partition the whole point cloud data into 3000 blocks containing approximately 10,000 points per block to achieve uniform processing. To improve the appearance of

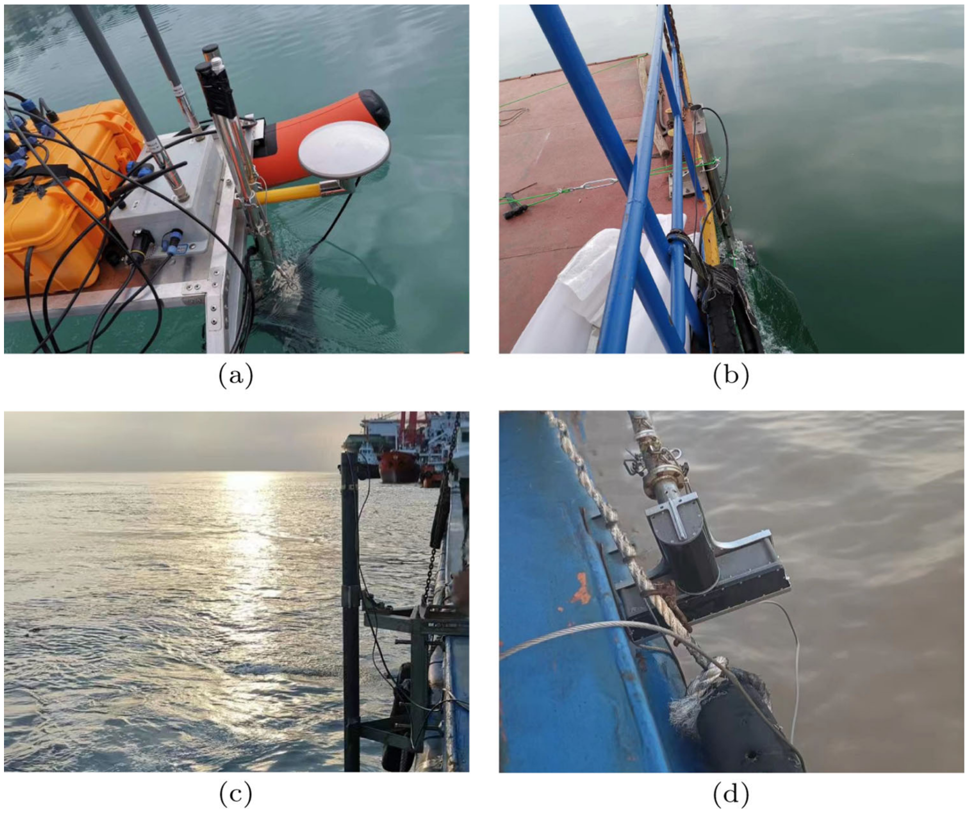


Fig. 6 Experimental environment and shipboard multibeam sonar instrumentation. On Songhua Lake, we performed the installation of experimental equipment and completed the data acquisition

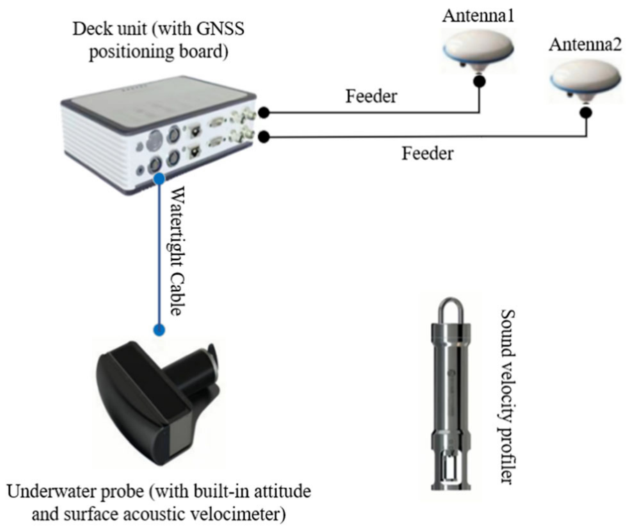

Deck unit (with GNSS positioning board)
Antenna1
Antenna2
Feeder
Feeder
Watertight Cable
Underwater probe (with built-in attitude and surface acoustic velocimeter)
Sound velocity profiler

