 https://doi.org/10.3311/PPee.14960
Creative Commons Attribution

Periodica Polytechnica Electrical Engineering and Computer Science, 64(3), pp. 274–285, 2020

Pedestrian Detection Based on Panoramic Depth Map Transformed from 3D-LiDAR Data

Guoqiang Chen[1*], Zhuangzhuang Mao[1], Huailong Yi[1], Xiaofeng Li[1], Bingxin Bai[1], Mengchao Liu[1], Hongpeng Zhou[1]

[1] School of Mechanical and Power Engineering, Henan Polytechnic University, 454003 Jiaozuo, Henan, China
[*] Corresponding author, e-mail: chengq@hpu.edu.cn

Received: 11 September 2019, Accepted: 17 December 2019, Published online: 23 April 2020

Abstract

Object detection is a crucial task of autonomous driving. This paper addresses an effective algorithm for pedestrian detection of the panoramic depth map transformed from the 3D-LiDAR data. Firstly, the 3D point clouds are transformed into panoramic depth maps, and then the panoramic depth maps are enhanced. Secondly, the grounds of the 3D point clouds are removed. The remaining point clouds are clustered, filtered and projected onto the previously generated panoramic depth maps, and new panoramic depth maps are obtained. Finally, the new panoramic depth maps are jointed to generate depth maps with different sizes, which are used as input of the improved PVANET for pedestrian detection. The 2D image of the panoramic depth map applied to the proposed algorithm is transformed from 3D point cloud, effectively containing the panorama of the sensor, and is more suitable for the environment perception of autonomous driving. Compared with the detection algorithm based on RGB images, the proposed algorithm cannot be affected by light, and can maintain the normal average precision of pedestrian detection at night. In order to increase the robustness of detecting small objects like pedestrians, the network structure based on the original PVANET is modified in this paper. A new dataset is built by processing the 3D-LiDAR data and the model trained on the new dataset perform well. The experimental results show that the proposed algorithm achieves high accuracy and robustness in pedestrian detection under different illumination conditions. Furthermore, when trained on the new dataset, the model exhibits average precision improvements of 2.8–5.1 % over the original PVANET, making it more suitable for autonomous driving applications.

Keywords

3D-LiDAR data, panoramic depth map, small object detection, improved PVANET

1 Introduction

Autonomous driving technology and advanced driver assistance system rely on accurate, real-time and robust perception of the environment, so it is increasingly important to detect and identify road objects accurately, real-time and robustly. Currently, the 3D-LiDAR scanner and the high-resolution camera are the main sensors that autonomous vehicles rely on for object detection. Generally speaking, the 3D-LiDAR scanner can detect vehicles, pedestrians, bicycles and other objects, while the high-resolution camera can detect a wide range of objects, such as traffic signs and license plates. The data acquired by the 3D-LiDAR scanner is informative and not easily affected by light, so the panoramic depth map transformed from it can still correctly represent the environment.

It is very popular to use Convolutional Neural Networks (CNNs) to perform object detection on RGB images. However, the object detection pipeline for autonomous driving based on RGB images is limited by illumination, and these pipelines cannot achieve ideal detection accuracy at night. Moreover, most image-based or LiDAR-based 3D object detection pipelines are limited to the front view of the sensor [1–4] and cannot detect moving objects behind. Therefore, this paper proposes an efficient and robust detection algorithm that can detect pedestrians around the sensor and is unaffected by illumination. The classic single-stage detection pipelines like YOLO [5], SSD [6] and the two-stage detection pipelines like Faster-RCNN [7] all detect large objects in RGB images, so these detection pipelines are not ideal for small object detection in RGB images. The proposed algorithm builds a new dataset for objects detection based on the 3D point cloud of KITTI [8]. Pedestrians, bicycles and even vehicles in the

Cite this article as: Chen, G., Mao, Z., Yi, H., Li, X., Bai, B., Liu, M., Zhou, H. "Pedestrian Detection Based on Panoramic Depth Map Transformed from 3D-LiDAR Data", Periodica Polytechnica Electrical Engineering and Computer Science, 64(3), pp. 274–285, 2020. https://doi.org/10.3311/PPee.14960

dataset become small objects due to the limitation of input tensor height. This paper adopts several improvements to PVANET [9], including optimizing Region Proposal Network (RPN) and learning rate.

The proposed algorithm is mainly divided into the following steps:

1. transforming 3D point clouds acquired by 3D-LiDAR into panoramic depth maps and enhancing the panoramic depth maps;
2. removing the ground, clustering and filtering from 3D point clouds, then projecting the remaining point clouds onto the panoramic depth maps to generate the Double Projection Panoramic Depth Map (DPPDM);
3. jointing several random, continuous, or identical DPPDMs to generate the Large-scale Double Projection Panoramic Depth Map (LDPPDM); and
4. using these LDPPDMs to build a new dataset, then training the new dataset by the CNN proposed in this paper to detect pedestrians.

The contributions of this paper are threefold:

1. This paper proposes a Clustering and Double Projection method to highlight the detection objects, and then builds a dataset composed of LDPPDMs.
2. This paper improves the model structure of PVANET with RPN and learning rate to increase its robustness to small objects, which leads to significant accuracy improvements of 2.8 % to 5.1 % for pedestrians.
3. This paper proposes an algorithm that can accurately detect pedestrians affected by illumination variations, and detect pedestrians outside the camera view but located around the LiDAR scanner.

2 Related work

2.1 Main object detection methods

At present, the modalities of object detection for autonomous driving are mainly classified into three types, which are based on cameras, 3D-LiDARS, and the fusion of the two.

The object detection method based on images obtained by the camera is relatively common. CNNs are mainly applied to 2D object detection [1, 7] and obstacle segmentation [2] in RGB images. The rapid progress on stereo and monocular depth estimation suggests that images could be used to 3D object detection. Wang et al. [10] got corresponding depth maps from monocular or stereo images, and then combined original images with the depth information to obtain the pseudo-LiDAR, which replaced the 3D-LiDAR for object detection. Ma et al. [11] used monocular depth estimation models and camera parameters to transform images into 3D point clouds, and then got the 3D box through 2D detection and RGB information. Szemenyei et al. [12] used the original shape image to arrange virtual objects in real-world scenes, which improved the quality of the arrangement.

Because the 3D point clouds obtained by 3D-LiDAR have precise 3D coordinates, CNNs based on LiDAR data for 3D object detection have been proposed. VoxelNet [13], LMNet [14], RT3D [15] overcame the shortcomings of traditional 3D CNNs [16, 17] in learning local features of different sizes. BirdNet [18] and PIXOR [19] projected 3D point clouds onto 3-channel BEV maps or 2-channel BEV maps, and maintained the spatial structure of the object through simplifying the network structure and improving the real-time performance of object detection. In addition, SqueezeSeg [3] and SqueezeSegV2 [4] segmented road objects based on 3D point clouds.

RGB-D data obtained by cameras and LiDARs have rich spatial information, and it is very popular to use CNNs for object detection of RGB-D data at present. Frustum PointNet [20] first used RGB images to find 2D region proposal of detection objects, and then located 3D bounding boxes of objects with the depth map generated by 3D LiDAR data in corresponding region. AVOD [21] achieved object detection in 3D space by integrating features extracted from RGB images and theirs corresponding bird's eye view maps in 3D space. PointFusion [22] extracted the feature of input RGB images and corresponding 3D point clouds with PointNet++ [23] and PointNet [24], and then obtained the 3D bounding boxes of objects in 2D images through the feature.

