Transformer-Based Global PointPillars 3D Object Detection Method

Lin Zhang, Hua Meng, Yunbing Yan * and Xiaowei Xu

Abstract: The PointPillars algorithm can detect vehicles, pedestrians, and cyclists on the road, and is widely used in the field of environmental awareness in autonomous driving. However, its feature encoding network only uses a minimalist PointNet network for feature extraction of point cloud information, which does not consider the global context information of the point cloud, and the local structure features are not sufficiently extracted, and these feature losses can seriously affect the performance of the object detection network. To address this problem, this paper proposes an improved PointPillars algorithm named TGPP: Transformer-based Global PointPillars. After the point cloud is divided into several pillars, global context features and local structure features are extracted through a multi-head attention mechanism, so that the point cloud after feature coding has global context features and local structure features; the two-dimensional pseudo-image generated by this feature is used for feature learning using a two-dimensional convolutional neural network. Finally, the SSD detection head is used to achieve 3D object detection. It is demonstrated that the TGPP achieves an average accuracy improvement of 2.64% in the KITTI test set.

Keywords: deep learning; 3D object detection; lidar point cloud; transformer

1. Introduction

The 3D object detection technology is an important part of the environment perception module in the automatic driving system. Accurately identifying objects such as vehicles, pedestrians, and cyclists on the road is the basis for vehicle planning and control. To better accomplish this goal, self-driving cars need to rely on a variety of sensors, among which lidar is one of the most important sensors. Lidar can measure the distance to the surrounding environment through a scanner, and directly generate sparse 3D point cloud information, which has inherent advantages in the task of 3D object detection. Traditional methods usually down-sample the point cloud information first, then remove the ground, and then use European, DBSCAN and other clustering methods combined with 3D bounding boxes to detect objects [1–5]. The traditional method requires cumbersome parameter adjustment work during the deployment process, making it difficult to apply in practice. With the rapid development of deep learning technology and parallel computing units, the end-to-end 3D object detection method based on deep learning has become the current key research content.

With the rapid development of computer vision and deep learning, 2D object detection technology has made great progress, but there are essential differences between the two data forms of the point cloud and image. In order, direct convolution of the point cloud will lead to severe distortion of the features [6], so the excellent 2D object detection algorithm cannot be directly applied to the 3D object detection task. In 2017, Qi et al. proposed PointNet [7] and PointNet++ [8] deep convolutional neural networks, which take the original point cloud as input and can be applied to point cloud point-by-point feature extraction, point cloud recognition, and point cloud semantics segmentation, and other fields also provide feature
extraction tools for 3D object detection tasks based on point cloud data. Subsequently, a point-based 3D object detection method was proposed. PointRCNN [9] is a more classic point-based method. The main idea is to extract point-by-point features from the PointNet network and predict 3D proposals to achieve 3D object detection. This type of method takes a lot of time to retrieve points, so the calculation is very large, and the detection efficiency is very low. In response to this problem, Zhou Y and others proposed the VoxelNet [10] algorithm, which is the earliest voxel-based method. This algorithm represents the point cloud as voxels, and the follow-up work in voxels reduces the amount of calculation and is based on the voxel method, which is more convenient for the extraction of target features, but due to the slow inference speed of the 3D convolutional neural network, its detection efficiency is still not ideal. As an upgraded version of VoxelNet, the SECOND [11] algorithm replaces the ordinary 3D convolution with sparse 3D convolution to speed up the reasoning time, but it still cannot eliminate the disadvantage of the slow calculation speed of 3D convolution. To this end, the PointPillars [12] algorithm proposes a novel encoder, which realizes end-to-end learning on 3D object detection tasks using only 2D convolutional neural networks. Its unique pillars-based encoding method greatly speeds up the detection speed. In addition, its simple algorithm framework can be easily deployed to a variety of laser radars. At present, it is one of the most widely used methods in engineering practice, and the research and improvement of the algorithm have practical application value and engineering significance.