Fig. 7 The multibeam bathymetric system devices. It mainly consists of three parts: deck unit, underwater probe, and sound velocity profiler

the point cloud data, we randomly selected colours to represent the processed point cloud, as shown in Fig. 11, and saved the result as the network's input. The point cloud is represented as a set of 3D points $\{P_i \mid i = 1, \ldots, n\}$, where each point P_i is a vector of (x, y, z).

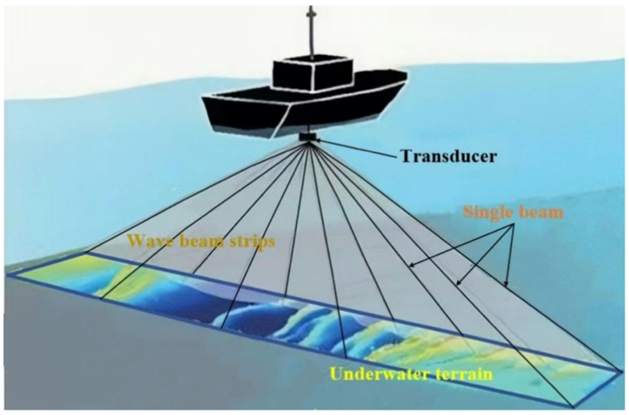


Fig. 8 Architecture of the multibeam bathymetry system

5.2 Experimental setup

We select 128 keypoints in one frame of the online point cloud for keypoint selection. The size of the offset geometry is set to $11 \times 11 \times 11$, while the steps of the x, y, and φ dimensions are 0.25 m, 0.25 m, and 0.5°, respectively. Therefore, the maximum tolerable offset of the predicted pose is approximately $\left(0.25 \times \frac{11-1}{2} = 1.25\ \mathrm{m}, 1.25\ \mathrm{m}, 2.5^{\circ}\right)$, which is sufficient for the application. In our implementation, the PointNet structure is $64 \times 32 \times 32$ MLP; 3D CNNs are $\mathrm{Conv3}d(16,1,1) - \mathrm{Conv3}d(4,3,1) - \mathrm{Conv3}d(1,3,1)$.

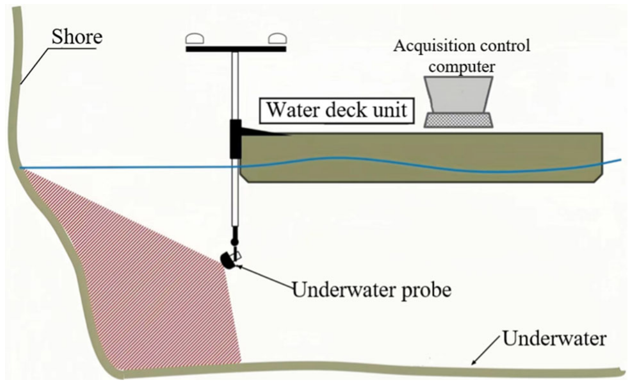


Fig. 9 The multibeam sonar bathymetry system acquisition data map

To achieve the best effect of the training model and address the limited computing power, a self-attention mechanism is added. The structure consists of three identical convolutional layers and a softmax function. This convolutional layer is Conv3$d(1, 1, 1)$ with a uniform step size of $(1, 1, 1)$. We also add forward propagation to the neural network of this module to obtain the loss error by acting on the input of each layer and calculating the output result layer by layer until the output layer.

We train the PointNet structure and the 3D convolutional neural network during the training phase. The loss is directly calculated from the probability vectors inferred from the probability offsets to achieve this. The batch size and learning rate were set to 1 and 0.01, respectively. To make the extracted features more robust, uniformly distributed random noise of in $x - y$ dimension [0–1.0]° and random noise in the *yaw* dimension [0–2.0]° were added to the input prediction poses. We randomly divided the dataset into training and validation sets during the training phase. There are 2400 point cloud blocks in the training set and 600 point cloud blocks in the validation set, a ratio of approximately 4:1. If there is no performance gain after 2000 epochs, the algorithm is stopped.