2.2 Small object detection methods

YOLO [5], SSD [6] and Faster-RCNN [7] are mainly used for object detection in general datasets similar to PASCAL VOC [25], whose object sizes occupy a large proportion in images. Most datasets contain images of different sizes, and the size of landscape images is about 500 × 375 while the size of portrait images is about 375 × 500. However, in some cases, the object is not easily detected because of its small size and proportion in the image. Consequently, some detection methods for small objects have emerged. With SSD [6] as the main framework and ResNet-50 as the basic network, ALFNet [26] generated multi-scale feature maps to detect pedestrians. DetNet [27] was generated by improving the basic framework of ResNet50, which increased the size of the feature map and has an obvious

effect on small object detection. The low-level and high-level features were simultaneously used for multi-scale object detection by improving Faster-RCNN [7], which improved the accuracy of small object detection in [28]. Pedestrian detection pipeline based on candidate regions and the Parallel Convolutional Neural Network (PCNN) was proposed in [29]. When Xu et al. [29] used CNN for feature extraction, the shallow network is added to form PCNN and extract feature output, which improved the detection accuracy of small objects.

3 Method description

The architecture of the proposed algorithm is presented in Fig. 1. The pipeline comprises two main stages:

1. transforming the 3D LiDAR data into a 2D dataset; and
2. using the proposed Convolutional Neural Network to detect pedestrians in the dataset.

3.1 Point cloud transformation

In the algorithm, the first step of making a new dataset is to transform the 3D-LiDAR data into panoramic depth maps. The number of points in each point cloud in the KITTI dataset is about 120,000. In order to improve the efficiency of transformation, the distance values of multiple similar point clouds in the horizontal direction are replaced by one pixel. As show in Fig. 2, a 3D point can be turned into a range value in a pixel using the image geometry.

According to the spherical coordinate system projection model, the coordinate value of pixels in the generated panoramic depth maps are determined by Eq. (1):

$$\begin{cases} \Delta\phi = 2\pi / n_c \\ \Delta\theta = (26.8°/180°)\pi / n_r \\ \theta = \arctan(y/x) \\ d = \sqrt{x^2 + y^2 + z^2} \\ \phi = \arcsin(z/d) \\ c = [\phi/\Delta\phi] \\ r = [\theta/\Delta\theta] \end{cases} \quad (1)$$

where $(x, y, z)^T$ denotes a 3D point, as shown by p_i in Fig. 2; n_c and n_r denote the number of columns and rows in the panoramic depth map, respectively; 26.8° is the vertical view of Velodyne HDL-64E; $\Delta\phi$ and $\Delta\theta$ are the average horizontal and vertical angle resolution between consecutive beam emitters, respectively; d is the distance between the point and 3D-LiDAR, as show in Fig. 2; θ and ϕ denote the azimuth and elevation angle of the point, as show in Fig. 2; c and r represent the position of the 3D point

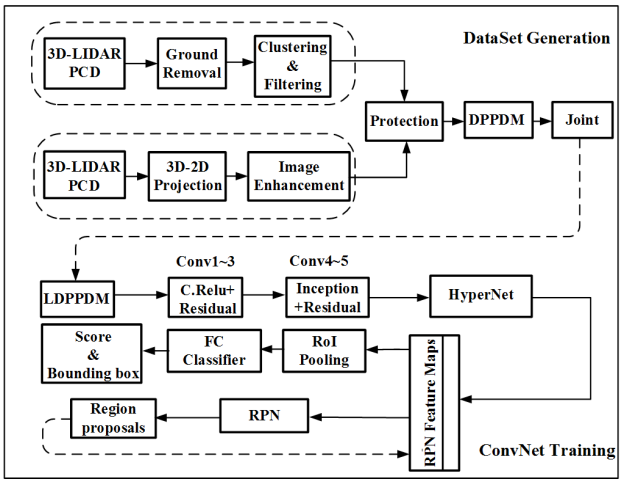


Fig. 1 Pedestrian detection algorithm based on 3D-LiDAR

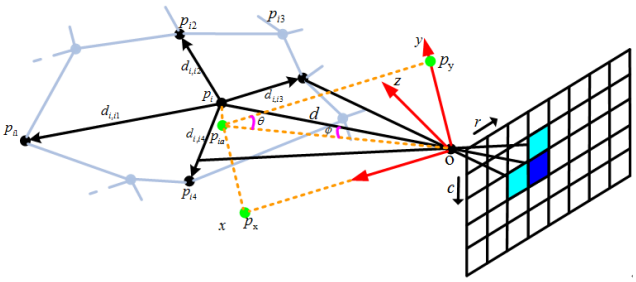


Fig. 2 Image geometric relationship between a 3D point and a range value in a pixel

projected onto the panoramic depth map. Algorithm 1 summarizes the procedure of obtaining a panoramic depth map from a 3D point cloud.

In the approach outlined in Fig. 2, the Euclidean distance between the point and 3D-LiDAR is stored as the pixel value of panoramic depth maps. In the process of 3D point cloud transformation, some coordinate values may be

Algorithm 1 Point cloud transformation to panoramic depth map

```
Result: I - depth image
Initialization
P - input point cloud
Main loop:
delta_horizon = 2 * pi / I.cols;
delta_vertical = 26.8 / 180 * pi / I.rows;
N = P.size;
for i = 0 : (N-1) do
    angle_cols = atan2 ( P[i].y, P[i].x);
    c = round (angle_cols / delta_horizon);
    distance = sqrt (P[i].x * P[i].x + P[i].y * P[i].y);
    angle_rows = atan2 (P[i].z, distance);
    r = round (angle_rows / delta_vertical);
    I[r][c] = sqrt (P[i].x * P[i].x + P[i].y * P[i].y + P[i].z * P[i].z);
end
return I;
```

lost, but the acquisition of main objects (vehicles, pedestrians, etc.) is not affected in the dataset made up of panoramic depth maps. Neighbors in the panoramic depth map implicitly define neighborhood relations between the 3D points. As show in Fig. 2, p_{i1}, p_{i2}, p_{i3}, and p_{i4} are the neighborhood of p_i; $d_{i,i1}$, $d_{i,i2}$, $d_{i,i3}$ and $d_{i,i4}$ represents the distance between p_{i1}, p_{i2}, p_{i3}, p_{i4} and p_i, respectively; and their neighborhood relations are also shown in the panoramic depth map.

The panoramic depth map transformed from the 3D-LiDAR data can be represented by 2D tensors with size 64 × 870, where 64 and 870 are the image height and width, respectively. Fig. 3 (a) is the visualization of a 3D point cloud acquired by 3D-LiDAR, and Fig. 3 (b) is the camera view of the frame corresponding to the point cloud. The original panoramic depth map transformed from the 3D point cloud is shown in Fig. 3 (c), and Fig. 3 (d) is the HSV visualization.

