

Received 26 January 2024, accepted 12 February 2024, date of publication 14 February 2024, date of current version 23 February 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3366180

RESEARCH ARTICLE

Optimal Layout Method for Roadside LiDAR and Camera

YAN LI^{1,2,3,4}, HAN ZHANG^{1,2,3}, ZHIHENG CHENG^{1,2,3}, ZIJIAN WANG^{1,2}, ZIYI ZHANG^{2,3}, AND XINPENG YAO^{1,2}

¹Shandong Hi-Speed Construction Management Group Company Ltd., Jinan 250117, China

²Shandong Key Laboratory of Smart Transportation (Preparation), Jinan 250002, China

³School of Qilu Transportation, Shandong University, Jinan 250061, China

⁴Shandong Hi-Speed Group Company Ltd., Jinan 250002, China

Corresponding author: Zhiheng Cheng (18853768146@163.com)


This work was supported in part by the Key Research and Development Program, China, under Grant 2022YFB2602102; in part by the Shandong Provincial Key Laboratory of Intelligent Transportation Open Subject Fund; and in part by the National Natural Science Foundation of China under Grant 52002224.

ABSTRACT With the widespread application of LiDAR and camera, the integration of LiDAR and camera has become an urgent issue. In this study, we proposed an optimal layout method for roadside LiDAR and camera. Firstly, the experimental design phase took into consideration various application scenarios, such as curved road sections and gradient road sections. Secondly, the video data and point cloud data collected from different experimental setups were subjected to object detection and recognition using YOLOv5s weights and PointPillars weights, respectively. These weights are applied to the video data under different layout schemes to output the mAP value of each scheme. By comparing the mAP values under different layout schemes, the optimal layout scheme for the road scene is determined. Thirdly, the four road scene parameters and six installation parameters from all scenarios were collected into a database. Furthermore, five machine learning algorithms were employed for optimal selection. Finally, the three regression algorithms with the highest accuracy are selected for the final prediction model based on different control groups. Through the field experiment, the results show that the optimized layout method can significantly reduce blind spot detection and vehicle occlusion problems for camera and LiDAR. The optimal deployment method can increase the Mean Average Precision (MAP) by over 4% through adjusting the installation parameters. The regression algorithm used can predict the optimal deployment scheme for roadside LiDAR and cameras in unknown scenarios with over 95% accuracy. This optimal deployment method improves the detection accuracy of roadside devices for road vehicles by changing the installation parameters and provides guidance for the future installation of roadside LiDAR and camera.

INDEX TERMS Optimal layout, YOLOv5, OpenPCDet, PointPillars, mAP, regression algorithms.

I. INTRODUCTION

Transportation is the key to national revitalization and the foundation of a strong country. With three significant characteristics of fundamental, pioneering, and strategic, transportation plays a significant role in the national economy. As a landmark product of the deep integration of new-generation information technology and transportation system, intelligent highway is changing the way people travel. LiDAR and

The associate editor coordinating the review of this manuscript and approving it for publication was Md. Moinul Hossain .

camera, an essential part of the vehicle-road coordination system and intelligent highway, have the advantages of high accuracy and a wide range of applications. The accuracy of camera is heavily affected by light. Its detection accuracy will be affected somewhat at a low light intensity. On the other hand, LiDAR [1], [2], [3], [4] can work well in nighttime conditions. However, it has not been possible to calculate the best layout method for other scenarios based on the data of known layout methods.

In terms of optimal deployment research for roadside perception devices, Kim et al. [5] proposed a method for

positioning a multi-sensor system in roadside infrastructure. They introduced a voxel grid and utilized a genetic algorithm to address the optimization challenge. Hussain et al. [6] presented a multi-objective optimization model to study the installation positions of perception devices on the roadside and proposed a hybrid evolutionary multi-objective algorithm to validate the model. Empirical results demonstrated that this algorithm outperformed existing ones in terms of both accuracy and stability. Twahirwa et al. [7] used a driving matrix scheme based on classical incremental strategies to optimize the deployment of intelligent roadside units. Simulation results showed that, under different vehicle density scenarios, this deployment of the travel matrix could reduce the number of roadside units while enhancing vehicle communication capabilities. Ahmad et al. [8] proposed a positioning scheme based on received signal strength. By detecting signals within the range of roadside devices, establishing communication, and developing an algorithm, their positioning algorithm helped determine the precise location of vehicles. A comparison with existing algorithms revealed that this algorithm exhibited superior performance.

Currently, research on roadside sensor layout is limited to the macro level. However, with the development of the concept of vehicle-road coordination, the optimal layout of sensors at individual pole locations is equally important. The essence of vehicle detection and recognition using roadside devices such as cameras and LiDAR lies in the number of pixels in video data and the number of point cloud points in point cloud data. Based on this point, this paper proposes an optimal method for roadside intelligent perception devices based on neural networks. This method uses the YOLOv5 algorithm and the PointPillars algorithm in the OpenPCDet framework as performance metrics for vehicle detection by roadside devices. This approach ensures that roadside devices can achieve optimal performance when integrating algorithms.

Regarding the YOLOv5 algorithm, Kumar et al. [9] proposed an intelligent and efficient solution based on YOLOv5 to address the detection and tracking of vehicles. The algorithm's outstanding results were verified through simulated dataset experiments. Scholars [10] integrated an attention mechanism module into the YOLOv5 network to reduce the number of parameters and floating-point operations per second, thereby reducing model parameters while ensuring detection accuracy. Another group of researchers [11] employed the YOLOv5 model and replaced the bottleneck portion of YOLOv5 with a convolutional block attention module based on attention mechanism to improve the model's accuracy and speed. Kumar et al. [12] enhanced the network's recognition capabilities by introducing channel and spatial attention modules into the YOLOv5 model. As for the network architecture under YOLOv5, Desta et al. [13] compared the YOLOv5 network models, including YOLOv5s, YOLOv5m, YOLOv5l, and YOLOv5x, and pointed out that as the model size increases, training accuracy increases, but so does training time. Hofinger et al. [14] compared various

network architectures of YOLOv5 and ultimately chose to use YOLOv5s as their research architecture. Subsequent experiments demonstrated that this network architecture could achieve superior results with less workload. Efrem et al. [15] integrated a coordinated attention module into the YOLOv5s backbone to effectively distinguish useful features by capturing channel and position information. Experimental results not only showed that the improved YOLOv5s model outperformed the baseline model trained only on real-world images in mixed datasets but also demonstrated its ability to address potential overlap and occlusion issues at various spatial scales. Nnadozie et al. [16] evaluated two versions of the YOLOv5 neural network, YOLOv5N and YOLOv5S, and found that YOLOv5s achieved the highest accuracy, while YOLOv5n had the fastest inference speed. In terms of false detection rate, compared to YOLOv5n, YOLOv5s achieved more accurate target counting under interference conditions.

In the context of laser radar data, Alaba and Ball [17] processed three-dimensional laser radar data using low-frequency and high-frequency coefficients as filters. Experimental results showed that this method can construct a lightweight network without significant performance loss. Silva et al. [18] proposed a dedicated model for roadside edge computing devices to perform airborne inference for roadside laser radar target detection algorithms, addressing the high equipment requirements. Experimental results demonstrated that this method exhibits high real-time performance under spatial and power constraints and achieves detection accuracy comparable to deep learning methods. Zhang et al. [19] introduced a feature fusion module with high attention and channel attention for feature fusion of multiple pseudo images. The method not only overcomes the sparsity of point clouds but also reduces interference caused by uneven point cloud distribution. Zhang et al. [20] proposed an improved PointPillars algorithm. After dividing the point cloud into several pillars, global context features and local structural features are extracted using a multi-head attention mechanism for feature encoding, and the generated two-dimensional pseudo images are used for feature learning with a two-dimensional convolutional neural network. Finally, a solid-state LiDAR is used for 3D object detection. The improved algorithm achieved a significant increase in average precision on a public test dataset. Singh [21] proposed an improved PointPillars algorithm leveraging the sparsity of point clouds and hierarchical feature learning of PointNet. Extensive experiments demonstrated that the algorithm has a significant advantage in accuracy over existing encoders while maintaining fast operational speed. Additionally, researchers [22] directly modified the three-dimensional object detection network based on laser radar in the PointPillars algorithm using information from multiple sensors. Experimental results showed a significant improvement in average precision (mAP) compared to the traditional PointPillars method.

