Sustainable Cities and Society 54 (2020) 102002

Contents lists available at ScienceDirect

Sustainable Cities and Society

journal homepage: www.elsevier.com/locate/scs

3D object detection based on sparse convolution neural network and feature fusion for autonomous driving in smart cities

Lei Wang[a,b], Xiaoyun Fan[a,1], Jiahao Chen[a], Jun Cheng[a,b,*], Jun Tan[c,d], Xiaoliang Ma[e]

[a] Shenzhen Institutes of Advanced Technology (SIAT) of the Chinese Academy of Sciences (CAS), Shenzhen 518055, China
[b] The Chinese University of Hong Kong, HK, China
[c] School of Mathematics, Sun Yat-sen University, Guangzhou 510275, China
[d] Guangdong Province Key Laboratory of Computational Science, Guangzhou 510275, China
[e] The College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China

ARTICLE INFO

Keywords:
Autonomous driving
3D object detection
Deep neural network
Feature fusion
Voxelization

ABSTRACT

People in cities are suffering from traffic congestion and air pollution in daily life partly due to a great number of private cars, and always face the danger of accidents, so autonomous driving is developed by many institutes and companies especially in recent years. Autonomous driving will play an import role in the future smart cities, reduce the time and economic cost of the whole society, and be helpful for the sustainability of the city and society. A significant task for autonomous driving is to detect surrounding objects accurately in real-time, including car, pedestrian, cyclist, etc. In this paper, we propose one end-to-end three dimensional (3D) object detection method based on voxelization, sparse convolution, and feature fusion. The proposed method exploits only point cloud as input, and it has two key components—small voxels and efficient feature fusion. Instead of utilizing extra networks to transform voxels, we directly average the points within each voxel as their feature representation. To enrich features for prediction, we have designed a two-step feature fusion method called *fusion of fusion network* that can combine information of multiple scales and 3D space. We have submitted to the official test server of the 3D detection benchmark—KITTI, and achieved state-of-the-art performance especially in the Cyclist class. Besides, detection speed of our method achieves 0.05 s/frame with a 2–4 fold runtime improvement against state-of-the-art methods due to its simple and compact architecture.

1. Introduction

Convenient traffic system is important for sustainable cities and society. However, the problem of traffic congestion is serious in most cities since there are a growing number of private cars. At the same time, thousands of traffic accidents are still happening every day which cause damage to families and economics. Another problem is that private cars have a high idle rate, but occupy a large amount of public space. These problems have lead to the decay of various functions of the society, as well as the continuous deterioration of the urban living environment, so they are considered as cities' chronic and stubborn diseases. Autonomous driving is researched by many institutes and companies especially in recent years (Chehri and Mouftah, 2019; Krueger et al., 2019; Hu et al., 2018b; Chen et al., 2018b; Kolarova et al., 2019; Herrenkind et al., 2019; Xiong et al., 2019), expected to solve the problems of traffic congestion and accidents.

According to recent studies, autonomous driving will reduce urban travel time as well as greenhouse emissions by a large partition, and the need for parking spaces will reduce significantly. Also it can reduce traffic accidents due to the safety design principles. So as a part of sustainable urban form, autonomous driving will be helpful for better environment and citizen lifestyles in the sustainable city. As a result, in future smart cities, city planning will pay more attention to individual and community's demand, instead of automobile traffic. To achieve the concept of sustainability in smart cities, the latest communication (such as 5G), information and sensor technologies can be utilized for infrastructures, roads, as well as vehicles. Thanks to the breakthrough of algorithms, big data and computer ability, artificial intelligence has made great advance in many areas such as machine translation, natural language processing, computer vision. These technologies have been researched and developed for autonomous driving, and we will focus on deep learning methods in this paper.

In autonomous driving, the central tasks include the detection of the lines separating the roadway's different lanes, and the detection of

* Corresponding author.
E-mail address: jun.cheng@siat.ac.cn (J. Cheng).
[1] Xiaoyuan Fan contributed equally for this paper.

https://doi.org/10.1016/j.scs.2019.102002
Received 7 November 2019; Received in revised form 26 November 2019; Accepted 3 December 2019
Available online 23 December 2019

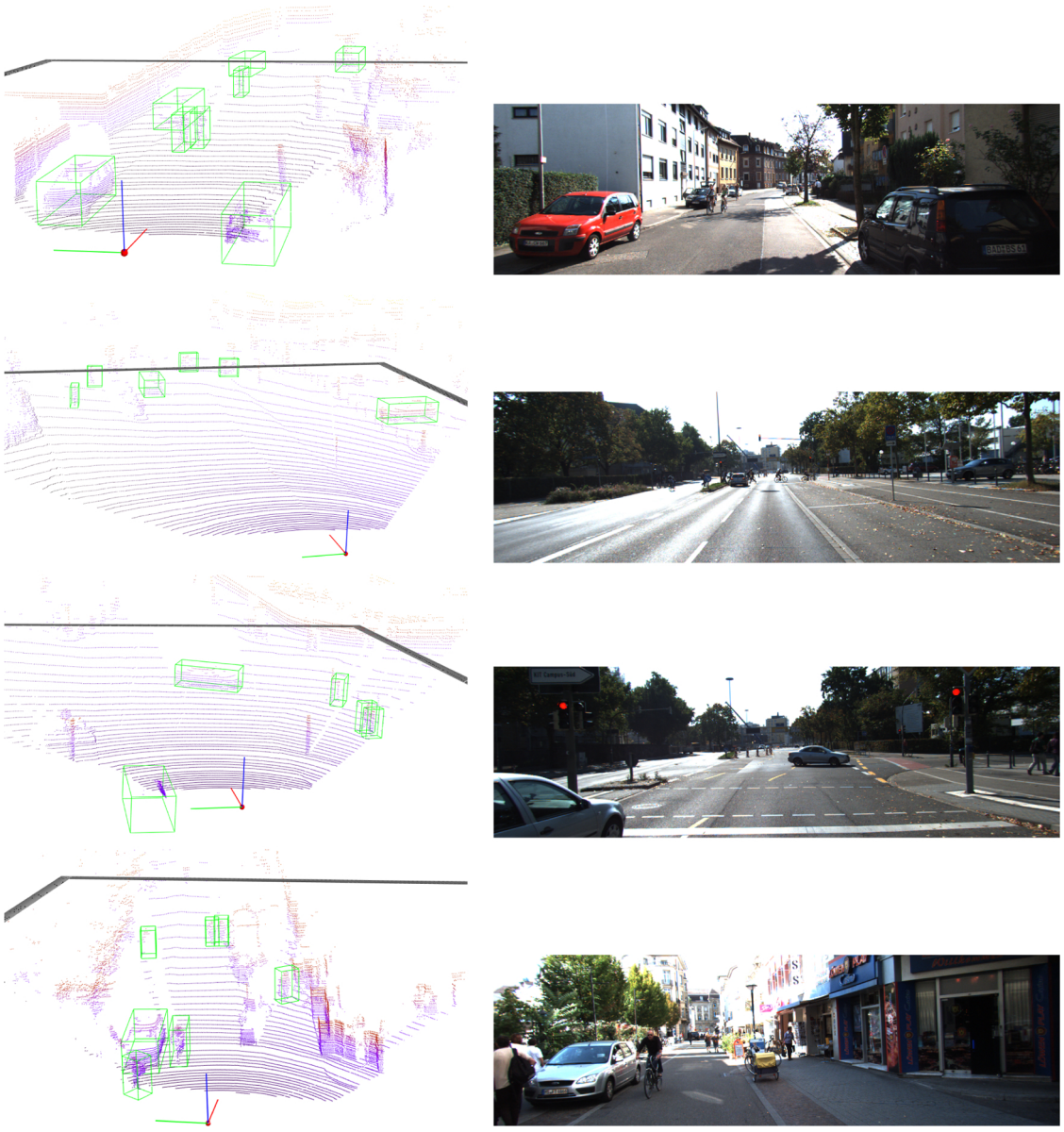

Fig. 1. Visualization of the sparse point cloud data. Left: Point Cloud Data, Right: RGB images. The objects in the green boxes need to be detected.