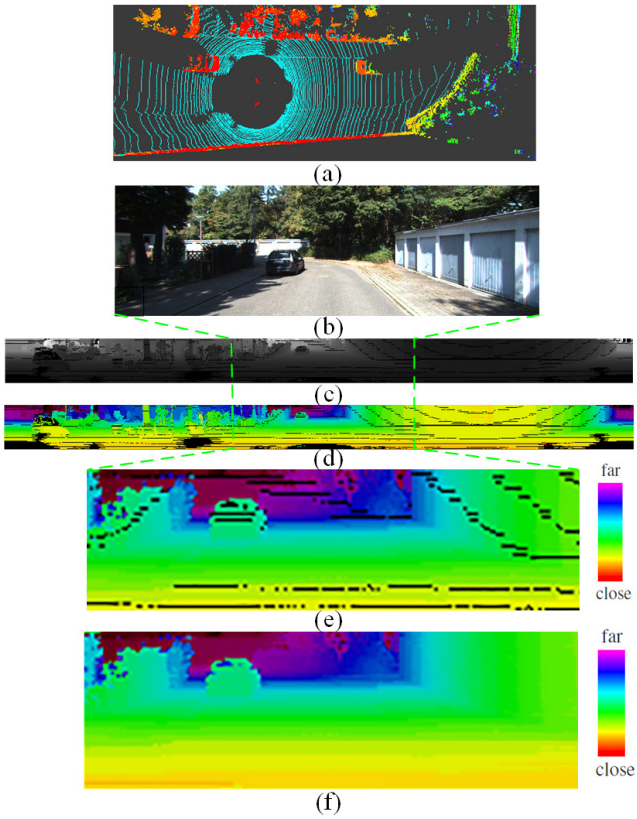


Fig. 3 Transformation of a 3D point cloud. (a) Visualization of the 3D point cloud. (b) Camera view of the frame corresponding to the point cloud. (c) Original panoramic depth map transformed from the 3D point cloud. (d) HSV visualization of the original panoramic depth map. (e) Enlarged part of the area corresponds to the view shown in (b). (f) Result of image enhancement.

3.2 Image enhancement

According to Fig. 3 (d), there are many gaps in the original panoramic depth map, which indicates that the pixel value here is abnormal. The measured distance is easily affected by noise, which may lead the pixel value of the panoramic depth map to be abnormal. In addition, when the point cloud is far away from 3D-LiDAR, the pixel may not contain valid measurement at all. From the above, the panoramic depth map should be enhanced.

As shown in Algorithm 2, firstly, the missing pixels in the panoramic depth map should be interpolated. It can be seen from Fig. 3 (d) the original panoramic depth map is poor in visualization due to the lack of pixels. The missing pixels at gaps will affect the feature extraction and object detection of the panoramic depth map, so it is necessary to interpolate gaps. Interpolation fills gaps first horizontally then vertically in a linear manner, which calculates the pixel value p_1 and p_2, respectively. The average value of p_1 and p_2 is taken as the final pixel value.

Secondly, the original panoramic depth map is interfered by noise, so the bilateral filter is used to smooth the distance value, which will reduce noise and improve image quality. Combined with the spatial weight and similar weight, the bilateral filter, which is based on the overall consideration of spatial distance and similarity degree, can be described as follows in Eq. (2):

$$\begin{cases} h(x) = k^{-1}(x)\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(\xi)c(\xi,x)s\big(f(\xi),f(x)\big)\mathrm{d}\xi \\ k(x) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} c(\xi,x)s\big(f(\xi),f(x)\big)\mathrm{d}\xi \\ s\big(f(\xi),f(x)\big) = \mathrm{e}^{-\frac{1}{2}\left(\frac{\sigma(f(\xi),f(x))}{\sigma_r}\right)^2} \\ \sigma\big(f(\xi),f(x)\big) = \sigma\big(f(\xi)-f(x)\big) = \|f(\xi)-f(x)\| \\ c(\xi,x) = \mathrm{e}^{-\frac{1}{2}\left(\frac{d(\xi,x)}{\sigma_d}\right)^2} \\ \mathrm{d}(\xi,x) = \mathrm{d}(\xi-x) = \|\xi-x\| \end{cases} \tag{2}$$

where $k(x)$ is the weight of bilateral filter $h(x)$; $s(f(\xi), f(x))$ and $c(\xi, x)$ are spatial weight and similar weight, respectively; $\sigma(\xi, x)$ and $\mathrm{d}(\xi, x)$ are radiation distance and Euclidean distance, respectively.

After the image is enhanced, as shown in Fig. 3 (f), the gaps are filled, and the objects are more obvious. The enhancement of the initial panoramic depth map not only can optimize its visualization, but also improve the accuracy of object detection in the later stage.

Algorithm 2 Image enhancement of panoramic depth map

```
Result: I_out - depth image after interpretation
Initialization:
I_in - original depth image
sigma - parameter for bilateral filter
filter_size - size for neighbour consideration
Main loop:
I_out = I_in;
for r = 0 : I_in.rows do:
    for c = 0 : I_in.cols do:
        if (I_in[r][c] == 0) do:
            p1 = linear_interpolation (I_in[r][c], I_in.row(r));
            p2 = linear_interpolation (I_in[r][c], I_in.col(c));
            I_out.[r][c] = (p1 + p2) / 2;
        end
    end
end
I_out = bilateral_filter (I_out, sigma, filter_size);
return I_out
```

3.3 Ground removal

This study uses the improved Ground Plane Fitting (GPF) algorithm [30] for fast ground removal. The advantage of this algorithm is that it is suitable for a variety of autonomous driving scenarios. According to Algorithm 3, the initial points are sorted in ascending order of height values. A set of lowest points is used to estimate the initial plane model of the ground. The ground model parameters are calculated by Eq. (3):

Algorithm 3 Ground removal of point cloud

```
Result: P_ground - point cloud with ground points
        P_noground - point cloud without ground points
Initialization:
P_in - original point cloud
N_ground - initial number of ground points
Thpoint - threshold for ground removal
Main loop:
sort (P, with_z_value);
P_ground = getLastPoints (P, N_ground);
[∑, μ] = computeMeanAndCovarianceMatrix (P_ground);
n = svd.matrixU (∑).col (2);
k = -n.transpose * μ.head <3> (0,0);
Thpoint = 0.5 - k;
point_matrix = Matrix (P_in.size, 3);
point_matrix << P_in;
d = point_matrix * n;
for r = 0 : d.rows do:
    if d[r] < Thpoint do:
        P_ground.push(P_in[r]);
    else
        P_noground.push(P_in[r]);
    end
end
return P_ground, P_noground
```

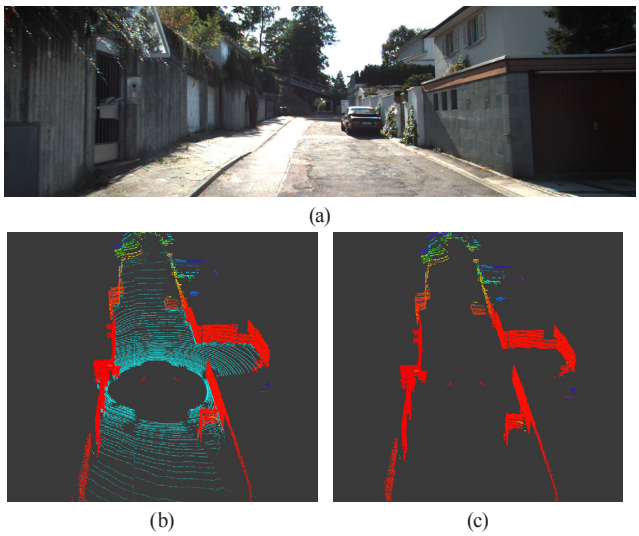
$$\begin{cases} ax+by+cz+k=0 \\ n=\begin{bmatrix} a & b & c \end{bmatrix}^T \\ p=\begin{bmatrix} x & y & z \end{bmatrix}^T \end{cases} \tag{3}$$

where n is the normal of the ground, p represents 3D point, and k is the distance from the point to its orthogonal projection. The parameters $\sum$ and μ represent the covariance matrix and mean of the initial ground points, respectively. The covariance matrix $\sum$ captures n by its Singular Value Decomposition (SVD). The threshold Th_{point} of the ground height is obtained by n and μ, and the point whose height value is lower than Th_{point} is considered as ground point. The distance value k is calculated by the ground model formula after the acquisition of n. If the distance value is lower than Th_{point}, it is the ground point, otherwise it is the non-ground point.