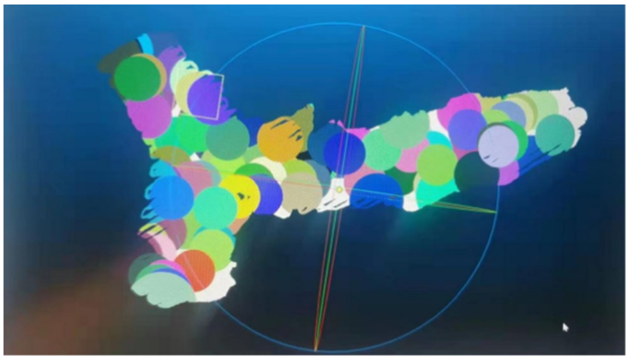


Fig. 11 Split the point cloud and render it with random colours

5.3 Quantitative analysis

TERCOM and ICP are both early classical point cloud matching algorithms. TERCOM mainly selects the corresponding point sequence in the terrain matching base map based on the submersible track point sequence, calculates the base map bathymetric sequence, and uses the position corresponding to this sequence as the matching result of the target. PPF-HoldNet and GeoTransformer are current point cloud matching algorithms based on deep learning. PPF-HoldNet is a deep learning-based algorithm for matching point clouds that uses the local 3D geometry encoded by a point pair feature. GeoTransformer encodes the distance between superpoint pairs and the angle information between superpoint triplets into a transformer for efficient global structure information learning.

Regarding performance, our proposed learning-based localization method is tested and compared to the conventional TERCOM and ICP methods and the learning-based localization methods PPF-FoldNet and GeoTransformer. In addition, we perform quantitative analysis for each available method by calculating the root mean square horizontal, latitude, longitude, and heading angles, the maximum values of the horizontal and heading angles.

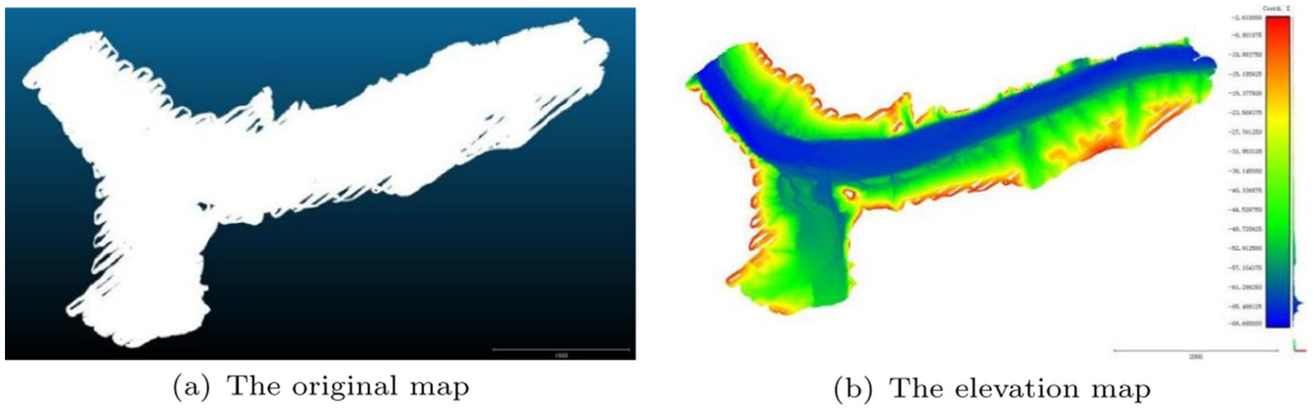


Fig. 10 Structure of the coder-decoder Introducing the attention mechanism

Table 1 demonstrates that our method offers a significant performance benefit.