objects around the autonomous vehicle. In this paper we study on real-time 3D object detection, the target of which is to recognize objects and predict their locations accurately and quickly (Chen et al., 2018a; Rubino et al., 2018; Zhang et al., 2019). LiDAR point cloud plays an important role and performs well in this task, with which we are able to measure the precise distance between an obstacle and the car, even when the obstacle has similar colors with surroundings (Adams, 2000; Brcs et al., 2017; Zhang et al., 2018; Zeng et al., 2018).

However, it is still a challenge to extract features from point cloud with a real-time system. Firstly, a LiDAR sensor equipped on a car scans at 10 frames/s (Geiger et al., 2012), which means there is only 0.1s processing time for each frame (Zeng et al., 2018). Curse of dimensionality always happens in 3D tasks, which places a huge burden on computation, so it is difficult to achieve this detection speed. Secondly, the point cloud data is sparse, as shown in Fig. 1, so a rich feature representation of point cloud is hard to be obtained. Besides, the geometric information of the sparse LiDAR point cloud is not prominent enough, since the adjacent points are scattered, so their relations are not strong enough to extract local geometric features.

Therefore, some previous works (Chen et al., 2017, 2016; Ku et al., 2018; Mousavian et al., 2017; Li et al., 2016) project point cloud into 2D views which have dense points to extract rich features easily. For example, the view-based method, MV3D (Chen et al., 2017) combines RGB images, the front view or top view of LiDAR point cloud as input, then exploits convolutional neural network (CNN) to extract features, and fuses the features to predict the bounding boxes of objects. However, the fusion of features from multiple views will cause information loss since their sizes are inconsistent and need to be normalized.

Some methods (Qi et al., 2018; Zaganidis et al., 2018; Shi et al., 2019) based on PointNet (Qi et al., 2017) directly handle point cloud data. For example, Frustum PointNet (Qi et al., 2018) utilizes PointNet as its basic feature extractor within frustums projected from 2D bounding boxes detected in advance by a 2D object detector, to obtain 3D detections. PointRCNN (Shi et al., 2019) generates proposals of bounding boxes directly from the segmented foreground point set, and then fine-tunes such proposals through transformation into canonical coordinates.

Another kind of point-cloud-based approach is voxel-based methods. Voxel Feature Extractor (VFE) is designed in VoxelNet (Zhou and Tuzel, 2018) to transform voxels into fixed-dimensional feature

vectors. Voxels of larger size are used in VFE, which successfully reduced the size of point cloud. Another method called SECOND (Yan et al., 2018), based on the structure of VoxelNet, uses Sparse CNN (Graham et al., 2018b) to replace normal 3D CNN which requires a lot of memory.

After feature extraction, region proposal is used to hypothesize object locations. Region Proposal Network (RPN) (Ren et al., 2015, 2017) has a bottom-up path to obtain high level features and a top-down path to intensify such features by which the predicted bounding boxes are mainly determined. However, features have only one route to pass through, thus information loss of the middle layers is unrecoverable. Many methods have been proposed to enrich the final features through feature fusion. For instance, SSD (Liu et al., 2016b) and DSSD (Fu et al., 2019) fuse features from different layers through concatenation of the channel dimension. These methods directly connect the coarse-grained features to the final feature maps, which actually add noise to the prediction.

Another representative method is Feature Pyramid Network (FPN) (Lin et al., 2017) with an encoding-decoding framework. The feature of the bottom-up path is transmitted to the top-down path through a lateral connection at the same scale, so that feature maps of the decoding path contain information of different resolutions. In particular, the final feature maps of FPN indirectly merge the features from coarse-grained to fine-grained, so that the model can better detect objects of different sizes.

In this paper, we propose one real-time end-to-end 3D object detection method based on voxelization, sparse convolution, and feature fusion. Considering the neatness of the voxelization features and the flexibility of voxel, we utilize voxelization as the initial step. However, instead of using an extra structure to transform the voxels, we average the vector representation of the points within a small voxel to represent it. In this step, little information is lost since the voxels are small enough, while transforming a large voxel will cause loss of local geometric information.

Secondly, the voxelized point cloud is processed with sparse CNN, transformed into 2D feature maps which will be used to generate proposals.

Thirdly, we propose a fusion of fusion network (FoFNet) for region proposal. As mentioned before, concatenation-based feature fusion methods can combine information of feature maps of different scales. However, 2D feature maps input to the RPN are transformed from 3D point cloud, so the one-step fusion methods can't efficiently fuse the special information that only exists in 3D space. Therefore, we propose FoFNet which provides two-step fusion of features. In the first step we use pyramidal network to fuse features of multiple scales. After amplifying their sizes to be uniform by deconvolution, we concatenate them in the channel dimension as the second step. As a result, our model can fuse more information including multi-scale and 3D space.

Our contributions can be summarized as follows:

- We average small voxels of the point cloud data as their vector representation to improve the efficiency of voxelization, followed by sparse convolution.
- We propose a two-step model for feature fusion to extract rich features of 3D space.
- Our method has achieved competitive performance compared with the most recently published works (Liang et al., 2019; Shi et al., 2019) on the 3D detection test board of KITTI (Geiger et al., 2012), while it can detect each frame within 0.05 seconds, which is much less than scanning time of LiDAR sensors.

2. Related work

We will briefly introduce exiting 3D detection methods based on point cloud from the following aspects.

2.1. Voxel

Recent research on 3D classification focuses on end-to-end trainable neural networks. As discussed earlier, due to the high computational and memory cost, some image-based methods (Chen et al., 2017, 2016; Ku et al., 2018; Mousavian et al., 2017; Li et al., 2016) extract features and infer 3D bounding boxes from 2D images, which usually have low accuracy because of depth localization. Some methods use point cloud data, while they can only consume point cloud data without converting point cloud data into intermediate representation, such as BEV formats. Most 3D-based methods either use point cloud data directly or require converting these data into 3D grids or voxels instead of generating BEV representations. Voxel-based methods have a simple architecture and perform well.