The ground removal of point clouds is mainly to prepare for point clouds clustering in the later stage. The visual comparison of a point cloud before and after ground removal is shown in Fig. 4.

3.4 DPPDM generation

The panoramic depth map transformed from the 3D point cloud is less visual than the RGB image, which is not conducive to object detection. By clustering, the same cluster is represented by one color, which highlights the detection object. However, the clustering based on depth map is not perfect. Therefore, this study runs clustering based on the 3D point cloud after ground removal.



(a)

(b) (c)

Fig. 4 Ground removal of a 3D point cloud. (a) Camera view of the frame corresponding to the point cloud. (b) Visualization of the point cloud with ground marked blue. (c) Visualization of the point cloud after ground removal

As shown in Algorithm 4, the original point cloud is classified according to the position of the 3D-LiDAR scan line where the point is located. The first point of each scan line is a new cluster and is assigned to a new label. When the distance between the latter point and the previous point is less than the set threshold $Th_{cluster}$, the two points are considered to be the same cluster; otherwise, the latter point represents a new cluster and is assigned a different label. In particular, the scanning range of a scan line is a circle. If the distance between the last point and the first point in the scan line is less than the threshold $Th_{cluster}$, the two clusters are considered to be the same one and assigned to the same label by updating. After the above clustering is completed with the point cloud of 64 scan lines, all clusters shall be updated. The distance between points on the same position on two adjacent scan lines should be calculated. Therefore, if the distance is less than the threshold $Th_{cluster}$, the two clusters are updated to the same one.

In order to improve the visualization of the detection object, the clustered point cloud should be filtered. By calculating the center position and scale of the cluster, the filtering conditions are set to remove objects that are farther away from the 3D-LiDAR and that have a large difference in geometry from the pedestrian. Two kinds of clusters are filtered in two corresponding steps:

1. the clusters too far away from the sensor, for example more than 50 m, have no detection significance in the actual and are filtered; and
2. the clusters significantly different from the size of the pedestrian are filtered.

For example, the clusters with a vertical center greater than 1.5 m or less than 0.5 m, and the clusters with an arbitrary size greater than 3 m or less than 0.3 m in the horizontal direction, are regarded as significantly different from the size of the pedestrian. In fact, the specific parameters are debugged according to the testing environment and requirements.

The filtered point cloud is projected onto the enhanced panoramic depth map to obtain the Double Projected Panoramic Depth Map (DPPDM), which is shown in Fig. 5.

3.5 Image join

The initial 3D point cloud data is from the open source dataset KITTI, which is collected by Velodyne HDL-64E LiDAR. The DPPDM can be represented by 3-dimentional tensors of size $H \times W \times 3$. The first two dimensions encode spatial position, where H and W are the image height and width, respectively. Because the Velodyne HDL-64E

Algorithm 4 Object clustering of point cloud without ground points

```
Result: clusters - all clusters
Initialization:
P_noground - point cloud without ground points
N - total scans of the Lidar sensor
Th_cluster - threshold for two points become a cluster in same scan
Th_scan - threshold for two points become
a cluster between two neighbour scans
Main loop:
scans = sort_pointcloud_with_scans(P_noground);
k = 0;
for s = 0 : N do:
    clusters = label_clusters (s, k);
    clusters = update_labels (s);
end
clusters = filter_clusters (clusters);
return clusters
function sort_pointcloud_with_scans (P_noground):
for p in P_noground do:
    scans[p.ring].push (p);
end
return scans
function label_clusters (s, k):
scans[s][0].label = k;
clusters[k].push (scans[s][0]);
for i = 0 : (sacans.size – 2) do:
    p_i = scans[s][i];
    p_j = scans[s][i+1];
     if distance (p_i, p_j) < Th_cluster do:
        scans[s][i+1].label = scans[s][i].label;
    else do:
                k = k + 1;
                scans[s][i+1].label = k;
    end
                clusters[k].push (scans[s][i+1]);
end
n = sacans[s].size;
if distance (scans[s][0], scans[s][n-1]) < Th_cluster do:
    merge_labels(scans[s][0].label, scans[s][n-1].label,
    clusters);
end
return clusters
function merge_clusters (l_1, l_2, clusters):
for c in clusters[l_2] do:
    c.label = l_1;
    clusters[l_1].push (c);
end
clusters[l_2].clear ();
return clusters
function update_labels (s, clusters):
for i = 0 : (scans[s].size-1) do:
    p_i = scans[s][i];
    for j = (s – 1) : 0 do:
        p_j = scans[j][i];
         if distance (p_i, p_j) < Th_scan do:
            merge_labels (p_i.label, p_j.label, clusters);
            p_i = p_j;
        end
    end
end
return clusters
```

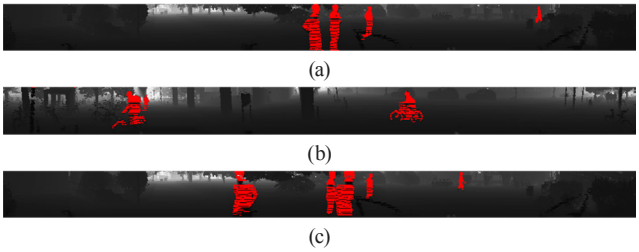


Fig. 5 DPPDM in different scenarios

LiDAR has 64 channels in the vertical direction, $H = 64$. In this paper, the panoramic depth map is divided into 870 grids, so $W = 870$. As shown in Algorithm 5, five identical, consecutive, or random DPPDMs are joined into one image to generate a LDPPDM with a size of 320 ×870 × 3. The reasons for image join are as follows:

1. more training data can be obtained;
2. the input image is not only applicable to the proposed CNN, but also applicable to Faster-RCNN [7] and PVANET [9], so as to facilitate the comparison of accuracy between different algorithms; and
3. the trained model can directly test DPPDM. The LDPPDM is shown in Fig. 6.

