Towards high-efficient object completion in cross modality 3D detection

by

Tianran Liu

Thesis submitted to the University of Ottawa
in partial fulfillment of the requirements for the degree of
Master of Applied Science
in
Electrical and Computer Engineering in Appied AI

Declaration of Authorship

I hereby certify that this thesis is entirely my own original work except where otherwise indicated. I am aware of the University of Ottawa regulations concerning plagiarism, including those regarding consequent disciplinary actions. Any use of the works of any other author, in any form, is properly acknowledged at their point of use.

Abstract

Multimodal detection is undoubtedly one essential task in the field of autonomous driving today. Considering the complexity of the scenarios in which self-driving vehicles operate, having LiDAR and images as joint inputs would provide a more accurate description of the environment. However, due to the inter-modal domain differences, models to process these data are in fact difficult to design. In this work, we will analyze the key factor that has enabled multimodal models to make continuous performance breakthroughs in recent years: object completion (densification).

In pure lidar-based 3D Perception, recent works have indicated the importance of object completion. Several methods have been proposed in which modules were used to densify the point clouds produced by laser scanners, leading to better recall and more accurate results. Pursuing in that direction, we present, in this work, a counter-intuitive perspective: the widely-used full-shape completion approach actually leads to a higher error-upper bound especially for far away objects and small objects like pedestrians. Based on this observation, we introduce a visible part completion method that requires only 11.3% of the prediction points that previous methods generate but has a higher error-upper bound. We in this way extend the object completion from a pure lidar-based to a cross-modality manner.

To recover the dense representation, we propose a mesh-deformation-based method to augment the point set associated with visible foreground objects. Considering that our approach focuses only on the visible part of the foreground objects to achieve accurate 3D detection, we named our method What You See Is What You Detect (WYSIWYD). Our proposed method is thus a detector-independent model that consists of 2 parts: an Intra-Frustum Segmentation Transformer (IFST) and a Mesh Depth Completion Network(MDCNet) that predicts the foreground depth from mesh deformation. This way, our model does not require the time-consuming full-depth completion task used by most pseudo-lidar-based methods. Our experimental evaluation shows that our approach can provide up to 12.2% performance improvements over most of the public baseline models on the KITTI and NuScenes dataset bringing the state-of-the-art to a new level.

Acknowledgements

Looking back on my two and a half years in Ottawa, I believe that my research training here has meant a lot to me, although I still have regrets on many things. Starting from code reading, we really tried to analyze the problems of existing frameworks and rethink about solutions, and at the end of the process, we finished this thesis. The person I should thank most for this process is my advisor Prof. Robert, without his help in funding and research, it is hard to imagine how difficult these years would have been. His optimism and tolerance are really impressive for me, and he never gave up supporting me even when the results of my research were not as good as I had hoped.

I also would like to appreciate all of my colleagues and friends in Ottawa, espeacilly for Zeping, Libo, Moterza and Jiekun, the work with you guys has taught me a lot of things and set my research style. Best wishes to all of you for discovering more interesting things in your future scientific explorations.

The end of my time in Ottawa today is only the beginning of a long research career ahead, and just as I thought when I first stepped into this lab in 2021, I'd like to end on the same note: do really interesting and valuable research, and always be skeptical of previous work.

"Hinc itur ad astra" - 由此路始，直抵星辰

List of Tables

List of Figures

Table of Contents

Chapter 1

Introduction

1.1 Problem definition

As one of the fundamental tasks of autonomous driving perception, multimodal 3D detection aims at obtaining a description of the position of an object in the real world using input signals from sensors. Considering the complexity of weather conditions, it is common to choose multiple sensors in order to obtain robust performance in different weather conditions. In this thesis we have chosen two complementary signals, lidar and images, as inputs, and cross-modality in the following also refers to these two modalities.

In this case of laser scanner(lidar), depth information is obtained by projecting a laser light onto the scene, which means that farther objects and highly occluded will be hit by a limited number of ray light. For closer objects, the richness of the captured data makes detection relatively easy and accurate. However, because of the sparsity of the signal for far or occulded objects, the detectin task from just few data point become highly unreliable. This is undoubtedly the really difficult part, i.e., how to utilize the dense semantic information of the image to compensate for the missing signals in distant or occluded objects. A growing body of work [27, 47, 72, 81] tries to address this difficulty with explicit depth estimation: as shown in Figure 1.1 below: the depth estimation module project the features to 3D space directly.

In this thesis, we argue that the completion (shape recovery) of an object is crucial for the final accuracy of 3D detection and more accurate and efficient depth estimation further improves the performance.

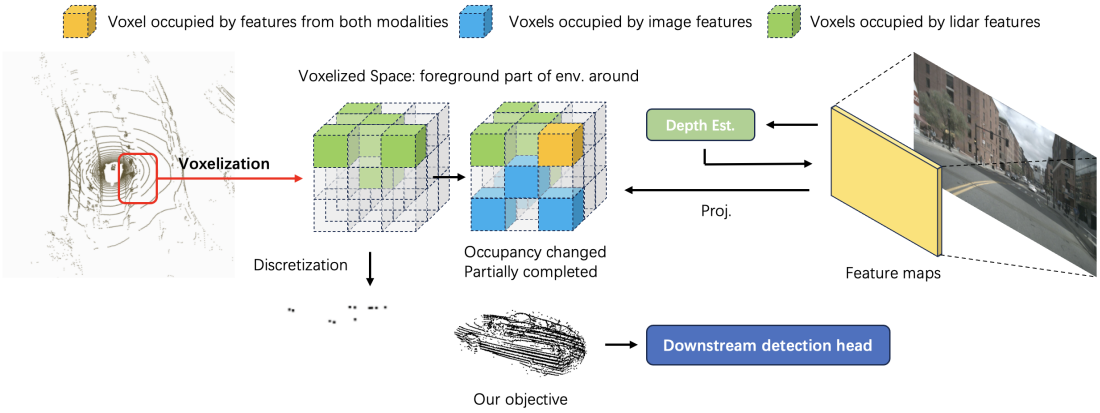


Figure 1.1: Statement of objectives: Our objective in this thesis is propose a new method which not only improve the quality of completion results but also eliminates many unnecessary computational burdens.

To summarize, this thesis aims to find a more efficient and better-performing object-completion scheme as the middleware a detection framework, and in this way to improve the performance of 3D detection.

1.2 Terms definition

It should be stated at the outset that, in view of the fluency of the text in various places, we have used different terms referring to the same meaning in a number of places below, and the explanations are given here first.

- Object completion: This step is essentially an augmentation of the original 3D point cloud by a denser representing of the object, specifically restoring the complete shape of the foreground object. We will also equivalently use terms such as shape completion or point cloud completion.
- Depth completion: Its a subtask of depth prediction and refers to the fact that when we project the lidar to the camera plane, not every pixel can get the corresponding depth, then the limited lidar depth information needs to be spread to the whole plane. The final output ensures that every pixel has a corresponding depth. After projecting the color into 3D space with this depth information, we can also obtain a point cloud, which will be called a pseudo-point cloud.

1.3 Background and motivation

For high-performance autonomous driving perception, lidar is probably the most critical sensor. Although we have recently witnessed an increasing amount of work based on lidar-image multimodality input, the lidar features extractor is still used as the mainstream branch in most network design, since it provides accurate depth information.

The main limitation of lidar is the sparsity of its representations: with the depth increasing, the density of the geometric features obtained from the point cloud decreases rapidly. For the lidar-based methods, recent works [40,56,69,76,77,79,82,86] have realized that better performance can be obtained by performing shape completion of objects with the network. However, it is important to point out that, most of these works are established under the premise that the first stage 3D Region of Interest(ROI) proposal is accurate enough, such that objects inside the proposed boxes can be refined and completed reliably. Considering that the Region Proposal Network(RPN) module at this stage still needs to face the sparsity of the lidar, especially for distant objects, this precondition can hardly be achieved. At the same time, even if the first stage detection fulfills the requirements, the completion task itself is inherently difficult for sparse inputs when relying only on lidar signal.