Some methods utilize a voxel grid representation on 3D object detection, followed by using 3D convolutions to compute 3D bounding boxes, such as 3D Voxel Pattern (Xiang et al., 2015). In Wang and Posner (2015), Engelcke et al. (2017), Song and Xiao (2014), Li (2017), the voxel is encoded with several statistics derived from the inside points or fused with local statistics or binary encoding. In Chen et al. (2017), Li et al. (2016), González et al. (2015), multiple perspective views projected from point cloud are used to compute the image-based grid representations. Sedaghat et al. (2017) takes a 3D voxel grid as input and effectively solve the problem of object orientation. Zhou and Tuzel (2018) proposed an end-to-end trainable deep architecture which can avoid information bottlenecks. A multimodal method based on voxel, MVX-Net (Sindagi et al., 2019), presents two effective approaches by leveraging the proposed VoxelNet architecture, which is PointFusion and VoxelFusion, and the results on KITTI benchmark demonstrate good performance.

2.2. Sparse convolutional network

Convolutional neural networks (CNNs) perform well in densely filled cases, while they are often limited by budget and time constraints when the data needed to process is sparse. Graham (2014) introduces spatially sparse convolution firstly. It improved the efficiency of networks by setting ground state. We can only calculate the values that differ from the ground state. Graham (2015) utilizes spatially sparse convolution to construct 3D convolutional neural networks. Based on previous literature on sparse convolution networks, Graham et al. (2018a) presents submanifold sparse convolutional networks (SSCNs), which have a strong performance in semantic segmentation. In traditional convolution frameworks, there are many inevitably limitations. To solve this problem, two slightly different convolution operations are proposed. Unlike the original convolution method, the active sites of output and input are the same, that is to say, only the central input is considered in the active process. This operation keeps parameters' sparsity in the process of deep convolution, optimizes calculation and improves efficiency.

2.3. Feature fusion

Many methods use RPN as their detection framework. And single shot multibox detector (SSD) is widely used to construct RPN architecture, while SSD abandons the shallow feature map in order to avoid using low-level features. Several methods Lee and Nam (2017), Hu et al. (2018a), Kong et al. (2016), Guan et al. (2019) concatenate multi-scale feature maps through connection of all or part of convolutional layers.

ParseNet (Liu et al., 2016a) can directly pool any layer of the network globally to get a feature map that represents the features of the whole map, and use this feature map to segment. FPN (Lin et al., 2017) proved to be more efficient in object detection than concatenation-based methods. FPN is a pyramid network that naturally utilizes the hierarchical features of CNNs and generates feature pyramids with strong semantic information on all scales. Its basic idea is to use both

Table 1
Automated driving companies.

Companies
Waymo, GM Cruise, Zoox, Aurora, Drive.ai, Phantom AI, NVIDIA, SF Motors, Telenav, CarOne/Udelv, Apple, Uber, Tesla, AIMotive, BMW, Mercedes Benz, Toyota, Nissan, Honda, Nuro, Pony.AI, Baidu, AutoX, Roadstar.AI, WeRide/JingChi, Nullmax, PlusAI, SAIC

low-level and high-level features to predict at the same time at different levels. This is because that there may be multiple targets of different sizes in an image, so different features are required to distinguish different targets. For simple targets, we only need shallow features to detect it. For complex targets, we need complex features. However, concatenation-based methods have one advantage over the pyramid structure that they provide a larger variance in the channel dimension, which is more conducive to the generalization of the model.

3. Automated driving

In the past few years, an increasing number of companies are studying related technologies about autonomous driving, an incomplete list of which has been shown in Table 1. And various autonomous vehicles have been developed, as shown in Fig. 2.

According to reports about autonomous driving from the California Department of Motor Vehicles (DMV), vehicle number and mileage of autonomous testing have increased by a large amount in last year. 48 companies have arranged 496 automated vehicle and driven for 3258074 miles. By the end of last year (Dec. 2018), there have been more than 120 cities in which autonomous driving is testing. Taking Waymo for instance, its automated vehicles need to be taken over manually one time every 17846.8 km on average. In Oct. 2018, a permit for Waymo to operate fully driverless cars (i.e. cars without human safety drivers) was issued by the California DMV.

Many studies show that autonomous driving will benefit on multiple aspects for sustainable cities and society (Chehri and Mouftah, 2019). We extract the data from Chehri and Mouftah (2019) in Fig. 3, from which we can see that autonomous driving will benefit the reduction of

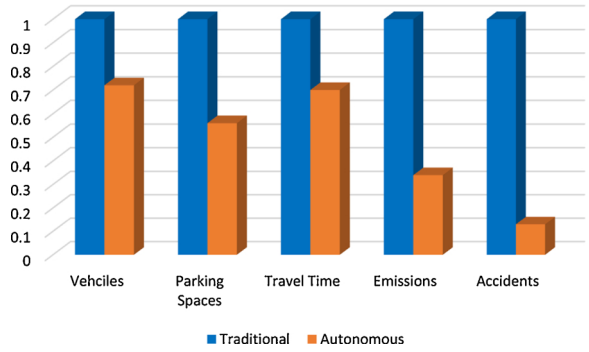

Fig. 3. Comparison between the traditional and the automated driving.

vehicles, parking spaces, travel time, greenhouse emissions, and accidents. Especially, the accidents can be reduced by nearly 90 percent, since driving safety can be increased when the degree of automation is high and inter-vehicle safety distances are more respected compared to manual driving.

Chan (2017) has introduced some advantages and benefits of automated driving system from three perspectives (vehicle user, transportation operation, society), and we add attributes including *safety, efficiency, humanity, economy, resource, environment,* and *sustainability* to the contributions, as shown in Fig. 4.

4. Approach for 3D object detection

In this section, we introduce the proposed approach for 3D object detection in details. As shown in Fig. 5, the main framework of the proposed method can be divided into three modules. The first module is the voxelization of point cloud. It is used to segment point cloud into voxels and transform the point set of each voxel into a feature vector of a uniform dimension. In the second module, the middle layers consisting of sparse convolution further extract features from the voxelized point cloud and transform to 2D feature maps. In the third module, FoFNet generates the proposals uniformly in advance and then passes feature maps through convolution and deconvolution layers to predict

Fig. 2. Automated vehicles.

	Citizen (user)	Road Infrastructure	Society
Safety	• Fewer collisions		• Reduce the accident rate and reduce societal losses.
Efficiency	• Saving individual time	• Less traffic congestion • Efficient real-time navigation • More efficient infrastructure through better vehicle control.	• More efficient in social operation since of less social transportation cost
Humanity	• Less stressful journey • Alternative mode of transportation	• More accessible, reliable, and flexible shared routes for transit and mobility services.	• Improve care and services dedicated to people with reduced mobility.
Economy	• Less demanding or unnecessary ownership for individuals	• More affordable mobility services and less subsidized transit operations by public services. • Improved economic returns and business models for private investors.	• Encourage the transition from personal ownership to carpooling services. • Reduce insurance and related property costs.
Resource	• Unnecessary for parking • Less resource consuming	• Saving resources for infrastructure (parking, roadway constructions)	• More sustainable since of less resource consuming
Environment	• Better environment for living	• Better environment since of better traffic	• Make vehicles and infrastructure more environmentally friendly. • Provide transport services that are increasingly capable of improving safety, reliability, security, and productivity.
Sustainability		• More sustainable for cooperation between vehicles and intelligent roadside system	• Better transport services that are increasingly capable of improving safety, reliability, security, and productivity.