After obtaining the optimal layout plans for LiDAR and camera using respective algorithms, a layout method for

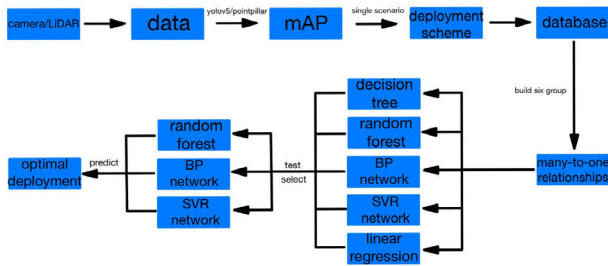


FIGURE 1. Flow chart of the optimal layout method.

roadside devices applicable in multiple scenarios is derived. Subsequently, the optimal layout plan data for various scenarios are collected in a dataset. This dataset is then utilized to predict layout plans for unknown scenarios based on the available scenario data. In this paper, five machine learning algorithms are introduced, namely the decision tree algorithm [23], random forest algorithm [24], backpropagation neural network algorithm [25], support vector regression algorithm [26], and linear regression neural network algorithm [27]. The fitting effects of these algorithms are compared, and the algorithm with better performance is selected for prediction. In this paper, the YOLOv5 algorithm and PointPillars algorithm are used to measure the accuracy and confidence of vehicle pixel points and point cloud points obtained by camera and LiDAR in the road environment. The overall process is shown in Figure 1.

The weights of YOLOv5s and PointPillar are applied to video and point cloud data, respectively, and the mAP values under different layout schemes in the scene are used to determine the optimal layout scheme. The road information and optimal installation parameter data are then inputted into a database. We constructed six groups of many-to-one data correspondence models and applied five different algorithms to fit and predict them respectively. The algorithm with the best fitting performance among different multiple-to-one relationships is selected based on the fitting rate of the data. The required algorithms are incorporated into the overall prediction algorithm framework. The regression algorithm with the highest fitting rate is employed in the prediction models of six dependent variables.

II. EXPERIMENTAL DESIGN

The experiment examines the relationship between those mentioned above and the independent and dependent variables. By systematically varying the lane width, road slope, curve radius, and distance from the road, we can observe the corresponding changes in the installation height, installation angles, and installation distance of the LiDAR and camera [28], [29]. Furthermore, the experiment acknowledges the influence of diverse road conditions on the performance of the LiDAR and camera systems. Consequently, the layout methods are adjusted to optimize the accuracy and effectiveness of data collection in this specific scenario. Ultimately, this experiment seeks insights into the optimal configuration and

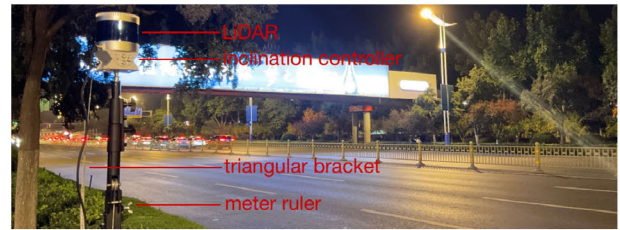


FIGURE 2. Roadside layout (LiDAR as an example).



FIGURE 3. Coordinate plot of LiDAR and camera.

layout strategies for LiDAR and camera systems in different road scenarios while enhancing their perception capabilities and data acquisition efficiency.

A. EXPERIMENTAL DESIGN OF ROADSIDE EQUIPMENT IN A STRAIGHT ROAD SCENARIO

In the experimental setup, a LiDAR model named Raishen C32 and a camera model named HY1080 were used. The experimental layout diagram is shown in Figure 2 (taking LiDAR as an example).

The installation height of LiDAR and cameras can directly result in the occurrence of vehicle occlusion and detection blind spot issues [30]. Adding rotation angles to both sensors and adjusting the installation distance from the road can effectively alleviate this issue. Excessive tilt angle of the LiDAR can cause point clouds to become too dense at close range and too sparse in the distance, resulting in lower detection accuracy. A higher installation height of the LiDAR leads to overall sparsity in point cloud data, thus reducing data accuracy.

Similarly, a larger tilt angle of the camera results in a decrease in the detection distance. In contrast, a higher installation height of the camera causes blurred target images in the collected data. In practical applications, the distance of roadside pillars is usually predetermined. Therefore, the distance from the road will be one of the dependent variables in this experiment. Similarly, the different widths of the road will also have an impact on the installation plan of the roadside sensors. Therefore, in this experiment, the width of the road will be set as the independent variable.

In the dependent variable section, the installation height of the LiDAR, the installation height of the camera, the installation tilt angle of the LiDAR around the X-axis and Y-axis, and the installation tilt angle of the camera around the X-axis and Y-axis were set. In all scenarios, the installation height range for the LiDAR and camera is 2.5m to 5m. The installation height increases by 0.5m during the actual installation process. At each height, the installation tilt angle

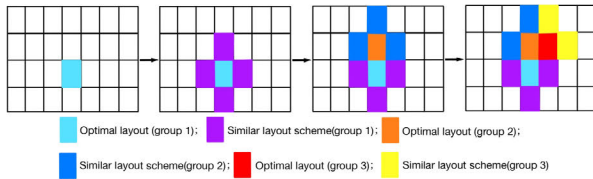


FIGURE 4. Layout of the optimization diagram.

of the camera and LiDAR around the X-axis is adjusted. The coordinate system of LiDAR and the camera is shown in Figure 3. The tilt range for the LiDAR installation angle is 0° to 12° , with a step size of 4° . The camera tilt angle range is also from 0° to 12° . The layout parameters and the data collected by the sensors for road vehicles are recorded for each detection scheme.

After acquiring video and point cloud data, target calibration and convolutional neural network construction are performed separately. To reduce the amount of data processing and expedite the determination of optimal layout solutions in different scenarios. The trained neural network models are then validated on other data in the same scene. The overall accuracy of target recognition is used to select the most reasonable set of current layout schemes. A similar set of schemes is also selected based on this group, with the current scheme as the center point but with changes in installation height and tilt angle. Data collection, neural network construction, and accuracy determination are then performed for similar layout schemes. If the accuracy of the center point scheme is the highest among the five groups, it is chosen as the optimal layout scheme. Otherwise, the group with the highest accuracy is selected again, and a similar set of schemes is tested repeatedly. The layout optimization process is shown in Figure 4.

Initially, the most accurate one is selected from 28 groups of neural networks as the optimal layout 1. Subsequently, this network is used as the center point in the graph for data collection, calibration, and validation steps for its similar layout 1. The accuracy of four layout schemes is then obtained. The layout scheme with the highest accuracy at this point is then chosen. If this layout scheme is the initial optimal layout 1, then this scheme is the optimal layout point. However, if the newly added layout scheme is currently the most accurate, this point is selected as the optimal layout 2. This step is repeated until the optimal layout three is more accurate than the similar layout points. At this point, this point becomes the optimal layout scheme for this scenario.

To validate the rationality of each proposed approach, we adopted mAP (mean Average Precision) as the evaluation metric for assessing the performance of neural networks under different schemes. In our experimental, we utilized the I7-13700kf CPU and NVIDIA GeForce RTX 4070Ti GPU for model training and testing. The evaluation criterion IoU was set to 0.5. We employed mAP to represent the detection results of each model and to quantify the average precision of the network under a given scheme. Recall, also known

as the positive rate, indicates the ratio of correctly detected objects (TP) to all that should have been detected (TP + FN). Precision, on the other hand, represents the ratio of correctly detected objects (TP) to all detected (TP + FP). The formulas for these evaluation metrics are provided below.

$$\text{precision} = \frac{TP}{TP + FP} \quad (1)$$

$$\text{Recall} = \frac{TP}{TP + FN} \quad (2)$$

$$\text{AP} = \int_0^1 P(R)d(R) \quad (3)$$

$$\text{mAP} = \frac{1}{N} \sum_{I=1}^N \text{AP}_i \quad (4)$$

Here, TP refers to true positives (objects correctly detected), FN refers to false negatives (objects that should have been detected but were missed), and FP refers to false positives (objects incorrectly detected). These metrics serve as comprehensive indicators of the network's performance in terms of both sensitivity and specificity.

B. EXPERIMENTAL DESIGN OF ROADSIDE EQUIPMENT IN SLOPE SCENARIO

In the scenario of a sloping road segment, the lane slope parameters can affect the detection range of the LiDAR and camera. For the LiDAR, when there is a significant uphill or downhill slope on the lane, the actual distance and height information may be affected. In the case of an uphill slope, one side of the LiDAR may detect a shorter distance while the other side may experience sparse point cloud due to the road inclination, leading to reduced detection accuracy. Conversely, in the case of a downhill slope, the LiDAR may experience reduced detection distance and sparse point cloud due to the angle of inclination, ultimately resulting in decreased accuracy in vehicle detection.

Similarly, changes in lane slope can also affect the camera's field of view and target detection. In uphill situations, the camera may face a higher angle, which could lead to a wider field of view or interference from the sky area. On the other hand, in downhill situations, the camera may face a lower angle, leading to limited visibility and difficulty in target detection.

Therefore, in the experimental design for sloped road sections, the camera and LiDAR are tilted around the Y-axis at an angle that matches the slope of the road in that particular scene. Subsequently, experiments were carried out with different heights and inclination angles.