Comparing the localization results in Tables 1 and 2, our localization method achieves high accuracy in various scenarios. Among them, the root mean square and maximum values of the horizontal offset and the root mean square latitude and longitude are numerically less than those of other location algorithms. Their values are 0.029, 0.131, 0.012, and 0.028, respectively, indicating that our proposed location method produces a model with less error after training. We also implement the verification of the root mean square and maximum values of the yaw angle, again by comparing with TERCOM, ICP, PPF-HoldNet, and GeoTransformer, and we see that Yaw.RMS and Yaw.Max are 0.021 and 0.057, respectively.

In addition, the accuracy ratio of both our horizontal localization offset and heading localization offset exceeded 70%, especially when the horizontal localization offset was less than 0.375 m and the heading localization offset was less than 0.3°, which is close to 80%. These values are significantly higher than those of the traditional TERCOM and ICP algorithms and are improved compared to those of the deep learning PPF-FoldNet and GeoTransformer algorithms, further demonstrating that our learning-based localization method is superior in terms of performance.

5.4 Accuracy analysis

We train the model further by combining the self-attention mechanism to test the method without the self-attention mechanism and the new method with the self-attention mechanism. We compare the accuracy ratios regarding the horizontal offset distance and heading offset angle. For the horizontal offset, we test the position offset at different distances $(0.125\ \mathrm{m}, 0.250\ \mathrm{m}, 0.375\ \mathrm{m})$ and count the accuracy ratio in these ranges; for to the heading offset, we test the position offset at different angles $(0.1^\circ, 0.2^\circ, 0.3^\circ)$ and count the accuracy ratio in these ranges, as shown in Table 3.

The experimental results are shown in Fig. 12a and b. After combining the self-attention mechanism, our new method improves even more, with accuracy ratios greater than 75% in all cases, with superior and stable performance.

We perform many tests on the model without the attention mechanism to better compare the horizontal offset distance and heading offset angle in different ranges. The new model, after adding the self-attention mechanism, randomly selects 10 sets of test results using the visual image method. We use a solid line to represent PL-Net and a dashed line to represent PL-Net (no attention) and highlight each set of test results. In addition, we use circles, triangles, and squares to indicate horizontal offset distances <0.125 m, <0.250 m, and <0.375 m (heading offset angles $<0.1^\circ$, $<0.2^\circ$, and $<0.3^\circ$), respectively, and use different colours to denote the various models. As shown in Fig. 13a and b, the new method combining the self-attention mechanism has an improved accuracy ratio, exceeding 80%, and excellent improved stability.

5.5 Runtime and error analysis

A runtime analysis was performed using a GTX4000 GPU, Inter Xeon(R) Gold 6226R CPU, and 64 GB of RAM, as shown in Table 4.

We separately tested and reported the times for TERCOM, ICP, PPF-HoldNet, GeoTransformer, and PL-Net. The whole time takes only 8.6 ms, which further indicates that the model works well and reflects the better real-time performance of our proposed localization method, as shown in Table 5.

Table 1 Comparison of TERCOM, ICP, PPF-FoldNet, GeoTransformer, and PL-Net (no attention) for each index in the horizon

Methods	Horiz.RMS	Horiz.Max	Long.RMS	Lat.RMS	< 0.125 m Pct	< 0.250 m Pct	< 0.375 m Pct
TERCOM	0.069	0.664	0.064	0.058	29.55%	34.58%	39.38%
ICP	0.045	0.448	0.029	0.045	34.34%	39.68%	47.58%
PPF-FoldNet	0.049	0.488	0.049	0.032	62.14%	65.99%	71.41%
GeoTransformer	0.032	0.180	0.018	0.029	71.23%	72.62%	74.80%
PL-Net (no attention)	0.030	0.131	0.012	0.028	72.39%	74.63%	78.87%

RMS represents the root-mean-square, Max represents the maximum offset, and < 0.125 m Pct. represents the proportion of the heading offset within the range of 0.125 m