At present, the detection rate of the PointPillars algorithm still has a large advantage, but its detection accuracy is inferior to the later excellent works. For example, Li Y et al. proposed the UVTR [13], which explicitly expresses and interacts with image and point cloud features in voxel space; Lai X et al. proposed the SphereFormer [14] method, which solved the problem of discontinuous information and limited receptive field. Therefore, in the past two years, some scholars have proposed some methods to improve PointPillars. For example, in 2021, Xinwei He et al. [15] proposed an intra-pillar multi-scale feature extraction module to enhance the overall learning ability of the PointPillars algorithm, thereby improving detection accuracy. This work improved the local structural feature extraction method of the point cloud, but still does not consider the global context feature information of the point cloud; in 2022, Dejiang Chen et al. [16] improved the 2D convolutional down-sampling module of the PointPillars algorithm based on Swin Transformer [17], optimized the original 2D convolutional neural network, and improved the Average Orientation Similarity (AOS) accuracy to a certain level. The improvement of this work optimizes the 2D convolutional neural network to improve the learning ability of point cloud features.

However, the above improvement scheme still does not make full use of point cloud features: its feature encoding process is to divide all point clouds into uniform pillars, where each pillar can be understood as a combination of voxels at the corresponding position on the z-axis, and then pass a minimalist PointNet network, which performs local feature extraction and uses Max Pooling to obtain points representing the features of each pillar, and finally generates a sparse 2D pseudo-image through position mapping. In this encoding process, the local feature extraction is insufficient and does not consider the global features causing a loss to the features of the point cloud.

To solve the appealing problem, this paper proposes an improved PointPillars algorithm named TGPP (Transformer-based Global PointPillars), based on Transformer [18], to improve the feature encoding network: after the point cloud is divided into pillars, the global position feature calculation and local structure feature calculation are performed based on the improved Transformer module, and each rich global context features and local features of the pillars enable the local features of the point cloud and accurate global position information to be preserved in the feature encoding process to improve the accuracy of the algorithm object detection.
2. TGPP Algorithm Network

TGPP is an improvement on PintPillars. The reasoning speed of the PointPillars algorithm is very fast, exceeding the scanning frequency of the radar, so its real-time detection is very good. The algorithm uses 3D point clouds as input, which can realize end-to-end learning and can detect road vehicles, pedestrians, and cyclists. These three common objects are identified.

The TGPP algorithm structure is shown in Figure 1 below. The algorithm can be divided into three main parts: (1) Pillar Feature Net: divide the 3D point cloud into pillars, and generate the 2D pseudo image. (2) Two-dimensional convolutional neural network: use multiple down-sampling of 2D pseudo-images to obtain feature maps of different resolutions, and then up-sample multiple feature maps after down-sampling to the same size for splicing to generate the final feature map. (3) Object detection head: generate a 3D detection frame and object classification for the feature map, and obtain the position and type of the object. The main difference between this method and the original method lies in the feature encoding network, and the structure of this algorithm will be introduced in detail below.

![Figure 1. TGPP algorithm network structure.](image)

2.1. Overall Algorithm Process

This algorithm takes the original point cloud data information as input, and first expresses the point cloud as a uniformly distributed pillar: the three-dimensional point cloud information is directly obtained from the top view, all points are discretized into a uniform square network on the x–y plane in the grid, and each pillar is cubic with infinite extension in the z-axis direction of each grid.

Due to the sparsity of the point cloud, most of the pillars are empty, and there are usually only a small number of points in the non-empty pillars. This sparsity is used to create a size density tensor (D, P, N), where D represents the feature dimension of each pillar, P represents the number of pillars, and N represents the maximum number of points in each pillar. When the points in the pillars exceed N, random sampling selects N points; when the points are less than N, it will be filled with 0 samples.

After obtaining a (D, P, N) tensor, the input is based on the improved Transformer feature an encoding network for feature extraction. First, we use the MLP (Multi-Layer Perceptron) operation for position encoding and dimension-up processing, changing from (D, P, N) tensors to (C, P, N) tensors, and C represents the feature dimension after dimension enhancement (256); then, based on the multi-head attention mechanism, we calculate the global context features for each pillars and calculate the local structural features for the points in each pillar, so that the point cloud information in each pillars has global context features and local structural features, in particular, in order to fully extract the local structural features of the point cloud, a combination of local and global position encoding is used; then, the maximum pooling is used to extract the feature points that best represent
the features of the pillars; finally, according to the pillars index, we remap the point cloud to the corresponding position of the original grid, and generate a 2D pseudo-image of size \((C, H, W)\), where \(H\) and \(W\) represent the height and width of the image.

