Hindawi
Wireless Communications and Mobile Computing
Volume 2021, Article ID 8351674, 10 pages
https://doi.org/10.1155/2021/8351674

Research Article

Subway Obstacle Perception and Identification Method Based on Cloud Edge Collaboration

Li Feng,[1] **Ronghui Yan**,[1] **Guangping Liu**,[2] **and Chen Shao**[3]

[1]*Department of Rail Transit Engineering, City College of Suzhou, Suzhou, Jiangsu, China*
[2]*School of Urban Rail Transportation, Soochow University, Suzhou, Jiangsu, China*
[3]*Suzhou Rail Transit Group Co., Ltd., Suzhou, Jiangsu, China*

Correspondence should be addressed to Li Feng; wzj022@suda.edu.cn

Received 1 August 2021; Accepted 6 September 2021; Published 15 October 2021

Academic Editor: Zhihan Lv

Copyright © 2021 Li Feng et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The traditional analysis method of train obstacle uses isomorphic sensors to obtain the state information and completes detection and identification analysis at the remote end of a network. A single data sample and more processing links will reduce the accuracy and speed analysis for subway encountering obstacles. To solve this problem, this paper proposes a subway obstacle perception and identification method based on cloud edge cooperation. The subway monitoring cloud platform realizes the training and construction of a detection model, and the network edge side completes the situation awareness of track state and real-time action when the train encounters obstacles. Firstly, the railroad track position is detected by cameras, and subway running track is identified by Mask RCNN algorithm to determine the detection area of obstacles in the process of subway train running. At the edge of network, the feature-level fusion of data collected by sensor cluster is carried out to provide reliable data support for detection work. Then, based on the DeepSort and YOLOv3 network models, the subway obstacle detection model is constructed on the subway monitoring cloud platform. Moreover, a trained model is distributed to the network edge side, so as to realize the fast and efficient perception and action of obstacles. Finally, the simulation verification is implemented based on actual collected datasets. Experimental results show that the proposed method has good detection accuracy and efficiency, which maintains 98.9% and 1.43 s for obstacle detection accuracy and recognition time in complex scenes.

1. Introduction

Urban rail transit is one of the most popular means of transportation for urban people, and its development speed is also very rapid [1]. Among them, the technology of fully automatic driverless metro train is a hot research content of urban rail transit [2–4], and its most key link is the rapid state analysis and emergency treatment when the train encounters obstacles.

According to the statistics of rail train operation safety accidents in recent years, there are many factors that will affect the subway train operation safety, mainly including management level, equipment reliability, and rail roadblocks [5, 6]. At the same time, because the subway traffic environment is mostly closed and low, the operating environment and lighting conditions are not enough to support the traditional detection methods to realize the identification of track obstacles. In addition, the fast running speed of subway trains poses a certain challenge to the safe and stable operation of subway, resulting in potential safety hazards during the running of trains [7]. Therefore, it is particularly important to develop a reasonable and efficient subway obstacle perception and recognition method.

The traditional method adopts contact detection method, and the train will be braked urgently only after the obstacle collides with the detection beam. The contact obstacle detection system can accurately find the target ahead and stop the train. But at the same time, the target was discovered, the rail train stopped running. The train may also be subject to a greater impact, so that the safety of subway and passengers cannot be guaranteed [8, 9].

With the development of sensor technology, state data acquisition is based on the installation of detection devices on specific tracks [10]. For example, a certain radar or RF device is installed at the front side of the subway train, which can collect the status data of the running track before not contacting the obstacles, upload it to the monitoring system platform for analysis and decision-making, realize effective and stable braking and deceleration, and greatly improve the operation safety.

However, there are still some problems in the noncontact train detection method: First, the detection device is a state acquisition device. Because of the differences in the nature of the sensors and the installation environment, simultaneous interpreting of objects by a single sensor can not guarantee the reliability of data and affect the accuracy of detection [11]. Second, there are too many links in obstacle identification and analysis. Relying on the detection and analysis of subway monitoring platform can improve the accuracy to a certain extent, but it can not meet the requirements of track foreign object identification for analysis speed.

2. Related Work

Due to the limited line of sight in the subway operation environment, it is sometimes difficult to distinguish the foreign objects in the track. The safety accidents caused by collision with obstacles often have the characteristics of large loss and serious harm. Therefore, it is particularly important to develop a fast and accurate obstacle autonomous recognition method for the safe operation of locomotive in case of obstacles.

The traditional obstacle detection method uses the contact obstacle detection system. The system installs a detection beam on the bottom of the train head and realizes the detection function when detection beam touches obstacles. The sensor detects deformation of the beam, and the train system prompts the train to brake train urgently according to sensors [12]. However, the contact obstacle detection system must break the train when detection beam touches obstacles. The speed of subway train is very fast. Although obstacles are detected, it will also cause damage to the train and cannot ensure the safety of train.

With the development of sensor technology, rail trains began to use radar detection, radiofrequency detection, or stereo camera to detect foreign objects. However, any single-sensor technology has shortcomings: for example, the detection effect of infrared camera is very poor when the temperature is high, the stereo camera can hardly collect data in bad weather, and the information collected by radar is also poor when the external environment is poor. A variety of heterogeneous sensors form sensor clusters at the edge of the network and fuse the actual data samples with each other, which can overcome the shortcomings of single-sensor technology, improve the detection results of the system, and support the stable operation of the train.