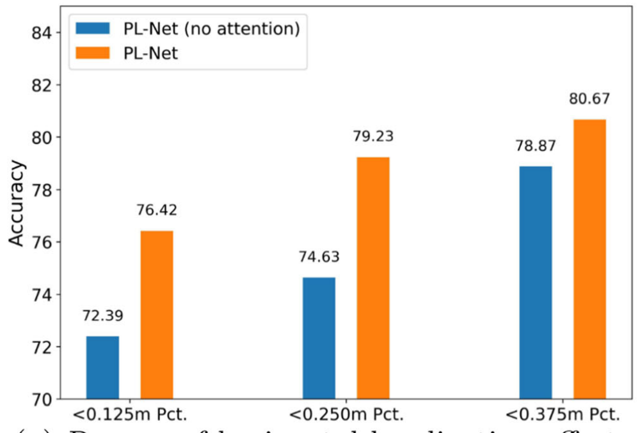
Table 2 Comparison of TERCOM, ICP, PPF-FoldNet, GeoTransformer, and PL-Net (no attention) for each index in the yaw

Methods	Yaw.RMS	Yaw.Max	$< 0.1°$ Pct	$< 0.2°$ Pct	$< 0.3°$ Pct
TERCOM	0.084	0.907	31.63%	47.15%	51.94%
ICP	0.075	0.750	40.83%	58.30%	61.39%
PPF-FoldNet	0.026	0.261	63.81%	66.43%	68.19%
GeoTransformer	0.019	0.165	73.28%	75.33%	76.01%
PL-Net (no attention)	0.019	0.165	76.35%	78.27%	79.45%

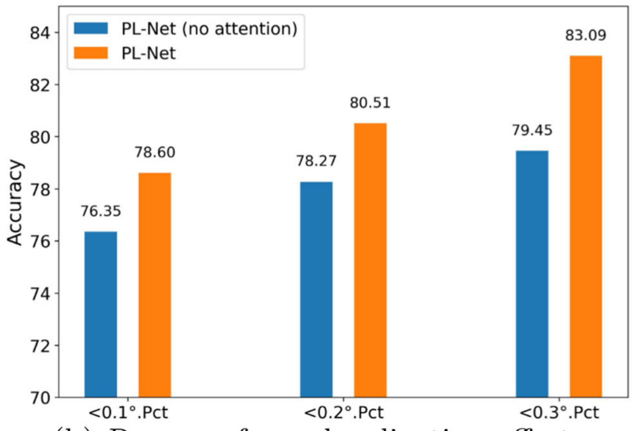
RMS represents the root-mean-square, Max represents the maximum offset, Yaw represents the yaw angle, and $<0.1°$ Pct. represents the proportion of the heading offset within the range of $0.1°$

Table 3 Different ranges of horizontal and yaw localization offsets

Methods	< 0.125 m Pct	< 0.250 m Pct	< 0.375 m Pct	$<0.1°$ Pct	$< 0.2°$ Pct	$< 0.3°$ Pct
PL-Net (no attention)	72.39%	74.63%	78.87%	76.35%	78.27%	79.45%
PL-Net	76.42%	79.23%	80.67%	78.60%	80.51%	83.09%