The generated 2D pseudo-image will be input into the 2D convolutional neural network for feature learning, and finally, the detection head based on the design of SSD (Single Shot Multibox Detector) [19] is used to realize the classification and regression of 3D object detection and generate a 3D object detection frame.

2.2. Feature Encoding Network Based on Transformer

The Transformer model is a deep learning model based on the attention mechanism, which has been widely used in natural language processing (NLP), image processing, and other fields. Its core idea is to split the input sequence into a set of vector representations, and then use the attention mechanism to learn the dependencies between positions. Through the multi-head attention mechanism, Transformer can perform more comprehensive and accurate feature extraction on point cloud data, and its application in 3D object detection tasks based on point cloud data has gradually become a trend. PCT [20], Point Transformer [21], SOE-Net [22], VoxSeT [23], FlatFormer [24] and other works have achieved good results. Therefore, it is feasible to improve the feature encoding network based on Transformer.

The feature encoding network structure of this algorithm is shown in Figure 2 below, which is a network structure based on the encoder–decoder. The input of the feature encoding network is the point cloud information represented by the pillar distribution, and a vector sequence is generated through position encoding, and input to the multi-head attention module for calculation; each element of the input sequence is compared with other elements in the sequence Elements interact and give different weights according to their relevance. This interaction is realized by calculating the attention weight matrix; then it is input into the feedforward neural network module, and the output of the attention layer is further nonlinearly transformed; additionally, in order to prevent degradation problems during the training process, we add ResNet residual neural network [25] and LN layer [26] (Figure 2: Add&Norm module); the difference between the decoder and the encoder is that there is add a masked multi-head attention module, whose input is the predicted output of the entire feature calculation process; and the final output layer converts the output of the decoder into the final probability distribution through a linear transformation and Softmax function for generating prediction results. \(Nx\) represents the number of encoders and decoders, that is, the number of layers of the Transformer. Each layer independently processes the input and passes its output to the next layer.

The core content of the feature encoding network is to use the multi-head attention module to calculate the global context feature of the pillars and the local feature calculation of the internal structure of the pillars:

(1) Global Feature Calculation:

The calculation formula of the global attention [18] is

\[
\text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)
\] (1)

In Formula (1), \(Q, K,\) and \(V\) are feature codes of point cloud columns. First, calculate the dot product of the \(Q\) matrix and \(K\) matrix, and divide it by the scale \(\sqrt{d_k}\) to prevent overflow of the dot product result, where \(d_k\) is the vector in the \(Q\) and \(K\) dimension. After calculating the dot product, use the Softmax function to normalize the dot product to a probability distribution, and finally multiply it by the matrix \(V\) to obtain the attention score matrix between different pillars.
The multi-head attention allows the model to simultaneously focus on information from different pillars and different locations. The representation of the multi-head attention [18] is as follows:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^o$$

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

(2)

Among them, $W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$, $d_v = d_k$, $d_{\text{model}} = d_k \times h$; $h$ is the number of attention heads.

All non-empty pillars perform global context feature calculation through the multi-head attention mechanism, which can add global attention to each pillar.

Figure 2. TGPP algorithm feature-encoded network structure.

(2) Local Feature Calculation:

In the process of calculating the global feature of the pillars, the local feature calculation is added, and the local geometric relationship between the center point and the adjacent point is used to effectively aggregate the local features by learning the attention weight. The specific method is to use the subtraction relationship, and at the same time add the local position information $\delta$ to the attention vector $\gamma$ and the feature vector $\alpha$ to aggregate features. The overall calculation expression [27] is

$$g_i = \sum_{\mu_j \in x(i)} \rho(\gamma(\phi(\mu_i) - \phi(\mu_j) + \delta))(\alpha(\mu_j) + \delta)$$

(3)
Among them, \( \mu = \{ f_i | i = 1, 2, \ldots, n \} \) is a set of feature vectors composed of points in the pillars, \( \mu(i) \in \mu \); \( g_i \) is the feature output after adding local attention; \( \varphi, \phi, \) and \( \alpha \) are point-by-point feature transformation functions, similar to linear projection functions; and \( \gamma \) is the attention-generating mapping function. The calculation formula of local position information \( \delta \) is

\[
\delta = \varepsilon (p_i - p_j)
\]  

(4)

Among them, \( p_i, p_j \) is the coordinates of the 3D point cloud; and \( \varepsilon \) is composed of two ReLU functions [28].