Thanks to the development of intelligent algorithms and big data technology, deep network technology is applied to the analysis of subway operation status. Based on the state data uploaded by sensors at the edge of network, through the continuous training and learning of multilayer network structure [13], the noncontact perception and recognition of obstacles in the track is realized. Reference [14] proposed a deep learning segmentation algorithm for railway detection based on RailNet network model. The multilayer network structure can be used to continuously extract the characteristics of a sample dataset to achieve noncontact recognition of foreign objects. Reference [15] improved the deep convolutional neural network (CNN) to construct a subway operation detection network. Besides, it used transfer learning technology to train facility images in subway tunnels to improve the performance of obstacle model detection. Reference [16] proposed a CNN-based railway area detection method to achieve pixel-level classification of track areas. Reference [17] combined the semantic segmentation algorithm with CNN to realize the accurate recognition of track area and forward train. Reference [18] introduced LeNet-5CNN to realize rail transit obstacle detection and provide intelligent early warning information for the train control system. The above method can realize obstacle perception and identification before the train comes into contact with obstacles. However, only relying on the single-state data uploaded by sensors to realize decision analysis has the problem of single unreliable data sample and the danger of missing valid data. On the other hand, overreliance on the subway monitoring cloud platform for detection can improve accuracy, but the real-time performance is not high [19]. It may lead to a slower braking action when encountering obstacles and the risk of car crashes and deaths.

To solve the above problems, under cloud edge collaboration architecture, this paper proposes a subway obstacle perception and identification method using deep learning. The innovations of this paper are as follows:

(1) Propose a track area identification method based on the Mask RCNN network model to meet the demand for autonomous and efficient identification of train running tracks in actual scenarios

(2) Overcome the incomprehensiveness of single-sensor data collection, realize feature-level data fusion of sensor cluster data at the edge of network to enhance the credibility of analysis data, and then improve the reliability of entire detection network system

(3) Based on reliable dataset support, use the YOLOv3 and DeepSort algorithms to train and establish detection network on the cloud analysis platform. At the cloud edge, the detection network is used to realize rapid perception and control, which greatly improves the safety and reliability of train operation

3. Method Framework

3.1. Overall Framework. The method architecture proposed in this paper combines cloud edge (metro monitoring cloud platform) decision-making, and edge side (train) monitoring. Under the condition of mutual cooperation between

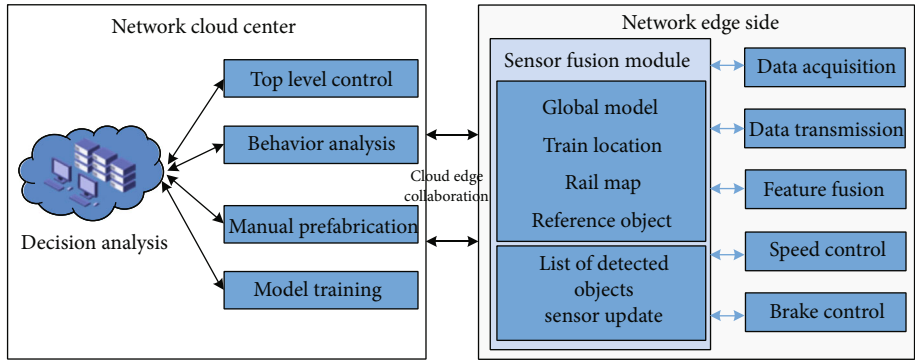


Figure 1: Overall framework of the proposed method.

the cloud edge and edge side, the efficient perception and identification of subway obstacles can be realized to support the safe and reliable operation of rail subway [20]. Figure 1 is the overall block diagram of the proposed method.

As shown in Figure 1, the method proposed in this paper supports the reliable operation of rail subways by cloud edge decision analysis-edge real-time control of cloud edge collaboration. First, the position of rails is detected by cameras. Based on deep learning for rail identification, we determine the detection area of obstacles in the process of subway trains. The edge layer is responsible for fusing multisensor data and executing the trained detection model issued by the subway monitoring cloud platform to detect obstacles in real time. The subway monitoring cloud platform is responsible for using deep learning methods to train and learn the track environment and obstacle characteristics in various scenarios, generate detection models, and periodically send them to the edge layer for execution.

3.2. Rail Perception Based on Deep Learning. Traditional analysis methods have certain limitations, and it is difficult to support the analysis requirements for autonomous rail identification and dangerous area division of rail trains. In this paper, the position of railroad track is detected by cameras, and based on the deep learning algorithm on cloud edge, the track area of subway train is drawn.

Firstly, the features of rail training samples are extracted based on CNN; then, the region proposal network (RPN) was used for training. Mask RCNN is responsible for rail detection and identifying dangerous areas [21]; as shown in Figure 2, a regional candidate network is selected to extract candidate frames in order to improve efficiency.

RPN network is a full convolution network specially used to extract candidate regions. It processes the previously extracted feature map, looks for candidate frames that may contain the target region, and predicts the category score of each frame.

Using CNN to directly generate candidate area frames is the core idea of the RPN network, which scans images by the sliding of window. The RPN network produces two outputs for each anchor point. One is the category of anchor points, for all anchor point boxes generated. After screening and filtering, the SoftMax classification function is used to judge whether the anchor point

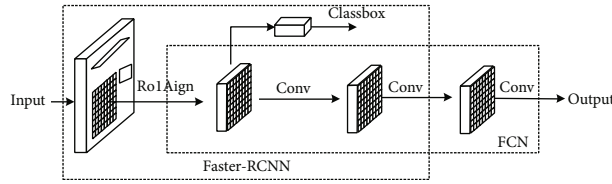


Figure 2: Mask RCNN structure.

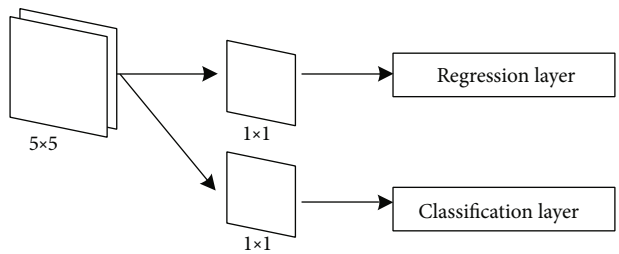


Figure 3: RPN partial structure.

belongs to the foreground or the background; that is, it is a railroad track or not a railroad track, so as to realize the identification of the railroad track. At the same time, the other is frame fine adjustment, which uses the bounding box regression function to correct the anchor point frame to form a more accurate candidate frame. After being extracted by CNN, the obtained feature map is input into RPN network, as shown in Figure 3.