In early multimodal detection networks [41,61,66,75], RGB features are used to decorate points or voxel features. With the research progressing, lifting 2D features to the pseudo lidar virtual points in 3D space has become increasingly popular. The success of recent cross-modality completion-based methods [27,71,72,90] confirms the appropriateness of this approach. Specifically, SFDNet [72] has pioneered an attempt to introduce external points to complement the objects. This augmented data from depth completion, is equivalent to interpolating the lidar signals of the objects surface. By fusing this information with real lidar signals, complete objects can thus be generated from sparse point clouds. This enriched 3D point representation has led to significant improvements in performance.

However, it is important to note that these pseudo-lidar representations are still plagued with inherent artifacts introduced by boundary depth dispersion problems. Considering that the ground truth used to train the depth completion network is semi-dense and the inherent smoothing properties of convolutional layers used in these models, the depth estimation of boundary pixels always tends to leak toward the background. As a matter of fact, this issue seriously affects the accuracy of subsequent detections, as it will later be demonstrated in section 2.2.2.

Although a recent work [71] has made it possible to reduce the inaccuracy of the depth estimation through a learnable discard module, models based on depth completion still

suffer from two other issues. First, full frame depth completion itself is time-consuming and most of the generated background depth information is useless for the later process. A typical depth completion method like PENet [22] needs 161ms [24] per frame when CUDA synchronization is used. In consequence, a complete system based on such a pre-processing network will be far from real-time detection. Second, even if the outlier depth points can be dropped accurately, this discard-based strategy will actually further dilute the already limited amount of semantic information available for small objects.

In this work, we revisit the fundamental issues related to the quality of foreground depth and our objective is to make the entire detector to get rid of time-consuming full frame depth completion networks. Overall, we only complete the foreground points instead of performing global depth completion. This process can be divided into two steps: **foreground points segmentation** and **object densification**.

Specifically, we demonstrate that visible part completion leads, in fact, to equivalent or even superior results than full shape completion used by most of the previous methods. This good performance also explains why methods like SFDNet [72] or VirConv [71] perform better than more traditional completion-based methods. To estimate the foreground region which needs to be densified, rather than relying on 3D RPNs that are insensitive to sparse objects, we choose to use the well-developed 2D instance segmentation networks. Within the 3D frustum of a specific segmented 2D object, considering the depth distribution of the point cloud, we design a lightweight Transformer, named IFST, to filter out the noise points.

Next, different from the existing models that use pseudo points from depth completion, here we chose to reconstruct the object directly from the lidar signal. The points from the IFST still contain some noise, so an ideal model should be able to control the shape and be invariant to noise as well. To fulfill such requirement, we integrated a mesh deformation approach to completion-based 3D detection for the first time. Through a lidar aggregation layer, we make the mesh learn the distribution of the lidar signal in a coarse to fine manner and restore the shape of the object progressively. Notably, the consistency of the resulting mesh is guaranteed and the vertex will not leak into the background benefiting from the specific guidance of the Laplacian loss.

It is also worth noting that although there are several works that claim to produce accurate point cloud completion [36, 82], they have all been tested in a noiseless or indoor environment. Few studies have demonstrated that they can be adapted to the case of sparse inputs.

1.4 Contribution

In summary, our contributions are as follows:

- We experimentally demonstrated the counter-intuitive conclusion that the visible portion of the completion is in fact more suitable for improved detection result. We believe that this can partly explain why several recent works were able to further improve the upper bounds on the SOTA accuracy.
- We introduce a novel lightweight Intra-Frustum Segmentation Transformer that utilizes the 2D location prior and 3D locations to extract foreground points. As the main component, a mesh-deformation-based completion module is proposed to learn the visible shape of an object from the lidar signal.
- By combining our modules with the publicly available 3D detector baseline, extensive experiments have demonstrated that it is possible to provide up to 12.2% performance improvement and obtain state-of-the-art performance, especially in the detection of smaller farther away objects.

1.5 Thesis outlier

In this thesis, we will first introduce a hidden clue in recent cross-modality 3D detection method: shape completion. With this concept, we will then elaborate the drawback of the exist completion method and finally propose our solution which introduced the mesh deformation into the real-time depth estimation for the first time.

Specifically, in Literature Review(Chapter 2), we will start from the introduction of encoders for different modalities input: ResNet and Vision Transformer will be introduced for the image signal and lidar encoders will be categorized to point-based and voxel-based. With these basic modules, the basic fusion-based 3D detectors will be introduced in section 2.1.3. Next, in 2.2.1, the concept of objects completion will be introduced. After that, we will point out that increasely more recent work focused on local-level completion, i.e., the foreground level completion, in 2.2.3. At the last of this chapter, we will indicate that the drawback of the exist local-level comepletion method.

In Chapter 3, we will analyze the dense-depth-completion model to give us a clear goal of the model we need to propose. We proved that the surface depth completion actually have a higher upper-boundary than the traditional methods first and the only thing we

need for downstream detection tasks is an accurate foreground depth estimation. Based on this, then we introduce our methods which consisted by IFST(Intra-Frustum Segmentation Transformer) and MDCNet (Mesh Depth Completion Net).

In Chapter 4, we tested our model on 2 famous benchmark: KITTI and nuScenes. Widely experiments have proved that by combination with our model, most of the baseline lidar detector can stand at the same level of the SOTA cross-modality method in 2023. A large number of ablation experiments have also demonstrated that our method can improve the detection performance of small and distant targets, such as pedestrians, dramatically.

In Chapter 5, we will summarize the content and contributions mentioned in the previous chapters and provide further discussion on the shortcomings of the proposed methods.

Chapter 2

Literature Review

In recent years we have witnessed the publication of a large number of multimodal perception models that continue to improve detection accuracy. It is worth noting that a clue that is often used in recent works is that model effectiveness can often be explained by completion of foreground regions.

From section 2.1, we will first introduce some basic modules and models which have been widely used in mutimodal 3D detection. Then from section 2.2.1, we will introduce model that are based on completion and use it to boost the performance. At the end of this chapter, we will conclude with the drawbacks of the existing models to transition to our solution.

2.1 Basic modules

Given P cameras, for each camera we can obtain a image with size $H \times W \times 3$, where H, W stand for height and width, so the image input can be represented as $\mathcal{F}_{io} \in \mathbb{R}^{P \times H \times W \times 3}$. For the lidar input, use N and M to denote the number of points per frame and the dimensions of single point, the signal can be represented by $\mathcal{F}_{lidar} \in \mathbb{R}^{N \times M}$. The frame of lidar and images mentioned here are already calibrated, so in the following we will ignore all effects due to the unsynchronized clock frequencies of these two sensors.

Although there has been some works [57, 67] looking to build a modality-independent encoder to process both types of information in a unified way, we present them separately here, given that the mainstream approach still maintains two types of encoders and that this design tends to have the best performance. In section 2.1.1 we will cover from the

ResNet to Vision-Transformer to illustrate the development of 2D encoders. Encoder of 2D detection model will be introduced in section 2.1.1. Next, to process the lidar signal, Point-based and Voxel-based backbone will be presented in section 2.1.2. After these, as a basic idea of fusion detection, some earlier published 3D detection models with multimodal inputs are presented in section 2.1.3.

2.1.1 Image encoders: From ResNet to Vision-Transformer

The first time neural networks showed amazing capabilities was the success of CNNs on the massive dataset ImageNet. As one of the typical CNN-based structures, ResNet [20] solved the problem of network degradation due to its very deep architecture. More recently, we have witnessed the massive rise of visual Transformers [46, 65]. In this section, we will briefly introduce the internal mechanisms and architectural design of these 2 classic image encoders.

Empirically, the depth of the network is critical to the performance of the model; When the number of network layers is increased, the network can perform the extraction of more complex features, so better results can theoretically be achieved when the model is deeper. But the question here is, can we just increase the number of layers? The answer is no, as shown in Figure 2.1 (from ResNet [20] paper), a 56-layer network is less effective than a 20-layer network. This would not be an overfitting problem because the training error of the 56-layer network is equally high. It is also unlikely that this situation would be caused by a vanishing gradient or gradient explosion, since the regularized data distribution would still cause this problem even after the addition of the BN [26] layer.