Fig. 4. Contributions of the automated driving.

the bounding boxes and classes of objects.

In our method, the basic unit is voxel. Voxelization is used to reduce the point cloud's dimension, and this is much better than inputing the whole point cloud of large size directly. So it helps to save memory usage, and the RPN of our model can use more complex and deeper structure to obtain richer features to improve the model's performance. Following VoxelNet (Zhou and Tuzel, 2018), a threshold T is also set, and up to T points in a voxel are randomly transformed into a feature vector.

4.1. Voxelization

First of all, we crop the valid area of the point cloud to be a standard cuboid. Assume that the range of point cloud along the Z, Y, X axes is D, H, W, respectively, and the voxel's size is v_D, v_H, v_W accordingly. Then, the voxelized 3D point cloud has a size of $D' = D/v_D$, $H' = H/v_H$ and $W' = W/v_W$.

Directly applying CNN to extract the feature of voxels will occupy a lot of memory. Therefore, VoxelNet designed VFE to transform a voxel into a feature vector, and it significantly reduced the size of point cloud. However, such structure will also cause the permanent loss of geometry information.

Unlike VoxelNet, we do not use extra structure to transform point cloud, but directly average the point set within the voxel into a single vector. There are two advantages. First, it reduces the parameters that need to be trained, so it greatly improves the efficiency of the

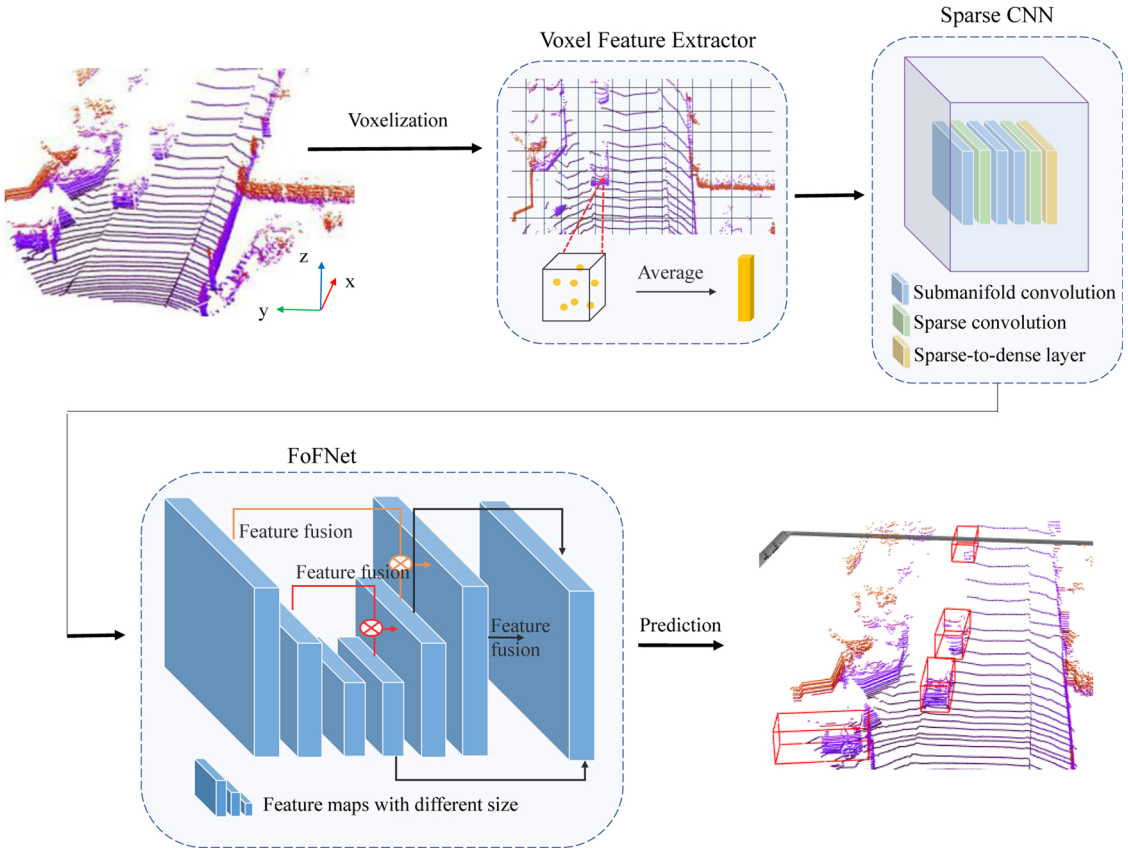

Fig. 5. Framework of the proposed method. The main components of the network are Voxel Feature Extractor, Sparse CNN, and FoFNet.

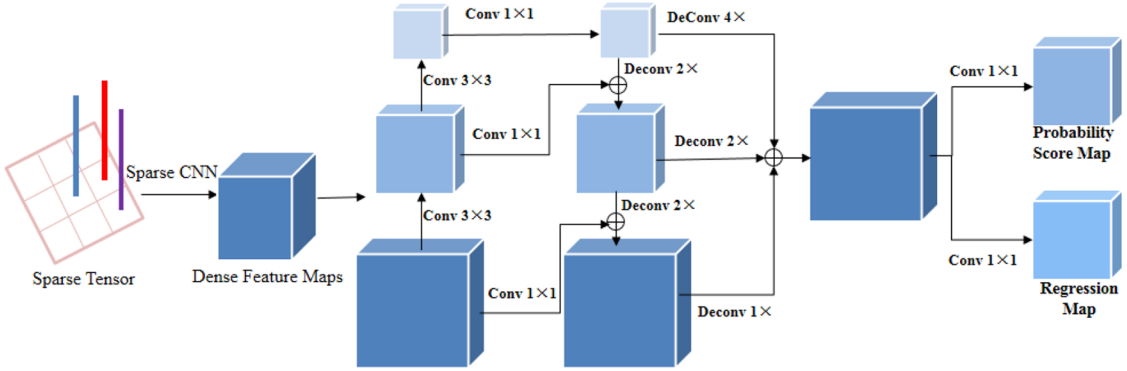

Fig. 6. Fusion of fusion network architecture.

computation. Second, the model is simpler and more stable. Besides, it reduces the loss of geometry information.

We define one non-empty point set within a voxel as

$$\mathbf{V} = \{P_i = [x_i, y_i, z_i, r_i]^T \in R^4\}_{i=1\ldots t}, \tag{1}$$

which randomly takes $t \leq T$ points (all if the number of points in the voxel is less than T), and the features of P_i include reflectivity r_i in addition to the Cartesian coordinates x_i, y_i, z_i. Then we compute the average of the points in the voxel as its feature representation:

$$\mathbf{V}_m = [\bar{x}, \bar{y}, \bar{z}, \bar{r}]. \tag{2}$$

4.2. Middle layers

After voxelization, we use sparse convolution (Graham et al., 2018b) instead of normal convolution to reduce the memory usage in a similar way with SECOND due to the sparsity of point cloud (Yan et al., 2018), which also provides a faster detection speed for our method. This approach preforms well in detection because the voxelization step mentioned above will generate 5k~8k voxels, while most of them are with a sparsity of nearly 0.005 (Yan et al., 2018).

