Real-time rail recognition based on 3D point clouds

Xinyi Yu¹, Weiqi He¹, Xuecheng Qian¹, Yang Yang¹, Tingting Zhang² and Linlin Ou¹,*

¹ Laboratory of Intelligent learning and Robotics, School of Information Engineering, Zhejiang University of Technology, Zhejiang 310000, People’s Republic of China
² School of Business, Shanghai University of Finance and Economics, Shanghai 200000, People’s Republic of China

E-mail: linlinou@zjut.edu.cn

Received 5 August 2021, revised 15 November 2021
Accepted for publication 31 May 2022
Published 20 July 2022

Abstract
Accurate rail location is a crucial part of safety monitoring in railway operational systems. Light detection and ranging can be used to obtain point clouds that contain three-dimensional (3D) information about the railway environment, especially in darkness and poor weather conditions. In this paper, a real-time rail recognition method based on 3D point clouds is proposed to solve the challenges, such as the disorder, uneven density and large volume of the point clouds. A voxel downsampling method is first presented to balance the density of the railway point clouds, and pyramid partitioning is designed to divide the 3D scanning area into voxels with different volumes. A feature encoding module is then developed to find the nearest neighbor points and to aggregate their local geometric features for the central point. Finally, a multiscale neural network is proposed to generate the prediction results for each voxel and the rail location. Experiments are conducted on nine sequences of 3D point cloud railway data. The results show that this method has good performance in detecting straight and curved rails and other complex rail topologies.

Keywords: rail recognition, railway safety, deep learning, LiDAR 3D point clouds

1. Introduction
Railway transportation is an essential type of mass transit, and is regarded as a comfortable, fast, and safe method of transportation. Safety monitoring is a crucial part of railway transportation automation. At present, most routes still rely on the judgment of drivers to ensure driving safety, for example, by identifying signal lights and obstacles on the rails. However, driver negligence and delayed reactions are inevitable and pose a significant challenge to train safety. Intelligent applications based on vehicular sensors, such as lane-keeping and auxiliary braking, have been developed to ensure safe driving for many years. Thus, this is an effective way to establish auxiliary train driving systems based on vehicular sensors [1–4]. Recognizing the rail location in front of the train is regarded as one of the essential functions of auxiliary driving systems. Based on this function, other security detection functions can be developed, such as signal lamp recognition [5], obstacle identification, collision prevention [6], and obstacle-free range detection [7].

The camera-based applications used in driver-assistance systems have many advantages, such as rich image information and direct visual feedback for train drivers. In previous studies, the rail extraction methods [2, 8] were based on two-dimensional (2D) images captured by vehicle cameras. However, these methods required good lighting conditions and were less robust in rainy, foggy, and night-time environments.
Light detection and ranging (LiDAR) is based on the use of an active sensor that transmits electromagnetic radiation and measures the backscattered signals. By measuring the attenuation of the incident light pulse, LiDAR detects the properties of the objects within its range. Recently, LiDAR has been considered as an efficient tool for remote sensing and environmental modeling [9]. It is widely used in autopiloted cars and crewless aerial vehicles to generate an accurate three-dimensional (3D) map of urban roads, dams, tunnels, and large buildings [10, 11]. Meanwhile, LiDAR has made significant progress in automatic driving. This allows the recognition of activities around the vehicle and plays a vital role in shaping decisions about the trajectory and characteristics of its motion. Furthermore, LiDAR has been introduced into railway applications, such as railway measurement [12], gap measurement [13], infrastructure reconstruction [14], and tunnel mapping [15].

As an essential function, rail recognition requires timely operation and judgment of trains running at high speed. Therefore, the accuracy and discrimination speed of rail recognition are two critical problems. However, the point cloud data produced by LiDAR has two drawbacks. On the one hand, a major challenge is the increased data volume compared with that of images. The data must be processed in a limited amount of time, which requires an expansion of the algorithm-processing capabilities. Previous studies [16–18] have used downsampling of point clouds to reduce the time cost, but they paid little attention to the density and sparseness of point clouds in outdoor scenes. Meanwhile, point clouds are discrete and variable in density, which increases the difficulty of extracting object features. On the other hand, LiDAR data is significantly affected by noise [19, 20], which deteriorates the signal-to-noise ratio. Although the noise is correlated with the LiDAR scanning distance and is occasionally handled as non-Gaussian, it makes it more difficult to recognize the rail with high accuracy. Additionally, the speed of the train together with the relative motion of the rail cause non-stationary behavior and add extra difficulty.

The rail area, including a pair of equidistant tracks and the rail bed, appears as an elongated strip in the 3D point clouds collected by LiDAR. The point clouds contain the specific reflection intensities of the tracks. The intensity data for the track surface appear as a slender line at the junction with the train wheels. However, many kinds of rail topology need to be located accurately in the railway scene. Thus, the following three problems have to be considered: (a) the number of point clouds in the railway scene is massive, and an appropriate downsampling method is needed to reduce the amount of calculation required for point feature extraction (PFE); (b) as the LiDAR scanning distance increases, the low density of the point clouds and the weak geometric characteristics of the rail pose great difficulty for recognition; (c) because of noise points and complex types of track arrangement (straight, curved, forks and intersections), rail recognition methods need good robustness.