The input of the RPN network is a picture of any size, and the network output is a series of candidate frames for different sizes. And the RPN network generates two outputs for each candidate frame, which are the probability value of identifying the target object and position information of target object equivalent to pictures. RPN network uses a 5×5 sliding window and the output of CNN to complete the convolution operation, and after the convolution operation, a low-dimensional matrix is obtained. Each anchor point can generate fifteen candidate boxes, and these fifteen candidate boxes are input to the regression layer and classification layer, which are used for bounding box regression and classification, respectively. The schematic diagram of the RPN structure is shown in Figure 4, where the candidate frame $k = 15$.

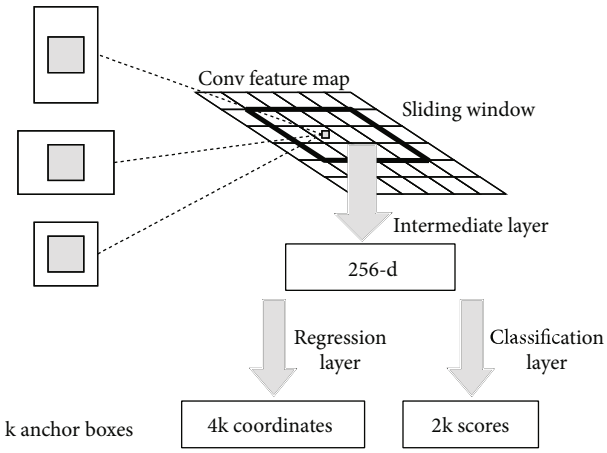


Figure 4: RPN structure.

If the intersection over union (IoU) value of the prediction box corresponding to the anchor point and the ground truth box is the largest, it is marked as a positive sample. If the IoU between the predicted frame and actual frame corresponding to the anchor point is greater than 0.33, it is marked as a positive sample. If the IoU is less than 0.33, it is marked as a negative sample. The rest are neither positive nor negative samples and do not participate in the final training. The loss function selects cross-direction objective function, and its expression is

$$C = -\frac{1}{n}\sum_{x}[y \lg a + (10 - y) \lg (10 - a)], \tag{1}$$

where x represents the selected sample and n indicates the number of samples selected.

Compared with the quadratic objective function, when the training error is larger, the gradient is larger, and the parameter adjustment speeds up, which makes the training faster and faster. The reasons are as follows:

Find the gradient of parameter w:

$$\frac{\partial C}{\partial w_j} = \sum_{x} x_j y(\sigma(z) - y), \tag{2}$$

where $\sigma(z) - y$ represents the error between the output value and the true value. In the same way, the gradient of b is

$$\frac{\partial C}{\partial b} = \sum_{x} (\sigma(z) - y). \tag{3}$$

The entire loss function is

$$L = L(\{t_i^*\}, \{t_i\}), \tag{4}$$

where $L(\{t_i^*\}, \{t_i\})$ is the loss function in Faster RCNN. The main $L(\{t_i^*\}, \{t_i\})$ should be composed of classification loss function and regression loss function, t_i represents the four parameter coordinates of predicted candidate frame

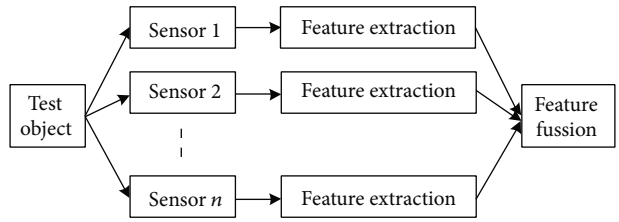


Figure 5: Feature-level fusion.

and t_i^* is the coordinate vector of selected frame when the sample is positive, that is,

$$\begin{cases} t_x = (x - x_a), \\ t_y = (y - y_a), \\ t_x^* = (x^* - x_a), \\ t_y^* = (y^* - y_a), \end{cases} \tag{5}$$

where x and y, respectively, represent the center coordinates and width and height of candidate frame predicted by the RPN network. Besides, x_a and y_a are the center coordinates and width and height of selection frame for positive samples.

3.3. Side-to-Side Multisensor Fusion. A single sensor has detection limitations. This paper uses sensor clusters to collect the train status when detecting rail train faults and highly integrates multiple status data to realize global situational awareness of fault status. The use of multisensor feature data fusion can greatly improve the system's ability to perceive environment; this improves the intelligence of the entire detection system platform [22].

As shown in Figure 5, the feature-level fusion used in this paper is an intermediate-level data fusion. To extract the feature vector contained in collected data, it can reflect the attributes of monitored physical quantity, which is the feature fusion of monitored objects. In the process of feature-level fusion, the representative features extracted from sensor data should be fused into a single feature vector. Then, we use the method of pattern recognition to process, and feature-level fusion realizes information compression, which is convenient for real-time processing. In this paper, the wavelet transform method is used to realize the data fusion of heterogeneous sensor cluster datasets.