Algorithm 5 Image joint

```
Result: I_joint - jointed depth images
Initialization
I_dp - depth images
mode - joint mode for SAME / CONTINUE / RANDOM
Main loop:
 if mode == SAME do:
    n = I_dp.size;
    for i = 0 : n do:
        for k = 1 : 5 do:
            I_joint[i].row ((k-1)*rows, k*rows)
            = I_dp[i] (0, rows);
        end
    end
else if mode == CONTINUE do:
    n = I_dp.size / 5;
    for I = 0 : n do:
        for k = 1 : 5 do:
            I_joint[i].row ((k-1)*rows, k*rows)
            = I_dp[5*i+k] (0, rows);
        end
    end
else if mode == RANDOM do:
    n = I_dp.size;
    for I = 0 : n do:
        for k = 1 : 5 do:
            I_joint[i].row ((k-1)*rows, k*rows)
            = I_dp[random (n)] (0, rows);
        end
    end
end
return I_joint;
```

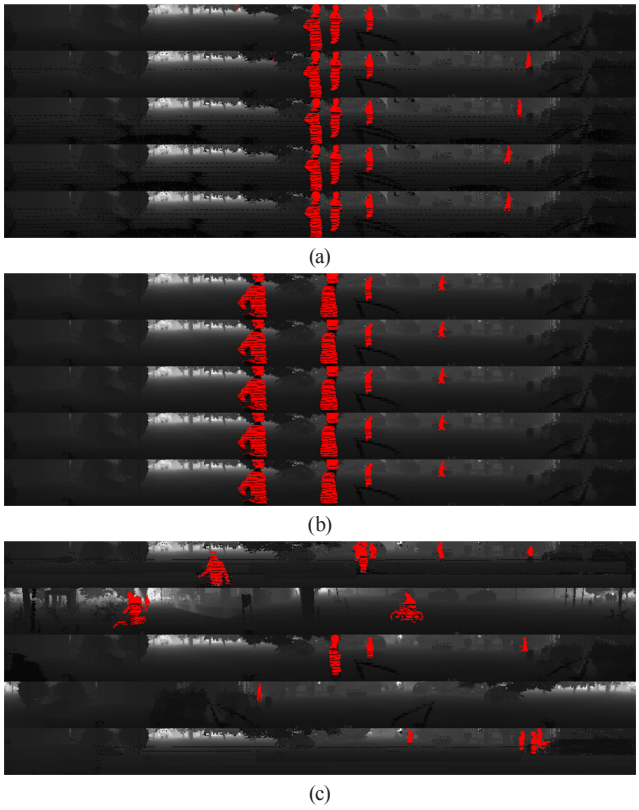


Fig. 6 LDPPDM with different jointed methods. (a) Jointed by 5 identical DPPDMs. (b) Jointed by 5 consecutive DPPDMs. (c) Jointed by 5 random DPPDMs.

3.6 Dataset establishment

The LDPPDM with a size of 320 × 870 × 3 is input of the improved PVANET. After transforming 3420 3D point clouds in KITTI dataset, the corresponding DPPDMs are obtained. Five pieces of identical, consecutive, or random DPPDMs are jointed together, which enhance the data. This jointed method can ensure the dataset is expanded or reduced in various forms as required. A new dataset is built and is split into a training set with 1644 frames and a validation set with 408 frames.

3.7 Network structure

PVANET has 25 anchors. The size of the anchor is determined by the basic size 16, 5 different scales (3, 6, 9, 16, 32) and 5 different ratios (0.5, 0.667, 1.0, 1.5, 2.0). In order to improve the detection accuracy, this study sets the scale, ratio and anchor number more suitable for the object in the new dataset by analyzing the size of the object boxes. The boxes of 25714 pedestrians in 2052 maps of the new dataset are analyzed. As can be seen from Fig. 7, the boxes range in size from 200 to 700, and the aspect ratio of the boxes is mainly distributed between 2:1 and 4:1. Further analysis of the height and width of object boxes

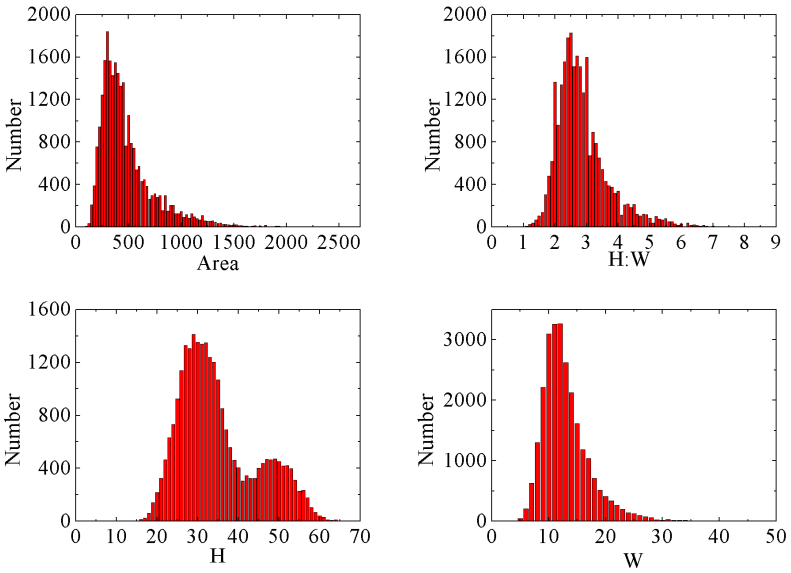


Fig. 7 Box information of pedestrians in new dataset

shows that the height is mainly distributed between 20 and 50, while the width is mainly distributed between 10 and 20. Therefore, in Fig. 7, the size of object boxes in the dataset changes little, and the average size of objects is small, which provides the basis for changing the scale and ratio to reduce the size and number of the anchor. Through analysis, the scale is set as (1, 2, 3, 5), and the ratio is set as (0.333, 0.5, 1, 2, 3, 4).

Learning rate is an important parameter in deep learning, which determines whether the objective function can converge to a local minimum and when it converges to a minimum. Since the learning rate is the weight of the negative gradient, the accuracy of the small object detection can also be increased by adjusting the learning rate. PVANET applies the learning rate strategy Plateau [31], which adjusts the learning rate by monitoring the change of loss value. However, the proportion of pedestrians in the LDPPDM is too small, which results in a large negative sample space and a slow convergence rate when training the model. When PVANET uses the strategy Plateau [31] to train the model, the learning rate remains unchanged at the initial learning rate of 0.001. Therefore, the original learning rate strategy is not applicable to the new dataset, and the learning rate strategy needs to be changed. According to [32], to find the adaptive batch size and initial learning rate, the batch size is set to 32, 64, 128, 256, 512 and the initial learning rate to 0.001 and 0.01 × *batch_size*/256, respectively. Fig. 8 shows the change of AP with different batch size and initial learning rate. It can be seen that when the batch size is 128, the initial learning rate of 0.005 can achieve the best detection result.