In this study, we propose a method to recognize rails in real time with high accuracy in complex scenes, such as curved rails and crossed rails. A new 3D space division method called pyramid partitioning is proposed to distribute the points evenly in each voxel. At the same time, the points discarded after voxel downsampling are only involved in the calculation performed during position encoding, which greatly reduces the amount of computation in the following steps. A point cloud semantic segmentation neural network is then introduced to adaptively learn the multiscale features of points. It shows good generalization performance and reduces the impact of point cloud noise on the recognition accuracy. Finally, the proposed method is tested on a dataset collected using solid-state LiDAR in a real-world railway environment, including straight, curved, and crossing rails. The major contributions of this study are as follows:

(a) This study proposes a balanced-density voxel downsampling method devised for the solid-state LiDAR scanning scope and solves the difficulties caused by the variable density of point clouds.
(b) This study proposes a PFE module, which can enrich the features of point voxel pairs by introducing the relative location and density information of points.
(c) An effective rail recognition neural network architecture is designed to achieve real-time rail recognition in various scenarios with good performance.

The structure of this paper is as follows: section 2 presents related studies of railway object recognition, including rail extraction based on images, object recognition based on mobile laser scanning data, and point cloud semantic segmentation based on deep learning. Section 3 presents the details of the proposed method, including downsampling with pyramid partitioning, the PFE module, and the multiscale network structure based on 3D sparse convolution. The experimental details and results are provided in section 4 and conclusions are presented in section 5.

2. Related studies

Cameras are widely used in railway systems because of their price advantage and the direct visual feedback they provide for humans. Image-based approaches include feature-matching and edge-detection methods, such as combining local gray-value gradients [1], inverse projective mapping (IPM), and sliding window methods [21, 22]. Nassu and Ukai [8] extracted rails by matching edge features to candidate rail patterns modeled as sequences of parabola segments. The robustness of long-distance rail matching algorithms is severely disrupted by rod-shaped objects, such as trees and telegraph poles. Selver [2] used 2D Gabor wavelets at multiple scales and directions as edge detectors for partitioned video frames. Although the robustness of rail recognition was enhanced, it was difficult to adjust the parameters for various railway scenes, and this method was susceptible to mechanical jitter. Wang et al [22] extracted rails by matching the edge features of IPM-based images to candidate rail patterns, which required a complex matrix solution and feature point selection. With the development of object detection [23, 24] based on deep learning, He et al [25] and Wang et al [26] regarded rail and obstacles as identification targets, and introduced convolutional neural
networks (CNNs) to realize end-to-end identification without manually extracting track features. However, images only have 2D information, which is the main reason for their poor robustness in complex railway environments and adverse weather conditions. Compared to the information provided by cameras, the 3D information provided by LiDAR data greatly improves spatial details.

LiDAR is widely accepted for environmental perception in autonomous vehicles [27, 28]. The corresponding railway applications [29, 30] include the detection, extraction, and modeling of objects and environments based on point clouds. Yang and Fang [31] implemented a moving-window filtering operator to search for elevation jumps in points in order to extract rails. The random sample consensus algorithm has been used to extract rough planes in large-scale road scenes [32]. It suppresses the influence of point cloud noise on feature segmentation with a high iterative calculation efficiency. However, manually setting the plane threshold for modeling in advance leads to its poor robustness. Lou et al [33] used the principal component analysis (PCA) method to cluster the rail points according to the mechanical structure characteristics of the relation between the rail and bed bulge. Rong et al [34] proposed a k-means clustering fused region-growing fitting algorithm to calculate the direction and endpoints of the rail section. Although they randomly sampled point cloud seeds without setting thresholds, their approach was vulnerable to noise points and the initial positions of the seeds during clustering. Huang et al [35] combined the repeated least trimmed squares idea with the smoothing fairness function to filter noise and fit an arc to noisy point clouds. The task of real-time rail recognition needs to take into account the large amount of data, noise points, and various railway environments. However, these manually designed rail feature processing methods only solve specific problems.

Nowadays, point cloud processing methods based on deep learning are continuously being developed. They provide a novel way to identify various objects in point cloud scenes. Qi et al [36] proposed PointNet based on a multi-layer perceptron (MLP) for the point cloud tasks of classification and semantic segmentation. The max-pooling layer adapted to the disorder of point clouds, but it caused the local details of high-dimensional features to be lost. An improved network, PointNet++ [37], added a hierarchical structure that extracted features from small regions and gradually extended to larger regions. Hu et al [17] proposed an efficient semantic segmentation method called RandLA-Net, which is built on multi-randm sampling and a local feature aggregation module to preserve complex local structures by progressively increasing the receptive field for each point. However, the random sampling used by these methods discards essential information about the railway, especially for objects with sparse points. 3D voxel partitioning is an important routine that encodes the features of point clouds. It converts a point cloud into a set of 3D voxels while mainly retaining the 3D geometric information. Huang and You [38] first divided a point cloud into voxels with the same volume for a labeling system. Furthermore, Li et al [39] designed a 3D backbone network by stacking multiple sparse 3D neural networks to save memory and accelerate computation using the sparsity of voxels. However, conventional 3D voxel partitioning methods obtain limited performance gains because they ignore the sparsity of the point clouds. Zhou et al [28] employed cylindrical partitioning for more robust varying density point clouds.