After precleaning the images collected by the multisensor cluster before fusion, the data sample set is divided into three bands R, G, and B according to the RGB model, and the three bands are wavelet decomposed, respectively:

$$\begin{cases} \text{Band}_R = LL_{R4} + \sum_{i=1}^{4}(HL_{Ri} + LH_{Ri} + HH_{Ri}), \\ \text{Band}_G = LL_{G4} + \sum_{i=1}^{4}(HL_{Gi} + LH_{Gi} + HH_{Gi}), \\ \text{Band}_B = LL_{B4} + \sum_{i=1}^{4}(HL_{Bi} + LH_{Bi} + HH_{Bi}). \end{cases} \tag{6}$$

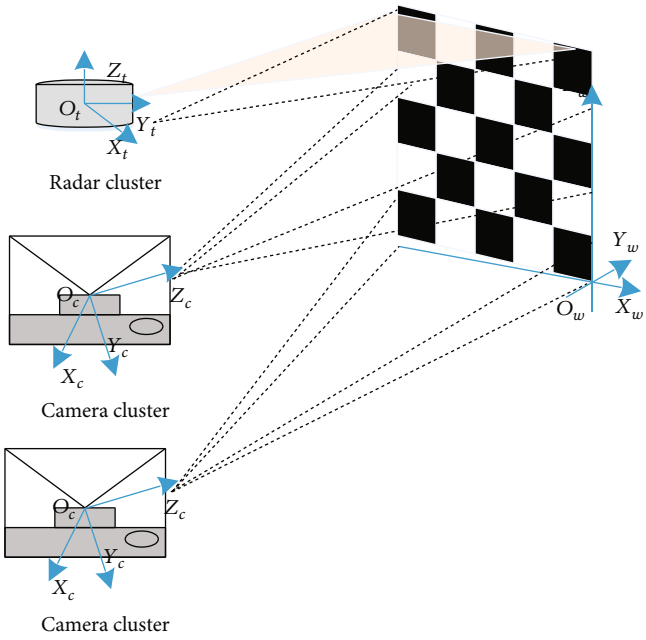


Figure 6: Joint calibration of the camera and radar.

Take the low-frequency coefficients LL_{R4}, LL_{G4}, and LL_{B4} decomposed by Band_R, Band_G, and Band_B and the $(\sum HL_i, \sum LH_i, \sum HH_i)$ reflecting the image edge detail elements for wavelet synthesis. The formula is as follows:

$$\begin{cases} \text{Band}'_R = LL_{R4} + \sum_{i=1}^{4}(HL_{Ri} + LH_{Ri} + HH_{Ri}), \\ \text{Band}'_G = LL_{G4} + \sum_{i=1}^{4}(HL_{Gi} + LH_{Gi} + HH_{Gi}), \\ \text{Band}'_B = LL_{B4} + \sum_{i=1}^{4}(HL_{Bi} + LH_{Bi} + HH_{Bi}). \end{cases} \tag{7}$$

RGB three-channel synthesis is used for the three band images to obtain the fused reliable dataset.

In the process of multisensor data fusion, sensor calibration is particularly important. In order to simplify the calculation, this paper selects the sensor coordinate system as a unified coordinate system. We obtain the external parameters jointly calibrated by the camera and LIDAR, so as to realize the unity between the two coordinate systems. In this paper, the point cloud data of LIDAR is mapped to the image coordinate system, which can complete the sensor spatial synchronization. Figure 6 is a schematic diagram of the joint calibration method.

The conversion formula for the joint calibration of LIDAR and camera is as follows:

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = \begin{bmatrix} R_t & T_t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_t \\ Y_t \\ Z_t \\ 1 \end{bmatrix}, \tag{8}$$

where (X_t, Y_t, Z_t) are the coordinates in the LIDAR coordinate system and R_t and T_t represent the rotation matrix and translation vector converted from LIDAR coordinate system to the camera coordinate system, respectively.

The relationship between the LIDAR coordinate system and pixel coordinates is as follows:

The joint calibration process is as follows:

(1) Run the camera and LIDAR node, start the camera and LIDAR sensor, and record and save the joint file of camera and LIDAR

(2) Restart the camera and LIDAR node and import the parameter file obtained from the previous calibration

(3) Adjust the angle of view of point cloud and then make sure that both the image and point cloud can see the complete calibration board and obtain multiframe images and point clouds

(4) Align the point cloud with the image, that is, extract the corresponding points in the point cloud and image, and obtain the external parameters jointly calibrated by the camera and LIDAR by calculation

3.4. Obstacle Recognition Based on Deep Learning on Cloud Edge. Based on the reliable dataset support provided by edge side sensor cluster, this paper uses the YOLOv3 and DeepSort algorithms on the subway monitoring cloud platform to iteratively learn the rail train status data in each scene to construct and improve the detection network model. The training network model is transferred to edge side equipment to complete the real-time rapid deceleration and avoidance operation when the train encounters obstacles.

The traditional CNN network has the problem of long detection time when processing a large amount of computational data. The YOLOv3 network model has a faster processing speed than the CNN model and is often used in real-time detection and analysis research. The YOLOv3 algorithm uses a network structure diagram that combines a multilayer convolutional network with a pooling layer and a fully connected layer. The input picture size has been expanded to 448×448 and then entered into the YOLOv3 network structure. After convolution feature extraction, pooling dimensionality reduction, and fully connected output, the predicted position and category probability of the target are obtained.

The YOLOv3 algorithm divides the input image into $S \times S$ rasters, and the output data of each raster is $(B \times 5 + C)$ dimension. Among them, $B \times 5$ is actually $B \times (4 + 1)$, and the 4-dimensional data in $(4 + 1)$ refers to x, y, w, and h as the predicted target position. The 1-dimensional data in $(4 + 1)$ refers to the confidence score. The C-dimensional data is a conditional class probability. Finally, the output is an $S \times S \times (B \times 5 + C)$-dimensional tensor.

The YOLOv3 algorithm divides the input image into grids. If there is a detection target in a detection grid, the detection grid is responsible for detecting the object. Each

grid cell predicts B regression frames and the scores of these regression frames. The score represents the predicted value of the output of the detection grid, predicting whether there is a target in the detection grid and the probability that the target belongs to a certain category. The score confidence is defined as

$$\text{Confidence} = \Pr(\text{Object}) * \text{IOU}_{\text{pred}}^{\text{trath}}. \tag{9}$$

If the target does not fall into the detection grid, Confidence = 0. If the target falls into the detection grid, the confidence is the IOU between the regression frame and the real area of the target. In other words, if the detection grid contains a target, Pr (Object) = 1; otherwise, Pr (Object) = 0. IOU is the intersection area between the predicted regression frame and the real area of the object.