After the above processing, the point cloud features after feature encoding will have global position features and local structure features, which reduce the feature loss caused by feature encoding, and the subsequent generated 2D pseudo-images are more conducive to subsequent feature learning to improve object detection accuracy.

### 2.3. 2D Convolutional Neural Network and SSD Detection Head

After the original point cloud information passes through the feature encoding network, a 2D pseudo image is generated, and the 2D convolutional neural network can be used very conveniently for feature learning. The structure of the 2D convolutional neural network is shown in Figure 1. The backbone network consists of two sub-networks: a top-down feature extraction network and an up-sampling and feature stitching network. The top-down sub-network uses a gradually decreasing spatial resolution to acquire features, and consists of a series of block structures, where each block structure contains three parameters \( (S, L, F) \), each block contains \( L \times 3 \times 3 \) 2D convolutional layers, \( F \) output channels, and the step size of the convolutional layer is \( S \). The network that performs up-sampling and feature splicing is responsible for up-sampling the features from the first sub-network and applying the BN Layer [29] and the ReLU function to form the final output features. The use of a 2D convolutional neural network avoids the disadvantages of slow inference speed of algorithms such as VoxelNet using a 3D convolutional neural network, simplifies the structure of the model, reduces the amount of calculation, has good detection accuracy, and greatly improves detection speed.

The SSD detection head is used to predict the position, category, and orientation of 3D objects. We use the 2D intersection-over-union ratio (IOU) to match the prior frame with the real label frame, regardless of the height information, but use it as an additional regression object, because in the real road object, all objects can be considered to be in the same plane of the three-dimensional space, the height difference between all categories of objects is not very large, and better results can be obtained by directly using the SmoothL1 function [30] for regression. At the same time, the FPN (feature pyramid network) [31] operation is also introduced in the detection head to handle objects of different sizes. By extracting features at different scales, objects of different sizes can be located more accurately.

### 3. Algorithm Implementation Details

#### 3.1. Details of Feature Encoding Network Structure Parameters

The cross-section of each pillar is a square with a side length of 0.16 m. In the actual feature encoding process, only the front view part is intercepted to generate a pseudo-image, because the real label information of the KITTI dataset is only in the front view captured by the camera. It is marked in the image, so the points of the original point cloud information in the negative direction of the x-axis should be discarded, and the points that are too far away should be removed. Refer to the original algorithm to take the maximum and minimum values of \( (x, y, z) \) in the point cloud space. It is min: \((0, -39.68, -3)\), max: \((69.12, 39.68, 1)\), in meters; the maximum value \( P \) of the number of pillars is 12,000, and the maximum value of point sampling in each pillar is \( N \), which is set to 32. The number of Transformer layers is 4, the number of heads is 2, and a 2-layer learnable MLP is used for position encoding.
3.2. Loss Calculation

This article uses the same loss calculation method as the original algorithm. Each real label box contains \((x, y, z, w, l, h, \theta)\) 7 parameters, where \((x, y, z)\) represents the three-dimensional coordinates of the object center; \((z, w, l)\) represents the label of the length, width, and height of the frame, and \(\theta\) represents the rotation angle. The regression residual of the positioning task between the prior box and the ground truth box is defined as

\[
\begin{align*}
\Delta x &= \frac{x^{gt} - x^a}{d^a}, \\
\Delta y &= \frac{y^{gt} - y^a}{d^a}, \\
\Delta z &= \frac{z^{gt} - z^a}{d^a}, \\
\Delta w &= \log \frac{w^{gt}}{w^a}, \\
\Delta l &= \log \frac{l^{gt}}{l^a}, \\
\Delta h &= \log \frac{h^{gt}}{h^a}, \\
\Delta \theta &= \sin (\theta^{gt} - \theta^a)
\end{align*}
\]

(5)

Among them, \(x^{gt}\) represents the \(x\) value of the label box; \(x^a\) represents the \(x\) value of the prior frame; \(y, z, w, l, h, \theta\) are the same; and \(d^a\) represents the diagonal distance between the length and width of the prior frame, defined as \(d^a = \sqrt{(w^a)^2 + (l^a)^2}\). The total localization loss is

\[L_{loc} = \sum_{b \in \{(x, y, z, w, l, h, \theta)\}} \text{SmoothL1}(\Delta b)\]