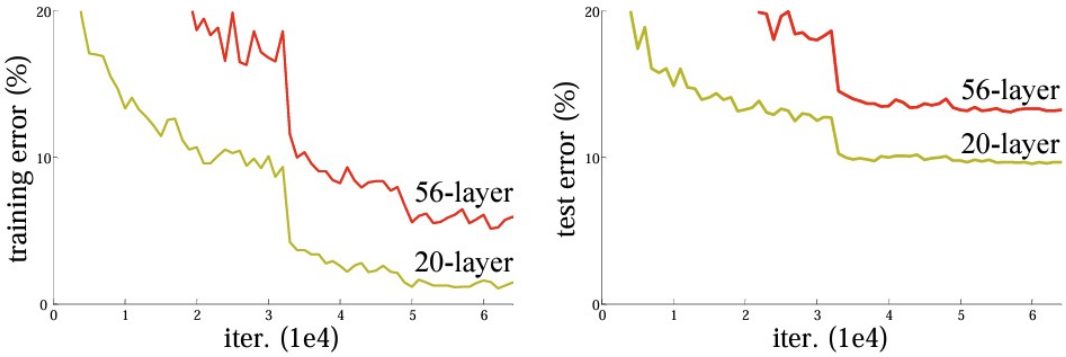


Figure 2.1: The perfomance comparsion of stacked CNNs with different number of layers on CIFAR-10 [32] dataset

So what's the problem? Actually, what we're dealing with here is the degradation of the network. More precisely, it is difficult for the network to make further progress on its already good performance. Another question at this point is why can't the network become a identity mapping through learning, so that even if it doesn't learn anything useful, its guaranteed that the accuracy won't degrade. In fact this problem can be explained by a nonlinear activation function. Since we want the mapping learned by the network to be nonlinear, most networks use nonlinear activation functions such as relu [1] or selu [30]. One of the major drawbacks of this use is that the previous inputs are almost irreversible for the later layers of the network, which means that it is very difficult to retain the complete information of the previous layers.

The solution proposed by ResNet is in fact quite straightforward: if it is difficult to learn more precise representations directly, then we can have the network learn only the residuals between representations. A typical residual unit can be expressed in the form of the following equation, here θ represent the trainable parameters and equivalent to W, x represent the features obtained from the original input $\mathcal{F}_{io}$.

$$\begin{aligned} y &= x + f(x, \theta) \\ f(x, \theta) &= BN(\max(Wx, 0)) \end{aligned} \tag{2.1}$$

When we consider the gradient from the loss and assume we use y to calculate directly, then we have:

$$\frac{\partial l}{\partial x} = \frac{\partial y}{\partial x}\frac{\partial l}{\partial y} = \left(I + \frac{\partial f(x, \theta)}{\partial x}\right)\frac{\partial l}{\partial y} \tag{2.2}$$

Here the I denote a unit matrix, which also represent this residual design can propagate gradient losslessly. Benefiting from this design, ResNet can more easily learn deeper features (in the deeper layers of the network).

ResNet has brought about a profound change in the field of deep learning. However, considering that it is still based on the CNN architecture, its position has nowadays tended to be replaced by the Visual Transformer in today's world of increasing performance demands.

Specifically, CNNs are designed based on the human a priori that neighboring regions should have higher correlation in semantic features. The appearance of the visual Transformer, on the other hand, breaks this conventional understanding, and in fact we now find that such a priori is instead what limits the model from further performance improvement. The story begins with the Google team's 2016 publication "Attention is all you need [65]", in which they proposed a new network architecture named Transformer to process the

feature in natual language processing.

Transformer was originally proposed for natural language tasks like language translation. So the overall structure as shown in Figure 2.2, is organized as a encoder-decoder structure: a sentence used as input and the model will extract the infomation in it, and then output a series of possiblity which might means different possibilities of words in the other language, depends on the specific tasks.

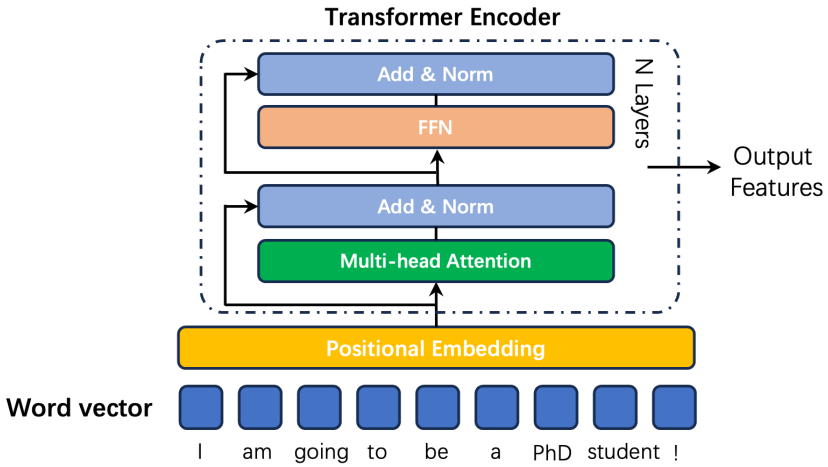


Figure 2.2: The overall structure of original Transformer in language translation task. Add and Norm stands for elements-wise add and LayerNorm [2]. The feed Forward module consist of several layers of plain MLP.

For a specific sentence, we first get the embedding of every words for example using the Word2Vec [50]. Then we need give words at different positions a special marking, as the same word at different position of the sentence may varies in meaning. Here positional encoding will be concat with the original embedding.

As shown in the orange block in Figure 2.2, one of the main contributions of Transformer is this multi-head self-attention module. We here first describe the calculation of a single head. For the feature of input sentence, 3 trainable parameter matrix W_k, W_q, W_v are used to get 3 different linear transformations K, Q, V of the input. Then the calculation of self-attention process can be presented by Equation (2.3), the d_k represent the dimension of feature K. Essentially, self-attention breaks down the features of each word into a weighted sum over the information of the whole sentence. And if we use different parameters to create seveal sets of single head attention and concatenate the final obtained features, this process is presented by the multi-head attention in the Figure 2.2.

In fact, When the Transformer architecture was introduced, people did not realize its potential in the fids of computer vision. However, the appearance of ViT [12] broke the mold.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.3}$$

If we want to apply Transformers to visual tasks such as classification, segmentation, etc., then we first need to solve the following problems

- Unlike linguistic tasks, the information in pictures has two dimensions instead of one in text, and the difference in input makes it difficult to apply the Transformer model directly.
- The output of the Transformer is sequential information that cannot be directly transformed into what the input sequence as a whole represents.

As the first work which use Transformer to handle the vision tasks successfully, the solution provided by ViT [12] was quite straightforward. First, if the shape required by Transformer is 2D, could we just squeeze the image from a 2D plane to 1D serial? The answer is yes, as the name of their paper suggests, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale", they divide the input image into 256 parts are reshape each part to a 1D tensor. After adding a positional embedding, the image can be used as the input of Transformer directly. This process is visualized in Figure 2.3.

The next question is how to get the result from the sequential output? The basic idea here is to only use the encoder part of Transformer structure. Then we add an extra input at the position 0 for every image, and the output at this position will be used as the final result for the classification.

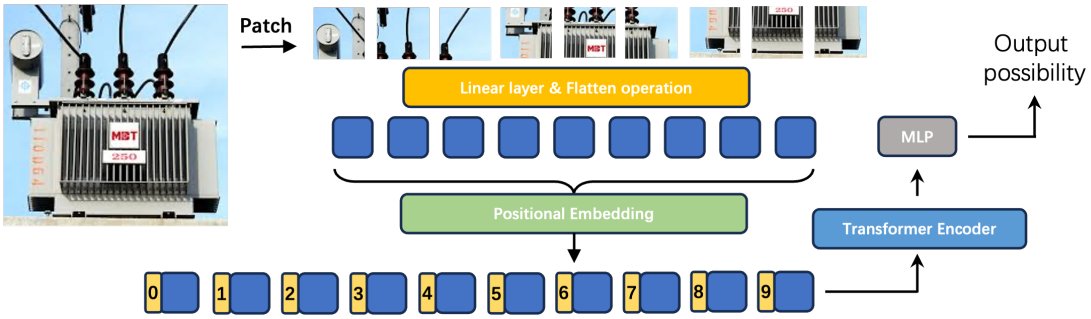


Figure 2.3: The ViT [12] overview, input are splited into 256 sub-images and the output at position 0 will be used as features to predict the required result.