In summary, methods based on neural networks improve the accuracy and robustness of point cloud applications. Moreover, the voxel representation naturally preserves the neighborhood structure of 3D point clouds and allows the direct application of 3D CNNs to achieve good performance. However, low-resolution voxel partitioning results in low accuracy and loss of detail from objects, while high resolution results in high computational costs. An appropriate resolution balances computational costs and acceptable accuracy, both of which are important for rail recognition tasks.

3. Method

The method proposed in this study is designed to recognize rails in 3D point clouds. Its framework is shown in figure 1. It includes four main parts: preprocessing, pyramid partitioning, feature extraction and multiscale 3D CNNs. The details of the method are given in the following subsections. First, invalid points are filtered during preprocessing. Railway 3D point clouds are divided into point–voxel pairs based on pyramid partitioning and downsampling. The PFE module then aggregates neighboring points to obtain local features. The most salient parts of the rail local features are extracted to produce the voxel-wise features using shared MLP and maxpooling layers. Finally, multiscale sparse 3D CNNs predict the class for each voxel. Three-dimensional rail points are segmented from origin point clouds according to the indexes of the point–voxel pairs.

3.1. Point cloud preprocessing

The data source for the proposed approach is a solid-state LiDAR, which is fixed in front of the train. The pitch and horizontal views of the LiDAR scanner are shown in figure 2, in which the dotted lines represent the rays of the LiDAR. In a location with a wide view, the LiDAR can effectively obtain the environmental point cloud data in front of the train.

In the 3D Cartesian coordinate system, taking the LiDAR position as the origin, a frame of the point cloud is represented as follows:

\[
F = \{ p_n(\bar{x}_n, \bar{y}_n, \bar{z}_n), i_n | n = 1, \ldots, N \} \tag{1}
\]

where \( N \) is the number of points, \( p_n \) is the \( n \)th point, \( \bar{x}_n, \bar{y}_n \), and \( \bar{z}_n \) are locations in the three dimensions, respectively, and \( i_n \) is the LiDAR reflection intensity.

As shown in figure 3, it can be observed that a cuboid contains a point cloud and that the area outside the cuboid hardly contains any rail points, but only useless background points from the point of view of rail recognition. Furthermore, there is much noise outside the region, even beyond the effective scanning range of the LiDAR.
Therefore, these invalid points are deleted in the preprocessing step in order to reduce the impact on rail recognition. Moreover, subsequent calculations reduce the consumption without engaging these points. Let $P$ be a set of points within the inspection area, which can be written as:

$$P = \{ p_{i}(x_{i}, y_{i}, z_{i}), i \in F | x_{\min} < x_{i} \leq x_{\max},$$
$$y_{\min} < y_{i} \leq y_{\max}, z_{\min} < z_{i} \leq z_{\max} \}$$

where $x_{\min}, x_{\max}, y_{\min}, y_{\max}, z_{\min},$ and $z_{\max}$ describe the cuboid parameters of the inspection area.

3.2. Pyramid partitioning and down-sampling

The solid-state LiDAR is installed at the head of the train and has a fixed viewing range. As shown in figure 4(a), the LiDAR scanning area is pyramid-shaped with limited horizontal and pitch angles. Figures 4(b) and (c) represent a point cloud after pyramid partitioning and down-sampling, respectively. Due to factors such as distance, occlusion, and relative position, the density of the point clouds in the nearby region is much larger than in the far region. In previous studies, the point clouds were divided into cuboid voxels [40], which led to disequilibrium in the point density of the voxels. A new 3D polar coordinate-based pyramid partitioning method is proposed to increase the voxel size in order to cover the remoter regions of the railway environment. The voxel volume is positively correlated with the distance range by considering the Euclidean distance, pitch, and horizontal angle as the resolution. The pyramid partitioning utilizes an increasing grid size to cover the farther-away 3D area. Thus, it evenly distributes the points across different voxels and produces a more balanced representation of varying densities and LiDAR scanning distances. The specific steps of our pyramid partition method include coordinate conversion, partitioning, and point grouping.
Given a point cloud, assume that the 3D space range is set to \( \alpha_{\min}, \alpha_{\max}, \beta_{\min}, \beta_{\max}, r_{\min}, \) and \( r_{\max}, \) which represent the minimum and maximum of the pitch angle, horizontal angle, and distance, respectively. The resulting 3D voxel grid then has a size of \( \Delta \alpha, \Delta \beta, \) and \( \Delta r. \) Therefore, the point cloud is partitioned into voxels, which can be written as follows:

\[
L = \frac{\alpha_{\max} - \alpha_{\min}}{\Delta \alpha}, \quad W = \frac{\beta_{\max} - \beta_{\min}}{\Delta \beta}, \quad H = \frac{r_{\max} - r_{\min}}{\Delta r},
\]

where \( L, W \) and \( H \) are the numbers of voxels in the \( \alpha, \beta, \) and \( r \) dimensions, respectively.

