Proceedings of the 5th International Conference on Computing and Data Science
DOI: 10.54254/2755-2721/13/20230703

Point cloud densification via symmetry

Yu Wei

Faculty of Computer Science, Harbin Institute of Technology, Harbin, China

ywei.binary@gmail.com

Abstract. Three-dimensional objects are usually represented by point cloud based on lidar reflection of the sensors. However, the point clouds are commonly sparse since the lidar reflection is restricted by the location where the machine scans the objects around it. Commonly, the point cloud data that we use in autonomous driving are basically concentrated on one side or two sides and can seldom depict the whole image of all the objects. In this paper, we propose methods based on symmetry to diminish this intrinsic problem. Our work uses *CenterPoint* as our backbone and we do some fine-tune on it to make the data augmented. Moreover, we use FutureDet as the detector and the predictor to see whether the results of the methods fit the design. We obtain the information of *CenterPoint* from the detector and use the position information of the center point to do symmetry so as to make the points exist on the other side. We are trying to address the issue by comparing the results of metrics from the *FutureDet* and the fine-tune model on nuScenes dataset.

Keywords: computer vision, 3D detection, point cloud, data augmentation.

1. Introduction

Object detection is what we use everyday through all kinds of devices. With the development of technology, 2D image detection becomes more and more mature. Many methods can do this job quickly and perfectly such as Yolo, Faster-RCNN and so on. However, these methods cannot meet our needs since we live in a 3D world. Strong 3D perception is an indispensable ingredient in many scenes of our real lives like autonomous diving [1]. Compared to 2D detection which is based on images, 3D detection mostly uses point cloud data as the input. However, this kind of data has some inherent issues. One of them is sparsity. Most regions of objects are not measured by the sensor and the points are on the surface of the objects assembling on the sides facing the lidar machine. In order to improve the performance of detection model and get more deep feature of the object, in this paper, we try to propose a new method using the thought of data augmentation to reduce sparse distribution of the point cloud data. Data augmentation is a useful and pragmatic technique that is common in all kinds of enhancement of the model, which can increases the amount of data and can remove the particularity of the data [2]. Thus, it seems necessary to propose a new data augmentation method to address the sparsity problem.

Flipping, rotation, clipping, adding noise [3,4] can both increase the amount of the date and diversify the variety of dataset so as to improve the generalization of the data and make the model robust when it meets all kinds of different situation. Nowadays, these operations are quite common in the data processing [5] which happens before training and inference. Thus, in our humble opinion,

DOI: 10.54254/2755-2721/13/20230703

when we deal with point cloud data, the same operations which we do on 2D images can also be used. our target is trying to use data augmentation to alleviate the problem of sparsity. We try to use data augmentation to get more features of the object [6], since most of the features are extracted from the point cloud accumulated on the surface of one side. More information of the object is needed [8] such as the points in the inner of the object and the points on the other side of the objects which fail to be detected by the sensor. To some extent, such data augmentation can improve the performance of the original model empirically [7].

In this paper, we propose a new point cloud augmentation method which use center symmetry to go deep into the objects to get more features. Our backbone model output heatmaps and centers that can depict the information of predicted trajectory and the rough location. Based on our methods, we do symmetry by using the centers' features which will augment the amount of the points in or on the surface of objects. More features can help regress more precise bounding box in the detection task.

To regress a better bounding box for the object, we cannot only use the first frame's data to do the symmetry. Points from different frames are mapped together to reduce the excessive dependence on the results of first frame. However, different points from different frames come from different moment. In order to map them together, we do the operation with the help of speed which can also get from the information of center points when we run forecasting of the model [9,10]. After we acquiring all of these center point features, we can compute bounding box by using our symmetry methods.

We try to test our proposed data augment strategy on nuScenes dataset and since the restriction of our resource, we only pay attention to the most important sub-type data in autonomous driving which is the vehicles.