In the $S \times S$ grids divided by the image, the probability of each grid prediction condition category: $\Pr(\text{Class}_i|\text{Object})$. $\Pr(\text{Class}_i|\text{Object})$ represents the target attribute and its probability value predicted to fall into the grid. In the test phase, we multiply the conditional category probability of each grid by the confidence of each regression frame:

$$\begin{aligned}&\Pr(\text{Class}_i|\text{Object}) * \Pr(\text{Object}) * \text{IOU}_{\text{pred}}^{\text{trath}} \\ &= \Pr(\text{Class}_i) * \text{IOU}_{\text{pred}}^{\text{trath}}.\end{aligned} \tag{10}$$

In this way, the confidence score of the specific category of each regression frame can be obtained. This product not only contains the probability information of the classification predicted in the regression frame but also reflects whether the regression frame contains objects and the accuracy of the coordinates of the regression frame.

The steps of using YOLOv3 for target detection are shown in Figure 7:

Step 1: input the input left-eye image frame into YOLOv3 network after size transformation and divide it into 5×5 raster $Bi(i = 1, 2, \cdots \cdots, 49)$.

Step 2: after each raster is processed by the YOLOv3 network, two prediction frames $\text{Re}(x, y, \text{Confidence})$ are output.

Step 3: determine whether the object falls into the grid. If the object does not fall into the grid, set Confidence to 0. If the object falls into the grid, the predicted Confidence value will be output, and the prediction frame $\text{Re}(x, y,$ Confidence$)$ will be updated.

Step 4: compare the predicted Confidence value with threshold T to remove the redundant window and retain high confidence value position window.

Step 5: determine whether the input target position of previous module falls into the reserved position window. If it falls into the reserved position window, output the recognition result. If it does not fall into the reserved position window, discard it.

However, it should be noted that rail trains are generally in high-speed motion. Adding the DeepSort algorithm framework to the obstacle recognition network, using the motion model and apparent information for data association, can achieve end-to-end multitarget visual fast tracking.

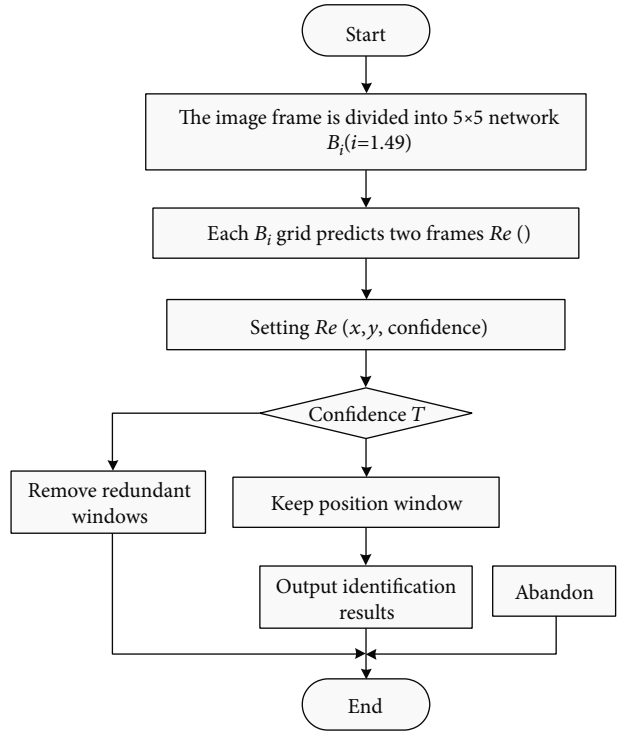


Figure 7: Flow chart of YOLOv3 target detection.

This enables the vehicle target to obtain a good tracking effect under complex conditions such as illumination, fast movement, and occlusion [23, 24].

The DeepSort algorithm has deep association features and is based on the improvement of Sort algorithm. Its tracking effect is based on the existing accurate detection results. The prediction module uses Wiener filtering, and the update module uses IOU to match the Hungarian algorithm. The tracking process is shown in Figure 8.

In order to prevent a target covering multiple targets or multiple detectors detecting a target in multitarget tracking, the DeepSort algorithm uses an eight-dimensional state space $(u, v, \gamma, h, \dot{u}, \dot{v}, \dot{\gamma}, \dot{h})$ to define the tracking scene, where (u, v) is the center position of bounding box, γ is the target rectangle aspect ratio, h is the height of rectangular frame, and $(\dot{u}, \dot{v}, \dot{\gamma}, \dot{h})$ is the motion information. The algorithm uses a linear observation model and standard Wiener filtering of uniform velocity model to predict the target trajectory in the next frame and uses a boundary coordinate (u, v, γ, h) as the direct observation of the object state. For each track k, the number of frames between the last successfully detected frame picture and the currently detected frame picture is recorded as a_k. This counter is incremented during Wiener filtering prediction period and is set to 0 when the trajectory is associated with the measurement. When a_k exceeds threshold $A_{\max}$, it is deemed that the track has left the scene and is deleted. When there is a detection in the detector that cannot be matched with the existing trajectory, a tentative trajectory is generated first, and if the trajectory cannot be rematched in three frames, it is deleted.