3.2.2. Point grouping. The points are grouped according to the pyramid voxels that they reside in. Let \( V^P \) represent a point cloud consisting of voxels, which can be discretized as follows:

\[
V^P = \left\{ \begin{array}{c} l = 1, 2, \ldots, L \\ w = 1, 2, \ldots, W \\ h = 1, 2, \ldots, H \end{array} \right\},
\]

where \( v_{l,w,h}^P \) is a voxel with many points. \( v_{l,w,h}^P \) can be written as follows:

\[
v_{l,w,h}^P = \left\{ \begin{array}{c} p_{n}(\alpha_n, \beta_n, r_n), n \left[ \frac{\alpha_n - \alpha_{\min}}{\Delta \alpha} \right] = l, \\ \left[ \frac{\beta_n - \beta_{\min}}{\Delta \beta} \right] = w, \left[ \frac{r_n - r_{\min}}{\Delta r} \right] = h \end{array} \right\}
\]

here, \([\cdot]\) represents a rounding operation.

The LiDAR 3D point cloud is sparse and has variable point density due to factors such as scan distance, occlusion, and non-uniform sampling. Thus, a point cloud is partitioned into many voxels that contain variable numbers of points. Typically, a railway point cloud is composed of 30 k–90 k points. The downsampling operation retains one point per voxel. This reduces the redundancy of the point cloud information and the amount of calculation required in the PFE process. The voxel–point pair can be written as:

\[
v_{l,w,h} = \arg \min \left( \| v_{l,w,h}^P - v_{l,w,h}^P \| \right),
\]

where \( v_{l,w,h} \) represents the point–voxel pair, \( v_{l,w,h}^P \) is the current voxel location in the negative direction, and \( \| \cdot \| \) is the Euclidean distance between two points. Note that the label must be determined when a voxel simultaneously includes rail and background points. If the number of rail points in the voxel is greater than the number of background points, the label of the current voxel is ‘rail’; otherwise, the label is ‘background’.

3.3. Point feature extraction

A new PFE module is proposed for each point to aggregate the geometric information from nearby points. First, the input of the module is a point with coordinates and feature information, which is downsampled from the origin point cloud by pyramid partitioning. The adjacent points of the input point
are then found from the origin points using the \( k \)-nearest neighbors (KNN) algorithm. The relative position coordinates are encoded according to the \( x-y-z \) coordinates so that each point carries its relative spatial position and geometric features. Otherwise, there is a significant difference in the density of point clouds collected by LiDAR in the railway scene. Thus, the Gaussian density function is introduced as an explicit feature to give each point a density feature. Finally, the whole network effectively learns the complex local structure in high dimensions through an MLP. The module is shown in figure 6, and the specific process is described in detail below.

First, we encode the point locations. A point contains \( x-y-z \) coordinates and other features, such as reflection intensity and 3D polar coordinates. The original point cloud is regarded as the support set, and the downsampled point clouds in section 3.2 are regarded as the query set for the search operation. For the \( n \)-th point \( p_n \) in a frame point cloud, the nearest \( K \) point set \( \{p_{n1}\,\ldots\,p_{nk}\} \) is found using the KNN algorithm. The relative point location for the nearest \( k \)-th point is encoded as follows:

\[
\hat{f}^k_n = (p_n \oplus p_{nk} \oplus (p_{nk} - p_n) \oplus f^k_n),
\]

where \( p_n \) and \( p_{nk} \) are the \( x-y-z \) coordinates of the points, \( f^k_n \) are the features (such as intensity and polar coordinate) of the \( k \)-th nearest point, and \( \oplus \) is the concatenation operation.

We then consider that the sparse and variable-density point clouds differ from the ordered arrangement of pixels in 2D images. The density of point clouds plays a crucial role in railway information. In this paper, a Gaussian distribution probability density function is introduced to determine the density of each point. Using the Euclidean distances of the near points obtained by the KNN algorithm, the density between the central point and the \( k \)-th nearest point is calculated as follows:

\[
\rho^k_n = \frac{e^{-r^2/2\sigma^2}}{\sqrt{2\pi\sigma}}
\]

where \( \rho^k_n \) represent the density of \( p_{nk}^k \), \( r \) is the Euclidean distance between \( p_n \) and \( p_{nk} \), and \( \sigma \) is a hyperparameter known as the receptive field. The density is then concatenated into \( \hat{f}^k_n \):

\[
\hat{f}^k_n = \rho^k_n \oplus \tilde{f}^k_n.
\]

The geometric information of the \( K \) nearest points is \( \tilde{F}_n = \{\hat{f}^1_n, \hat{f}^2_n, \ldots, \hat{f}^k_n, \ldots, \hat{f}^K_n\} \). Point-wise features are then obtained from adjacent points using the MLP. The most significant features \( \tilde{F}_n \) are now extracted from the adjacent points by the max-pooling layer, which can be written as:

\[
F_n = \max \left( \text{MLP} \left( \tilde{F}_n \right) \right).
\]

Eventually, the output of the PFE module is a high-rank feature that explicitly encodes the local geometric structures for the central point and the voxel–point pair. After these steps, we define the 3D pyramidal feature representation as \( \mathbb{R} \in C \times H \times W \times L \), where \( C \) denotes the feature dimension of the voxel, and \( H, W, \) and \( L \) represent the pitch angle, horizontal angle, and distance, respectively.