2. Related work

2.1. Center-based network

Different from anchor-based network relying on massive number of anchors to search on the image, center-based net work is an anchor-free detecting network. It is faster and more accurate than the traditional anchor-based network like faster-RCNN [11]. Most of the well-known object detection networks describe the target as a bounding box, and then give each bounding box a corresponding category, which generally comes with the disadvantages of inefficient operation and additional post-processing. In this case, the center-based network uses the center point to describe the target, and then returns to different target attributes (such as: size, 3D position, orientation, and attitude) according to the task, and this is Centernet [12]. Compared with the traditional object detection network can only do the two-dimensional object detection of the target, Centernet only needs to make simple modifications to adapt to the two-dimensional object detection, three-dimensional object detection and key point detection and other tasks, and has a good performance in detection efficiency and accuracy.

2.2. Data augmentation

Data augmentation is really important to some tasks since data amount can always influence the performance of one model. When we talk about 2D data augmentation, traditional ways [13] like flipping, symmetry, rotation, and mosaic are really common. These can be used in all kinds of tasks to increase the amount and the diversity of data and alleviate the problem of overfitting. However, in 3D point cloud world, thing become more difficult. It is not only because that the point cloud is more complicated on the data organization, but also because it is separate in the 3D space. As a result, when you try to make more points, the depth and location of every points can influence our operation. The traditional methods may also be useful under some certain circumstance. We still need to propose new methods. For example, in the recent work "Multimodal Virtual Point 3D Detection" [14], the authors successfully use 2D RGB images to create virtual 3D points. With the help of depth information, the method can generate 3D-point cloud near the target objects.

DOI: 10.54254/2755-2721/13/20230703

2.3. Object detection

The 2D object detection is mature, and the representative works include 2 categories: The first category is Faster-RCNN, Mask-RCNN and so on, which use RPN structure. The second category is one shot model like YOLOv1-YOLOv3. 3D object detection has spewed since 2019. The task's target is to regress three dimensional bounding boxes of the objects which need to be detected [15-19]. Different from 2D detectors, we ask more information from 3D detectors including three more angles of each axis and the size of one extra dimension. It is also complicated because of the complex environment of the real world. Thus, 3D object detection is an important and tough task. In this field some excellent works have been done recently. We usually divide these methods into two large categories: one-stage method and two-stage method. The one-stage method performs object detection by directly regressing the class label and position coordinates of the object, while the two-stage method has two steps, first the algorithm predicts a series of 3D candidate boxes, and then further sample classification. In the paper *CenterPoint* [20], the authors proposed a brand new method to detect 3D objects. The method is a classic two-stage detector. The first stage is to detect the centers of objects and regress to other attributes, including 3D size, 3D orientation, and velocity. In a second stage, it refines these estimates using additional point features on the object. The also use the connection between different centers from different frames to predict the trajectory. The center-based 3D detection is good at detect objects in all kinds of anchors without anchors and this can help the downstream tasks like track trajectory and so on.

2.4. Object tracking

Object tracking means that once the initial position of the target object is defined, you can estimate or predict the position of the target object in every successive frame. Object detection, on the other hand, is the process of detecting a target object in every single frame of video. Object detection works only if the target image is visible on the given input. In autonomous driving task, 3D object tracking is an important part of the whole process. We usually use LiDAR data such as point cloud data as our input since the lidar machine can generate points to fix an object's location. The point cloud data is disorder and sparse, the tracking task of this kind of data is different from 2D images in video. Many great works have made some great process in this task. For example, in the paper *FutureDet* [22,23], the authors propose a new method to predict the trajectory. By linking future and current locations in a many-to-one manner, the method is able to predict multiple futures, which is different from traditional end-to-end approaches. We are also inspired by FutureDet to achieve our work.

3. Dataset

The nuScenes dataset is a large-scale dataset for models of autonomous driving tasks to do training and test. The 3D object dataset and its annotations are developed by the team at Motional. NuScenes data collection was mainly carried out in Boston and Singapore, and the vehicles used for acquisition were equipped with 1 spinning LIDAR, 5 long range radar sensors and 6 cameras. The amount of data annotation is more than 7 times higher than that of KITTI. Typically, each set of data is extracted from a continuous 20-second long sequence of frames representing sensor data at a certain point in time. The dataset includes high-definition (HD) maps, sensor data, and annotations collected from real-world driving scenarios in various urban environments. It focuses on complex and diverse urban driving scenes and provides rich and diverse sensor data, including lidar point clouds, camera images, radar data, and calibrated sensor information.