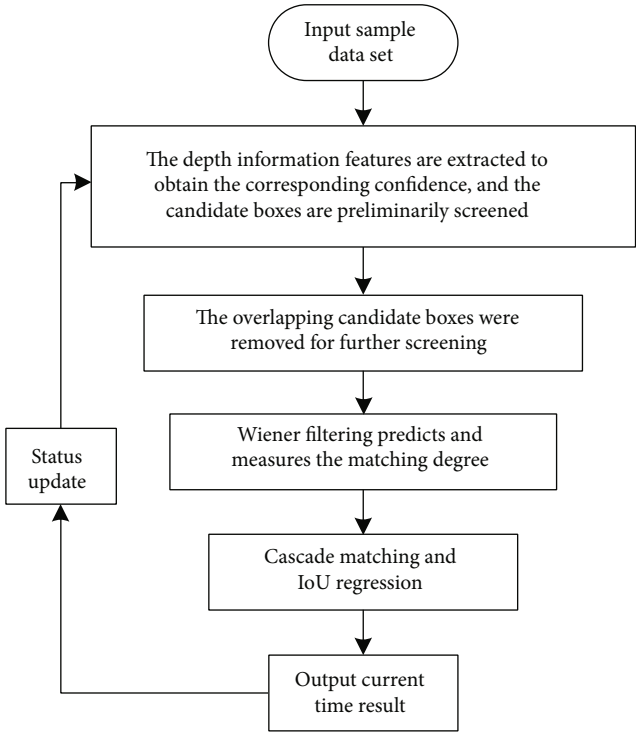


Figure 8: DeepSort tracking process.

The Mahalanobis distance indicates the degree of deviation of the detection target from the average position of target trajectory; the Mahalanobis distance can be used to measure the degree of matching between the target state predicted by Wiener filtering and detection value. We use y_i to represent the target prediction frame position of the i tracker, and d_j as the j detection frame position. S_i is the covariance matrix between the detection position and tracking position. The formula for calculating Mahalanobis distance is

$$d(i,j) = \left(d_j - y_i\right)^T S_i \left(d_{j+1} - y_{i+1}\right). \tag{11}$$

The left and right detected targets are screened by the Mahalanobis distance, and threshold $t = 11.526$ is set. If the associated Mahalanobis distance d is less than the threshold, the set motion state association is successful, and the indicator function is

$$b_{i,j} = l[d(i,j) < t]. \tag{12}$$

When the motion uncertainty is very low, the Mahalanobis distance can be a good measure of the relationship between the detected target and trajectory. But when the camera shakes violently, the association method fails. Thus, CNN is introduced for correlation. We obtain feature vector r_j of each detection target d_j, and $\|r_j\| = 1$.

The trained YOLOv3 detector is used for train obstacle detection in complex environments, and the obstacle detection model trained by YOLOv3 is used. The abnormal target detection result is used as the real-time input of DeepSort tracker, thus making up for the own shortcomings of DeepSort.

Table 1: Operation scenarios of simulation experiment.

Software environment	Operating system	Windows 10
	Deep learning framework	PaddlePaddle
	Program editor	PyCharm
Hardware environment	CPU	Intel Core i7 9700
	GPU	GeForce GTX-1650
	Running memory	32 GB

Table 2: Network parameter setting.

Parameter	Value
Weight attenuation silver	0.0012
Momentum parameter	0.97
Initial learning rate	0.001
Maximum learning rate	0.027
Training batch	200

4. Experiment and Comparative Analysis

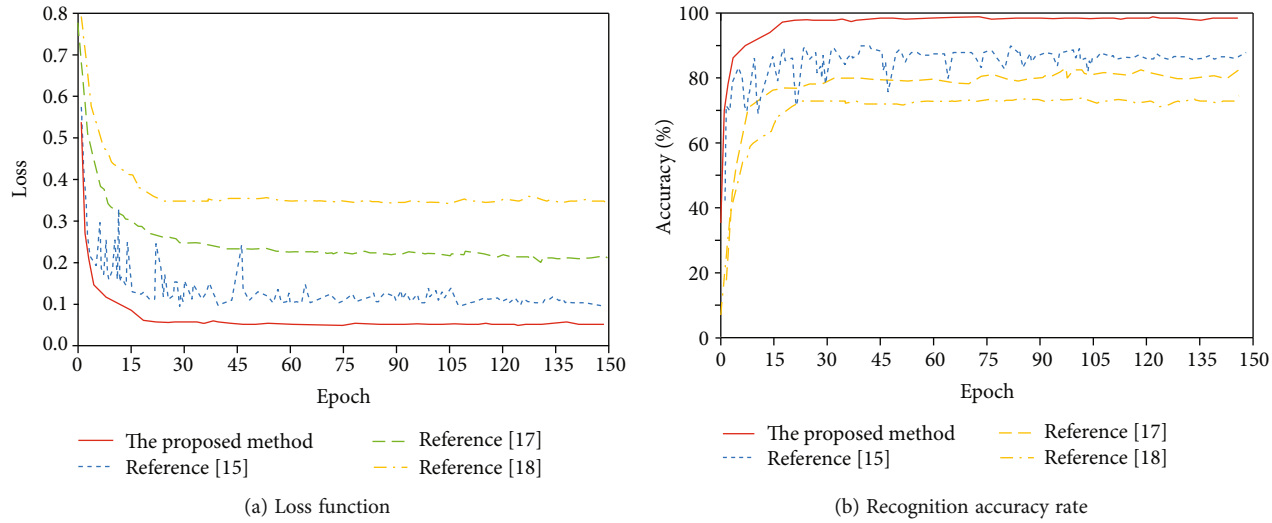
In order to verify the feasibility and accuracy of the proposed method for the detection and identification of subway track obstacles, this paper uses references [15], [17], and [18] as comparison methods. The proposed method and the comparison method are set in the same experimental scene for simulation verification. The experimental scene settings are shown in Table 1.

The experimental dataset uses the actual subway operation dataset of a city in China in 2020. The dataset randomly extracts the rail train operating status data on a certain day in July. The data sample parameter is 30 frames/s, and pixel size is 1080×720. The dataset format was converted to VOC format, then the format labeling information to a TXT file in YOLO format. The recognition category in YOLOv3.cfg file is changed to 1. In view of the small sample data, cross-validation is used to train 200 epochs.

The main network parameters of the subway obstacle analysis method proposed in this paper are shown in Table 2.