4.2.1. Strategy of sparse convolution

For a D-dimensional network, the input corresponds to D-dimensional sites. A site is active if there is no feature vector in *ground state* (a state that feature vectors consisting of channels on a coordinate are all 0), otherwise, inactive. Inactive sites have the same feature vectors. When training forward pass, these inactive sites only need to be calculated once, which brings the saving in computation and storage. In other words, sparse convolution only activates the corresponding output field when the input field is active, so it greatly reduces the number of activated neurons in the network when the data is very sparse.

4.2.2. Algorithm of sparse convolution

To simplify the proceeding, we start from a 2D case. For an elements whose spatial location is (u, v), we use l and m to denote input and output channels respectively. Then $F_{u,v,l,m}$ represents the filtered elements and $I_{u,v,l}$ represents image elements.The convolution output can be written as follows:

$$Y_{x,y,m} = \sum_{u,v \in P(x,y)} \sum_{l} F_{u-u_0,v-v_0,l,m} I_{u,v,l}, \tag{3}$$

where x and y represent output spatial indexes, u_0 and v_0 represent the bias of the kernel and l represents the input channel. The function $P(x, y)$ generates the input locations by the provided output locations. Then, after the calculation through General Matrix Multiply (GEMM) algorithm, we can construct a matrix and the function $Y_{x,y,m}$ can be given by the following formula:

$$Y_{x,y,m} = \sum_{l} F_{*,l,m} \tilde{I}_{P(x,y),l}, \tag{4}$$

where $F_{*,l,m}$ denoted $F_{u-u_0,v-v_0,l,m}$ in GEMM form. For the sparse data in the images, we use i instead of (u, v) and j instead of (x, y), Eq. (3) can be written as follows:

$$Y_{j,m}{}' = \sum_{i \in P'(j)} \sum_{l} F_{k,l,m} I_{i,l}{}', \tag{5}$$

where k is the kernel offset which corresponds to the bias results of u_0 and v_0. To sum up, a new representation of Eq. (3) is given as follows:

$$Y_{j,m}{}' = \sum_{l} F_{*,l,m} \tilde{I}_{P'(J),l}{}'. \tag{6}$$

Due to sparsity of the data, we need to avoid the redundant computation as far as possible. We use $R_{k,l}$ instead of the correspondence between i and j, which represents the input index i given the kernel offset k and the proposed output index l. So we can convert Eq. (5) into the following formula:

$$Y_{j,m}{}' = \sum_{k} \sum_{l} F_{k,l,m} \tilde{I}_{R_{k,l},k,l}{}'. \tag{7}$$

4.2.3. Sparse convolutional layers

Considering that the height dimension is much smaller than the width and depth of the tensors, and has less information, so we set larger stride along the height dimension. As a result, the 4D tensors including channel dimension are transformed into 3D tensors which do not include height dimension. They actually become 2D feature maps with channels. Taking these denser feature maps as input, region proposal network performs better compared to that taking 3D feature tensors as input.

4.3. Fusion of fusion network

In the FoFNet, we make necessary improvements on the RPN to fuse information of multi-resolution and multi-height in the extracted features, making the model more suitable for point cloud.

As shown in Fig. 6, FoFNet is basically an encoding-decoding framework, where *Conv* represents a 2D convolution, 3×3 means the convolution kernel's size, and *DeConv* 2× represents a deconvolution kernel with stride 2.

4.3.1. Down-top path

The input to our FoFNet is a dense feature map obtained from Sparse CNN. Features maps are usually getting smaller and smaller after computation of convolution. For our feature pyramid, multi-level feature maps are obtained by down-sampling with a scaling step of 2.

4.3.2. Top-down path

As for the fusion, feature maps from the convolutional layers are

transferred to the top-down path, concatenating with feature maps which are up-sampled from the above layers through deconvolution. There are also some feature layers whose input is in the same original size, which is regarded as the same stage. We concatenate features in the same stage. Multiple down-sampling operations produce position information errors, so we combine high-level features with low-level features by lateral connections, by doing which we can obtain accurate location information.

In the second step, feature maps from the top-down path are firstly transformed to be with uniform size, and then merged through concatenation. Next, they are passed through two 1×1 convolutional layers to generate prediction results.

4.4. Regression target

The output of FoFNet includes a probability score map and a regression map. In our model, we use only one anchor size, following VoxelNet (Zhou and Tuzel, 2018), since the output feature map contains multi-resolution information. The following box encoding functions are used for regression targets,

$$x_t = \frac{x_g - x_a}{D_a}, \quad y_t = \frac{y_g - y_a}{D_a}, \quad z_t = \frac{z_g - z_a}{D_a}, \tag{8}$$

$$W_t = \log\left(\frac{W_g}{W_a}\right), \quad L_t = \log\left(\frac{L_g}{L_a}\right), \quad H_t = \log\left(\frac{H_g}{H_a}\right), \tag{9}$$

$$\theta_t = \theta_g - \theta_a, \tag{10}$$

where x, y, and z are coordinates of the bounding box's center. Subscripts t, a, and g represent encoding value, predicted bounding box, and ground truth, respectively. D_a is the diagonal of the predicted bounding box, defined as

$$D_a = \sqrt{W_a^2 + L_a^2}. \tag{11}$$

And W, L, and H represent the width, length, and height, respectively. θ represents the rotation angle around the Z-axis.

4.5. Loss function and training

The loss function for 3D object detection is one multi-task loss, which is defined as (Yan et al., 2018)

$$\text{Loss} = \lambda_1 L_{\text{cls}} + \lambda_2 (L_{\text{reg}-\text{LD}} + L_{\text{reg}-\theta}) + \lambda_3 L_{\text{dir}}, \tag{12}$$

where L_{cls} is the classification loss, $L_{\text{reg}-\text{LD}}$ is the regression loss for location and dimension, $L_{\text{reg}-\theta}$ is the angle loss, and L_{dir} is the direction loss. λ_1, λ_2, λ_3 are constant coefficients.

We train the proposed model with Nvidia Titan XP GPU. During training, the Adam (adaptive moment estimation) optimization algorithm is used, with initial learning rate of 0.0002 and momentum of 0.9. For cars, we only consider assigning anchors to the ground-truth if the intersection-over-union (IoU) threshold of objects is more than 0.6 and if their IoUs are less than 0.45, we regard it as background. We ignore anchors with IoUs between 0.45 and 0.6. For pedestrians and cyclists, we use values of 0.35 for the non-matching threshold and 0.5 for the matching threshold.

5. Experiments

Fig. 7

We evaluate our method on the 3D object detection benchmark of KITTI (Geiger et al., 2012) dataset. First, we evaluate our improvements including voxelization and feature fusion. Second, we compare with state-of-the-art methods and contrast the detection speed. For all the experiments, we choose three categories— *Car*, *Pedestrian*, and *Cyclist* as our target classes. Besides, more challenging dataset have been proposed, such as TOR4D by the Uber Advanced Technologies Group (Yang et al., 2018), and H3D by Honda Research Institute (Patil et al., 2019). In this paper we adopt the most popular KITTI benchmark so as to compare with other methods. In the 3D object detection task, KITTI dataset contains 7481 training samples and 7518 testing samples. We follow the approach in Chen et al. (2017) to divide the training samples into *train* split (3712 samples) and *val* split (3769 samples).