(6)

Since it is not possible to completely distinguish between two a priori boxes with completely opposite directions during angle regression, it is necessary to add direction classification to the a priori box. The direction classification loss function uses the Softmax function, denoted as \(L_{dir}\) [11]. The object classification loss function uses Focal Loss [32]:

\[L_{cls} = -\lambda_a (1 - p^a)^r \log p^a\]

(7)

Among them, \(p^a\) represents the probability that the predicted prior box belongs to the positive class, \(\lambda = 0.25, r = 2\). Finally, the total loss function is obtained as

\[L = \frac{1}{N_{pos}} (\beta_{loc} L_{loc} + \beta_{cls} L_{cls} + \beta_{dir} L_{dir})\]

(8)

Among them, \(N_{pos}\) is the number of correct prior frames. The values of \(\beta_{loc}, \beta_{cls}\) and \(\beta_{dir}\) we refer to SECOND algorithm, so \(\beta_{loc} = 2, \beta_{cls} = 1, \beta_{dir} = 0.2\).

4. Testing Results

4.1. KITTI Dataset Division

The training and testing of the model use KITTI’s 3D object detection dataset [33], which consists of lidar point clouds and image samples. It is only trained on the lidar point cloud, but the lidar point cloud and image fusion are used. The method to realize the comparison between the prior frame and the true value. The sample data have 7481 training samples. For the convenience of comparison, the same data set division method as the PointPillars algorithm is used: the training samples are divided into 3712 training samples and 3769 testing samples.

4.2. Experiment Analysis

4.2.1. Model Training

The computer environment used for the training and testing of this algorithm is Ubuntu 20.04 system, the processor is Intel® Core™ i9-9900 CPU @ 3.10 GHz × 16, the graphics card is Nvidia A40, and the video memory is 48G. TGPP is improved based on the PointPillars algorithm model in the OpenPCDet framework and written in Python3.8.

OpenPCDet is an open-source point cloud object detection algorithm library based on Pytorch. The PointPillars algorithm in this framework adopts more advanced data enhancement methods, optimizers, learning strategies, and other methods to optimize the model. The trained model has better detection accuracy.
The optimizer selected during training is Adam_onecycle, and the maximum learning rate LR is 0.002. This algorithm and PointPillars are trained under the same conditions. During the training process, the loss change curve before and after the model improvement is shown in Figure 3. It can be seen from the figure that the TGPP has a stronger feature learning ability.

![Loss change curve.](image)

**Figure 3.** Loss change curve.

### 4.2.2. Model Training

The test is performed on the trained model based on KITTI’s 3D object detection testing set. The test scenarios are divided into three types: simple, medium, and difficult. The test mainly uses the average precision (AP) of 3D object detection as the evaluation index. During the test, the detection of vehicles adopts the standard of IoU = 0.7, and the detection of pedestrians and cyclists adopts the standard of IoU = 0.5.

1. **Compared with PointPillars:**

   The test results of this algorithm and the PointPillars algorithm are shown in Table 1. It can be seen from Table 1 that compared with PointPillars, this method has improved the 3D object detection performance of vehicles, pedestrians, and cyclists. The vehicle detection AP in the three difficulty scenarios increased by 2.68%, 1.84%, and 2.62%, respectively; the pedestrian detection AP increased by 4.84%, 3.97%, and 3.42%, respectively; the cyclist detection AP increased by 1.41%, 2.12%, and 2.24%, respectively. To better evaluate the overall detection performance, the mAP of vehicles, pedestrians, and cyclists detected under medium difficulty are calculated. The TGPP mAP is 63.56%, and the PointPillars mAP is 60.92%. TGPP has improved by 2.64% mAP in the testing set, which is equivalent to a performance improvement of about 4.3% for PointPillars.