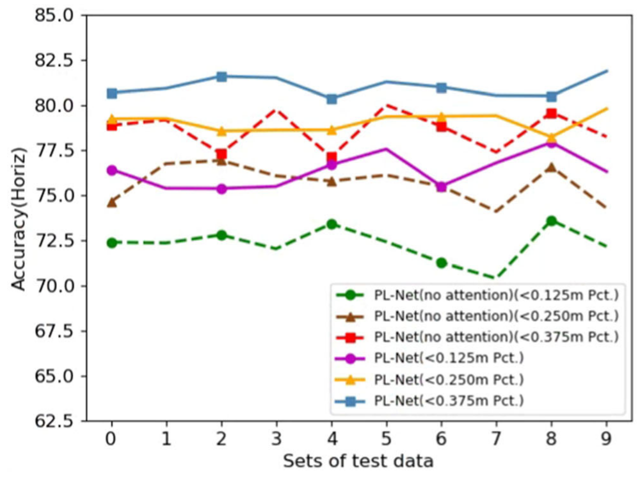


(a) Ranges of horizontal localization offsets

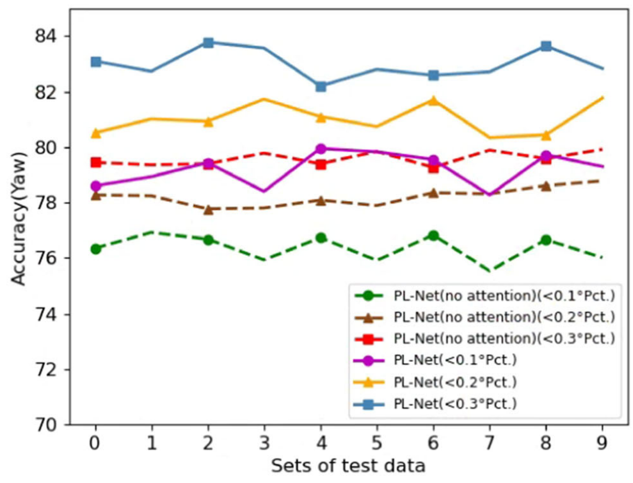


(b) Ranges of yaw localization offsets

Fig. 12 Accuracy ratio



(a) The horizontal localization offsets



(b) The yaw localization offsets

Fig. 13 Accuracy ratio of PL-Net (no attention) and PL-Net

Table 4 Computer performance configuration

Performance configuration	Parameter
Inter processor	Inter Xeon(R) Gold 6226R CPU
Random access memory	64 GB
Central processing unit	32 core
Graphic processing unit	Nvidia GeForce RTX3090*2
Hard disk drive	4TB
CPU clock speed	2.9 GHz

Table 5 Real-time performance comparison of different algorithms (TERCOM, ICP, PPF-FoldNet, GeoTransformer, and PL-Net)

Methods	TERCOM	ICP	PPF-FoldNet	GeoTransformer	PL-Net
Runtime (ms)	569	455	23	10.3	8.6

The predicted location of the point cloud and ground truth are obtained by the network, and the offset between them is calculated to select a specific area and mark the significant points in it. As shown in Fig. 14a, b, and c, we build a point cloud map of all of Songhua Lake, compare our method with the latest PPF-FoldNet and GeoTransformer alignment algorithms, and select the point clouds containing significant features. The green line represents the standard offset of the point cloud, whereas the red line represents the abnormal offset. Our network is highly efficient and achieves the expected results.

The loss function is the key to determining the learning quality of the network. If the premise of the network structure remains unchanged, the inappropriate selection of a loss function will lead to consequences such as poor model accuracy. We calculate the loss values of the deep learning-based localization algorithms separately and compare the model accuracy of these algorithms.

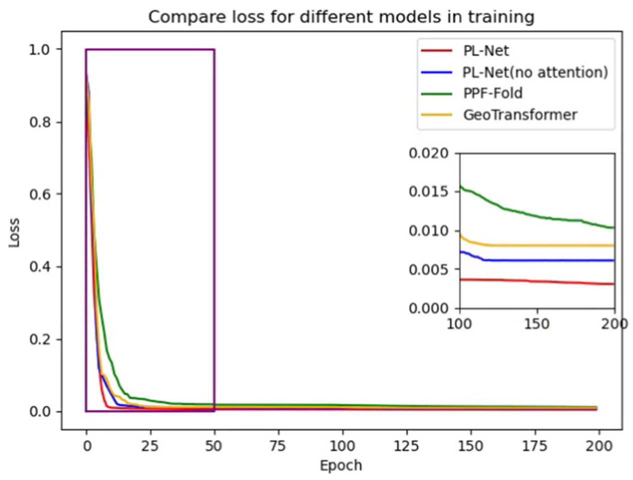


Fig. 15 Comparison of loss for different models (local enlargement of specific regions)