C. EXPERIMENTAL DESIGN OF ROADSIDE EQUIPMENT IN CURVED ROAD SCENARIO

Curved road segments also have specific effects on the detection of vehicles by LiDAR and camera. The specific impact depends on the radius of curvature of the bend. For LiDAR, when vehicles pass through a curve, the laser beams may not align perfectly with the curvature of the curve, leading

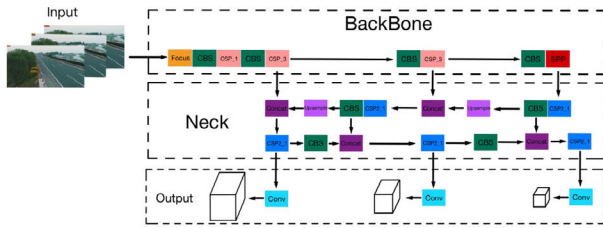


FIGURE 5. YOLOv5 architecture diagram.

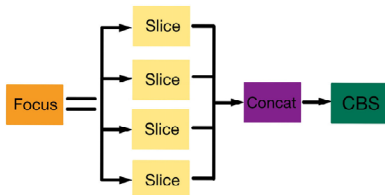


FIGURE 6. Schematic diagram of the Focus layer.

to distance measurement errors. The LiDAR on the outside of the curve may detect shorter distances, while the LiDAR on the inside of the curve may detect longer distances. This can affect the perception of the driving scenario and the estimation of vehicle positions.

For camera, curved road sections can cause changes in the field of view. Camera located on the outside of the curve may experience greater changes in the field of view, leading to distortion in object shapes and sizes. Camera located on the inside of the curve may experience smaller changes in the field of view, which may result in object occlusion or partial omission.

Therefore, in experiments, it is necessary to modify the installation distance of sensors from the road and determine whether the camera and LiDAR should be deployed on the outer or inner side of the road, building upon the existing experimental setup.

III. NEURAL NETWORK CONSTRUCTION BASED ON OBJECT RECOGNITION

To determine the accuracy of vehicle detection under different installation schemes, it is necessary to train corresponding neural network models for each layout scheme. For the camera, the YOLOv5 algorithm is employed to process the two-dimensional video data, and the YOLOv5s model is used as the detection weight. For the LiDAR, the PointPillar algorithm within the OpenPCDet framework is utilized to process the three-dimensional point cloud information, and the open-source PointPillar model is employed as the detection weight. The mAP values are then calculated for each scene, and based on these values, the optimal layout method for each scene is determined.

A. VIDEO DATA PROCESSING BASED ON YOLOV5

YOLOv5, proposed in 2020, is an efficient object detection algorithm. Compared to its previous versions, YOLOv5

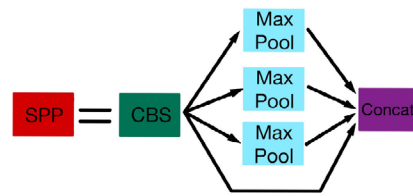


FIGURE 7. SPP architecture diagram.

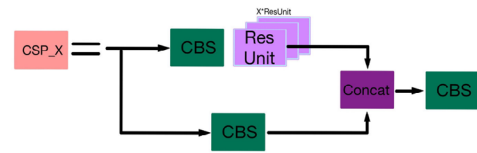


FIGURE 8. Schematic diagram of the CSP_X module.

adopts a simpler architecture while optimizing detection speed, accuracy, and model size. This algorithm employs an anchor-free approach for object detection. Additionally, it utilizes the SPP (Spatial Pyramid Pooling) structure and PAN (Path Aggregation Network) module for feature extraction. Moreover, sampling strategies and training methods are optimized, leading to further improvement in detection performance. Figure 5 shows the network architecture diagram of YOLOv5.

The Backbone module primarily focuses on feature extraction, extracting object-related information from the images using convolutional networks for subsequent object detection. The Neck module blends and combines these features to enhance the network’s robustness and strengthen its object detection capabilities. These features are then passed to the Convolutional layer for prediction.

The principle of the Focus layer is similar to that of the PassThrough layer. It utilizes slicing operations to divide high-resolution images into multiple low-resolution sub-images/feature maps, accomplished through column-wise sampling and concatenation.

The SPP architecture utilizes spatial pyramid pooling to transform feature maps of arbitrary sizes into fixed-size feature vectors.

The backbone is a relatively deep network, and the addition of CSP_X modules can increase the gradient values in backpropagation between layers, preventing gradient disappearance caused by deepening of the network. This allows for more fine-grained feature extraction without concerns of network degradation.

In the YOLOv5 model, the Bounding Box Loss function is used to optimize the alignment between the ground truth detection boxes and the model’s predicted output boxes, which is then employed for backpropagation to optimize the model.

The principle of the Focus layer is similar to that of the PassThrough layer. It utilizes slicing operations to divide high-resolution images into multiple low-resolution

TABLE 1. The map value of object detection in the video.

Height(m) Tile angle(°)	2.5	3	3.5	4	4.5	5
0	0.773	0.795	0.779	0.798	0.766	0.765
4	0.772	0.789	0.787	0.795	0.766	0.753
8	0.781	0.732	0.774	0.801	0.768	0.761
12	0.769	0.791	0.773	0.792	0.765	0.764

sub-images/feature mAP, accomplished through column-wise sampling and concatenation. The SPP architecture utilizes spatial pyramid pooling to transform feature mAPs of arbitrary sizes into fixed-size feature vectors. The backbone is a relatively deep network, and the addition of CSP_X modules can increase the gradient values in backpropagation between layers, preventing gradient disappearance caused by the deepening of the network. This allows for more fine-grained feature extraction without concerns of network degradation.

Overall, YOLOv5 is a single-stage object detection algorithm. This algorithm incorporates several new improvements on top of the original algorithm, resulting in significant performance improvements in speed and accuracy. These improvements include Mosaic data augmentation at the input stage, adaptive anchor box calculation, and adaptive image scaling operations. Additionally, the algorithm utilizes the Focus and CSP structures at the backbone stage, SPP and FPN+PAN structures at the neck stage, and GIOU_Loss as the loss function at the output stage, along with DIOU_nms for prediction box filtering.

The following is a method for selecting the optimal camera layout scheme based on the YOLOv5 algorithm in the scenario of a one-way three-lane road. Firstly, 28 image data groups were collected in a one-way, three-lane road with a lane width of 15.5m. The collection rule was to increase the collection height by 0.5m each time and to increase the rotation angle of the camera by 4° at each layout height. The accumulated data was labeled and trained. The initial training effect is shown in Figure 9.

After obtaining the neural network for each scene, various metrics were calculated to evaluate their performance. The mAP values obtained are shown in Table 1:

Based on the mAP values, the preliminary optimal layout scheme was determined to be a mounting height of 4m and an installation angle 8°. Similar layout points were selected, including four layout schemes with mounting heights of 4m and installation angles of 7°, 9°, as well as a mounting height of 4.1m and an installation angle of 8°, and a mounting height of 3.9m and an installation angle of 7°. Subsequently, data collection, annotation, network construction, and mAP value output were performed. After comparing the results, it was found that the layout scheme with the highest accuracy was a mounting height of 4.1m and an installation angle of 8°, with an mAP value of 0.808. Following the steps mentioned above, the optimal layout scheme for this scene was determined to

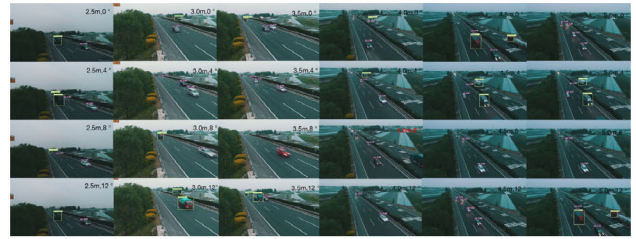


FIGURE 9. Object detection picture of camera.

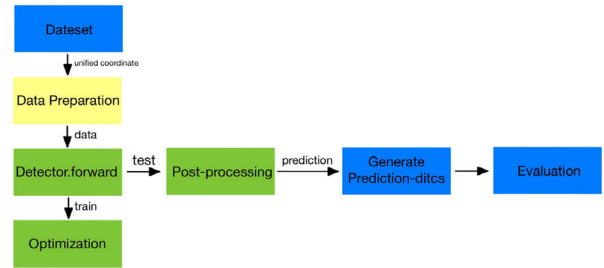


FIGURE 10. PCDet data-model separation code frame.

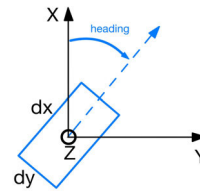


FIGURE 11. Standardized 3D inspection frame used by PCDet.

be a mounting height of 4.3m and an installation angle of 7°, with an improved mAP value of 0.811.