**Table 1.** Comparison of average precision of 3D object detection (%).

<table>
<thead>
<tr>
<th>Method</th>
<th>Runtime (ms)</th>
<th>Car (IoU = 0.7)</th>
<th>Pedestrian (IoU = 0.5)</th>
<th>Cyclist (IoU = 0.5)</th>
<th>mAP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Easy Mod.</td>
<td>Easy Mod.</td>
<td>Easy Mod.</td>
<td>Easy Mod.</td>
<td></td>
</tr>
<tr>
<td>PointPillars</td>
<td>16</td>
<td>85.06 76.05</td>
<td>52.08 45.88</td>
<td>80.05 62.95</td>
<td>60.92</td>
</tr>
<tr>
<td>TGPP</td>
<td>21</td>
<td>87.74 77.89</td>
<td>56.92 49.85</td>
<td>45.09 35.12</td>
<td>63.56</td>
</tr>
</tbody>
</table>

In terms of detection speed, the average time for PointPillars to process a frame of point cloud data is only 16 ms. Compared with PointPillars, the detection speed has decreased, and the average time is 21 ms, which is 47 Hz when converted into Hz. Considering the
vehicle-mounted laser, the scanning frequency of the lidar is usually 10–20 Hz, so this method can still meet the real-time detection requirements.

(2) Comparison with Other 3D Object Detection Methods:

Comparing this method with the more excellent methods in recent years, as shown in Table 2, the 3D detection performance of other methods is derived from their own papers, some of which did not give the detection speed, taken from KITTI’s 3D object detection method performance leaderboard. It can be seen from the table that this method is compared with commonly used methods based on the fusion of image and point cloud data such as MV3D [34], RoarNet [35], AVoD-FPN [36], and F-PointNet [37]. There are no small advantages in the speed or detection of AP. Among lidar-based methods, this method also has certain advantages compared with voxel-based methods. For example, compared with VoxelNet, SECOND, TANe [38], and PSA-Det3D [39] the mAP is 14.51%, 7.17%, 2.93%, and 2.43% higher; the detection accuracy of the point-based method is usually higher, but this method also has advantages compared with it, such as PointRCNN and STD [40], where the mAP is 4.51% and 2.85% higher, respectively. At the same time, this method is superior to all the methods mentioned above in terms of detection speed.

Table 2. Comparison of 3D object detection accuracy with other methods (%).

<table>
<thead>
<tr>
<th>Method</th>
<th>Runtime (ms)</th>
<th>Car (IoU = 0.7)</th>
<th>Pedestrian (IoU = 0.5)</th>
<th>Cyclist (IoU = 0.5)</th>
<th>mAP Mod.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Easy</td>
<td>Mod.</td>
<td>Hard</td>
<td>Easy</td>
</tr>
<tr>
<td>MV3D</td>
<td>360</td>
<td>71.29</td>
<td>62.68</td>
<td>56.56</td>
<td>-</td>
</tr>
<tr>
<td>RoarNet</td>
<td>100</td>
<td>83.71</td>
<td>73.04</td>
<td>59.16</td>
<td>-</td>
</tr>
<tr>
<td>AVoD-FPN</td>
<td>100</td>
<td>81.94</td>
<td>71.88</td>
<td>66.38</td>
<td>50.80</td>
</tr>
<tr>
<td>F-PointNet</td>
<td>169</td>
<td>81.20</td>
<td>70.39</td>
<td>62.19</td>
<td>51.21</td>
</tr>
<tr>
<td>VoxelNet</td>
<td>220</td>
<td>77.47</td>
<td>65.11</td>
<td>57.73</td>
<td>39.48</td>
</tr>
<tr>
<td>TANet</td>
<td>35</td>
<td>83.81</td>
<td>75.38</td>
<td>67.66</td>
<td>54.92</td>
</tr>
<tr>
<td>PSA-Det3D</td>
<td>80</td>
<td>87.46</td>
<td>78.80</td>
<td>74.47</td>
<td>49.72</td>
</tr>
</tbody>
</table>

In summary, this method maintains the advantages of the PointPillars algorithm in detection speed and is also superior to the current mainstream methods in detection accuracy. Therefore, it is proved that the feature encoding network improvement scheme proposed in this paper is feasible and practical.