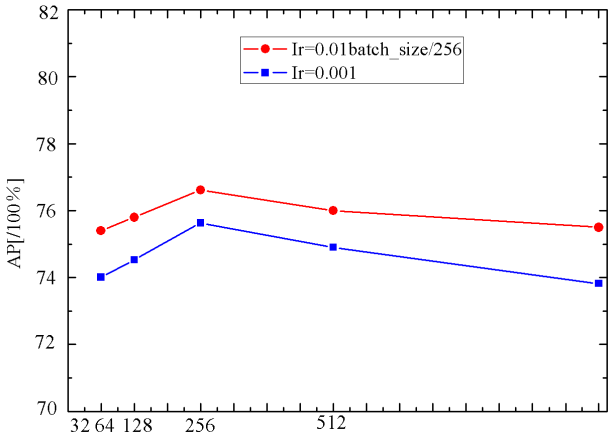


Fig. 8 AP changes with different batch size and initial learning rate

The CNN proposed in this paper is improved from PVANET, and the network structure of the proposed CNN is shown in Fig. 9. The proposed CNN is a lightweight feature extraction network structure for object detection, so it achieves real-time object detection performance without losing accuracy. The upper part of Fig. 9 shows the feature extraction network, from which feature maps are generated through C.ReLU [33], Inception [34], and HyperNet [35]. Meanwhile, batch normalization [36] and residual connections [37] optimize the performance of the model. By optimizing the output node, C.ReLU [33] reduces the number of output channels, and achieves the weight reduction of network structure, which leads to speed-up of the training without losing accuracy. Due to the convolution combination of multiple receptive fields, Inception [34] can capture both small objects and large objects in the input image. In order to detect small objects in the new dataset, 1 × 1

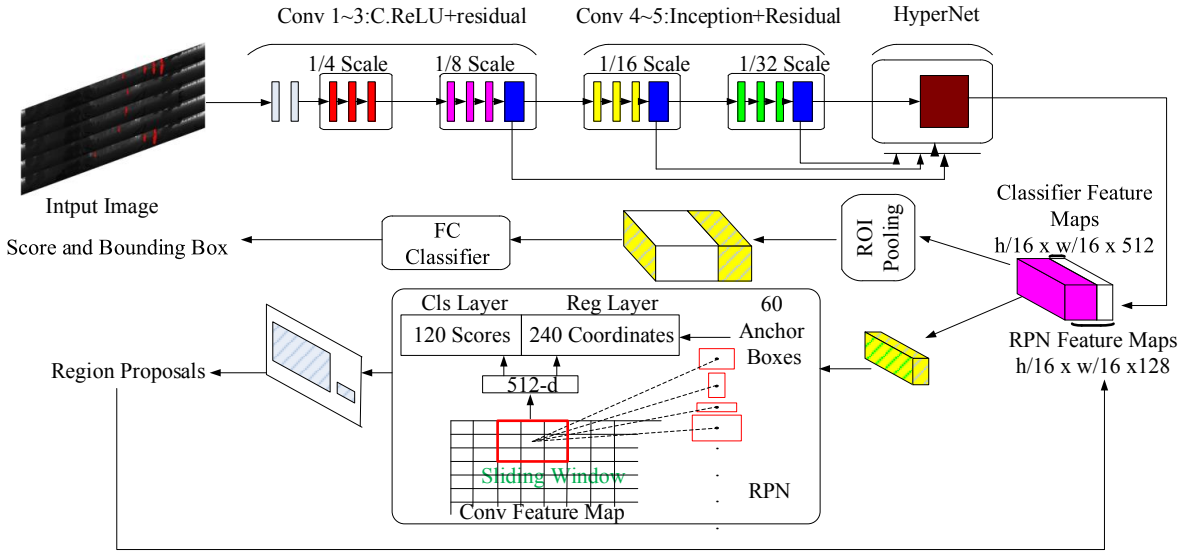


Fig. 9 Network structure of proposed CNN

convolution achieves the purpose of improving the detection accuracy of small objects by slowing down the growth of perception fields of some output features. HyperNet [35] collects 1/8, 1/16 and 1/32 of the feature maps in conv3 for feature association, which increases multi-scale information in the final feature map and realizes effective detection of multi-scale objects. The structure of the final RPN network is shown in the lower part of Fig. 9.

4 Experiment

4.1 Evaluation metrics

Average Precision (AP) and Recall are used as the main evaluation indexes to measure the performance of the model. AP and Recall are defined as follows:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{AP} = \int_0^1 \text{P}(\text{R})\,\text{dR}$$

where TP (True Positive) refers to the correct classification of positive cases as positive cases, FN (False Negative) refers to the correct classification of positive cases as negative cases, TN (True Negative) refers to the correct classification of negative cases as negative cases, FP (False Positive) refers to the correct classification of negative cases as positive cases, and P(R) refers to the corresponding relationship between the two represented by the Precision-Recall curves.

4.2 Experimental setup

The new dataset is transformed from KITTI dataset [8], which consists of a training set with 1644 frames and a validation set with 408 frames. Since the LDPPDM is jointed by 5 identical, consecutive, or random DPPDMs, there is no temporal correlation between them.

This study mainly compares the proposed CNN with Faster-RCNN and PVANET for pedestrian detection on the new dataset, and the AP, Recall (R), Frames Per Second (FPS) are used to evaluate the comprehensive performance of these models. It should be noted that the VGG16 [38] network is used for Faster-RCNN. In order to highlight the innovation of the proposed dataset obtained by Clustering and Double Projection (CDP) based on 3D point cloud in object detection, the study compares our model with the model trained without CDP.

4.3 Experimental results

Performance comparison of several algorithms is summarized in Table 1, which shows our AP is higher than PVANET and Faster-RCNN, both on the dataset with CDP and the dataset

Table 1 Detection performance of various algorithms

Method	AP/%	R/%	FPS
Ours(w/ CDP)	0.7844	0.4785	3.33
Ours(w/o CDP)	0.7608	0.5276	3.45
PVANET(w/ CDP)	0.7563	0.5219	3.21
PVANET(w/o CDP)	0.7382	0.4747	3.17
Faster-RCNN(w/ CDP)	0.3856	0.3261	3.80
Faster-RCNN(w/o CDP)	0.3142	0.3074	3.88

without CDP. The RPN feature of Faster-RCNN [7] is only obtained from the last convolutional layer. From [39], the RPN feature of Faster-RCNN [7] is not conducive to the detection of small objects, which results in the AP of Faster-RCNN [7] being significantly smaller than PVANET. As mentioned above, the proposed algorithm setting reasonable anchors and initial learning rate, and these improvements together boosted the accuracy by 2.2 % and 2.8 % in pedestrian detection both on the dataset with CDP and the dataset without CDP. In addition, the objects of the dataset with CDP are more obvious. Therefore, as shown in Table 1, models trained on the dataset with CDP perform better than that trained on the dataset without CDP.

On a ZOTAC GTX 1070 GPU, our model can process 3.33 pictures per second. Because the LDPPDM in the validation set is composed of 5 DPPDMs, the actual processing speed of a frame of DPPDM is faster, which is conducive to the actual operation of automatic driving.

Fig. 10 (a) illustrates the proposed algorithm can detect the pedestrians within the camera view that are not easily detected due to the influence of illumination. Fig. 10 (b) illustrates that the proposed algorithm can detect pedestrians outside the camera view, which is highly desirable for autonomous driving applications. Figs. 10 (c) and (d) illustrate the model trained on the dataset with CDP can detect the objects that cannot be detected by the model trained on the dataset without CDP.