B. POINT CLOUD DATA PROCESSING BASED ON OPENPCDET

Different from image processing, the diversity of datasets in point cloud 3D object detection results in a large amount of data and difficulties in processing. To address this issue, PCDet defines a unified normalized 3D coordinate representation that runs through the entire data processing and model calculation, completely separating the data module from the model processing module. The advantage is that when developing different structured models, a standardized 3D coordinate system is used for various related processes (such as calculating loss, RoI pooling, and post-processing) without considering the differences in coordinate representations of different datasets. Moreover, when adding new datasets, only a tiny amount of code is required to convert the original data to the standardized coordinate definition. PCDet will automatically perform data augmentation and adapt to various models. The top-level design of PCDet's data-model separation enables easy adaptation of various models to different point cloud 3D object detection datasets, eliminating the problem of getting lost in 3D coordinate transformations

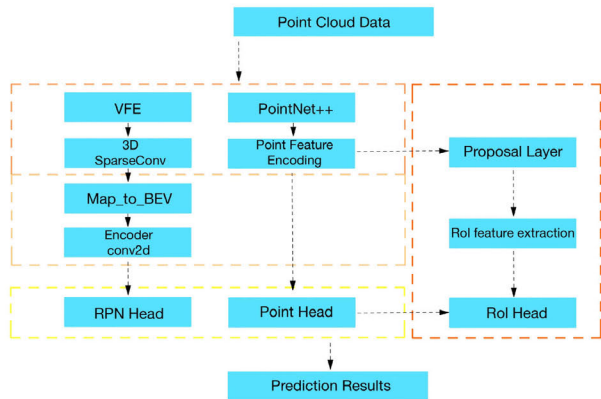


FIGURE 12. 3D object detection framework based on PCDet modularity.

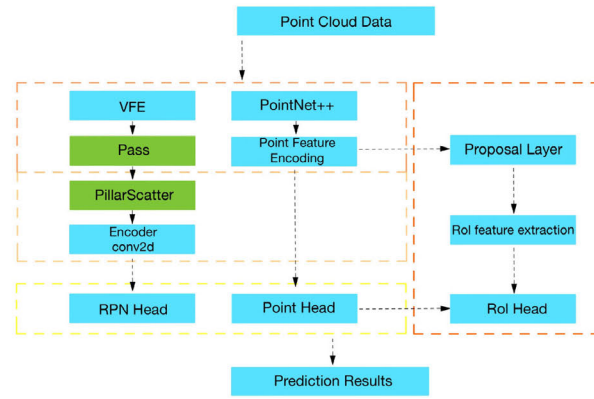


FIGURE 13. Schematic of the operation of the PointPillar algorithm in the PCDet framework.

during model development. The data-model separation code framework of PCDet is shown in Figure 10.

Different point cloud datasets often have inconsistent coordinate systems and definitions for 3D bounding boxes. Therefore, a fixed unified point cloud coordinate system is used in PCDet, as shown in Figure 11, and a more consistent definition for 3D detection bounding boxes is used to ensure consistency throughout the entire data augmentation, processing, model computation, and post-detection processing pipeline.

Among them, (cx, cy, cz) represents the geometric center position of the 3D bounding box, and (dx, dy, dz) represents the lengths of the 3D bounding box along the $x, y,$ and z directions when the heading angle is 0. The heading angle refers to the orientation angle of the object in the top-down view (0 degrees along the x -axis direction, increasing counterclockwise from x to y).

With the standardized 3D bounding box definition adopted by PCDet, there is no need to worry about whether it represents the center of the object or the bottom center of the object. There is also no need to worry about the arrangement of the object’s three-dimensional dimensions in terms of length-width-height or width-length-height. Furthermore, there is no need to be concerned about the exact orientation of the 0-degree heading angle and whether it increases in a clockwise or counterclockwise direction. Based on the flexible and comprehensive modular design shown in Figure 12, building a 3D object detection framework in PCDet only requires writing a configuration file to clearly define the required modules. PCDet will then automatically combine these modules in the topological order to construct a 3D object detection framework for training and testing purposes.

In the OpenPCDet framework, PointPillars algorithm is selected as the detection algorithm for target detection. A schematic diagram of the PointPillar algorithm in operation in the PCDet framework is shown in Figure 13.

The PointPillars algorithm transforms point cloud data into two-dimensional images using a Pillar structure, significantly reducing computational complexity and memory consumption. Compared to other point cloud-based algorithms,

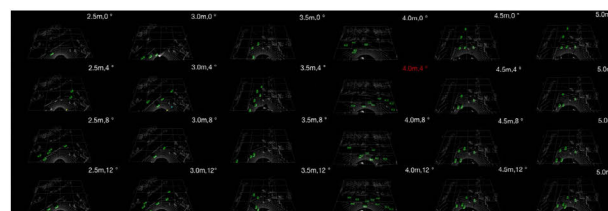


FIGURE 14. Object detection picture of LiDAR.

TABLE 2. The map value of object detection in the point cloud.

Height(m) Tile angle(°)	2.5	3	3.5	4	4.5	5
0	0.773	0.795	0.779	0.798	0.766	0.765
4	0.772	0.789	0.787	0.795	0.766	0.753
8	0.781	0.732	0.774	0.801	0.768	0.761
12	0.769	0.791	0.773	0.792	0.765	0.764

PointPillars achieves higher detection speed while maintaining accuracy. The algorithm employs a unique point cloud encoding method to extract feature information from the point cloud effectively. Additionally, multiple convolutional neural networks are utilized to capture objects of different sizes and shapes better. PointPillars algorithm is adaptable to various scenarios and targets as it does not require complex preprocessing or prior knowledge input. Therefore, the PointPillars algorithm is selected as the target detection algorithm in this study.

The set of M schemes includes 3D bounding boxes $B = \{b_i\}_{i=1}^M$, where $b_i = \{x, y, z, l, h, w, \theta\}$. Here, $x, y,$ and z represent the center position of the box, $l, h,$ and w represent the dimensions (length, height, and width) of the box, and θ represents the orientation angle of the box. And confidence scores $S = \{S_i\}_{i=1}^M$ obtained from any one-stage detectors. By utilizing the point cloud P and camera image $I = \{I_i \in \mathbb{R}^{3 \times H_i \times W_i}\}_{i=1}^T$, the detection results can be enhanced.

$$(B_r, S_r) = R(B, P, I) \tag{5}$$

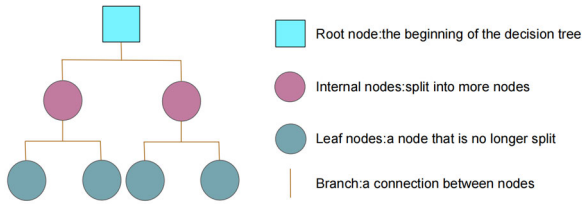


FIGURE 15. Structure diagram of the decision tree.

In the equation, B_r and S_r represent the corrected bounding boxes and confidence scores, respectively. R represents the proposed network.

In this chapter, point cloud data was collected in a deployed scenario, and the PointPillars algorithm from OpenPCDet was used as the network training and inference tool. The example scenario considered is a bidirectional eight-lane road. Firstly, 28 sets of image data were collected on a unidirectional three-lane road with a lane width of 16.3m. The data collection followed the same rules as the camera setup, with the height incrementing by 0.5m each time and the rotation angle of the LiDAR increasing by 4° at each height setting. The accumulated data was labeled and used for training. The detection diagram is shown in Figure 14.

After obtaining the neural network for each scenario, various metrics were calculated. Table 2 shows the calculated mAP values.

Based on the mAP values, the preliminary optimal layout scheme is determined to be a mounting height of 3.5m and an installation angle of 4° . The similar layout points selected are: mounting height of 3.5m with installation angles of 3° and 5° , mounting height of 3.4m with an installation angle of 4° , and mounting height of 3.6m with an installation angle of 4° . Subsequently, through data collection, annotation, network construction, and mAP value output, a comparison is made, and the layout scheme with the highest accuracy is identified as a mounting height of 3.6m and an installation angle of 4° , with an mAP value of 0.595. Following these steps, the final optimal layout scheme for this scenario is determined to be a mounting height of 3.7m and an installation angle of 6° , resulting in an mAP value of 0.595.

The road parameters are independent variables, including lane width, slope, curve radius, and installation distance from the road. The installation parameters of roadside devices include the installation height of LiDAR, the installation height of the camera, the installation tilt angle of LiDAR (around the X-axis), the installation tilt angle of the camera (around the X-axis), the installation tilt angle of LiDAR (around the Y-axis), and the installation tilt angle of the camera (around the Y-axis). The dataset records the optimal layout solutions for 360 road scenarios. Accurately predicting layout solutions for different road scenarios based on existing data is a pressing issue.

IV. EVALUATION

To achieve optimal layout compatibility of camera and LiDAR in different scenarios, this study conducted data

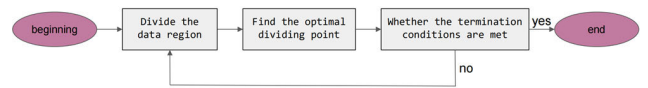


FIGURE 16. Decision tree flowchart.

regression and prediction processing on the experimental database obtained. The processed dataset includes four independent variables and six dependent variables. To make more accurate predictions for the combined layout schemes, this paper selected five popular linear regression algorithms for comparative experiments: random forest algorithm, decision tree algorithm, BP neural network algorithm, SVR neural network algorithm, and linear regression algorithm. The operational parameters of each algorithm were adjusted to achieve optimal fitting accuracy. Then, a quarter of the independent variable data was extracted from the dataset and inputted into the algorithm models. The primary evaluation metrics for the performance of the regression prediction model include mean squared error (MSE), root mean squared error (RMSE), mean absolute error (MAE), and coefficient of determination (R^2). In this study, MSE, MAE, and R^2 are utilized to gauge the model's performance.