As one of the pioneer work in visual transformers, ViT [12] has solved the problem of model unavailability, but it is worth noting that its performance still does not surpass that of the most advanced CNN detectors of the time, such as EfficientNet [62] or ResNeXt [73]. In addition to this, the ViT [12] network is also much slower compared to traditional CNNs, considering that the attention operation is not as optimization-friendly as convolution.

Because of this, a series of optimizations were proposed: Swin-transfomer [46] improves the problem that ViT [12] can only reason on a single-size feature map (one 256th of the original map) by using multi-scale features built with a sliding window. This has made the Transformer-based Vision model to reach the SOTA performance. EffcientViT [6, 45] propose a cascaded group attention operation and linear-attention to replace the original self-attention operation, and thus accelerating the model by tens of times.

2.1.2 Lidar encoders

After presenting the basic works and development of vision models represented by ResNet and ViT, in this section we will introduce point cloud encoders. Unlike cameras, lidar can directly obtain a 3D description of the world. The goal here is get a robust representation from the lidar signal, the point cloud, which will be used to detect later.

Broadly speaking, in order to be able to deal with point clouds, 3D encoders can be categorized as either point-based, which handles point data directly, or voxel-based, which takes a higher level of abstraction of the point cloud, describing the point cloud distribution in terms of small volumetric elements: the voxels. The following sub-sections introduce each of these two processing methods.

2.1.2.1 Point-based solution

PointNet [54] and its successor, PointNet++ [55], are typical point-based approaches, and as early work in point cloud processing, they have utilized various properties of the point cloud to design networks with good performance. In particular, point cloud data structures have the following properties:

- Disorderedness: the order in which the point cloud is arranged does not affect the description of the objects
- Points are not isolated individuals: their local structure needs to be considered
- Affine transformations of point clouds do not affect the properties of the objects they describe.

To deal with a disorganized input, a simple way is to find a symmetric function: for the same point cloud, the output representation is the same regardless of the order of input. Assuming that the input $\mathcal{F}_{lidar} = p_{i0}, p_{i1}, ..., p_{iN}$, the method to construct this function in PointNet can be presented as Equation 2.4, where MaxPooling operation are used as this function to process features from Multi-Layer Perceptron(MLP).

$$MaxPooling\left(MLP\left(p_{i0}\right), \ldots, MLP\left(p_{iN}\right)\right) \tag{2.4}$$

Next, to get the local description of a given point cloud, here PointNet adopted a local+global solution. Specifically, PointNet adds the process of concatenating the local and global features of each point to obtain point-wise features that are simultaneously aware of both local and global information to improve the characterization effect. Finally, to make the obtained point cloud features invariant to affine transformations, PointNet proposes a simple T-Net. The idea of T-Net is fairly straightforward: align all input point sets to a uniform point set space. Operationally, we directly predict a transformation matrix (3*3 or 64*64 depend on the feature channels) to restore the coordinates of the input points. Since there will be data augmentation operations, doing so provides some assurance that the network will learn transform-independence, i.e., obtain the same features no matter how they are rotated or scaled. The overall framework of PointNet is shown in Figure 2.4.

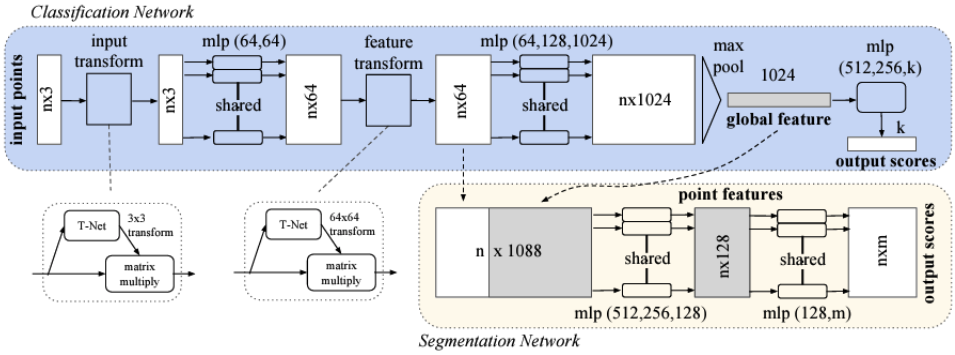


Figure 2.4: The PointNet architecture: taking the point cloud consisting of n points as input, we first extract the global feature, which can be directly used as the output of the classification task. For tasks such as segmentation, both global and local feature splicing scheme is used. In this process, T-net provides the role of restoring the point cloud affine transform and aligning the features. Figure from PointNet [54] paper directly.

In PointNet, the network does a low-dimensional to high-dimensional mapping for each point for feature learning, and then finally represents the global features through maximum pooling. In essence, it either operates on one point or on all points, and there is practically no local context, so it is difficult to learn some fine features which describe the detail of the points cloud. Second, the T-Net in PointNet is in fact not applicable to complex scenes, given that the rotation matrix in T-Net is not adapted to translation operations: for a single object we can translate it to the center of the coordinate system in the preprocessing to avoid this issue, while it is difficult to do so in complex scenes.

These shortcomings were the motivation of Pointnet++ [55]. In PointNet++, the authors use the distance metric in which they divide (partition) the set of points into local regions with overlap. Based on this, Pointnet is first used in small regions to extract local features (shallow features) from the geometric structure, and then the scope is expanded to extract higher-level features on top of these local features until global features are extracted for the whole point set.

Specifically, we first obtain a sparse version of the original point cloud by a downsampling method such as FPS(Farthest Point Sampling), and these points will be used as core points to aggregate features of points around them. Different points share the same PointNet when aggregated within a neighborhood. One such downsampling + aggregation accomplishes one abstraction of the point cloud, and several such combinations yield a high-level representation of the point cloud. Considering that such a representation is derived from a hierarchical aggregation of different local features, its ability to represent

the details of the point cloud is much better than the global feature approach mentioned before.

PointNet++ achieves decent results in point cloud representation task, but still has one obvious drawback: run-time efficiency. Indeed, the time complexity of FPS and Neighborhood Search in PointNet++ are proportional to the square of the number of points. This makes it difficult to apply in scenarios such as autonomous driving where the number of points in a single frame exceeds 200k.

In the next section, we will focus on the voxel-based scheme based on submanifold sparse convolution, which has been widely used in various types of scenarios with high requirements for efficiency.

2.1.2.2 Voxel-based solution

The difficulty in efficiently processing point cloud signals lies in their discrete nature, and the essence of the Voxel-based approach lies in the fact that we voxelize the space and represent the point cloud as an occupancy of a spatial grid. Based on this representation, we can treat the whole space as an image, where the three dimensions of width, height, and channel in the traditional image correspond to XYZ here. With inputs similar to image, can we just use the network we previously used for image processing? The answer is no, because here the network will lead this 3D spatial representation lose its sparsity.

Specifically, the CNN's algorithm dictates that it tends to blur the signal, i.e., spread the information on one pixel to the surrounding pixels. This is not a problem in RGB images, as the vast majority of pixels are not empty, and such diffusion instead facilitates the learning of features within the sensory field. However, the situation is very different in voxelized point cloud images: most of the voxels are empty, and the occupied/empty pixels are essentially the most important information describing the presence of an object in the space, and we don't want this information to be blurred after convolution. To better illustrate this concept, we show an example in Fig 2.5 of why traditional convolution cannot be introduced directly into the processing of lidar voxel maps.

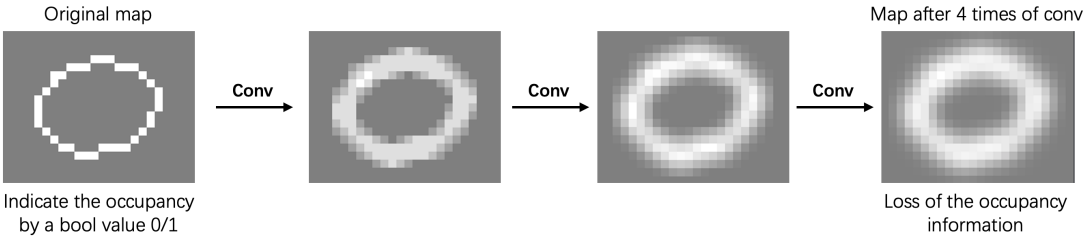


Figure 2.5: Spatial occupation information spreads out gradually with convolution, whereas we need the image to retain a portion of the information

In 2017, submanifold sparse convolution [17, 18] was proposed to overcome this issue. Specifically, an ordinary convolution operation responds to all input pixels and computes the output as soon as the convolution kernel covers an input point. In submanifold sparse convolution, we only compute when the center of the convolution kernel covers the signal. Such a convolution ensures that pixels that are 0 in the input image, what the authors call non-active sites, are likewise 0 in the output map.