### 3.4 Multiscale sparse 3D CNNs

In the railway scene point clouds, the rail presents a continuous and equal width strip area, and the beginning of the rail is located at the center of the LiDAR view. Inspired by the application of U-net [41], which offers good performance in 2D image semantic segmentation, a multilayer residual network structure is designed to fuse the features of the point clouds. Figure 7 shows the proposed rail recognition method, whose semantic segmentation network follows the widely used encoder–decoder architecture with skip connections. The encoding and decoding layers include four down-sampling modules and four upsampling modules based on the block structure of Mobilenet [42]. After partitioning, there are many empty voxels. Thus, sparse 3D convolution [43] is used in the \( L, W, \) and \( H \) dimensions of the point clouds to construct the down-sampling and upsampling modules. Moreover, recent literature [28] also shows that low-rank kernels build rich high-rank context, according to tensor decomposition theory [44]. In this way, three rank-1 sparse 3D convolution kernels predict the semantic label for each voxel.
4. Experimental details

In this section, we first introduce the dataset, including point cloud collection, samples, and labeling. Second, the experimental settings and metrics are described, and we present the testing of our method in terms of its real-time performance and accuracy for rail recognition using this dataset. Third, ablation experiments are described that verify the effectiveness of each innovation used in this method. Finally, the experimental results show that our work accurately recognizes the rail and achieves real-time performance.

4.1. Data preparation

Since there are almost no track scenes in the existing public datasets, we collected data from China Railway’s test environment and manually created the datasets used for training and validation. The point cloud data of the railway were collected using a solid-state LiDAR, as shown in figure 8, which was mounted at the head of a train with a forward view.

The hardware equipment used to collect the point clouds was the innovusion Jaguar Gen-1 LiDAR; its parametric information is shown in table 1.

| Parameter          | Value                                   |
|--------------------|-----------------------------------------|
| Scanning frequency | 6–10 Hz, adjustable                      |
| Pitch angle        | 40°                                     |
| Horizontal angle   | 65°                                     |
| Range accuracy     | <3 cm                                   |
| Detection range    | 200 m                                   |

The dataset contains more than 200 million points and 6000 frames. Each point has $x$–$y$–$z$ and intensity information, representing the location in 3D coordinates and the reflection intensity of the object, respectively. Based on the characteristics of the railway environment and the rails, the dataset was manually classified into six scenes, namely straight rails, curved rails, intersection rails, multirails, departure, and arrival. Examples of railway 3D point clouds are shown in figure 9.

The dense point-wise annotations for the rails were labeled manually. The standard rail spacing is 1435 mm. The rail area is extended to a width of 1550 mm during labeling in order to obtain the integrity of the rail and the geometric characteristics of the upper and lower drop. In order to reduce dataset redundancy and the workload generated by manual annotation, continuous point cloud frames were downsampled to five frames per second. Table 2 shows nine sequences of detailed railway data information, including point cloud frames, point quantities, and rail types. The number of rail points in the railway scenes is one-tenth of all railway points. In the open scenes, a frame of the point cloud is composed of about 40 000 points. However, in dense scenes, such as the
4.2. Experimental setup

All the experiments were performed on a computer with a 6-core 3.4 GHz CPU and an NVIDIA 2080Ti GPU. The deep learning environment consists of the ubuntu-18.04 system, pytorch-1.6, spconv-1.2.1, etc. The KNN algorithm is implemented in C++, which obviously saves memory.

The major parameter settings used in the pyramid and downsampling processes are listed in table 3. Data enhancement was then used to reduce the phenomenon of overfitting during network model training. The point clouds were rotated $\pm 5^\circ$ around the $z$-axis and scaled in three dimensions by $\pm 0.05$ times. In addition, Gaussian noise was added to the point clouds. During model training, zero to six sequences were used as the training dataset. The dynamic learning rate was set to reduce by ten times every ten epochs, and a cross-entropy loss function was introduced to estimate the error.
Table 3. Parameter set used in the experiments.

| Parameter          | Value(s)                                      |
|--------------------|-----------------------------------------------|
| Inspection area    | [6.5, 70, −30, 30, −5.5, 7]                  |
| Grid size of voxel | [0.3°, 0.5°, 0.5 m]                           |
| Receptive field    | 0.5                                           |
| MLP structure      | [13, 64, 64, 16]                              |
| K                  | 4                                             |
| Learning rate      | 0.001                                         |
| Epoch              | 60                                            |

Table 4. The calculated IoU, precision, and recall for the test dataset.

| Sequence | IoU (%) | Precision (%) | Recall (%) |
|----------|---------|---------------|------------|
| 7        | 96.03   | 98.28         | 97.66      |
| 8        | 93.88   | 96.34         | 97.35      |
| 9        | 95.52   | 98.60         | 96.83      |
| Mean     | 95.14   | 97.74         | 97.28      |

4.3. Metrics

The evaluation metrics employed in our experiments are the methods commonly used in point cloud semantic segmentation [10, 17], which are as follows:

\[
\text{IoU} = \frac{TP}{TP + FP + FN}
\]

\[
\text{precision} = \frac{TP}{TP + FP}
\]

\[
\text{recall} = \frac{TP}{TP + FN}
\]

where IoU is the intersection over the union, TP denotes the number of true positive rail points, FP denotes the number of false positive points (for which background points in the ground truth are predicted to be rails), and FN denotes the number of false negative points (rail points in the ground truth that are predicted to be background).