A. DECISION TREE

Decision tree is a classification and regression algorithm based on a tree-like structure [31]. Its principle involves recursively partitioning the dataset selecting the optimal feature at each node for splitting until a predefined termination condition is met. For a given overall training set T , consisting of N samples, the random selection of N samples with replacement (since it is with replacement, it is impossible to traverse all samples) is performed. These selected N samples are then used to train a decision tree. Firstly, from the total N samples in T , N samples are randomly selected with replacement. These selected N samples are used to train a decision tree as the samples at the root node of the tree. Secondly, when each sample has M attributes and a node in the decision tree needs to split, m attributes are randomly chosen from the available M attributes, satisfying the condition $m \ll M$. Then, a specific strategy is employed to select one attribute from these m attributes as the splitting attribute for that node [32]. The process of forming the decision tree continues, with each node undergoing the split step, following the procedure above, until further splitting is no longer possible. The structure of the decision tree is shown in Figure 15.

Initially, the voting process starts from the initial position and all the data is partitioned into a single node. Then, go through the following two steps: If the data is an empty set, exit the loop. If the node is the root node, return null. If the node is an internal node, label it with the class that appears most frequently in the training data. If all samples belong to the same class, exit the loop. The flowchart for decision tree regression is shown below.

The algorithm begins by partitioning the sample set into the root node. Then it searches for the optimal split point, which

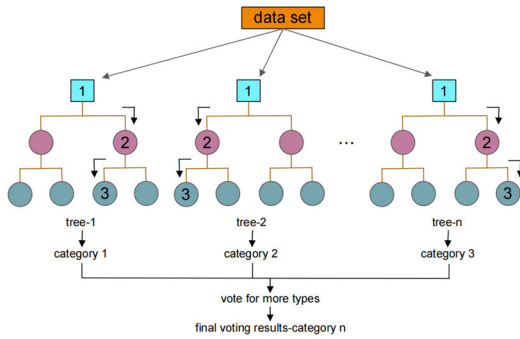


FIGURE 17. Structure diagram of a random forest.

refers to a certain value (s) of a certain attribute (j). The data is then divided into child nodes using the optimal split point (j, s), with the attribute values less than s assigned to one node and those greater than s assigned to another node. The output value of each node is the average of the samples within it. The regression decision tree seeks the optimal splitting point (j, s) by utilizing the least squares method. The following objective function is defined:

$$\min_{j,s} \left[\sum_{x_i \in R_1(j,s)} (y_i - c_1)^2 + \sum_{x_i \in R_2(j,s)} (y_i - c_2)^2 \right] \quad (6)$$

The algorithm will iterate over each attribute and its corresponding values (j, s). When reaching attribute j and value s, the sample set will be divided into two subsets ($x \leq s$ and $x > s$), denoted as R1 and R2. The algorithm aims to find the optimal values C1 and C2 that minimize the sum of squared errors for R1 and R2, respectively. It can be mathematically proven that C1 and C2 should be chosen as the mean values of y in subsets R1 and R2, respectively. The equation can also be written as follows:

$$\min_{j,s} \left[\sum_{x_i \in R_1(j,s)} (y_i - c_1)^2 + \sum_{x_i \in R_2(j,s)} (y_i - c_2)^2 \right] \quad (7)$$

For each (j, s), we obtain a numerical value based on the aforementioned equation. Subsequently, we select the (j, s) pair that minimizes the value of the equation as the optimal splitting point. The algorithm then checks if the termination conditions are met; if so, it ends, otherwise it continues to partition. There are various termination conditions, such as the minimum number of samples in a child node or the maximum depth of the decision tree. Finally, the decision tree is formed.

B. RANDOM FOREST

Random forest is a specific implementation of the bagging method that trains multiple decision trees and combines the results to produce the final output [33]. For regression problems, the predicted result of random forest is the mean output of all decision trees. Compared to a single decision tree algorithm, random forest performs well on datasets. The introduction of two sources of randomness makes it less likely to overfit, although overfitting is still possible

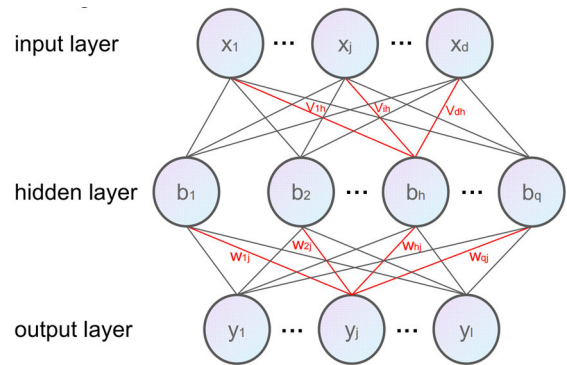


FIGURE 18. BP neural network architecture diagram.

for small datasets. Furthermore, random forest can handle high-dimensional data without feature selection, adapt to datasets strongly, and process discrete and continuous data. The data sets being processed do not need to be normalized [34]. Random Forest requires the combination of multiple decision trees to achieve regression tasks. It involves constructing multiple decision trees and combining them into a regression model. Firstly, a random subset of samples is selected as the training set for each decision tree. Then, a portion of features (the square root of the total number of features) is randomly chosen as the feature set for each decision tree. Consequently, decision trees are built based on the training set and feature set until a predetermined number of leaf nodes is reached or further division is not possible. This procedure is repeated to build multiple decision trees. For a new sample, it is inputted into each decision tree, resulting in multiple prediction outcomes. Finally, the average of the multiple prediction outcomes is calculated to obtain the final prediction result. The algorithm formula is based on the decision tree regression model. The prediction function for each decision tree can be represented as shown in Equation (8):

$$f_k(x) = \sum_{j=1}^{J_k} c_{kj} I(x \in R_{kj}) \quad (8)$$

In the equation, “k” represents the k-th decision tree, “X” denotes the input sample, “L” represents the number of leaf nodes in the k-th decision tree, “Y” signifies the predicted value of the i-th leaf node in the k-th decision tree, and “D” represents the sample set associated with the i-th leaf node in the k-th decision tree. The prediction function for multiple decision trees can be expressed as:

$$f(x) = \frac{1}{K} \sum_{K=1}^K f_k(x) \quad (9)$$

where K denotes the number of decision trees. A schematic diagram of the structure of the random forest is shown in Figure 17.

Numerous studies have shown that ensemble classifiers outperform individual classifiers regarding classification

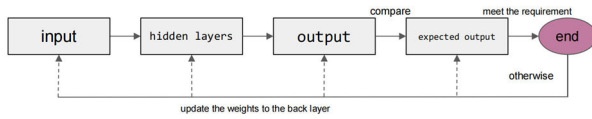


FIGURE 19. BP neural network flow diagram.

performance. Random Forest is a method that utilizes multiple classification trees to discriminate and classify data. It not only performs classification on the data but also provides importance scores for each variable, evaluating their respective contributions to the classification process.

C. BP NEURAL NETWORK

The BP neural network can be divided into two components, namely backpropagation (BP) and the neural network itself [35]. BP stands for Back Propagation. The BP network is capable of learning and storing a large number of input-output pattern mappings without the need for revealing explicit mathematical equations describing these mappings in advance. Its learning rule employs the method of steepest descent, adjusting the network’s weights and thresholds continuously through backward propagation to minimize the sum of squared errors [36]. The infrastructure of the BP neural network is shown in Figure 18.

The input layer serves as the entry point for information. In contrast, the hidden layer acts as the processing unit and allows for the specification of its number of layers. Finally, the output layer serves as the information output. The weights from the input layer to the hidden layer and from the hidden layer to the output layer are denoted as v and w , respectively. For the neural network model depicted in the figure with only one hidden layer, the process of the BP neural network can be divided into two stages. The BP neural network flowchart is shown in Figure 19.

$$f(x) = \frac{1}{K} \sum_{k=1}^K f_k(x) \tag{10}$$

The first stage is the forward propagation of signals, which starts from the input layer, passes through the hidden layer and reaches the output layer. The second stage is the backward propagation of errors, which flows from the output layer, through the hidden layer, and ultimately reaches the input layer. During this stage, the weights and biases between the hidden layer and the output layer, as well as those between the input layer and the hidden layer, are adjusted sequentially.

D. SVR NEURAL NETWORK

SVR regression finds a regression plane that makes the distance between all data points in a set and the plane as close as possible. Unlike general regression models, the support vector regression (SVR) allows for a certain degree of bias in the model [37]. The points within the bias range are not considered problematic by the model, while those outside are included in the loss calculation. Therefore, for

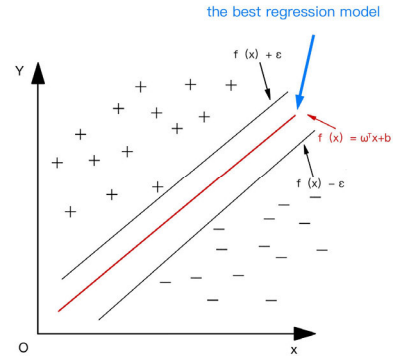


FIGURE 20. SVR neural network schematic.