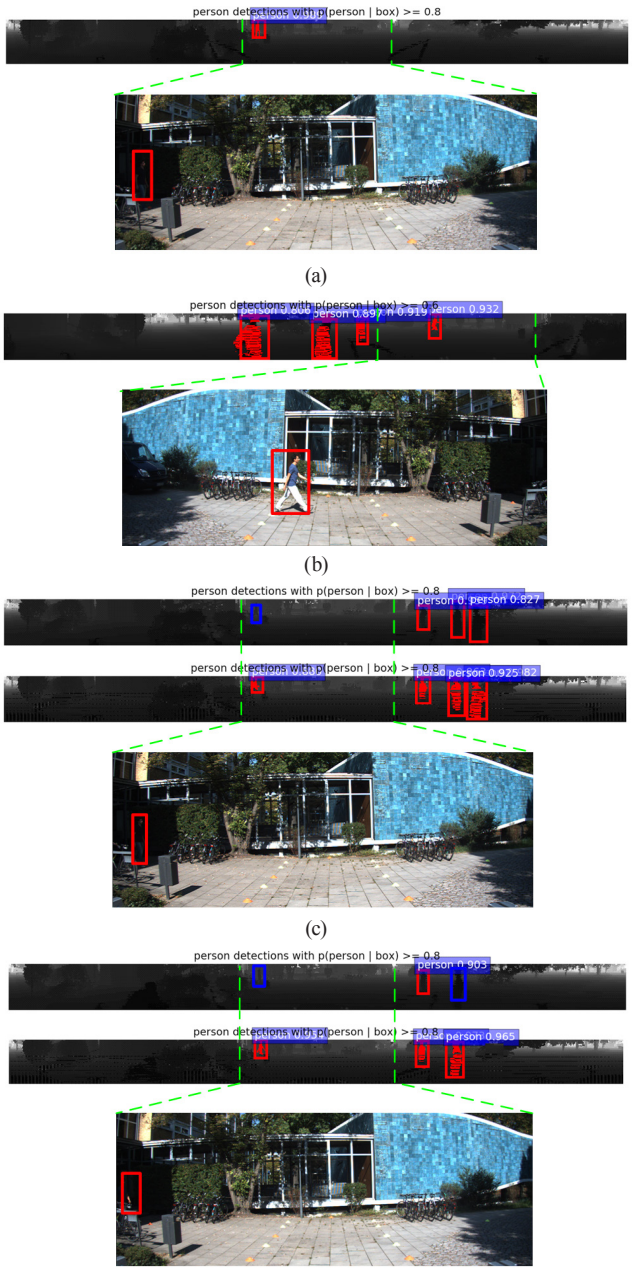


Fig. 10 Detection advantages of the proposed algorithm. (a) Not affected by light. (b) Not limited to the camera view. (c) and (d) Higher detection accuracy.

5 Conclusions

An effective object detection algorithm is proposed in this paper for pedestrian detection based on 3D LiDAR data. The pipeline of the algorithm:

1. transforms 3D LiDAR data to 2D image to ensure detection accuracy is not affected by illumination variations;
2. builds a new dataset so that pedestrians outside the camera's field of view can be accurately detected, which improves the safety of autonomous driving;
3. uses clustering and filtering to make pedestrians in the dataset more prominent, so that the detection object is separated from the background; and
4. proposes a Convolutional Neural Network based on PVANET, which increases the accuracy in pedestrian detection.

The proposed algorithm has been successfully tested in our dataset and its performance has been verified. The proposed algorithm achieves high accuracy and robustness of pedestrian detection, which has been supported by the experimental evaluation and is more suitable for autonomous driving. In the future, the proposed algorithm can be improved to apply to the detection of multiple objects in more scenarios.

Acknowledgment

This work is partially supported by the Key Technology R&D Program of Henan Province of China (No. 172102310664), the Fundamental Research Funds for the Universities of Henan Province (No. NSFRF170913)

and National Natural Science Foundation of China (No. U1304525). The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation.

References

[1] Girshick, R., Donahue, J., Darrell, T., Malik, J. "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation", In: 2014 IEEE Conference Computer Vision and Pattern Recognition, Columbus, OH, USA, 2014, pp. 580–587. https://doi.org/10.1109/CVPR.2014.81

[2] He, K., Gkioxari, G., Dollár, P., Girshick, R. "Mask R-CNN", In: 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 2017, pp. 2980–2988. https://doi.org/10.1109/ICCV.2017.322

[3] Wu, B., Wan, A., Yue, X., Keutzer, K. "SqueezeSeg: Convolutional Neural Nets with Recurrent CRF for Real-Time Road-Object Segmentation from 3D LiDAR Point Cloud", In: 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, Australia, 2018, pp. 1887–1893. https://doi.org/10.1109/ICRA.2018.8462926

[4] Wu, B., Zhou, X., Zhao, S., Yue, X., Keutzer, K. "SqueezeSegV2: Improved Model Structure and Unsupervised Domain Adaptation for Road-Object Segmentation from a LiDAR Point Cloud", In: 2019 International Conference on Robotics and Automation (ICRA), Montreal, Canada, 2019, pp. 4376–4382. https://doi.org/10.1109/ICRA.2019.8793495

[5] Redmon, J., Divvala, S., Girshick, R., Farhadi, A. "You Only Look Once: Unified, Real-Time Object Detection", In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 2016, pp. 779–788. https://doi.org/10.1109/CVPR.2016.91

[6] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., Berg, A. C. "SSD: Single Shot MultiBox Detector", In: European Conference on Computer Vision, Amsterdam, Netherlands, Springer, Cham, Switzerland, 2016, pp. 21–37. https://doi.org/10.1007/978-3-319-46448-0_2

[7] Ren, S., He, K., Girshick, R., Sun, J. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", IEEE Transactions on Pattern Analysis and Machine Intelligence, 39(6), pp. 1137–1149, 2017. https://doi.org/10.1109/TPAMI.2016.2577031

[8] Geiger, A., Lenz, P., Urtasun, R. "Are we ready for autonomous driving? The KITTI vision benchmark suite", In: 2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, 2012, pp. 3354–3361. https://doi.org/10.1109/CVPR.2012.6248074

[9] Kim, K. H., Hong, S., Roh, B., Cheon, Y., Park, M. "PVANET: Deep but Lightweight Neural Networks for Real-time Object Detection", Computer Science: Computer Vision and Pattern Recognition (arXiv:1608.08021), pp. 1–7, 2016. [online] Available at: https://arxiv.org/abs/1608.08021 [Accessed: 30 September 2016]

[10] Wang, Y., Chao, W. L., Garg, D., Hariharan, B., Campbell, M., Weinberger, K. Q. "Pseudo-LiDAR from Visual Depth Estimation: Bridging the Gap in 3D Object Detection for Autonomous Driving", Computer Science: Computer Vision and Pattern Recognition (arXiv:1812.07179), pp. 1–16, 2019. https://doi.org/10.1109/CVPR.2019.00864

[11] Ma, X., Wang, Z., Li, H., Zhang, P., Fan, X., Ouyang, W. "Accurate Monocular Object Detection via Color-Embedded 3D Reconstruction for Autonomous Driving", Computer Science: Computer Vision and Pattern Recognition (arXiv:1903.11444), pp. 1–11, 2019. [online] Available at: https://arxiv.org/abs/1903.11444 [Accessed: 12 August 2019]

[12] Szemenyei, M., Vajda, F. "3D Object Detection and Scene Optimization for Tangible Augmented Reality", Periodica Polytechnica Electrical Engineering and Computer Science, 62(2), pp. 25–37, 2018. https://doi.org/10.3311/PPee.10482

[13] Zhou, Y., Tuzel, O. "VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection", In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 2018, pp. 4490–4499. https://doi.org/10.1109/CVPR.2018.00472