4.4. Results evaluation

After 60 epochs of training, the model was verified using sequences 7, 8, and 9, and the results are shown in table 4. Note that sequence 9 is straight rail, and sequences 7 and 8 are curved rail. The mean IoU of rail recognition is more than 95% for the proposed method.

In addition, the representative semantic segmentation methods Pointnet [36] and RandLA-Net [17] were employed to evaluate the accuracy using our datasets and further compared with our methods. The results are shown in table 5, which shows that our method achieves the highest IoU, which is 95.14%. Although the time cost and GPU memory usage are less than optimal, they are within the acceptable range. The time cost of our method is more than those of Pointnet and RandLA-Net: the processing time of the proposed method is 80.9 ms, which is faster than the LiDAR collection speed.

Table 5. Comparison of the rail recognition results.

| Method        | IoU (%) | Time (ms) | GPU memory (MB) |
|---------------|---------|-----------|-----------------|
| Pointnet      | 71.88   | 17.6      | 8203            |
| RandLA-Net    | 83.42   | 34.5      | 1175            |
| Our method    | 95.14   | 80.9      | 1669            |

Sample visualizations of rail recognition results are shown in figure 10, where gray, blue, red, and green points represent background point clouds, and TP, FN, and FP represent rail points, respectively. It can be observed that Pointnet only recognizes the straight track well, and the curved and crossed rails show a lot of error points, which means that Pointnet has poor robustness for complex rail topologies. RandLA-Net uses multilayer random downsampling, and it can be seen that its point clouds are obviously sparser than the others. Although the local spatial encoding and attentive pooling modules improve its performance for curved and crossing rails, a few FP points are present on both sides of the rail. It can be seen that the points corresponding to straight rails, curved rails, and crossed rails can be accurately identified by our method, which is the closest to the ground truth. Enlarged example visualizations are shown in figure 11. The blue points are correctly recognized, occupying 95.14% of the rail, which confirms the good performance of the proposed method. It can be observed that a few error points are present in the middle of the rail; this occurs because the PFE module performs a full extraction of the local geometric features of the rail area.

The red and green points indicate the error points, which are distributed on both sides and at the end of the rail. The point
clouds on both sides of the rail are at the junction of the subgrade and the rail, which contain both rail geometry information and roadbed information. Thus, the points on both sides of the rail are easily misidentified. In addition, one of the components in the proposed method is pyramid partitioning, which may cause points with different categories to be divided into the same cell, leading to information loss, especially at the junction of the rail and the background.

Due to the scanning range limit of LiDAR, the points at the end of the rail are too sparse to recognize, and the geometric features between the point clouds are very weak. Thus, there are a few crossed rails with error points.

From figure 12, it can also be observed that the time cost has a positive correlation with the point quantity in a frame point cloud. In the actual railway point cloud collection, the number of points in a frame reaches 70 000 in the densest scenes, such as train arrival and departure, and the number of points is more than 25 000 in sparse scenes, such as single rails and plain scenes. The experimental results show that the average processing time for a frame of the point cloud is 80.9 ms. When the train runs at 140 km s$^{-1}$, the distance it covers during the processing time of the algorithm is 3.2 m. The maximum scanning frequency of the Jaguar Gen-1 LiDAR is 10 Hz, as shown in table 1. This means that the time required for the LiDAR to collect a frame of the point cloud is fixed at 100 ms by the hardware attributes. This indicates that the rail position of the current frame has been recognized when the LiDAR begins to collect the next frame of the point cloud. Thus, each frame of point cloud data can be processed in real-time, and our method can satisfy the requirement for real-time performance.

4.5. Ablation study

In this section, we describe additional experiments that were conducted to evaluate the effectiveness of each aspect of our method. The results obtained using the railway dataset are reported in table 6. Our method consists of an algorithm that uses pyramid partitioning, a PFE module and multiscale sparse 3D convolution networks. First, the pyramid partitioning is replaced with cube partitioning to divide the railway space into voxels. It can be observed that pyramid partitioning performs better than cube partitioning, showing a 3.9% IoU gain. Second, we retain the integral point clouds without down-sampling. This almost doubles the time consumption and approximately triples the GPU memory usage. Although the IoU rises by about 0.6%, this approach hardly meets the real-time requirements. Third, the PFE module also significantly

![Figure 11. Enlarged results of rail recognition, including nine sub-pictures. The sub-pictures in the top, middle and bottom rows correspond to the straight, curved, and crossed rails in figure 10, respectively. The sub-figures in the left, middle, and right columns represent the near, medium-distance, and distant rails, respectively.](image)

![Figure 12. Relationship between our algorithm’s time cost and point quantity in a frame point cloud.](image)

| Method                      | IOU (%) | Time (ms) | GPU memory (MB) |
|-----------------------------|---------|-----------|-----------------|
| Cube partitioning           | 91.23   | 73.9      | 1431            |
| Removal of downsampling     | 95.75   | 135.6     | 4163            |
| Replacing PFE with MLP      | 92.50   | 67.1      | 1403            |
| Removing density            | 94.02   | 79.5      | 1607            |
| Our method                  | 95.14   | 80.9      | 1669            |
boasts the performance by about 3% compared to the MLP, which demonstrates that it is essential to aggregate the local information of the point clouds. Moreover, density estimation also improves the recognition accuracy by about 1.1%. Overall, it can be seen that both the voxel downsampling based on pyramid partitioning and the PFE module are crucial for rail recognition; they not only improve the performance but also reduce the calculation required.