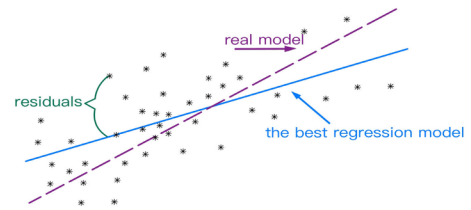


FIGURE 21. Schematic diagram of a linear regression algorithm.

SVR, the points within the support vector range affect the model, while the points outside the support vector range are used to calculate the loss. This regression method is effective in solving classification and regression problems with high-dimensional features and has good performance even when the feature dimension exceeds the number of samples. For databases with non-massive data samples, SVR has high classification accuracy and strong generalization ability [38]. A schematic of regression for the SVR neural network is shown in Figure 20.

The formula for Support Vector Regression (SVR) is represented as $\omega^T x + b = 0$, with ϵ denoting the fitting accuracy. In SVR, values within the dashed line are considered correctly predicted, and the loss is calculated only for values outside the dashed line. Taking into account the scenario of linear inseparability in SVM, the introduction of slack variables $\epsilon_i \geq 0$ and $\epsilon_i^* \geq 0$ leads to the final optimization problem of Support Vector Machine Regression:

$$\min_{(\omega, b, \epsilon)} \frac{1}{2} \|\omega\|^2 + C \sum_{i=1}^n (\epsilon_i + \epsilon_i^*) \tag{11}$$

$$\omega^T x_i + b - y_i \leq \epsilon + \epsilon_i^* \tag{12}$$

The linear regression function can be obtained by introducing Lagrange multipliers and solving a series of dual problems. The resulting function is given by:

$$f(x) = \omega^T x + b = \sum_{i=1}^n (\alpha_i + \alpha_i^*) (x, x_i) + b \tag{13}$$

In the aforementioned equations, α_i and α_i^* represent the Lagrange multipliers. By introducing a kernel function, we have the following expression:

$$f(x) = \omega^T x + b = \sum_{i=1}^n (\alpha_i + \alpha_i^*) K(x, x_i) + b \tag{14}$$

TABLE 3. Evaluation table for different algorithm models.

Models	LiDAR height			LiDAR inclination(X)			LiDAR inclination(Y)			Camera height			camera inclination(X)			camera inclination(Y)		
	R2	MSE	MAE	R2	MSE	MAE	R2	MSE	MAE	R2	MSE	MAE	R2	MSE	MAE	R2	MSE	MAE
Decision tree	0.95	0.97	0.86	0.93	1.04	0.93	0.92	1.61	1.38	0.93	1.95	0.99	0.93	0.98	0.71	0.97	0.64	0.61
Random forest	0.97	0.63	0.43	0.95	0.89	0.73	0.98	0.64	0.61	0.97	0.71	0.43	0.88	1.82	1.43	0.95	1.32	1.12
BP	0.93	1.13	0.98	0.95	0.71	0.57	0.93	1.77	1.66	0.96	0.98	0.78	0.98	0.53	0.55	0.99	0.30	0.29
SVR	0.86	1.56	1.04	0.98	0.59	0.52	0.89	2.26	1.94	0.89	3.54	1.21	0.95	0.66	0.61	0.88	1.86	1.65
Linear regression	0.91	1.27	1.18	0.91	1.65	1.02	0.97	0.89	0.72	0.82	4.27	3.93	0.97	0.67	0.81	0.93	1.09	0.97

Therefore, the advantage of the SVR neural network lies in its ability to handle nonlinear and high-dimensional data while exhibiting good generalization performance. By setting appropriate parameters, the SVR neural network can flexibly control the model’s complexity and tolerance, thus adapting to different regression tasks. Furthermore, the SVR neural network can help address common regression issues such as overfitting and underfitting, thereby improving prediction accuracy.

E. LINEAR REGRESSION

Regression is a widely employed predictive modeling technique where the core aspect lies in predicting continuous variables [39]. It is an essential problem in supervised learning, aiming to predict the relationship between input and output variables. Specifically, it investigates how the values of the output variable change with variations in the input variables. A regression model represents the function mapping from input variables to output variables. Learning a regression problem is equivalent to function fitting: selecting a function curve that fits known data well and predicts unknown data accurately. In statistics, linear regression is a regression analysis that models the relationship between one or more independent variables and a dependent variable using a linear regression equation, also known as the least squares function. This function is a linear combination of model parameters called regression coefficients. The case with only one independent variable is called simple regression, while the scenario with multiple independent variables is known as multivariate regression [40]. A schematic of linear regression is shown in Figure 21.

When modeling income datasets, it is necessary to construct different lines, forming a family of parameters. Within this parameter family, there exists an optimal combination that can statistically describe the dataset in the best possible way. The process of supervised learning can be defined as the task of finding the best parameters given the data, enabling the model to fit the data better. Linear regression is a function that predicts the outcome by utilizing a linear combination of attributes, which can be expressed as follows:

$$f(X) = \omega_1 x_1 + \omega_2 x_2 + \dots + \omega_d x_d + b \tag{15}$$

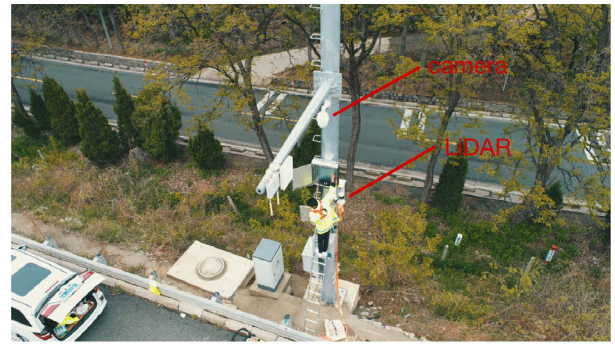


FIGURE 22. Installation diagram of roadside equipment pile points.

In the above equation, we are given an example $X = (x_1; x_2; \dots; x_d)$ described by d attributes, where x_i represents the value of x for the i -th attribute. In vector form, it can be expressed as:

$$f(x) = W^T x + b \tag{16}$$

The meaning of $W = (\omega_1; \omega_2; \dots; \omega_d)$ can be interpreted as the weight parameter of each feature. To find the optimal fitting line, machine learning employs a loss function to measure this problem. The loss function, also known as the cost function, quantifies the discrepancy between the model’s predicted values and the actual values. The training of a linear regression model aims to make the predicted score values close to the true values. The key to this process lies in defining a loss function that measures the difference between the predicted values and the true values. Mean squared error (MSE) is the most commonly used performance metric in regression tasks, which can be defined as:

$$\text{loss} = \frac{1}{m} \sum_{i=1}^m (f(x) - y)^2 = \frac{1}{m} \sum_{i=1}^m (y - wx - b)^2 \tag{17}$$

Thus, we can attempt to minimize the mean square error and solve for the parameters w and b that minimize the loss value. A more significant loss function indicates a poorer model performance and a remarkable inability to fit the data.

The decision tree algorithm can deal with nonlinear relationships and exhibits robustness in handling missing values

and outliers. However, its susceptibility to overfitting, especially in complex datasets, should be noted. In contrast, the random forest algorithm effectively handles many features and samples. The BP neural network regression algorithm is well-suited for addressing complex problems and pattern recognition tasks due to its high flexibility and generalization ability. However, it requires a substantial amount of data and careful selection of hyperparameters, as well as a slower training speed. SVR regression, on the other hand, is effective in handling high-dimensional data and nonlinear problems. It demonstrates robustness in the presence of a small amount of noise and outliers, but training on large-scale datasets may be time-consuming. The regression equation algorithm is simple and easy to understand, allowing for quick model construction and predictions. However, its ability to fit nonlinear relationships is relatively limited. Additionally, it should be noted that different algorithms may produce varying results when applied to the same database. The R2, MSE, and MAE parameters for different algorithms on the same dataset are shown in Table 3. The R2 represents the coefficient of determination, with a higher R2 indicating a better fit for the model. Meanwhile, MSE represents the mean squared error and MAE represents the mean absolute error. Smaller values of these parameters indicate lower error rates for the model. This paper employs these three evaluation metrics as standards for assessing the model's performance. According to Table 3, random forests have the highest fit rate for the installation height of the LiDAR and camera, as well as the tilt angle of the LiDAR (around the Y-axis). The BP neural network regression algorithm has the best performance in handling the installation tilt angle data of the camera. The SVM neural network algorithm performs best in handling the tilt angle of the LiDAR (around the X-axis). In this study, these three algorithms are used to process data from different groups to obtain the most accurate prediction results.