We compared four algorithmic models, as shown in Fig. 15: PPF-FoldNet, GeoTransformer, PL-Net (no attention), and PL-Net. Generally, they are all stable in the end with a loss below 0.01, but our model training process is more stable and superior. In addition, when we zoom in partially on the different loss comparison graphs, the selected purple boxes indicate the loss of different models in the initial stage, and obviously, PL-Net works best. In addition, to demonstrate the results of the final stabilized loss, the loss images with epochs within $(100, 200)$ are magnified. The final loss of our model is the lowest, reaching 0.004, which further validates this model effect and emphasizes the small error and precise localization of this localization method.

6 Conclusion

We present PL-Net, a 3D sonar point cloud method for underwater terrain matching localization, for the first time. Deep learning-based detection and localization methods

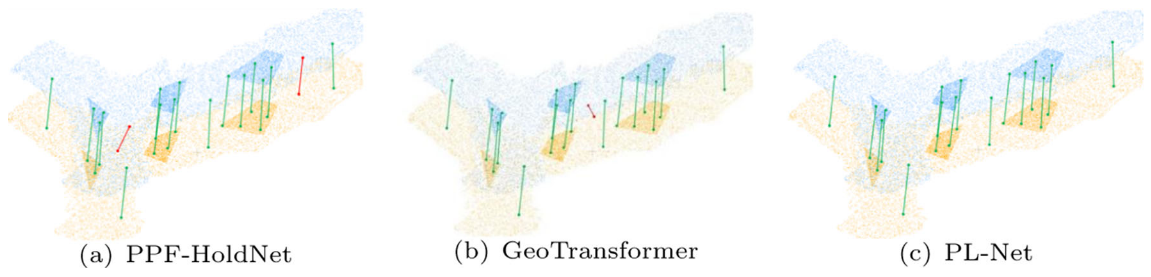


Fig. 14 The point cloud localization offset map

have replaced hand-crafted modules in current localization methods and are now industrially viable. Our method achieves high efficiency in detecting keypoints and does not require manual feature extraction, avoiding situations in which traditional algorithms fail due to environmental and equipment factors. In summary, we achieve improved matching, reduced errors, and precise location.

Data availability Data supporting the findings of this study are not publicly available due to the sensitive information related to the national underwater terrain. Upon reasonable request, data can be provided by sending a message to the author's email address, zhangyang13d@nudt.edu.cn.

Declarations

Conflict of interest No potential conflict of interest was reported by the authors.

Consent for publication Informed consent was obtained from all participants included in the analysed studies.