5.1. Ablation study

We investigate the voxelization and feature fusion by fixing one of them while validating the other. For voxelization, we mainly test the voxel's size. For small voxels, we use the simple average strategy in the proposed method. For large voxels, we utilize an extra structure VFE of VoxelNet (Zhou and Tuzel, 2018) to further transform the voxels into feature vectors. Since objects of the *Pedestrian* class are the most difficult to detect among all classes in the KITTI dataset, which is better for us to analyze the efficiency of detecting small objects, so we do the ablation study on this class. Results have been given in Table 2.

5.1.1. Voxel, small or large

No matter which fusion method is used, models with small voxels perform much better than that with large voxels, meaning that using small voxels can preserve more information. Although large voxels can be further transformed by specific networks, the computation is much complex and loss of geometric information can not be avoided.

5.1.2. Feature fusion

Obviously, FoFNet performs much better than the other two feature fusion methods. It demonstrates that our method suits better for 3D data, e.g. point cloud. FoFNet actually combines advantages of feature pyramid network and concatenation-based methods, which enables our model to fuse the information of multi-scale and multi-height.

5.2. Results on KITTI

5.2.1. Evaluation on KITTI Test Split Set

By submitting to the official test server of KITTI[2], we get the results on the *test* split set. As shown in Table 3, compared with methods that utilizing both RGB images and LiDAR point cloud, our method outperforms all the previous methods except UberATG-MMF (Liang et al., 2019) on the *Car* class and F-PointNet (Qi et al., 2018) on the *Pedestrian* class, while our method only takes LiDAR point cloud as input. Compared with methods that using LiDAR, our method is better than the previous except that slightly worse than PointRCNN (Shi et al., 2019) on the *Car* class. On the *Cyclist* class, our method performs better than state-of-the-art methods.

5.2.2. Evaluation on KITTI Val Split Set

As shown in Tables 4 and 5 , our method outperforms all the others in both 3D object detection and bird's eye view (BEV) detection for all difficulty settings. Especially in the hard setting, our method has an advantage of more than 7% compared to the previous best average precision (AP) both in 3D object detection and BEV detection, which demonstrates that our model performs better in detecting occluded objects.

5.3. Algorithm's speed

In autonomous driving, real-time detection is very important, so we compare the detection speed of different methods. As can be seen in Table 3, our method only takes 0.05 seconds for inference in average, faster than the others. And more important, while ensuring speed, the accuracy rate is guaranteed. The fast detection speed is due to the

[2] Website: http://www.cvlibs.net/datasets/kitti.

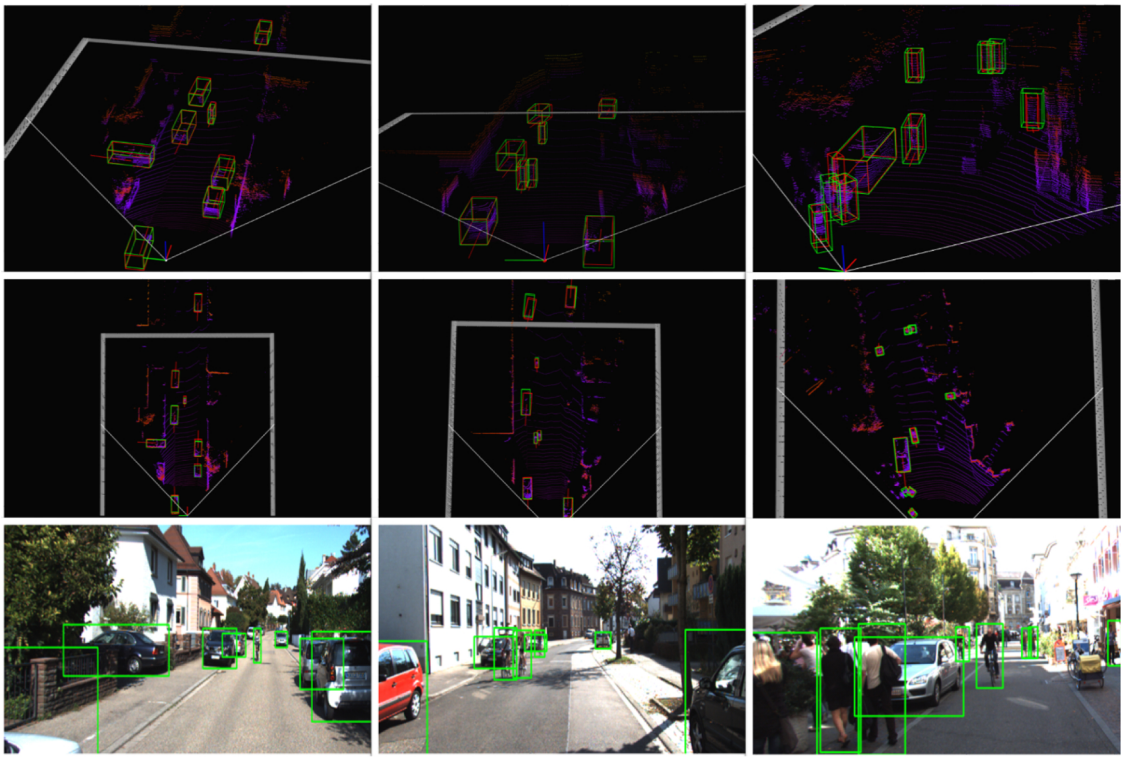

Fig. 7. Result Visualization. This example shows detection results of three different scenes. The first row shows 3-D detection in point cloud and the second row shows the result of the 3-D localization on the xy-plane. The last row shows 2-D ground-truth on images. Ground-truth are shown in green and our results are shown in red.

Table 2
Ablation study for 3D object detection on the Pedestrian class of KITTI *val* split set in average precision(AP) (in %).

Method	Voxel	Feature Fusion	Easy	Moderate	Hard
Ours	large	concatenation	53.96	47.95	44.64
Ours	large	pyramid	55.18	48.77	44.47
Ours	large	fusion of fusion	57.67	53.50	49.08
Ours	small	concatenation	58.88	51.42	44.98
Ours	small	pyramid	60.19	53.30	50.70
Ours	small	fusion of fusion	**61.36**	**57.94**	**51.58**

Table 4
Comparison of 3D object detection on the Car class of KITTI *val* split set in average precision (AP) (in %).

Method	Time (s)	Easy	Moderate	Hard
MV3D (Chen et al., 2017)	0.36	71.29	62.68	56.56
VoxelNet (Zhou and Tuzel, 2018)	0.23	81.98	65.46	62.85
SECOND (Yan et al., 2018)	0.05	87.43	76.48	69.10
F-PointNet (Qi et al., 2018)	0.17	83.76	70.92	63.65
AVOD-FPN (Ku et al., 2018)	0.1	84.41	74.44	68.65
Ours	**0.05**	**88.53**	**78.26**	**76.16**

Table 3
Performance comparison of 3D object detection on KITTI*test* split set in average precision (AP) (in %) (Mod. is the abbreviation of Moderate).