[14] Minemura, K., Liau, H., Monrroy, A., Kato, S. "LMNet: Real-time Multiclass Object Detection on CPU Using 3D LiDAR", In: 2018 3rd Asia-Pacific Conference on Intelligent Robot Systems (ACIRS), Singapore, Singapore, 2018, pp. 28–34. https://doi.org/10.1109/ACIRS.2018.8467245

[15] Zeng, Y., Hu, Y., Liu, S., Ye, J., Han, Y., Li, X., Sun, N. "RT3D: Real-Time 3-D Vehicle Detection in LiDAR Point Cloud for Autonomous Driving", IEEE Robotics and Automation Letters, 3(4), pp. 3434–3440, 2018. https://doi.org/10.1109/LRA.2018.2852843

[16] Engelcke, M., Rao, D., Wang, D. Z., Tong, C. H., Posner, I. "Vote3Deep: Fast object detection in 3D point clouds using efficient convolutional neural networks", In: 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, Singapore, 2017, pp. 1355–1361. https://doi.org/10.1109/ICRA.2017.7989161

[17] Li, B. "3D fully convolutional network for vehicle detection in point cloud", In: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, Canada, 2017, pp. 1513–1518. https://doi.org/10.1109/IROS.2017.8205955

[18] Beltrán, J., Guindel, C., Moreno, F. M., Cruzado, D., García, F., De La Escalera, A. "BirdNet: A 3D Object Detection Framework from LiDAR Information", In: 2018 21st International Conference on Intelligent Transportation Systems (ITSC), Maui, HI, USA, 2018, pp. 3517–3523. https://doi.org/10.1109/ITSC.2018.8569311

[19] Yang, B., Luo, W., Urtasun, R. "PIXOR: Real-time 3D Object Detection from Point Clouds", In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 2018, pp. 7652–7660. https://doi.org/10.1109/CVPR.2018.00798

[20] Qi, C. R., Liu, W., Wu, C., Su, H., Guibas, L. J. "Frustum PointNets for 3D Object Detection from RGB-D Data", In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 2018, pp. 918–927. https://doi.org/10.1109/CVPR.2018.00102

[21] Ku, J., Mozifian, M., Lee, J., Harakeh, A., Waslander, S. L. "Joint 3D Proposal Generation and Object Detection from View Aggregation", In: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 2018, pp. 5750–5757. https://doi.org/10.1109/IROS.2018.8594049

[22] Xu, D., Anguelov, D., Jain, A. "PointFusion: Deep Sensor Fusion for 3D Bounding Box Estimation", In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 2018, pp. 244–253. https://doi.org/10.1109/CVPR.2018.00033

[23] Qi, C. R., Yi, L., Su, H., and Guibas, L. J. "Pointnet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space", Advances in Neural Information Processing Systems, pp. 5099–5108, 2017.

[24] Qi, C. R., Su, H., Mo, K., Guibas, L. J. "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation", In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 2017, pp. 77–85. https://doi.org/10.1109/CVPR.2017.16

[25] Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., Zisserman, A. "The PASCAL Visual Object Classes (VOC) Challenge", International Journal of Computer Vision, 88(2), pp. 303–338, 2010. http://doi.org/10.1007/s11263-009-0275-4

[26] Liu, W., Liao, S., Hu, W., Liang, X., Chen, X. "Learning Efficient Single-Stage Pedestrian Detectors by Asymptotic Localization Fitting", In: European Conference on Computer Vision (ECCV), Munich, Germany, 2018, pp. 643–659. https://doi.org/10.1007/978-3-030-01264-9_38

[27] Li, Z., Peng, C., Yu, G., Zhang, X., Deng, Y., Sun, J. "DetNet: Design Backbone for Object Detection", In: European Conference on Computer Vision, Munich, Germany, 2018, pp. 339–354. https://doi.org/10.1007/978-3-030-01240-3_21

[28] Huang, J. P., Shi, Y. H., Gao, Y. "面向小目标的多尺度Faster_RCNN检测算法" (Multi-scale Faster-RCNN algorithm for small object detection), Journal of Computer Research and Development, 56(2), pp. 319–327, 2019. (in Chinese) https://doi.org/10.7544/issn1000-1239.2019.20170749

[29] Xu,Z.,Wang,Y.H."基于候选区域和并行卷积神经网络的行人检测" (Pedestrian detection based on candidate regions and parallel convolutional neural network), Computer Engineering and Applications, pp. 1–12, 2019. https://doi.org/10.3778/j.issn.1002-8331.1902-0004

[30] Zermas, D., Izzat, I., Papanikolopoulos, N. "Fast Segmentation of 3D Point Clouds: A Paradigm on LiDAR Data for Autonomous Vehicle Applications", In: 2017 IEEE International Conference on Robotics and Automation (ICRA), Marina Bay Sands, Singapore, 2017, pp. 5067–5073. https://doi.org/10.1109/ICRA.2017.7989591

[31] Li, J., Peng, K., Chang, C. C. "An Efficient Object Detection Algorithm Based on Compressed Networks", Symmetry, 10(7), Article number: 235, 2018. https://doi.org/10.3390/sym10070235

[32] Mishkin, D., Sergievskiy, N., Matas, J. "Systematic evaluation of convolution neural network advances on the ImageNet", Computer Vision and Image Understanding, 161, pp. 11–19, 2017. https://doi.org/10.1016/j.cviu.2017.05.007

[33] Shang, W. L., Sohn, K., Almeida, D, Lee, H. "Understanding and improving convolutional neural networks via concatenated rectified linear units", Computer Science: Machine Learning (arXiv:1603.05201), pp. 2217–2225, 2016. [online] Available at: https://arxiv.org/abs/1603.05201 [Accessed: 19 July 2016]

[34] Szegedy, C., Liu, W., Jia, Y. Q., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A. "Going deeper with convolutions", In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 2015, pp. 1–9. https://doi.org/10.1109/CVPR.2015.7298594

[35] Kong, T., Yao, A. B., Chen, Y. R., Sun, F. C. "HyperNet: Towards Accurate Region Proposal Generation and Joint Object Detection", In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 2016, pp. 845–853. https://doi.org/10.1109/CVPR.2016.98

[36] Ioffe, S., Szegedy, C. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift", Computer Science: Machine Learning (arXiv:1502.03167), pp. 1–11, 2015. [online] Available at: https://arxiv.org/abs/1502.03167 [Accessed: 2 March 2015]

[37] He, K. M., Zhang, X. Y., Ren, S. Q., Sun, J. "Deep Residual Learning for Image recognition", In: Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR), Nevada, USA, 2016, pp. 770–778. https://doi.org/10.1109/CVPR.2016.90

[38] Simonyan, K., Zisserman, A. "Very Deep Convolutional Networks for Large-Scale Image Recognition", Computer Science: Computer Vision and Pattern Recognition (arXiv:1409.1556), 2014, pp. 1–14. [online] Available at: https://arxiv.org/abs/1409.1556 [Accessed: 10 April 2015]

[39] Hosang, J., Benenson, R., Dollár, P., Schiele, B. "What Makes for Effective Detection Proposals?", IEEE Transactions on Pattern Analysis and Machine Intelligence, 38(4), pp. 814–830, 2016. http://doi.org/10.1109/TPAMI.2015.2465908