4.3. Comparison of Actual Road Environment Test Results

We use this method and the original method to test the effect of target detection in the same road environment, as shown in Figure 4. In Figure 4 (scenario a), it can be seen that the false detection rate of the original method is higher, and many non-object point clouds are recognized as vehicles and cyclists; in Figure 4 (scenario b), it can be seen that the original method has a higher impact on pedestrians. The false detection rate is high, and the point cloud of non-pedestrians is recognized as pedestrians. From this, it can be seen that the detection accuracy of this method is better than that of the original method in the actual road environment test.
4.3. Comparison of Actual Road Environment Test Results

We use this method and the original method to test the effect of target detection in the same road environment, as shown in Figure 4. In Figure 4 (scenario a), it can be seen that the false detection rate of the original method is higher, and many non-object point clouds are recognized as vehicles and cyclists; in Figure 4 (scenario b), it can be seen that the original method has a higher impact on pedestrians. The false detection rate is high, and the point cloud of non-pedestrians is recognized as pedestrians. From this, it can be seen that the detection accuracy of this method is better than that of the original method in the actual road environment test.

5. Ablation Experiments

In order to verify the effectiveness of the improved Transformer-based feature encoding network, an ablation experiment is performed. The hyperparameters of the feature encoding network include the number of Transformer layers and the number of heads. Change these two parameters to observe the impact on detection performance. In order to avoid the influence of random number seeds, each set of parameters was trained five times. For the convenience of the experiment, Epoch was set to 120, and the average mAP under medium difficulty was used as the evaluation index. The results are shown in Table 3. It can be seen from the table that when the number of layers and heads is small, the detection performance of the original method is better.
performance is not as good as PointPillars. Increasing the number of layers and heads within a certain range can improve the detection performance. When the number of layers is 4 and the number of heads is 2, the effect is the best. Continuing to increase the number of layers and the number of heads will degrade the detection performance.

Table 3. Results of ablation experiments.

<table>
<thead>
<tr>
<th>Group</th>
<th>Transformer Layer</th>
<th>Transformer Head</th>
<th>Epoch</th>
<th>Training Time</th>
<th>mAP Mod.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 (PointPillars)</td>
<td>--</td>
<td>--</td>
<td>120</td>
<td>5</td>
<td>60.42</td>
</tr>
<tr>
<td>2 (TGPP)</td>
<td>1</td>
<td>1</td>
<td>120</td>
<td>5</td>
<td>59.76</td>
</tr>
<tr>
<td>3 (TGPP)</td>
<td>2</td>
<td>1</td>
<td>120</td>
<td>5</td>
<td>59.95</td>
</tr>
<tr>
<td>4 (TGPP)</td>
<td>2</td>
<td>2</td>
<td>120</td>
<td>5</td>
<td>61.34</td>
</tr>
<tr>
<td>5 (TGPP)</td>
<td>4</td>
<td>1</td>
<td>120</td>
<td>5</td>
<td>62.26</td>
</tr>
<tr>
<td>6 (TGPP)</td>
<td>4</td>
<td>2</td>
<td>120</td>
<td>5</td>
<td>63.32</td>
</tr>
<tr>
<td>7 (TGPP)</td>
<td>6</td>
<td>1</td>
<td>120</td>
<td>5</td>
<td>62.81</td>
</tr>
<tr>
<td>8 (TGPP)</td>
<td>6</td>
<td>2</td>
<td>120</td>
<td>5</td>
<td>62.94</td>
</tr>
</tbody>
</table>

6. Conclusions

In this paper, for the 3D object detection algorithm, an improved PointPillars feature encoding network based on Transformer is proposed. This improved PointPillars algorithm is named TGPP. The improved feature encoding network uses a multi-head attention mechanism to extract global context features and local structure features from pillars. The feature extraction ability of the original algorithm in the feature encoding process is improved, and the feature loss is reduced. Experimental results prove that this algorithm has better object detection performance than PointPillars, and the average object detection accuracy on the KITTI testing set has increased by 2.64%, which is also competitive with other methods in recent years.

Author Contributions: Conceptualization, L.Z. and H.M.; methodology, H.M.; software, H.M.; validation, L.Z., H.M., Y.Y. and X.X.; formal analysis, L.Z. and H.M.; resources, Y.Y. and X.X.; writing—original draft preparation, L.Z. and H.M.; writing—review and editing, Y.Y. and X.X.; project administration, L.Z. and Y.Y.; funding acquisition, L.Z. and Y.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by National Natural Science Foundation of China [51975428], “Chunhui Plan” Cooperative Scientific Research Project of the Education Department of China [HZKY2022(330)] and Guidance Project of Scientific Research Plan of the Education Department of Hubei Province of China [B2022027].

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Conflicts of Interest: The authors declare no conflict of interest.

References


Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.