4.1. Accuracy Analysis of Track Recognition. In order to verify the feasibility of the proposed method for subway track recognition, we build a proposed detection network model based on the above parameters and reproduce the methods in references [15], [17], and [18] in the same experimental scenario. Figure 9 shows the detection and analysis results of subway tracks under each method.

As shown in Figure 9, at the 45th iteration of proposed detection method, the network loss function drops to 0.06. At the same time, the detection network's orbit recognition accuracy has increased to 98.9%; its value is almost close to 100% and remains stable. References [15], [17], and [18] achieved a stable network performance at 120 times, 90 times, and 60 times, respectively. However, it can be seen that the comparative reference not only has a certain disadvantage compared with the proposed method in terms of



(a) Loss function

(b) Recognition accuracy rate

Figure 9: Subway track recognition performance under different methods.

analysis speed. Moreover, the analysis accuracy is slightly inferior to the recognition ability of the proposed method. Reference [15] is 11.6% lower than proposed method, reference [17] is 21.8% lower than the proposed method, and the analysis accuracy of reference [18] is 72.3%.

4.2. Performance Analysis of Obstacle Detection. The detection and processing of obstacles before the subway encounters obstacles is particularly important. Therefore, we also discuss the performance of the detection method in obstacle recognition and analysis. Figure 10 is a discussion of obstacle detection performance under different recognition methods.

As shown in Figure 10, the method proposed in this paper can effectively distinguish obstacles in the 50th iteration with a recognition accuracy of 98.9%. However, the accuracy of reference [15] is 11.2% lower than proposed method, and the accuracy of reference [18] is 14.6% lower than proposed method. Reference [17] has not yet found the optimal solution in the iterative analysis process. The reason is that we implement feature-level fusion of sensor cluster data on edge side to provide reliable and complete data support for detection network model. The comparison literature only carries out simple data preprocessing on the collected samples. For the deep network, the quality of the dataset samples will determine the accuracy of obstacle recognition to a certain extent. At the same time, reference [17] combines the semantic segmentation network and deep learning network, which has the possibility of local optimization due to the complex network structure, which limits the analysis and recognition.

At the same time, we also analyzed the calculation efficiency of different methods, and the results are shown in Table 3.

According to Table 3, with the help of edge computing for fast and efficient action control at the edge of network, the method proposed in this paper can complete the detection of obstacles in track within 1.43 s. The comparison methods all have a certain time delay. The detection time

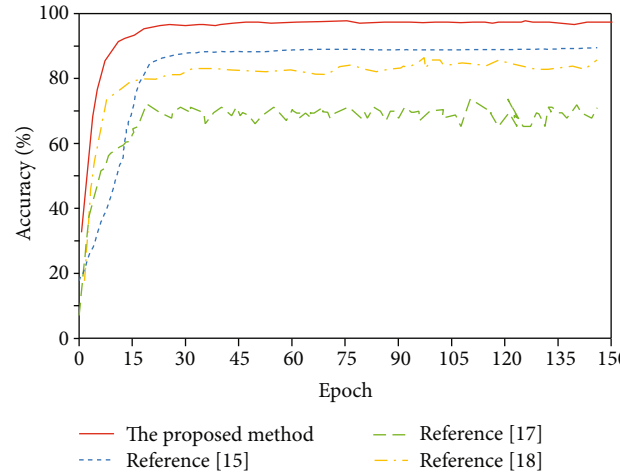


Figure 10: Subway track recognition performance under different methods.

Table 3: Statistical table of target detection experiment results.

Method	Analysis time (s)
The proposed method	1.43
Reference [15]	2.79
Reference [17]	—
Reference [18]	5.42

of reference [15] is 2.79 s, the time of reference [18] is 5.42 s, and reference [17] did not complete the reliability of subway track obstacles within the set time. At the same time, the YOLOv3 network used in this paper is essentially a one-step solution, which can realize direct and efficient feature extraction for the sample dataset, while the CNN network used in the comparative literature needs to classify the sample dataset first and then realize feature extraction. Therefore, it is proved that the proposed method has the ability of an efficient and rapid obstacle analysis.

Table 4: Statistical table of multitarget tracking analysis results.

Method	Actual number of obstacles	Number of obstacles detected	Accuracy (%)
The proposed method	100	96	96
Reference [15]	100	91	91
Reference [17]	100	72	61
Reference [18]	100	64	64

4.3. Target Tracking Analysis. At the same time, we also analyze the performance of multitarget tracking. Table 4 shows the performance of multitarget tracking analysis under different methods.

As shown in Table 4, due to the introduction of DeepSort algorithm, the proposed method can effectively achieve multitarget visual fast tracking at the edge of network, and the recognition accuracy can reach 96%. The comparison method is obviously not as good as the proposed method. The recognition accuracy of references [15], [17], and [18] is 91%, 61%, and 64%.

In summary, the proposed method can meet the needs of efficient identification for obstacles in actual subway operation. Compared with the current analysis methods, it has better image feature mining and analysis capabilities, which achieves reliable support for stable operation of rail trains.

5. Conclusion

An efficient and accurate obstacle identification method is very important for the stable and safe operation of the subway. Based on cloud edge cooperation mode and deep learning technology, this paper proposes a fast and effective rail transit obstacle recognition method. In this method, Mask RCNN algorithm is applied to the route identification of a metro rail transit, which can provide route guarantee for the safe directional operation of trains. Based on the local fast computing mode of edge computing, the state perception and foreign object recognition of running track are realized on the edge side of the network based on the YOLOv3 and DeepSort algorithms. Through the simulation analysis, it can be seen that the method proposed in this paper can achieve more rapid and accurate track obstacle analysis in the actual complex scene.