Method	Modality	Time (s)	Car			Pedestrian			Cyclist		
			Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard
MV3D (Chen et al., 2017)	RGB + LiDAR	0.36	71.09	62.35	55.12	-	-	-	-	-	-
AVOD-FPN (Ku et al., 2018)	RGB + LiDAR	0.1	81.94	71.88	66.38	46.35	39.00	36.58	59.97	46.12	42.36
F-PointNet (Qi et al., 2018)	RGB + LiDAR	0.17	81.20	70.39	62.19	**51.21**	**44.89**	**40.23**	71.96	56.77	50.39
UberATG-ContFuse (Liang et al., 2018)	RGB + LiDAR	0.06	82.54	66.22	64.04	-	-	-	-	-	-
UberATG-MMF (Liang et al., 2019)	RGB + LiDAR	0.08	**86.81**	**76.75**	**68.41**	-	-	-	-	-	-
VoxelNet (Zhou and Tuzel, 2018)	LiDAR	0.23	77.47	65.11	57.73	39.48	33.69	31.51	61.22	48.36	44.37
SECOND (Yan et al., 2018)	LiDAR	0.05	83.13	73.66	66.20	51.07	42.56	37.29	70.51	53.85	46.90
PointRCNN (Shi et al., 2019)	LiDAR	0.1	85.94	75.76	68.32	49.43	41.78	38.63	73.93	59.60	**53.59**
Ours	LiDAR	**0.05**	84.15	74.45	66.97	49.44	41.21	36.42	**75.36**	**59.65**	53.03

method's design. First, we improve the VoxelNet by averaging the point set within the voxel into a single vector. Secondly, Sparse CNN is adopted to replace general convolution to extract features, and it greatly reduces the number of activated neurons in the network when the data is very sparse. The detailed running time of our method on one single frame has been given in Table 6. The time is measured on a NVIDIA Titan XP GPU.

6. Conclusion

In this paper, we first reviewed about automated driving, including current situation, companies, and benefit for sustainable cities and society. Then we proposed a deep learning method to detect 3D objects including the most common–car, pedestrian, cyclist. We designed an efficient feature fusion network, which proves to be better than

Table 5
Comparison of bird's eye view (BEV) detection on the Car class of KITTI *val* split set in average precision (AP) (in %).

Method	Time (s)	Easy	Moderate	Hard
MV3D (Chen et al., 2017)	0.36	86.55	78.10	76.67
VoxelNet (Zhou and Tuzel, 2018)	0.23	89.60	84.81	78.57
SECOND (Yan et al., 2018)	0.05	89.96	87.07	79.66
F-PointNet (Qi et al., 2018)	0.17	88.16	84.02	76.44
Ours	**0.05**	**90.18**	**87.82**	**86.96**

Table 6
The benefit of autonomous driving.

	Voxel feature extraction	Middle Forward	RPN Forward	Prediction
time (ms)	0.216	31.662	14.267	6.214

pyramidal methods and concatenation-based methods. In addition, we simplify the process of transforming voxels and improve the efficiency by using small voxels. Experiments on the KITTI benchmark show that our model has a state-of-the-art performance while the detection speed can be 0.05s/frame. In the future, we will verify the proposed method on different datasets including indoor 3D objects. Also, RGB images contain more appearance features, so utilization of both RGB and LiDAR data for object detection will be next work.

Automated driving is one of the most important technology in the future which would change current transportation system's paradigm for citizens, infrastructures, and society. Although the fully automated driving traffic is still a number of years away, more commercial automated driving service will be available in recent years.

Acknowledgments

This work was supported in part by the National Key R&D Program of China (2018YFB1308000), the National Natural Science Foundation of China (61772508, U1713213, 61976143), in part by Shenzhen Technology Project (JCYJ20170413152535587, JSGG201708230 91924128, JCYJ20170307164023599), CAS Key Technology Talent Program, Guangdong Technology Program (2016B010108010, 2016B010125003, 2017B010110007), Shenzhen Engineering Laboratory for 3D Content Generating Technologies (NO. [2017] 476), CAS Key Laboratory of Human-Machine Intelligence-Synergy Systems, Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences (2014DP173025).