In the pursuit of advancing computer vision and autonomous driving, the nuScenes dataset can work as a vital resource. With a focus on supporting autonomous driving research, this extensive dataset offers a comprehensive collection of diverse driving scenes in two traffic-challenging urban environments—Boston and Singapore. By meticulously selecting 1000 unique driving sequences, we think that nuScenes dataset captures all kinds of possible traffic scenes and diverse intricate driving situations, presenting researchers with rich data for experiment.

DOI: 10.54254/2755-2721/13/20230703

Since nuScenes recognized the significance of encompassing various road scenarios and potential occurrences, the dataset incorporates an extensive array of object types and road conditions, ensuring a thorough representation of driving environments. Meticulously annotates 23 object categories, one of the standout features of nuScenes is its precise annotation of object classes. These annotations manifest as accurate 3D bounding boxes, consistently provided at a rate of 2Hz throughout the entire dataset.

The nuscenes dataset schema comprises 13 components, each representing a specific type of information with its unique token. Motional, to assist users in accessing and developing the dataset, offers a dedicated Python library known as the nuscenes devkit. This library encompasses various functionalities such as data retrieval, coordinate transformation, visualization, and more. By utilizing the nuscenes devkit, users can conveniently access and manipulate the dataset, making it easier to work with the data and develop applications based on it. The library acts as a valuable resource for efficiently navigating and utilizing the nuscenes dataset.

In our paper this time, due to the limited time and computing resources, we only use vehicle category as a test of our methods.

4. Methods

We use existing 3D detection method *3D CenterPoint* and *FutureDet* as our backbones. What we contribute in this paper is proposing a new point acquisition method and new point symmetry operations. The two types of methods can help us finish the data augmentation operation.

4.1. Introduction to backbone

Generally, we use only 4 numerical value in the point cloud data, which are x,y,z,r. The x,y,z is the location of the point. The r represent reflection intensity which can help us know the depth of the point. In object detection, what we need to do is to find which bounding box the point belongs to. The parameters of the bounding box are u, v, o, w, l, h, θ. The first three parameters are the location of center point. The next three parameters are the size of bounding box. The final parameter is the yaw rotation along z axis.

In one of our backbones *Centerpoint*, the authors use two methods which are VoxelNet [24] and PointPillars [25] as their backbones. In our experiments, we choose VoxelNet as our detector, which divides different points into certain regular bins. VoxelNet then extract features from all the points inside the bins. The output of 3D convolutions is a feature map.

Simultaneously, within the output layer, we employ various detection techniques to estimate distinct attributes like the center point, velocity, and box size. Moreover, it is important to highlight that speed computation is based on the center point of bounding boxes in consecutive frames, along with the time interval separating them. Given the independence of each frame, we can then regress the bounding box (including size and rotation) and center point for each vehicle within every individual frame.

In order to predict object trajectories, we employ a method that involves shifting the center point of each object to the subsequent frame based on its speed. Subsequently, we calculate the distances from the detected object's bounding box center in the next frame. By selecting the closest distance, we can determine the object's anticipated location in the following frame, effectively accomplishing trajectory prediction. For this particular experiment, we can decide to set the number of time step which is a hyper-parameter, meaning that we utilize n consecutive frames to forecast the object's trajectory.

4.2. Symmetry point acquisition and symmetry operation

To address the issue of point cloud sparsity, particularly when there is limited representation of distant objects, we propose employing point symmetry as a solution. This approach aims to mitigate the problem of sparsity and enhance the number of point representations for distant objects.

As we all know, in our real life almost all the cars and other vehicles are center-symmetrical cuboids which gives us the hint to do the symmetry operation. The point cloud data is obtained by the

DOI: 10.54254/2755-2721/13/20230703

reflection of lidar waves. As a result, the points are mainly accumulated on only one or two sides of objects which facing lidar sensors. We try to do symmetry to get the points on the other sides of objects which can be helpful when we compute the information of bounding box and so on.