Nowadays submanifold sparse conv has been widely utilized in various lidar-based 3D perception works [11, 58]. Its regular organization is three layers of submanifold sparse convolution followed by a layer of ordinary convolution. These 4 layers consist a block and the last layer of ordinary convolution will use 2 as stride so the final output of every block will be half of the original size.

So far we have introduced the point cloud and the image encoder in their conventional form and basic principles. In the next section we will combine them to obtain a vanilla backbone for fusion-based 3D detector, which has been used in early works such as AVOD [33].

2.1.3 Basic fusion-based 3D detection

The core of the multimodal detection task, is how to utilize dense RGB features to compensate for the sparsity of the lidar signal. To solve this issue, different models at different times proposed different answers. One basic idea could be to connect 2D and 3D features after we get 3D proposals by the detection head. Then these 2 feature maps can be fused together to further refine this 3D proposal to a final prediction box. This method, as shown in Fig 2.6, and called IOU-based fusion, is actually very common in early 3D detection models.

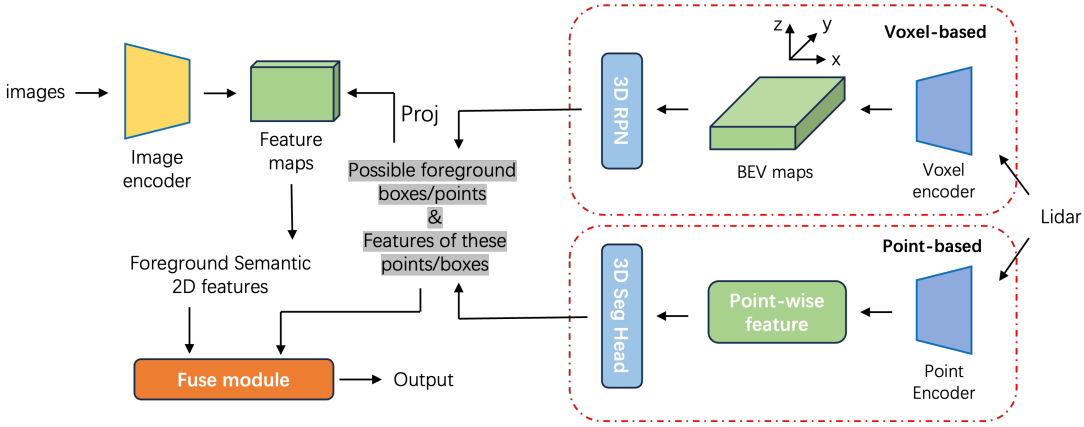


Figure 2.6: The classic IOU-based two-stage 3D detection pipeline: if we use a voxel-based encoder to process the lidar, as final BEV image, the RPN [16] can be used directly to generate proposals(boxes). If point-based encoders are used here, we need a point segmentation head to provide the possible foreground points instead of boxes. After we get these intermediate candidates, we need to project them back to a 2D image plane to get the semantic features, then fuse them together. The fused features will be used to refine the first-stage proposals to get the final results

Based on such a framework, two works are now presented: Voxel-based and Point-based approaches mentioned in Figure 2.6.

For the former, Voxel-based solution, MVXNet [61] is one of the most representative works. Overall it follows the architecture mentioned in Fig 2.6, with the only difference being that it chooses to rely on the results of 2D detection as opposed to a detection head that relies on 3D point cloud segmentation. Specifically, a pre-trained 2D detection module is used to provide 2D boxes. Only the points that can be projected to this part of image can get fused with RGB features. For Point-based solution, Multi-feature Fusion VoteNet [70] follow this idea and extend the detection to an indoor conditions.

We have presented the basic components and principles of 3D multimodal models that have made the joint detection of multiple sensors possible. Unfortunately, these solutions compare poorly with today's models both in terms of accuracy and algorithmic real-time requirements. So what makes these models so significantly different from today's SOTA solutions? In the next section, we will start by analyzing an implicit clue hidden in the work of recent years which constitutes a core element to improve 3D detection: object completion in 3D detection.

2.2 Object Completion: from global to local

2.2.1 Why object completion?

Like we mentioned, the core of cross-modality 3D detection is on how to utilize dense RGB features to compensate for the sparsity of lidar signal. Early research often looked to a late fusion-based, i.e., IOU-based method to associate semantic features with the lidar signals. However, in terms of spatial representation, the obtained RGB signals do not change the spatial occupancy of the voxel. So although the point cloud signal is enhanced by the images, this richer semantic feature can not solve the sparsity issue of lidar.

A better approach should rather be to enrich the point cloud early representation using a densification process. To extend this further, for the voxel-based and points-based approaches, this compensation process changes the occupancy of the voxel or generates virtual points respectively. In fact we can summarise **most** of the recently published work on 3D detection follow this basic framework of shape completion.

As shown in Figure 2.7, we categorise most of the relevant models using two metrics: the mode and scope of virtual points or voxel generation.

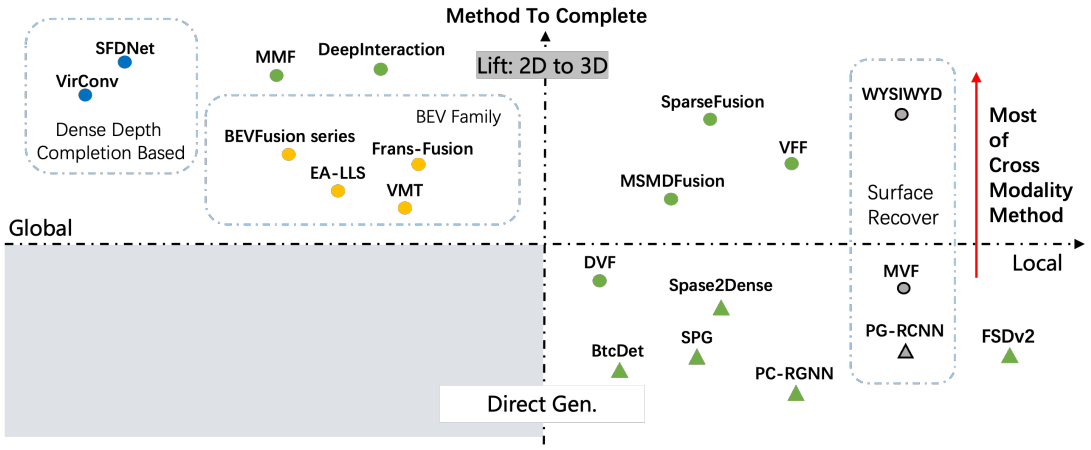


Figure 2.7: The classification of completion-based methods. Triangles represent single modal and circles stand for image-rgb multimodal approaches or plug-to-play methods which can be directly used in cross-modality models. Only the most representative models were selected for the figure.

On the horizontal axis, models were classified by the scope of the virtual points they generated: the more to the left of the axis the more they tend to generate points globally,

whereas the more to the right the models were more concerned with the densification of the foreground points, i.e., generate points locally. To be more specific, the "global-level" here means when we unproject 2D features to 3D space, the features of foreground and background are not distinguishable. In the other word, we estimate a depth distribution for every pixels on the feature map. Considering the image's inherent dense representation, most cross-modal schemes use the lift scheme to unproject features to 3D space, while lidar-based solutions tend to generate the points directly in the 3D space, since the lack of RGB sementic features.

On the vertical axis, we categorized all the work in the Y-axis according to the way the virtual points were generated, and most of the articles in the positive half-axis of the Y-axis used explicit depth estimation of the pixels, i.e., the color information was lifted from the 2D plane to the 3D space. Depending on the granularity of this lifting operation we assign a height in the coordinates of the positive half-axis of the Y-axis. The methods for the negative half-axis of the Y-axis have all adopted the use of the network to directly generate foreground point solutions. Considering that this is not compatible with the concept of a "global" range, the third quadrant is left empty. In the fourth quadrant, the smaller the value of the Y-axis coordinates, the more virtual points are added to the model.