References

Adams, M. D. (2000 Dec). Lidar design, use, and calibration concepts for correct environmental detection. *IEEE Transactions on Robotics and Automation, 16*(6), 753–761.
Brcs, A., Nagy, B., & Benedek, C. (2017 July). Instant object detection in lidar point clouds. *IEEE Geoscience and Remote Sensing Letters, 14*(7), 992–996.
Chan, C.-Y. (2017). Advancements, prospects, and impacts of automated driving systems. *International Journal of Transportation Science and Technology, 6*, 208–216.
Chehri, A., & Mouftah, H. (2019 Nov). Autonomous vehicles in the sustainable cities, the beginning of a green adventure. *Sustainable Cities and Society, 51* Article 101751.
Chen, X., Kundu, K., Zhang, Z., Ma, H., Fidler, S., & Urtasun, R. (2016). Monocular 3D object detection for autonomous driving. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* 2147–2156.
Chen, X., Ma, H., Wan, J., Li, B., & Xia, T. (2017 July). Multi-view 3D object detection network for autonomous driving. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* 6526–6534.
Chen, X., Kundu, K., Zhu, Y., Ma, H., Fidler, S., & Urtasun, R. (2018a]). 3D object proposals using stereo imagery for accurate object class detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 40*(5), 1259–1272.
Chen, Y., Zhao, D., Lv, L., & Zhang, Q. (2018b]). Multi-task learning for dangerous object detection in autonomous driving. *Information Sciences, 432*, 559–571.
Engelcke, M., Rao, D., Wang, D. Z., Tong, C. H., & Posner, I. (2017). Vote3deep: Fast object detection in 3D point clouds using efficient convolutional neural networks. *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA),* 1355–1361.
Fu, C.-Y., Liu, W., Ranga, A., Tyagi, A., Berg, A. C. "DSSD: Deconvolutional single shot detector," in arXiv preprint arXiv:1701.06659.
Geiger, A., Lenz, P., & Urtasun, R. (2012). Are we ready for autonomous driving?. the KITTI vision benchmark suite. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* 3354–3361.
González, A., Villalonga, G., Xu, J., Vázquez, D., Amores, J., & López, A. M. (2015). Multiview random forest of local experts combining RGB and lidar data for pedestrian detection. *IEEE Intelligent Vehicles Symposium (IV),* 356–361.
Graham, B., Engelcke, M., & Maaten, L. V. D. (2018a]). 3D Semantic Segmentation with Submanifold Sparse Convolutional Networks. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT,* 9224–9232.
Graham, B., Engelcke, M., & van der Maaten, L. (2018b]). 3D semantic segmentation with submanifold sparse convolutional networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* 9224–9232.
Graham, B., "Spatially-sparse convolutional neural networks," CoRR, vol. abs/1409.6070, 2014. [Online]. Available: http://arxiv.org/abs/1409.6070.
Graham, B. (2015). Sparse 3D convolutional neural networks". *Proceedings of the British Machine Vision Conference (BMVC)* pages 150.1-150.9.
Guan, W., Wang, T., Qi, J., Zhang, L., & Lu, H. (2019 Jan). Edge-aware convolution neural network based salient object detection. *IEEE Signal Processing Letters, 26*(1), 114–118.
Herrenkind, B., Nastjuk, I., Brendel, A., Trang, S., & Kolbe, L. (2019 September). Young people's travel behavior - Using the life-oriented approach to understand the acceptance of autonomous driving. *Transportation Research Part D: Transport and Environment, 74*, 214–233.
Hu, J., Chen, Z., Yang, M., Zhang, R., & Cui, Y. (2018a]). A multiscale fusion convolutional neural network for plant leaf recognition. *IEEE Signal Processing Letters, 25*(6), 853–857.
Hu, X., Chen, L., Tang, B., Cao, D., & He, H. (2018b]). Dynamic path planning for autonomous driving on various roads with avoidance of static and moving obstacles. *Mechanical Systems and Signal Processing, 100*, 482–500.
Kolarova, V., Steck, F., & Bahamonde-Birke, F. J. (2019 November). Assessing the effect of autonomous driving on value of travel time savings: A comparison between current and future preferences. *Transportation Research Part A: Policy and Practice, 129*, 155–169.
Kong, T., Yao, A., Chen, Y., & Sun, F. (2016). Hypernet: Towards accurate region proposal generation and joint object detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* 845–853.
Krueger, R., Rashidi, T. H., & Dixit, V. (2019 November). Autonomous driving and residential location preferences: Evidence from a stated choice survey. *Transportation Research Part C: Emerging Technologies, 108*, 255–268.
Ku, J., Mozifian, M., Lee, J., Harakeh, A., & Waslander, S. L. (2018). Joint 3D proposal generation and object detection from view aggregation. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS),* 1–8.
Lee, J., & Nam, J. (2017 Aug). Multi-level and multi-scale feature aggregation using pretrained convolutional neural networks for music auto-tagging. *IEEE Signal Processing Letters, 24*(8), 1208–1212.
Li, B., Zhang, T., & Xia, T. (2016). Vehicle detection from 3D lidar using fully convolutional network. *Proceedings of Robotics: Science and Systems (RSS)*.
Li, B. (2017). 3D fully convolutional network for vehicle detection in point cloud. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS),* 1513–1518.
Liang, M., Yang, B., Wang, S., & Urtasun, R. (2018). Deep continuous fusion for multi-sensor 3D object detection. *Proceedings of the European Conference on Computer Vision (ECCV)* (pp. 641–656).
Liang, M., Yang, B., Chen, Y., Hu, R., & Urtasun, R. (2019). Multi-task multi-sensor fusion for 3d object detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* 7345–7353.
Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., & Belongie, S. (2017). Feature pyramid networks for object detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* 2117–2125.
Liu, W., Rabinovich, A., & Berg, A. C. (2016a]). Parsenet: Looking wider to see better. *International Conference on Learning Representation (ICLR)*.
Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. (2016b]). SSD: Single shot multibox detector. *Proceedings of the European Conference on Computer Vision (ECCV), Cham,* 21–37.
Mousavian, A., Anguelov, D., Flynn, J., & Kosecka, J. (2017). 3D bounding box estimation using deep learning and geometry. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* 7074–7082.
Patil, A., Malla, S., Gang, H., & Chen, Y. (2019). The H3D Dataset for Full-Surround 3D Multi-Object Detection and Tracking in Crowded Urban Scenes. *2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada,* 9552–9557.
Qi, C. R., Su, H., Mo, K., & Guibas, L. J. (2017). Pointnet: Deep learning on point sets for 3D classification and segmentation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* 652–660.
Qi, C. R., Liu, W., Wu, C., Su, H., & Guibas, L. J. (2018). Frustum pointnets for 3D object detection from rgb-d data. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* 918–927.
Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards real-time object detection with region proposal networks. *Advances in Neural Information Processing Systems (NIPS),* 91–99.
Ren, S., He, K., Girshick, R., & Sun, J. (2017 June). Faster R-CNN: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 39*(6), 1137–1149.
Rubino, C., Crocco, M., & Del Bue, A. (2018 June). 3D object localisation from multi-view image detections. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 40*(6), 1281–1294.

Sedaghat, N., Zolfaghari, M., Amiri, E., & Brox, T. (2017). Orientation-boosted voxel nets for 3D object recognition. *British Machine Vision Conference (BMVC)*.

Shi, S., Wang, X., & Li, H. (2019). PointRCNN: 3D object proposal generation and detection from point cloud. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–779.

Sindagi, V. A., Zhou, Y., & Tuzel, O. (2019). MVX-Net: Multimodal VoxelNet for 3D Object Detection. *2019 International Conference on Robotics and Automation (ICRA)*, 7276–7282.

Song, S., & Xiao, J. (2014). Sliding shapes for 3D object detection in depth images. *Proceedings of the European Conference on Computer Vision (ECCV)* (pp. 634–651).

Wang, D. Z., & Posner, I. (2015). Voting for voting in online point cloud object detection. *Proceedings of the Robotics: Science and Systems (RSS), 1*(3) pp. 10-15 607.

Xiang, Y., Choi, W., Lin, Y., & Savarese, S. (2015 June). Data-driven 3d voxel patterns for object category recognition. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Xiong, W., Lu, Z., Li, B., Wu, Z., & Xuan, X. (2019 May). A self-adaptive approach to service deployment under mobile edge computing for autonomous driving. *Engineering Applications of Artificial Intelligence, 81*, 397–407.

Yan, Y., Mao, Y., & Li, B. (2018). SECOND: Sparsely embedded convolutional detection. *Sensors, 18*(10), 3337.

Yang, B., Luo, W., & Urtasun, R. (2018). PIXOR: Real-time 3D Object Detection from Point Clouds. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT*, 7652–7660.

Zaganidis, A., Sun, L., Duckett, T., & Cielniak, G. (2018 Oct). Integrating deep semantic segmentation into 3-d point cloud registration. *IEEE Robotics and Automation Letters, 3*(4), 2942–2949.

Zeng, Y., Hu, Y., Liu, S., Ye, J., Han, Y., Li, X., & Sun, N. (2018 Oct). RT3D: Real-time 3-D vehicle detection in lidar point cloud for autonomous driving. *IEEE Robotics and Automation Letters, 3*(4), 3434–3440.

Zhang, Y., Wang, J., Wang, X., & Dolan, J. M. (2018 Dec). Road-segmentation-based curb detection method for self-driving via a 3D-LiDAR sensor. *IEEE Transactions on Intelligent Transportation Systems, 19*(12), 3981–3991.

Zhang, Y., Xiang, Z., Qiao, C., & Chen, S. (2019). Accurate and Real-Time Object Detection Based on Bird's Eye View on 3D Point Clouds. *2019 International Conference on 3D Vision (3DV)* (pp. 214–221).

Zhou, Y., & Tuzel, O. (2018). Voxelnet: End-to-end learning for point cloud based 3d object detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 4490–4499.