References

1. Cheng CQ, Hao XY, Zhang ZJ et al (2016) Robust integrated navigation algorithm of terrain aided navigation. INS J Chin Inert Technol 24:202–207
2. Song ZQ, Bian HY, Zielinski A et al (2016) Underwater terrain navigability analysis based on image processing and fuzzy decision. Inertial Technol 24(2):164–169
3. Han Y, Wang B, Deng Z et al (2016) An improved TERCOM-based algorithm for gravity-aided navigation. IEEE Sens J 16(8):2537–2544
4. Zhang J, Shen J, Li H et al (2015) Research and application progress on underwater terrain aided navigation technology. J Natl Univ Def Technol 37:128–135
5. Hollowell J (1990) Heli/SITAN: a terrain referenced navigation algorithm for helicopters. Sandia National Lab. (SNL-NM) Albuquerque NM(United States)
6. Li P, Zhang X, Xu X (2010) Novel terrain integrated navigation system using neural network aided Kalman filter. In: Sixth international conference on natural computation, pp. 445–448. IEEE
7. Anonsen KB, Hallingstad O, Hagen OK (2007) Bayesian terrain-based underwater navigation using an improved state-space model. In: Symposium on underwater technology and workshop on scientific use of submarine cables and related technologies, pp 499–505. IEEE
8. Nygren I, Jansson M (2004) Terrain navigation for underwater vehicles using the correlator method. IEEE J Ocean Eng 29(3):906–915
9. Golden JP (1980) Terrain contour matching (TERCOM): a cruise missile guidance aid. In: Image processing for missile guidance, pp 10–18. SPIE
10. Mahmoud Zadeh S, Abbasi A (2022) Uninterrupted path planning system for Multi-USV sampling mission in a cluttered ocean environment. Ocean Eng 254:111328
11. Zhi L, Zhu Y, Wang H (2016) A recurrent neural network for modeling crack growth of aluminium alloy. Neural Comput Applic 27:197–203
12. Besl PJ, McKay ND (1992) Method for registration of 3-D shapes. In: Sensor fusion IV: control paradigms and data structures, pp 586–606. Spie
13. Kamgar-Parsi B (1997) Matching sets of 3D line segments with application to polygonal arc matching. IEEE Trans Pattern Anal Mach Intell 19(10):1090–1099
14. Yuan G, Zhang H, Yuan K et al (2011) A combinational underwater aided navigation algorithm based on TERCOM/ICCP and Kalman filter. In: Fourth international joint conference on computational sciences and optimization, pp 952–955. IEEE
15. Qi CR, Su H, Mo K et al (2017) Pointnet: deep learning on point sets for 3d classification and segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 652–660
16. Qi C R, Yi L, Su H, et al (2017) Pointnet++: deep hierarchical feature learning on point sets in a metric space. Adv Neural Inf Process Syst
17. Zeng A, Song S, Nießner M et al (2017) 3dmatch: learning local geometric descriptors from rgb-d reconstructions. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 1802–1811
18. Deng H, Birdal T, Ilic S (2018) Ppfnet: global context aware local features for robust 3d point matching. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 195–205
19. Saleh M, Dehghani S, Busam B, et al (2020) Graphite: Graph-induced feature extraction for point cloud registration. In: International conference on 3D vision (3DV), pp 241–251. IEEE
20. Kipf T N, Welling M (2016) Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907
21. Rosten E, Drummond T (2006) Machine learning for high-speed corner detection. In: Computer vision-ECCV 2006: 9th European conference on computer vision proceedings Part I. Springer, Berlin, pp 430–443
22. Li J, Lee GH (2019) Usip: unsupervised stable interest point detection from 3d point clouds. In: Proceedings of the IEEE/CVF international conference on computer vision, pp 361–370
23. Streiff D, Bernreiter L, Tschopp F, et al (2021) 3D3L: deep learned 3D keypoint detection and description for lidars. In: IEEE international conference on robotics and automation (ICRA), pp 13064–13070. IEEE
24. Tangelder JWH, Veltkamp RC (2008) A survey of content based 3D shape retrieval methods. Multimed Tools Appl 39:441–471
25. Lian Z, Godil A, Bustos B et al (2013) A comparison of methods for non-rigid 3D shape retrieval. Pattern Recognit 46(1):449–461
26. Uy MA, Lee GH (2018) Pointnetvlad: deep point cloud based retrieval for large-scale place recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 4470–4479
27. Cheein FAA, Guivant J (2014) SLAM-based incremental convex hull processing approach for treetop volume estimation. Comput Electron Agric 102:19–30
28. Lu W, Zhou Y, Wan G et al (2019) L3-net: towards learning based lidar localization for autonomous driving. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 6389–6398
29. Aoki Y, Goforth H, Srivatsan RA et al (2019) Pointnetlk: robust and efficient point cloud registration using pointnet. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 7163–7172
30. Huang X, Mei G, Zhang J (2020) Feature-metric registration: a fast semi-supervised approach for robust point cloud registration without correspondences. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 11366–11374

31. Du J, Wang R, Cremers D (2020) Dh3d: deep hierarchical 3d descriptors for robust large-scale 6dof relocalization. In: Computer vision-ECCV 2020: 16th European conference proceedings part IV. Springer International Publishing, pp 744–762
32. Qin Z, Yu H, Wang C et al (2022) Geometric transformer for fast and robust point cloud registration. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 11143–11152