V. SCENARIO VERIFICATION

To validate the feasibility of the proposed installation method, we selected a single three-lane road scenario and conducted measurements on the road parameters in this scenario. The road width was 11.5m, the distance between the pillars and the road was 0.7m, the turning radius was 0m, and the road slope was 2.3°. The aforementioned data were separately input into the random forest, BP neural network, and SVR neural network models to predict six installation parameters. The predicted installation height, tilt angle around the X-axis, and tilt angle around the Y-axis for the camera were 4.1m, 7.1°, and 2.3°, respectively. The predicted installation height, tilt angle around the X-axis, and tilt angle around the Y-axis for the LiDAR were 3.22m, 6.5°, and 2.4°, respectively. The LiDAR and camera were installed on the roadside according to this layout scheme, as shown in Figure 22.

Under this installation scheme, the YOLOv5 algorithm and the PointPillars algorithm in the OpenPCDet framework were separately used to perform target detection on the collected video and point cloud data of passing vehicles. The obtained

mAP values for the video and LiDAR were 0.823 and 0.627, respectively. Following the experimental process in Chapter 2 and the optimal installation method in Chapter 3, the optimal installation scheme for this scenario was determined. The installation height, tilt angle around the X-axis, and tilt angle around the Y-axis for the camera were 4.1m, 7°, and 2.3°, respectively. The installation height, tilt angle around the X-axis, and tilt angle around the Y-axis for the LiDAR were 3.2m, 6°, and 2.3°, respectively. Moreover, under this installation scheme, the mAP values for the camera and LiDAR were 0.824 and 0.626, respectively. The experimental results demonstrate that the proposed optimal installation method for roadside LiDAR and camera can quickly and accurately output the optimal layout scheme at the centimeter level.

VI. CONCLUSION

In this study, we investigate the optimal layout of roadside LiDAR and cameras. Firstly, experiments are designed to compare the factors of lane slope and turning lane with the straight lane, considering the actual road construction process. Then, parameter measurements are conducted on the road under different scenarios. Different installation schemes for LiDAR and camera are applied on the road, and point cloud data and video data are collected separately. Subsequently, the Intersection over Union (IOU) value for object detection is set to 0.5. Object detection is then performed using the PointPillars algorithm based on the OpenPCDet framework on the Ubuntu system, as well as the YOLOv5 algorithm on the Windows 11 system. The mAP values of the neural networks are obtained for each layout scheme. The average accuracy of vehicle detection is used as a criterion to measure the priority of different layout schemes. A unique optimal layout scheme is determined for each road scenario. The layout schemes, including the four independent and six dependent variables, are summarized in a database to construct a high-precision combined layout model. In this paper, the decision tree algorithm, random forest algorithm, BP neural network algorithm, SVR neural network algorithm, and regression linear algorithm are introduced. Multiple regression models, each with one independent variable and six dependent variables, are constructed. One-fourth of the independent variable data is randomly selected as input for each model. The output data is compared with the original data in the database. In this study, the performance of the model is evaluated using Mean Squared Error (MSE), Mean Absolute Error (MAE), and R-squared (R2). For each predicted installation parameter, the model with the maximum R2 value and the minimum MSE and MAE values is selected as the single-parameter prediction model. For example, in the case of the installation height parameter of the laser radar, the Random Forest algorithm with an R2 value of 0.97, MSE value of 0.63, and MAE value of 0.43 was chosen. Similarly, the best-performing Random Forest algorithm was selected for the installation height of the laser radar, rotation angle around the Y-axis, and camera precision. The SVR algorithm was chosen for the rotation angle around the X-axis of the

laser radar. The BP neural network algorithm was selected for the camera rotation angle parameter. These selected models were integrated into an overall prediction model, thus completing the method for the layout of the combination of laser radar and camera. Subsequently, experiments are conducted in the actual field to deploy LiDAR and cameras, and the mAP values for vehicle detection are obtained. The experimental process described in Chapter 2 and Chapter 3 of this paper is replicated in this specific scene. The layout schemes and mAP values obtained from the experiments are then compared with the layout schemes predicted by the prediction model and the mAP values obtained from the actual scene. The results indicate that the optimal layout solutions for laser radar and camera predicted by the optimal deployment method achieved mAP values of 0.823 and 0.627, respectively, for road vehicle target detection. Compared to the optimal layout solutions determined in chapters two and three, the mAP value errors do not exceed 1%. This experiment demonstrates that the proposed optimal deployment method for laser radar and camera not only improves target detection accuracy by changing installation parameters but also accurately predicts the optimal deployment solutions in unknown scenarios based on known scene layouts. It provides new ideas and guidance for the optimal deployment of future roadside intelligent perception devices.

However, there are certain aspects that can be improved in this study. For instance, the data acquisition and processing of roadside intelligent perception devices in this research require significant human and material resources. Additionally, using YOLOv5s and PointPillars as the computational weights for object detection may lead to performance variations and hinder obtaining more accurate optimal layout strategies in different scenarios. Training weights for different installation conditions can greatly enhance overall accuracy. Furthermore, the prediction model is limited to fixed models or similar parameter combinations of LiDAR and cameras, making it unsuitable for all LiDAR and camera configurations. Moreover, although this optimal deployment method considers various road scenarios such as sloping sections and curved sections, it falls short of being applicable to various other scenarios like merging/diverging areas and intersections. We believe that this research approach can also be extended to other roadside sensors such as RSUs and millimeter-wave radars.

REFERENCES

- [1] J. Wu, H. Xu, Y. Tian, Y. Zhang, J. Zhao, and B. Lv, "An automatic lane identification method for the roadside light detection and ranging sensor," *J. Intell. Transp. Syst.*, vol. 24, no. 5, pp. 467–479, Sep. 2020.
- [2] J. Wu, Y. Tian, H. Xu, R. Yue, A. Wang, and X. Song, "Automatic ground points filtering of roadside LiDAR data using a channel-based filtering algorithm," *Opt. Laser Technol.*, vol. 115, pp. 374–383, Jul. 2019.
- [3] J. Wu, Y. Zhang, and H. Xu, "A novel skateboarder-related near-crash identification method with roadside LiDAR data," *Accident Anal. Prevention*, vol. 137, Mar. 2020, Art. no. 105438.
- [4] Y. Cui, J. Wu, H. Xu, and A. Wang, "Lane change identification and prediction with roadside LiDAR data," *Opt. Laser Technol.*, vol. 123, Mar. 2020, Art. no. 105934.
- [5] T.-H. Kim, G.-H. Jo, H.-S. Yun, K.-S. Yun, and T.-H. Park, "Placement method of multiple lidars for roadside infrastructure in urban environments," *Sensors*, vol. 23, no. 21, p. 8808, Oct. 2023, doi: 10.3390/s23218808.
- [6] M. M. Hussain, A. T. Azar, R. Ahmed, S. U. Amin, B. Qureshi, V. D. Reddy, I. Alam, and Z. I. Khan, "SONG: A multi-objective evolutionary algorithm for delay and energy aware facility location in vehicular fog networks," *Sensors*, vol. 23, no. 2, p. 667, Jan. 2023, doi: 10.3390/s23020667.
- [7] E. Twahirwa, M. Laha, J. Rwigema, and R. Datta, "Travel matrix enabled delta-based roadside units deployment for vehicular ad hoc networks: A case of Kigali city," *J. Adv. Transp.*, vol. 2022, pp. 1–12, Mar. 2022, doi: 10.1155/2022/2809184.
- [8] W. Ahmad, G. Husnain, S. Ahmed, F. Aadil, and S. Lim, "Received signal strength-based localization for vehicle distance estimation in vehicular ad hoc networks (VANETs)," *J. Sensors*, vol. 2023, pp. 1–15, Mar. 2023, doi: 10.1155/2023/7826992.
- [9] S. Kumar, S. K. Singh, S. Varshney, S. Singh, P. Kumar, B.-G. Kim, and I.-H. Ra, "Fusion of deep sort and Yolov5 for effective vehicle detection and tracking scheme in real-time traffic management sustainable system," *Sustainability*, vol. 15, no. 24, p. 16869, Dec. 2023, doi: 10.3390/su152416869.
- [10] R. Arifando, S. Eto, and C. Wada, "Improved YOLOv5-based lightweight object detection algorithm for people with visual impairment to detect buses," *Appl. Sci.*, vol. 13, no. 9, p. 5802, May 2023, doi: 10.3390/app13095802.
- [11] J. Mun, J. Kim, Y. Do, H. Kim, C. Lee, and J. Jeong, "Design and implementation of defect detection system based on YOLOv5-CBAM for lead tabs in secondary battery manufacturing," *Processes*, vol. 11, no. 9, p. 2751, Sep. 2023, doi: 10.3390/pr11092751.
- [12] N. Kumar, Nagarathna, and F. Flammini, "YOLO-based light-weight deep learning models for insect detection system with field adaption," *Agriculture*, vol. 13, no. 3, p. 741, Mar. 2023, doi: 10.3390/agriculture13030741.
- [13] D. S. Prasvita, D. Chahyati, and A. M. Arymurthy, "Automatic detection of oil palm growth rate status with YOLOv5," *Int. J. Adv. Comput. Sci. Appl.*, vol. 14, no. 3, pp. 529–537, 2023, doi: 10.14569/ijacsa.2023.0140361.
- [14] P. Hofinger, H.-J. Klemmt, S. Ecke, S. Rogg, and J. Dempewolf, "Application of YOLOv5 for point label based object detection of black pine trees with vitality losses in UAV data," *Remote Sens.*, vol. 15, no. 8, p. 1964, Apr. 2023, doi: 10.3390/rs15081964.
- [15] E. Y. Obsie, H. Qu, Y.-J. Zhang, S. Annis, and F. Drummond, "YOLOv5s-CA: An improved YOLOv5 based on the attention mechanism for mummy berry disease detection," *Agriculture*, vol. 13, no. 1, p. 78, Dec. 2022, doi: 10.3390/agriculture13010078.
- [16] E. C. Nnadozie, O. N. Iloanus, O. A. Ani, and K. Yu, "Detecting cassava plants under different field conditions using UAV-based RGB images and deep learning models," *Remote Sens.*, vol. 15, no. 9, p. 2322, Apr. 2023, doi: 10.3390/rs15092322.
- [17] S. Y. Alaba and J. E. Ball, "WCNN3D: Wavelet convolutional neural network-based 3D object detection for autonomous driving," *Sensors*, vol. 22, no. 18, p. 7010, Sep. 2022, doi: 10.3390/s22187010.
- [18] A. Silva, D. Fernandes, R. Névoa, J. Monteiro, P. Novais, P. Girão, T. Afonso, and P. Melo-Pinto, "Resource-constrained onboard inference of 3D object detection and localisation in point clouds targeting self-driving applications," *Sensors*, vol. 21, no. 23, p. 7933, Nov. 2021, doi: 10.3390/s21237933.
- [19] J. Zhang, J. Wang, D. Xu, and Y. Li, "HCNET: A point cloud object detection network based on height and channel attention," *Remote Sens.*, vol. 13, no. 24, p. 5071, Dec. 2021, doi: 10.3390/rs13245071.
- [20] L. Zhang, H. Meng, Y. Yan, and X. Xu, "Transformer-based global PointPillars 3D object detection method," *Electronics*, vol. 12, no. 14, p. 3092, Jul. 2023, doi: 10.3390/electronics12143092.
- [21] A. R. Singh, *PointPillars++: An Encoder for 3-D Object Detection and Classification From Point Clouds*, document 28747751, ProQuest Diss. Theses Global; ProQuest Diss. Theses Global A&I: Humanities Social Sci. Collection; ProQuest Diss. Theses Global A&I: Sci. Eng. Collection (2595127407), 2021. [Online]. Available: <https://www.proquest.com/dissertations-theses/pointpillars-encoder-3-d-object-detection/docview/2595127407/se-2>
- [22] K. Hariya, H. Inoshita, R. Yanase, K. Yoneda, and N. Suganuma, "ExistenceMap-PointPillars: A multifusion network for robust 3D object detection with object existence probability map," *Sensors*, vol. 23, no. 20, p. 8367, Oct. 2023, doi: 10.3390/s23208367.