The first thing we need to do is to obtain the scope of points to do symmetry which is the point acquisition operation. After that we are required to find the center for symmetry operation and decide how to do symmetry.

In this paper, we introduced two symmetry operations. The first operation involves determining another point on the same straight line as the original point and the center point which we can get from the output of model. This is a simple and naive symmetry method. The center point is computed as the midpoint of the line connecting the generated point and the original point. The second operation involves using the original point as one vertex and the center point as the center of a cube. By symmetrically generating points, we can obtain the other seven vertices of the cube.

With the features and information obtained from the 3D object detector, including the object's center and bounding box size, we can leverage this data effectively. To determine which points require symmetry and which points serve as the centers of symmetry, we explored four different methods.

Method 1: The first method involves calculating the length of the body diagonal of the bounding box cuboid based on its size. Using half of this diagonal length as a threshold, a symmetry operation is performed on points within this threshold range from the center point of each object. By applying this operation, seven symmetrically generated points are obtained for each qualifying point.

Method 2: Similar to the first method, the half-diagonal length serves as the threshold. However, in this method, only one symmetry operation is allowed for points within the threshold range of an object. The center of symmetry is determined by the object's closest center point. This ensures that each point near the center of an object is exclusively associated with that particular object. Additionally, seven points are generated through the second symmetry operation for each original point.

Method 3: This method follows the same approach as the second method, but with a distinction. It employs the first symmetry operation to generate a single point for each original point.

Method 4: Considering that the symmetry range thresholds in the previous methods (half-body diagonals) may include points unrelated to the object, the fourth method aims to address this issue. By reducing the threshold size, it is set to the minimum value among the length, width, and height of the bounding box. This adjustment ensures that symmetrical points generated must belong to the object, providing a more accurate and reliable symmetry operation.

By employing these four methods, we explore different approaches to determine the points that require symmetry and establish the centers of symmetry. Each method offers its own advantages and considerations in achieving precise and effective symmetry operations.

5. Experiments and results

Our experiment is basically use the backbone *CenterPoint* for 3D detection and *Futuredet* for Trajectory.

For 3D object detector, we use *VoxelNet* as the backbone architecture. For the casual data augmentation, we do the same operation of *CenterPoint* , use global random noise scaling and global random rotations.

For dataset, we only use the data of vehicles and try to divide them into three sub-categories: static car, linearly moving car and non-linearly moving car. The foundations of the division are as below:

We consider IoU for every target objects in first and last frames. If the IoU is greater than 0, we think the object as static. We then utilize the speed of the first frame of a vehicle to extrapolate its bounding box position from the first frame to the last frame. By comparing this extrapolated bounding box with the detected bounding box in the last frame, we determine if there is an overlap. If there is an overlap, we classify the car's trajectory as linear, indicating that it is moving in a straight line. On the other hand, trajectories that do not exhibit overlap and are not classified as static are considered nonlinear trajectories, indicating that the vehicle is moving in a non-straight path. This classification allows us to differentiate between linear and nonlinear motion patterns in the dataset.

DOI: 10.54254/2755-2721/13/20230703

We also use some evaluation indicators in this task which are common in autonomous driving topics listed as below:

1) Mean Average Precision (MAP): This metric calculates the average accuracy across all categories. It measures the precision of the model's predictions for each category and takes the mean value to provide an overall assessment of the model's performance.

2) Mean Average Recall (MAR): Similar to MAP, this metric calculates the average recall across all categories. It measures the model's ability to correctly identify instances of each category and computes the mean value to provide an overall evaluation of the model's recall performance.

3) Average Displacement Error (ADE): This metric quantifies the average difference between the predicted points and the corresponding ground-truth points. It measures the accuracy of the model's predictions by calculating the average displacement error for all predicted points.

4) Final Displacement Error (FDE): FDE calculates the distance between the predicted position and the corresponding ground-truth value at the end of the prediction period. It provides an indication of the model's accuracy in predicting the final position of an object or trajectory.