Although most cross-modality models adopt the lifting method, the **granularity** of this lift operation can be quite different. Next we will begin with BEV-based method and illustrate why it can be explained by completion of objects. After we have introduced the bird-eye-view-based global-completion scheme represented by BEVFusion, more advanced local-completion based solutions will be introduced.

2.2.2 Global-level objects completion

When BEV perception was first mentioned by Tesla [64] in 2020, this intuitive and simple solution immediately attracted widespread attention in the academic community. The key of BEV-based methods is the RGB-BEV transformation, which allow the lidar and RGB to be processed in a homogenous space.

To begin with, a very classic work related to 2D-3D lifting is the LSS (Lift, Splat, Shoot: Encoding Images from Arbitrary Camera Rigs by Implicitly Unprojecting to 3D) [52], which is always seen as the predecessor of the BEV-based method. This work proposed an explicit depth estimation scheme for end-to-end training. The core step in this paper is the outer product. With the image features $\mathcal{F}_i \in \mathbb{R}^{H\times W\times C}$ and estimated depth $\mathcal{F}_d \in \mathbb{R}^{H\times W\times D}$ (C for image channels and D for depth bins), the outer product can construct a dense tensor $F_{3d} \in \mathbb{R}^{H\times W\times D\times C}$ to present the distribution of image features in 3D space. This lifting

method adequately brings image features into 3D space as a kind of compensation for the lidar signal: it **changes the occupancy of the voxelized space** and if we assume that the depth information is accurate, in fact, this method can completely reconstruct the target surface information.

This step designed by LSS was utilized by most BEV-based cross-modality methods including the BEVFusion [42] or vision-centric methods like SA-BEV [83], DepthBEV [39]. Considering BEVFusion as an example, by completing the perspective transformation in the way mentioned above, then compressing the Z-axis of the lidar signal after sparse convolution to get a lidar tensor, and finally merging these two features into the detection header, a standard multimodal BEV pipeline has been obtained, as shown in Fig 2.8.

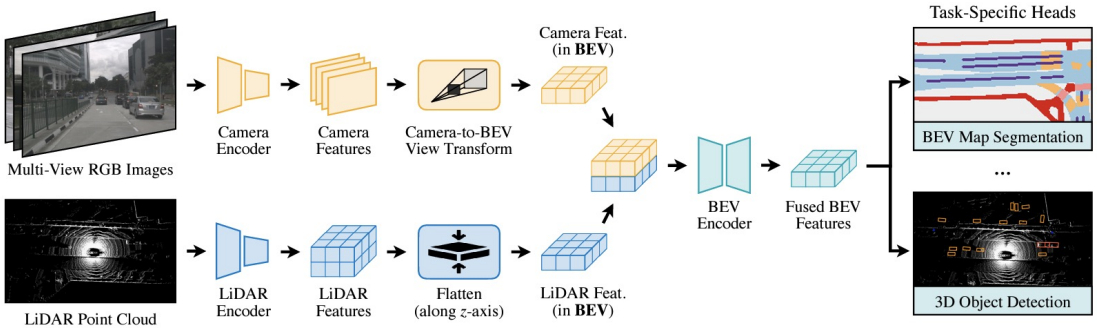


Figure 2.8: A classical pipeline for BEV-based Fusion solution, BEVFusion [42] used as an example here. The obtained features in BEV space can be used as the input to a series downstream tasks such as segmentation or detection.

Similar structural diagrams can be seen in more multimodal models, such as the fusion of radar, lidar, and camera in the BEVGuide [49]. To get a representation for lidar and RGB image in BEV format, some of the models also replace the modality-related encoder with a uniform transformer [67].

However, no matter how we design networks to obtain increasingly high-quality multi-modal features, this class of models always shares the same dilemma: the scale of image features is in fact highly correlated with accuracy and efficiency.

The essence of the problem is that we need to estimate the depth distribution for every pixel on the feature maps. After we downsample the RGB image, the number of features unprojected to 3D space is depends on the size of the feature maps. As shown in Figure 2.9: if we choose a higher resolution for the features maps to better describe the depth especially for distant or small objects, the model will lose the real-time efficiency. However, if we

choose the opposite, there will be a dramatic decrease in the accuracy with the increase of downsample times.

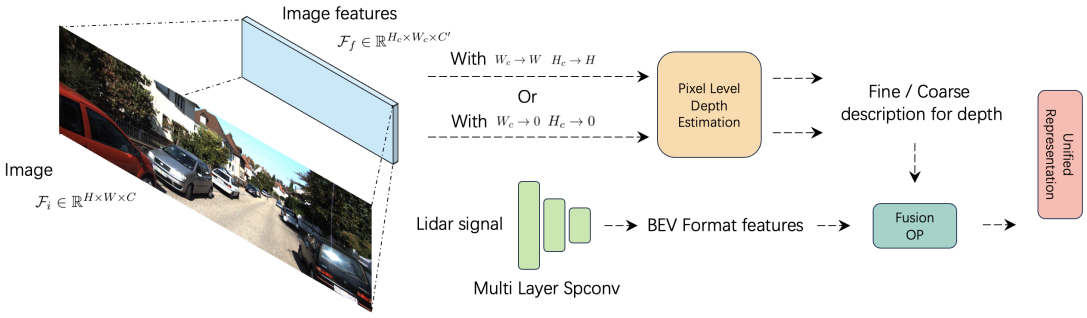
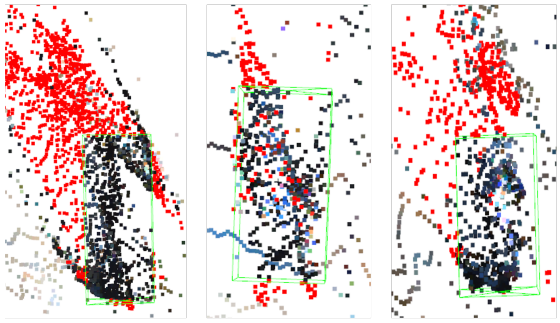


Figure 2.9: The dilemma for the BEV-based method: speed or accuracy? With $H_c \to H$ and $W_c \to W$, more details can be added to the 3D space to get a better result, with an increased calculation burden. However, $\mathcal{F}_f$ with too low resolution cannot help the densification of the lidar signal.

SFDNet [72] in 2022 was the pioneer of this approach, by utilizing the depth completion model as the pre-posed model to provide pseudo point. Specifically, to make a depth prediction on the image, we need to introduce an additional depth-completion network. Here our model is not quite the same as in the BEV scheme for the unprojection of color features: we need to project the lidar signal onto the 2D image first, and then completion using a features from image, instead of relying solely on the semantic information to make the depth estimation. This process is often accomplished by a pre-trained EPNet [23] or TWISE [25]. Another difference from the BEV scheme is that in our case we directly put the predictions of this pre-posed network into 3D space, rather than a distribution along a specific ray.

Although such a scheme allows as many points as possible to be added, the downside is that: the depth dispersion effect of the points predicted by the completion network is very pronounced, i.e. there is a lot of noise in the hundreds of thousands of points that are added. To be more specific, in the KITTI dataset, the size of a frame is 375 pixels high and 1242 pixels wide, which means that more than 460k pseudo-points are added to the 3D space. In Figure 2.10, we visualize this boundary depth dispersion problem, from which we can see the points added are not always accurate and most errors in depth estimation are located in the boundary of the objects. This can in fact be attributed to the fact that convolutional networks in fact tend to smooth the signal rather than output discrete values.



(a) Pseudo points (depth estimation) visualization from BEV

(b) Visualization of error depth estimation from image plane

Figure 2.10: Boundary depth leak problem, ground truth 3D box shown in green, the estimated depth outside of the 3D box is marked in red. Most of the misprediction pixels occur along the boundary.

To solve this issue, VirConv [71] proposed a discard module to eliminate the noise, which also makes their method to rank first in KITTI [15] benchmark in 2024. However, this runtime discard method is just partial alleviation of the problem rather than a completely solve it.

Despite the strong detection performance demonstrated by the depth-completion-based model, its performance in terms of compute time is very unsatisfactory: for SFDNet, counting the pre-posed depth completion network, the processing time for a single frame of the entire model on the KITTI dataset is close to 200ms [24] [23] with an RTX2080Ti test environment. The main reason for this dilemma is the fact that a large number of virtual points/voxels unprojected into 3D space by these methods are not in the foreground region, and a large amount of computation is the wasted.