The nature of edge computing is lightweight on-site computing. However, the memory and computing power of smart devices at the edge of network are greatly restricted under the condition of limited hardware costs. In order to further reduce the difficulty of computing and solution, the lightweight processing research will be carried out on the deep learning detection network model in the future. Furthermore, it can save network memory and reduce computational complexity and realize sensitive and efficient identification of track obstacles in actual complex scenes.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare no conflicts of interest.

Acknowledgments

The work described in this paper was fully supported by a grant from the Natural Science Foundation of colleges and universities of Jiangsu Province (No. 18KJD510009).

References

[1] C. Wu, X. Qiang, Y. Wang, C. Yan, and G. Zhai, "Efficient detection of obstacles on tramways using adaptive multilevel thresholding and region growing methods," *Proceedings of the Institution of Mechanical Engineers, Part F: Journal of Rail and Rapid Transit*, vol. 232, no. 5, pp. 1375–1384, 2018.

[2] D. Li, L. Deng, and Z. Cai, "Design of traffic object recognition system based on machine learning," *Neural Computing and Applications*, vol. 33, no. 14, pp. 8143–8156, 2021.

[3] S. Q. Guo and Y. Dong, "Research on obstacle detection method in front of train running on straight track based on radar," *Journal of Railway Science and engineering*, vol. 17, no. 1, pp. 224–231, 2020.

[4] P. Pavel and O. Andrey, "Autonomous train - the Russian perspective," *Automation, Communication and Informatics*, vol. 8, no. 1, 2019.

[5] X. K. Ding, X. D. Hu, and Q. Wei, "High speed train line safety monitoring technology based on optical measurement," *Journal of Railway Science and engineering*, vol. 15, no. 9, pp. 2224–2231, 2018.

[6] J. Li, F. Zhou, and T. Ye, "Real-world railway traffic detection based on faster better network," *IEEE Access*, vol. 6, no. 1, pp. 68730–68739, 2018.

[7] S. Shi, "Review of active obstacle detection in rail transit system," *Mechanical and electrical engineering technology*, vol. 50, no. 6, pp. 212–216, 2021.

[8] G. R. Zhai, C. H. Hu, and L. J. Zhang, "Reliability design of obstacle and derailment detection device," *Heilongjiang sci tech information*, vol. 34, pp. 155–155, 2014.

[9] H. Mukojima, D. Deguchi, and Y. Kawanishi, "Moving camera background-subtraction for obstacle detection on railway tracks," in *2016 IEEE International Conference on Image Processing (ICIP)*, pp. 1–5, Phoenix, USA. IEEE, 2016.

[10] X. Zhang, M. Zhou, P. Qiu, Y. Huang, and J. Li, "Radar and vision fusion for the real-time obstacle detection and identification," *Industrial Robot: the international journal of robotics research and application*, vol. 46, no. 3, pp. 391–395, 2019.

[11] L. B. Chang, S. B. Zhang, and H. M. Du, "Position-aware lightweight object detectors with depthwise separable convolutions," *Journal of Real-Time Image Processing*, vol. 18, no. 3, pp. 857–871, 2021.

[12] S. T. Ding and S. R. Qu, "Traffic target region of interest detection based on deep learning," *Chinese Journal of highways*, vol. 31, no. 9, pp. 167–174, 2018.

[13] T. Ye, Z. Zhang, X. Zhang, and F. Zhou, "Autonomous railway traffic object detection using feature-enhanced single-shot detector," *IEEE Access*, vol. 8, no. 1, pp. 145182–145193, 2020.

[14] Y. Wang, L. Wang, Y. H. Hu, and J. Qiu, "RailNet: a segmentation network for railroad detection," *IEEE Access*, vol. 7, no. 1, pp. 143772–143779, 2019.

[15] D. He, Z. Jiang, J. Chen, J. Liu, J. Miao, and A. Shah, "Classification of metro facilities with deep neural networks," *Journal of Advanced Transportation*, vol. 2019, no. 1, 2019.

[16] Z. Wang, X. Wu, G. Yu, and M. Li, "Efficient rail area detection using convolutional neural network," *IEEE Access*, vol. 6, no. 1, pp. 77656–77664, 2018.

[17] Q. Zhang, F. Yang, and B. Zhang, "Application of intelligent obstacle detection system in the automatic operation of Beijing new airport line," *Railway rolling stock*, vol. 39, no. 6, pp. 114–118, 2019.

[18] Y. Z. Deng and M. Lin, "Recognition method of rail transit obstacles based on improved LeNet-5," *Industrial control computer*, vol. 33, no. 1, pp. 63–66, 2020.

[19] M. A. Albreem, A. M. Sheikh, M. H. Alsharif, M. Jusoh, and M. N. M. Yasin, "Green internet of things (GIoT): applications, practices, awareness, and challenges," *IEEE Access*, vol. 9, no. 1, pp. 38833–38858, 2021.

[20] T. Qiu, J. Chi, X. Zhou, Z. Ning, M. Atiquzzaman, and D. O. Wu, "Edge computing in industrial internet of things: architecture, advances and challenges," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 4, pp. 2462–2488, 2020.

[21] G. Han, J. P. Su, and C. W. Zhang, "A method based on multi-convolution layers joint and generative adversarial networks for vehicle detection," *KSII Transactions on Internet and Information Systems*, vol. 13, no. 4, pp. 1795–1811, 2019.

[22] P. F. Alcantarilla, S. Stent, G. Ros, R. Arroyo, and R. Gherardi, "Street-view change detection with deconvolutional networks," *Autonomous Robots*, vol. 42, no. 7, pp. 1301–1322, 2018.

[23] Z. X. Li, W. Sun, and M. M. Liu, "Research on vehicle detection and tracking algorithm in traffic monitoring scene," *Computer engineering and application*, vol. 57, no. 8, pp. 103–111, 2021.

[24] X. Lin, C.-T. Li, V. Sanchez, and C. Maple, "On the detection-to-track association for online multi-object tracking," *Pattern Recognition Letters*, vol. 146, no. 9, pp. 200–207, 2021.