5) Miss Rate (MR): The miss rate metric represents the rate of missed detection results in the model's detection results. It indicates the proportion of instances or objects that were not detected by the model. A lower miss rate signifies a higher level of detection accuracy.

These evaluation metrics provide quantitative measures to assess different aspects of the model's performance, including precision, recall, displacement accuracy, and detection rates. They help in understanding the strengths and weaknesses of the model's predictions and are commonly used in object detection and tracking tasks.

The performance evaluation results are presented in the following tables:

Table 1. Model performance using mini dataset.

CLASS	mAP	mAR	ADE	FDE	MR
static_car	0.025	0.096	13.96	27.05	0.974
linear_car	0.0	0.124	13.96	27.05	0.974
nonlinear_car	0.0	0.025	1.0	1.0	1.0

Table 2. Model performance using full dataset(Futuredet method).

CLASS	mAP	mAR	ADE	FDE	MR
static_car	0.749	0.954	0.385	0.614	0.065
linear_car	0.365	0.985	0.332	0.600	0.073
nonlinear_car	0.956	0.962	0.315	0.598	0.076

Table 3. Model performance using symmetry method 1.

CLASS	mAP	mAR	ADE	FDE	MR
static_car	0.433	0.697	2.629	5.012	0.356
linear_car	0.004	0.926	2.552	4.944	0.338
nonlinear_car	0.758	0.800	2.537	4.980	0.340

Table 4. Model performance using symmetry method 2.

CLASS	mAP	mAR	ADE	FDE	MR
static_car	0.492	0.745	2.215	4.207	0.292
linear_car	0.004	0.926	2.144	4.116	0.280
nonlinear_car	0.727	0.800	2.145	4.173	0.289

DOI: 10.54254/2755-2721/13/20230703

Table 5. Model performance using symmetry method 3.

CLASS	mAP	mAR	ADE	FDE	MR
static_car	0.679	0.927	0.472	0.800	0.088
linear_car	0.175	0.971	0.457	0.793	0.088
nonlinear_car	0.959	0.957	0.452	0.808	0.096

Table 6. Model performance using symmetry method 4.

CLASS	mAP	mAR	ADE	FDE	MR
static_car	0.699	0.937	0.389	0.642	0.069
linear_car	0.182	0.971	0.356	0.625	0.081
nonlinear_car	0.930	0.950	0.393	0.692	0.087

6. Conclusion and future work

Because of time constraints and limited computing resources in this experiment, the results did not meet our initial expectations. We can see from the number of assessment indicator, despite implementing the most comprehensive symmetric strategy, it did not outperform the baseline method, *Futuredet*. Analyzing the results, we observed that the deviations between the predicted center point and the ground truth center point can easily influence our final results. These deviations can introduce some inaccuracies in our point acquisition. Also when we do symmetry, the most important information is the location of centers, but it seems not so accurate, thereby affecting the final bounding box regression.

Nevertheless, after a horizontal comparison of our methods, we can still find consistent improvement in our model's performance, indicating that the symmetry operations still work in some way. However, there are still several areas we can focus on for the future work.

The first and foremost one we need to do is to effectively address the deviation caused by inaccurate prediction of the center point during symmetry operations.

Additionally, our paper explores the idea of combining information from the preceding and subsequent frames to predict the trajectory and velocity and other information. The goal is to map corresponding points of objects in these frames, in order to supplement the available points. However, we encountered a challenging problem in this aspect. When performing adjacent frame mapping, not only do we require the center point for each frame of the object, but we also need the bounding box information. If our predictions for the bounding box or center point are inaccurate, it can lead to incorrect superposition of objects. Consequently, we find the method too reliant on the initial output of the network.

The third is to improve trajectory prediction for non-linearly moving vehicles, as the current model tends to perform better with linearly moving or stationary cars. Moreover, achieving multi-future predictions and generating diverse, multi-modal predictions remains an open challenge. These areas present valuable avenues for further investigation and careful study in future research endeavors.