Here, we summarize the drawbacks of global completion as follows:

- A front-loaded depth-completion network takes a lot of time, which makes the whole network far from being able to satisfy real-time requirements.
- Most of the pixels in the image are background and the depth of foreground objects, which are most important points in detection, accounts for only a small fraction of the estimate. Thus a large amount of the estimated depth is in fact not utilized.
- In these few effective foreground depths, we observe a severe boundary depth dispersion effect, which further limits the available valid depths

To solve these problems, models proposed in recent years have worked on the completion of the foreground region implicitly or explicitly.

2.2.3 Local-level completion

As shown in Fig. 2.7, all models on the right side of the horizontal axis focus only on local completion, i.e., foreground object completion, which makes them better in inference speed than the previously mentioned models (left side of the horizontal axis). These models are again divided into two parts according to the way they generate virtual points: directly or using the lift scheme as well.

For these models, we only want to densify the signal locally, so the core problem that needs to be solved for models here is how to estimate a relatively accurate foreground area to densify. To be more specific, 2 questions that should be answered here are "how to densify" and "where to densify". In Table 2.1, are listed and elaborated under these two different indexes. Next, we will introduce the lidar-based local-completion models and the extend to the multimodal ones.

Table 2.1: Local completion models: how and where to generate virtual points. L and R-L stand for models take lidar or camera-lidar signal as input.

Paper	Refer	Modality	Where to densify	How to densify
SPG [77]	ICCV-2021	L	Missing voxels prediction	Foreground voxels expansion
PC-RGNN [86]	AAAI-2021	L	3D proposals from point-level features based RPN	Multi-Resolution graph encoder and decoder
BtcDet [76]	AAAI-2022	L	Occuled voxels in cylinder coordinate system	Offset to center of empty voxel
Sparse2Dense [69]	NIPS-2022	L	3D proposals from RPN after spconv backbone	Similarity optimisation in hidden space
FSD-V2 [14]	ArXiv-2308	L	Foreground candidates from 3D points segmentor	Add objects center voted by foreground points back to feature maps
PG-RCNN [31]	ICCV-2023	L	3D proposals from RPN after spconv backbone	Offset to ROI grid points to recover the boundary
MVF [81]	NIPS-2021	R-L	Voxels in 2D instance mask from segmentation model	Sampling and nearset neighbor points matching
DVF [48]	WACV-2023	R-L	Voxels in 2D boxes from detection model	Voxels center in different scales used as virtual points
VFF [38]	CVPR-2022	R-L	Same as the MVF	Expand the lidar features to voxels go through by rays
MSMDFusion [27]	CVPR-2023	R-L	Same as the MVF	K Nearest neighbor retrieval
SparseFusion [74]	ICCV-2023	R-L	3D proposals from both RGB and Lidar branch	Concatenate sparse candidates from 2 backbone
WYSIWYD [44]	ArXiv-2310	R-L	Same as the MVF	Estimate pixel-wise dense depth for object masks

2.2.3.1 Lidar-based local-completion model

For most of the single modality model, local completion still needs a 3D backbone to provide candidate proposals and then densify the points inside. PC-RGNN [86] is an typical example for this kind of models. After generate the 3D proposals, PC-RGNN

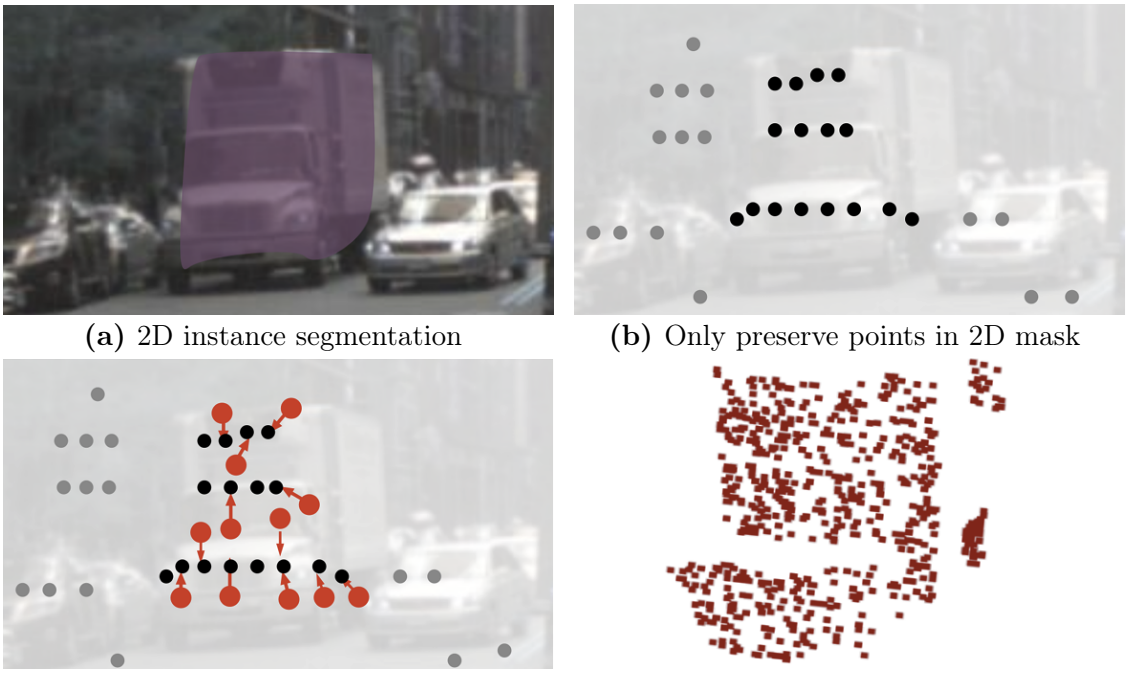
proposed a multi-resolution graph encoder to recover the full shape of the objects. This process is guided by a adversarial loss that discriminates between real and fake objects. For PG-RCNN [31], after using ROI Grid Pooling to extract the proposal-wise features, points can be generated around the 3D proposals. A similar strategy appears in SPG [77], which, given a foreground region, designs an unsupervised extension scheme to recover the object shape.

However, this process heavily relies on a 3D backbone for which the distant objects can be easily ignored due to signal sparsity. Therefore, considering that 3D boxes proposed might be diffcult in a sparse scenario, lots of work published recently foucs on how to recover missing lidar signal by a point-based or voxel-based methods.

For example, by voxelizing the space with a cylinder coordinate system, Btcdet [76] attempts to recover signal loss due to occlusion directly in 3D space. This step is directly predicted by a subnetwork, so theoretically Btcdet's scheme can be merged with any detector. Compared to a task as intuitively difficult as recovering the full shape of a target, the FSD-V2 [14] strategy is much more elegant. First of all, unlike the traditional 3D backbone + RPN combination, the FSD-V2 uses a segmentation head to get the foreground points. Following the voting mechanism introduced in VoteNet [53], FSD-V2 also votes for the center of objects. However, the difference is that the voted center of objects are added back to the space, and in this way, objects are densified at the center. Considering the point cloud segmentation task will be intuitively simpler than proposal generation, this is probably the main reason why that FSD-V2 performs better than the previous work.

Sparse2Dense [69], on the other hand, offers a different path from the previous explicit densification: it trains a teacher-student model to densify the foreground region directly in the hidden space. Specifically, Sparse2Dense contains two training stages: first, an independent network is trained with the pre-densified lidar signal, to get the dense representation for the proposals. Next, these features are used to supervise the features obtained from real lidar signal.

Considering the lack of dense semantic information, although numerous ways have been tried to improve 3D monitors with lidar as input, it is still inherently difficult to localize sparse objects at a distance. It is also for this reason that we need to introduce RGB color information to help us better locate these objects and help us recover the shape of them better.