5. Conclusions

Accurate 3D rail location is a necessary part of railway operational systems. In this paper, a new semantic segmentation method based on point clouds is proposed for rail recognition. Pyramid partitioning matches the 3D scanning area of the LiDAR, allowing disordered point clouds to be divided into regular voxels. It balances the density expression for the point clouds. The voxel downsampling that takes place after pyramid partitioning reduces the GPU memory footprint by reducing the data volume input to the CNNs. The PFE module aggregates valuable railway information to point–voxel pairs and improves the accuracy of rail recognition. The multiscale sparse 3D CNNs adaptively identify railway scenes with multiple topologies. The fast inspection speed (80.9 ms per frame) and the high IOU (95.14%) of the rail recognition performed in our experiments demonstrate the efficiency of our methods.

In a future study, we will examine point clouds scanned by LiDAR, which may contain motion distortion when the train runs at high speed. The time-dimension information can be fused into point clouds to solve the motion distortion problem and further increase the features used for rail recognition. In addition, data from both spatial and temporal dimensions can also be used to improve rail recognition accuracy. Rail curve fitting and template-matching methods may be good ways to predict distant rail locations.

Data availability statement

The data generated and/or analyzed during the current study are not publicly available for legal/ethical reasons but are available from the corresponding author on reasonable request.

Acknowledgments

The research is supported by the National Key Research and Development Program of China (Grant No. 2018YFB1308400), and National Natural Science Foundation of Zhejiang Province, China (Grant No. LY21F030018).