The results indicate a consistent deviation of tens of centimeters between the output of the first network, the predicted bounding box, and the center point of the box, compared to the ground truth. This discrepancy introduces the possibility of transmitting incorrect information. We have yet to devise an effective solution to address this challenge.

References

[1] Mayank Bansal, Alex Krizhevsky, and Abhijit Ogale. Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. RSS, 2019.

[2] Suorong Yang, Weikang Xiao, Mengcheng Zhang, Suhan Guo, Jian Zhao and Furao Shen. Image Data Augmentation for Deep Learning: A Survey. CVPR, 2022. arXiv:2204.08610

[3] A survey on Image Data Augmentation for Deep Learning. J Big Data 6, 60 (2019). https://doi.org/10.1186/s40537-019-0197-0
[4] Neehar Peri, Jonathon Luiten, Mengtian Li, Aljoˇsa Oˇsep, Laura Leal-Taix´e, Deva Ramanan. Forecasting from LiDAR via Future Object Detection. CVPR, 2022.
[5] Hsu-kuang Chiu, Antonio Prioletti, Jie Li, and Jeannette Bohg. Probabilistic 3d multi-object tracking for autonomous driving. arXiv:2001.05673, 2020. 2, 3, 6, 8
[6] Chenxu Luo, Xiaodong Yang, Alan Yuille. Exploring Simple 3D Multi-Object Tracking for Autonomous Driving. ICCV, 2021.
[7] C. Shorten, T.M Khoshgoftaar. A survey on Image Data Augmentation for Deep Learning. J Big Data 6, 60 (2019). https://doi.org/10.1186/s40537-019-0197-0
[8] Jason Wang and Luis Perez. The Effectiveness of Data Augmentation in Image Classification using Deep Learning. CVPR, 2017, arXiv.1712.04621
[9] Xinshuo Weng and Kris Kitani. A Baseline for 3D Multi-Object Tracking. IROS, 2020.2,3,5, 6C. Shorten, T.M Khoshgoftaar.
[10] J. Redmon, A. Farhadi. YOLOv3: An Incremental Improvement. CVPR, 2018, arxiv:1804.02767
[11] Tianwei Yin, Xingyi Zhou and Philipp Kr¨ahenb¨uhl. Multimodal Virtual Point 3D Detection. NeurIPS. 2021
[12] Tianwei Yin, Xingyi Zhou and Philipp Kr¨ahenb¨uhl. Center-based 3D Object Detection and Tracking. CVPR, 2021
[13] Jun-Yan Zhu, Taesung Park, Phillip Isola and Alexei A Efros. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networkss. ICCV. 2017
[14] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. CVPR, 2012.
[15] Alex H. Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. CVPR, 2019.
[16] Ming Liang, Bin Yang, Yun Chen, Rui Hu, and Raquel Urtasun. Multi-task multi-sensor fusion for 3d object detection. CVPR, 2019.
[17] Charles R Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J Guibas. Frustum pointnets for 3d object detection from rgb-d data. CVPR, 2018.
[18] Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. Sensors, 2018.
[19] Zetong Yang, Yanan Sun, Shu Liu, and Jiaya Jia. 3dssd: Point-based 3d single stage object detector. CVPR, 2020.
[20] Zetong Yang, Yanan Sun, Shu Liu, Xiaoyong Shen, and Jiaya Jia. Std: Sparse-to-dense 3d object detector for point cloud. ICCV, 2019.
[21] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. CVPR, 2018.
[22] K. He, G. Gkioxari, P. Doll´ar and R. Girshick, ”Mask R-CNN,” 2017 IEEE International Conference on Computer Vision (ICCV), 2017, pp. 2980-2988, doi: 10.1109/ICCV.2017.322.
[23] Mayank Bansal, Alex Krizhevsky, and Abhijit Ogale. Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. RSS, 2019.
[24] Ren S, He K, Girshick R, et al. Faster r-cnn: Towards real-time object detection with region proposal networks[J]. Advances in neural information processing systems, 2015, 28.
[25] Duan K, Bai S, Xie L, et al. Centernet: Keypoint triplets for object detection[C]//Proceedings of the IEEE/CVF international conference on computer vision. 2019: 6569-6578.