- [23] J. Yang, S. Han, and Y. Chen, "Prediction of traffic accident severity based on random forest," *J. Adv. Transp.*, vol. 2023, pp. 1–8, Feb. 2023.
- [24] V. Jain and A. Phophalia, "M-ary random forest—A new multidimensional partitioning approach to random forest," *Multimedia Tools Appl.*, vol. 80, nos. 28–29, pp. 35217–35238, Nov. 2021, doi: [10.1007/s11042-020-10047-9](https://doi.org/10.1007/s11042-020-10047-9).
- [25] H. Zhang, Y. Li, and L. Yan, "Prediction model of car ownership based on back propagation neural network optimized by particle swarm optimization," *Sustainability*, vol. 15, no. 4, p. 2908, Feb. 2023.
- [26] J. D. Borrero and J. Mariscal, "Elevating univariate time series forecasting: Innovative SVR-empowered nonlinear autoregressive neural networks," *Algorithms*, vol. 16, no. 9, p. 423, Sep. 2023.
- [27] S. Chaudhary, A. Bhardwaj, and P. Rana, "Image enhancement by linear regression algorithm and sub-histogram equalization," *Multimedia Tools Appl.*, vol. 81, no. 21, pp. 29919–29938, Sep. 2022.
- [28] J. Wu et al., "A variable dimension-based method for roadside LiDAR background filtering," *IEEE Sensors J.*, vol. 22, no. 1, pp. 832–841, Jan. 2022, doi: [10.1109/JSEN.2021.3125623](https://doi.org/10.1109/JSEN.2021.3125623).
- [29] J. Wu, H. Xu, J. Zheng, and J. Zhao, "Automatic vehicle detection with roadside LiDAR data under rainy and snowy conditions," *IEEE Intell. Transp. Syst. Mag.*, vol. 13, no. 1, pp. 197–209, Spring. 2021, doi: [10.1109/MITS.2019.2926362](https://doi.org/10.1109/MITS.2019.2926362).
- [30] J. Wu, H. Xu, Y. Tian, R. Pi, and R. Yue, "Vehicle detection under adverse weather from roadside LiDAR data," *Sensors*, vol. 20, no. 12, p. 3433, Jun. 2020, doi: [10.3390/s20123433](https://doi.org/10.3390/s20123433).
- [31] A. I. Weinberg and M. Last, "Selecting a representative decision tree from an ensemble of decision-tree models for fast big data classification," *J. Big Data*, vol. 6, no. 1, pp. 1–17, Dec. 2019, doi: [10.1186/s40537-019-0186-3](https://doi.org/10.1186/s40537-019-0186-3).
- [32] S. Ma and J. Zhai, "Big data decision tree for continuous-valued attributes based on unbalanced cut points," *J. Big Data*, vol. 10, no. 1, p. 135, Aug. 2023, doi: [10.1186/s40537-023-00816-2](https://doi.org/10.1186/s40537-023-00816-2).
- [33] S. Ratnasingam and J. Muñoz-Lopez, "Distance correlation-based feature selection in random forest," *Entropy*, vol. 25, no. 9, p. 1250, Aug. 2023, doi: [10.3390/e25091250](https://doi.org/10.3390/e25091250).
- [34] M. R. Camana, C. E. Garcia, T. Hwang, and I. Koo, "A REM update methodology based on clustering and random forest," *Appl. Sci.*, vol. 13, no. 9, p. 5362, Apr. 2023, doi: [10.3390/app13095362](https://doi.org/10.3390/app13095362).
- [35] Q. Zhang, Y. Shen, Y. Pei, X. Wang, M. Wang, and J. Lai, "Determination of integrity index K_1 in CHN-BQ method by BP neural network based on fractal dimension D_f ," *Fractal Fractional*, vol. 7, no. 7, p. 546, Jul. 2023, doi: [10.3390/fractalfrac7070546](https://doi.org/10.3390/fractalfrac7070546).
- [36] H. Chen, "Research on modeling and dynamic characteristics of complex biological neural network model considering BP neural network method," *Adv. Multimedia*, vol. 2021, pp. 1–8, Dec. 2021, doi: [10.1155/2021/7646121](https://doi.org/10.1155/2021/7646121).
- [37] S. R. Jondhale, V. Mohan, B. B. Sharma, J. Lloret, and S. V. Athawale, "Support vector regression for mobile target localization in indoor environments," *Sensors*, vol. 22, no. 1, p. 358, Jan. 2022, doi: [10.3390/s22010358](https://doi.org/10.3390/s22010358).
- [38] Y. Liu, "Prediction of structural damage trends based on the integration of LSTM and SVR," *Appl. Sci.*, vol. 13, no. 12, p. 7135, 2023, doi: [10.3390/app13127135](https://doi.org/10.3390/app13127135).
- [39] N. Gündüz and B. Torsney, "D-optimal designs for binary and weighted linear regression models: One design variable," *Mathematics*, vol. 11, no. 9, p. 2075, Apr. 2023, doi: [10.3390/math11092075](https://doi.org/10.3390/math11092075).
- [40] J. Liu, "The advance of underdetermined linear regression optimization based on implicit bias," *J. Phys., Conf. Ser.*, vol. 2580, no. 1, Sep. 2023, Art. no. 012008, doi: [10.1088/1742-6596/2580/1/012008](https://doi.org/10.1088/1742-6596/2580/1/012008).

YAN LI is currently a Researcher with Shandong Hi-Speed Group Company Ltd. His research interests include traffic safety analysis and traffic control.

HAN ZHANG is currently a Researcher with Shandong Hi-Speed Construction Management Group Company Ltd. His research interests include intelligent transportation systems and traffic control.

ZHIHENG CHENG is currently a Research Assistant with Shandong University. His research interests include intelligent transportation systems and traffic safety analysis.

ZIJIAN WANG is currently a Researcher with the Shandong Hi-Speed Group Company Ltd. His research interests include traffic safety analysis and traffic control.

ZIYI ZHANG is currently a Research Assistant with Shandong University. His research interests include connected vehicles and traffic simulation.

XINPENG YAO is currently a Researcher with Shandong Hi-Speed Group Company Ltd. His research interests include traffic safety analysis and traffic control.

• • •