ORCID ID

Weiqi He https://orcid.org/0000-0002-8128-0870

References

[1] Wohlfell J 2011 Vision based rail track and switch recognition for self-localization of trains in a rail network 2011 IEEE Intelligent Vehicles Symp. (IV) (IEEE) pp 1025–30
[2] Selver M A, Er E, Belenlioglu B and Soyaslan Y 2016 Camera based driver support system for rail extraction using 2D Gabor wavelet decompositions and morphological analysis 2016 IEEE Int. Conf. on Intelligent Rail Transportation (ICIRT) (IEEE) pp 270–5
[3] Lenior D, Janssen W, Neerinck M and Schreiber K 2006 Human-factors engineering for smart transport: decision support for car drivers and train traffic controllers Appl. Ergon. 37 479–90
[4] Huang W, Zhang W, Du Y, Sun B, Ma H and Li F 2013 Detection of rail corrugation based on fiber laser accelerometers Meas. Sci. Technol. 24 094014
[5] Kastrinaki V, Zervakis M and Kalaitzakis K 2003 A survey of video processing techniques for traffic applications Image Vis. Comput. 21 359–81
[6] Ruder M, Mohler N and Ahmed F 2003 An obstacle detection system for automated trains IEEE IV 2003 Intelligent Vehicles Symp. Proc. (Cat. No. 03TH8653) (IEEE) pp 180–5
[7] Maire F and Bigdeli A 2010 Obstacle-free range determination for rail track maintenance vehicles 2010 11th Int. Conf. on Control Automation Robotics & Vision (ISV) pp 2172–8
[8] Nassu B T and Utkai M 2012 A vision-based approach for rail extraction and its application in a camera pan–tilt control system IEEE Trans. Intell. Transp. Syst. 13 1763–71
[9] Selver A M, Ataç E, Belenlioglu B, Dogan S and Zoral Y E 2017 Visual and LiDAR data processing and fusion as an element of real time big data analysis for rail vehicle driver support systems Innovative Applications of Big Data in the Railway Industry (Engineering Science Reference) p 395
[10] Geiger A, Lenz P, Stiller C and Urtasun R 2013 Vision meets robotics: the kitti dataset Int. J. Robot. Res. 32 1231–7
[11] Hackel T, Savinov N, Ladic V, Wegner J D, Schindler K and Pollefeys M 2017 Semantic3D.net: a new large-scale point cloud classification benchmark (arXiv:1704.03847)
[12] Yi B, Yang Y, Yi Q, Dai W and Li X 2017 Novel method for rail wear inspection based on the sparse iterative closest point method Meas. Sci. Technol. 28 125201
[13] Taheri Andani M, Peterson A, Munoz J and Ahmadian M 2018 Railway track irregularity and curvature estimation using doppler LiDAR fiber optics Proc. Inst. Mech. Eng. F 232 63–72
[14] Arastounia M 2015 Automated recognition of railroad infrastructure in rural areas from LiDAR data Remote Sens. 7 14916–38
[15] Daoust T, Pomerleau F and Barfoot T D 2016 Light at the end of the tunnel: high-speed lidar-based train localization in challenging underground environments 2016 13th Conf. on Computer and Robot Vision (CRV) (IEEE) pp 93–100
[16] Nezhadarya E, Taghavi E, Razani R, Liu B and Luo J 2020 Adaptive hierarchical down-sampling for point cloud classification Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition pp 12956–64
[17] Hu Q et al 2020 RandLA-net: efficient semantic segmentation of large-scale point clouds Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition pp 11108–17
[18] El-Sayed E, Abdel-Kader R F, Nashaat H and Marei M 2018 Plane detection in 3D point cloud using octree-balanced density down-sampling and iterative adaptive plane extraction IET Image Process. 12 1595–605
[19] Bolkas D and Martinez A 2018 Effect of target color and scanning geometry on terrestrial LiDAR point–cloud noise and plane fitting J. Appl. Geod. 12 109–27
[20] Duan Y, Yang C, Chen H, Yan W and Li H 2021 Low-complexity point cloud denoising for LiDAR by PCA-based dimension reduction Opt. Commun. 482 126567
[21] Wang Z et al 2015 An inverse projective mapping-based approach for robust rail track extraction 2015 8th Int. Congress on Image and Signal Processing (CISP) (IEEE) pp 888–93
[22] Wang Z et al 2016 Geometry constraints-based visual rail track extraction 2016 12th World Congress on Intelligent Control and Automation (WCICA) (IEEE) pp 993–8
[23] Duan K, Bai S, Xie L, Qi H, Huang Q and Tian Q 2019 Centernet: keypoint triplets for object detection Proc. IEEE/CVF Int. Conf. on Computer Vision pp 6569–78
[24] He K, Gkioxari G, Dollár P and Girshick R 2017 Mask r-cnn Proc. IEEE Int. Conf. on Computer Vision pp 2961–9
[25] He D, Li K, Chen Y, Miao J, Li X, Shan S and Ren R 2021 Obstacle detection in the dangerous area of railway track based on convolutional neural network Meas. Sci. Technol. 32 105401
[26] Wang Z, Wu X, Yu G and Li M 2018 Efficient rail area detection using convolutional neural network IEEE Access 6 77656–64
[27] Thomas H, Qi C R, Deschaud J-E, Marcotegui B, Goulette F and Guibas L J 2019 Kpconv: flexible and deformable convolution for point clouds Proc. IEEE/CVF Int. Conf. on Computer Vision pp 6411–20
[28] Zhou H et al 2020 Cylinder3D: an effective 3D framework for driving-scene lidar semantic segmentation (arXiv:2008.01550)
[29] Sahebdivani S, Areli H and Maboudi M 2020 Rail track detection and projection-based 3D modeling from UAV point cloud Sensors 20 5220
[30] Chen C, Zhang T, Yan K, Li S and Jin G 2020 A rail extraction algorithm based on the generalized neighborhood height difference from mobile laser scanning data SPIE Future Sensing Technologies vol 11525 (International Society for Optics and Photonics) p 115250N
[31] Yang B and Fang L 2014 Automated extraction of 3D railway tracks from mobile laser scanning point clouds IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens. 7 4750–61
[32] Qiu K, Sun K, Ding K and Shu Z 2016 A fast and robust algorithm for road edges extraction from LiDAR data Int. Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences vol 41
[33] Lou Y, Zhang T, Tang J, Song W, Zhang Y and Chen L 2018 A fast algorithm for rail extraction using mobile laser scanning data Remote Sens. 10 1998
[34] Zou R, Fan X, Qian C, Ye W, Zhao P, Tang J and Liu H 2019 An efficient and accurate method for different configurations railway extraction based on mobile laser scanning Remote Sens. 11 2929
[35] Huang S, Ming C, Lu S, Chen S and Zha Y 2021 A novel algorithm: fitting a spatial arc to noisy point clouds with high accuracy and reproducibility Meas. Sci. Technol. 32 085004
[36] Qi C R, Su H, Mo K and Guibas L J 2017 Pointnet: deep learning on point sets for 3D classification and segmentation Proc. IEEE Conf. on Computer Vision and Pattern Recognition pp 652–60
[37] Qi C R, Yi L, Su H and Guibas L J 2017 Pointnet++: deep hierarchical feature learning on point sets in a metric space (arXiv:1706.02413)
[38] Huang J and You S 2016 Point cloud labeling using 3D convolutional neural network 2016 23rd Int. Conf. on Pattern Recognition (ICPR) (IEEE) pp 2670–5
[39] Shi S et al 2020 PV-RCNN: point-voxel feature set abstraction for 3D object detection Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition pp 10529–38
[40] Zhou Y and Tuzel O 2018 Voxelnet: end-to-end learning for point cloud based 3D object detection Proc. IEEE Conf. on Computer Vision and Pattern Recognition pp 4490–9
[41] Ronneberger O, Fischer P and Brox T 2015 U-net: convolutional networks for biomedical image segmentation Int. Conf. on Medical Image Computing and Computer-assisted Intervention (Springer) pp 234–41
[42] Sandler M, Howard A, Zhu M, Zhmoginov A and Chen L-C 2018 MobileNetV2: inverted residuals and linear bottlenecks Proc. IEEE Conf. on Computer Vision and Pattern Recognition pp 4510–20
[43] Graham B 2015 Sparse 3D convolutional neural networks (arXiv:1505.02890)
[44] Chen W et al 2020 Tensor low-rank reconstruction for semantic segmentation European Conf. on Computer Vision (Springer) pp 52–69