(a) 2D instance segmentation

(b) Only preserve points in 2D mask

(c) Add points to densify the foreground objects

(d) The object after densification

Figure 2.11: The foreground part densification process proposed by MVF [81]

2.2.3.2 Fusion-based local-completion model

The first model covered in this part is MVF [81], which is the base of most local-completion-based multimodal detection models. The core idea here is using a 2D task to guide the densification process. In MVF, a 2D instance segmentation algorithm is first applied to get a foreground mask. Then only lidar points which can be projected inside the mask are preserved. Finally, more points can be added according to the depth of their neighbour points by pre-defined algorithm. This process are visualized in the Figure 2.11. Based on this work, the following work MSMDFusion [27] optimizes the points generation method from a neighbour-based solution to a weighted KNN model.

The VFF [38] and DVF [48] provide other ways to densify the objects in the 3D space. For VFF, it proposes a ray-wise fusion scheme to replace the original point-wise or voxel-wise fusion. This method actually change the one-to-one relationship between voxel and pixel to into a one-to-many relationship between a pixel and multiple voxels on a ray. Therefore, the occupancy of objects changed for the points on its surface yielding to a

denser representation. A similar strategy is used in DVF, where voxels of different scales with RGB information. The object from a point cloud describing only the surface to a solid object in 3D space. SparseFusion [74] takes a slightly different approach, splicing two proposals from different branches in an attempt to densify this representation.

So far, we have introduced the local object completion scheme in the lidar-RGB fusion model, although this field has made long progress in recent years, there are still several problems to be solved. Most works still rely on MVF as their virtual point generation module, but this module has the following problems.

- Not all points that can be projected onto the 2d masks can be used as reference points for virtual point generation. A lot of background noise is projected onto the edge positions, which affects the estimation of the position of the newly generated points.
- Essentially the points added are proportional to the points that can be projected into the 2d mask. This means that we are adding fewer points for sparse objects, which is in fact in opposition to the problem we are hoping to solve. In other words, such schemes are in fact detrimental to small objects such as pedestrians

Having observed these issues, starting from the next chapter, we will try to introduce a completely new method to 3D detection: depth-aware mesh deformation, to alleviate and solve the mentioned problems in both local-completion and global-completion methods.

Chapter 3

Methodology

If we were to summarize the problem mentioned in 2.2.3.2 in one sentence, its would be: how do we get more and better quality foreground attractions to complement the target. Especially for small targets, the number of these newly added points should far exceed the number of true lidar points the target contains. Of all the schemes we have mentioned, only the dense-depth-completion method in fact provides a sufficient number of points, since the number of points it provides is in fact equal to the 2D pixel occupied by the object. However, here we stilll have 2 problems to prove before we can improve the quality of the points based on this method.

- Dense-depth-completion based method in fact provides depths for all points including background and foreground, so is it true that only foreground depths are valid for subsequent 3D detectors?
- Dense-depth-completion based method in fact provide only depth completion of the object surface. Is it optimal for downstream detectors compared to the full shape completion commonly used in pure point cloud schemes? If not, how else can it be optimized?

In section 3.1, we will address these issues step by step and give a clear goal for the model we need to propose.

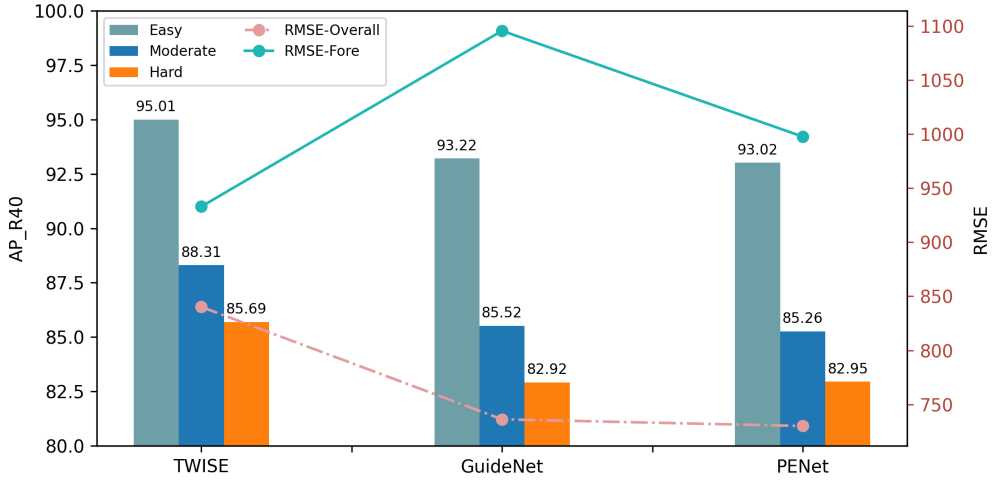


Figure 3.1: SFDNet 3D AP with different depth completion methods. TWISE has the highest overall RMSE (dotted orange line) but the lowest RMSE for foreground objects as shown by plain green line.

3.1 Deep analyze of dense-depth-completion method

A typical pseudo-points-based detector method needs an upstream depth completion module, in which the depth of every pixel is estimated. Here we use SFDNet [72] as the testbed to explore the relation between the accuracy of foreground points and 3D detection. The authors of SFDNet mentioned PENet [22] and TWISE [25] in their paper and implementation but only adopted the latter in the published code. Consequently, we first simply replace this completion module with different depth completion methods to verify their effectiveness inside the downstream 3D detection.

As shown in Fig. 3.1, the highest 3D detection performance occurs when TWISE is used. But when applying RMSE(Root Mean Squared Error, the lower the better), the criterion for evaluating the performance of depth completion, TWISE obtains the worst results (as shown by the pink line). Considering that the RMSE evaluates both foreground and background depth, while the foreground is more critical to 3D detection, we speculate that TWISE obtain a better performance in foreground depth prediction. To verify this hypothesis, we need to generate the dense ground truth of foreground depth, since the lidar points cannot guaranteed that all pixels in the 2D foreground region can get a depth estimate. This generation process is introduced in section 3.1.1 and will later be used to estimate the RMSE of the foreground objects in section 3.1.2.

3.1.1 Visible part dense depth generation

With the annotated 3D boundary boxes and 2D masks [21], the points that can be projected to the specific mask of objects in the KITTI dataset can be extracted from the lidar file, denoted by $P = \{p_1, p_2, \cdots, p_Q\}$ $p_i \in \mathbb{R}^{(n_i \times 3)}$, Q objects in total, n_i points for the object i. The masks of all objects are denoted as $\mathcal{O} = \{\Omega_1, \Omega_2, \cdots, \Omega_Q\}$ $\Omega_i \in \mathbb{N}^{(m_i \times 2)}$, m_i pixels for the mask i.

The set P and $\mathcal{O}$ are equinumerous, in the other word, every image has its corresponding lidar points set. However, the lidar signal is not dense enough in most cases. In this subsection, our objective is generate a dense depth ground truth that can make $\forall i$, $n_i = m_i$, which means for every image segments, we need to guarantee the number of pixels in the image segements is equal to the number of points in its lidar points set.

In order to achieve this, we have to perform roughly three steps, as shown in Figure 3.2. Firstly, we need to perform full shape completion, then we need to reconstruct the surface based on the complementary point cloud, and finally we need to filter the occluded parts according to the image information. In the following, we describe in detail the specifics of completing this process based on the data in KITTI dataset.

Before performing the whole process, It's worth noting that we first need maintain a pool of objects that contain the object p_i extracted by 3D boxes, only if $n_i > 20$ for cars and 10 for pedestrians. This filtering is quite useful to avoid unrealistic estimation since for those objects are very sparse, the shape information we can obtain are quite limited.

To generate the full shape completion, we will start by using symmetry for completion. Considering that the lidar signal only covers a part of the object and the objects we need to detect, such as vehicles and pedestrians, are highly symmetric in terms of their shape profile, we firstly perform a mirroring operation on every object p_i in the objects pool P to recover its backside signal. The output cloud points in this step will be denoted by A. At the same time, considering that most of the objects in the same category are very close in appearance, we can use a heuristic, denoted by $\mathcal{H}$, to match different objects one by one. This way, we can obtain a best match B for every A, and then combine them together to further complete A.

The heuristic to find out B, denoted by $\mathcal{H}(A, B)$ is proposed in BtcDet [76]. The heuristic is shown in Equation 3.1, where P_A and P_B are the points set for object A and candidate B and D_A and D_B are their bounding boxes. The $\sum_{x \in P_A} \min_{y \in P_B} \|x - y\|$ is half Chamfer Distance measures the overlap between A and B , the term $\alpha \operatorname{IoU}(\mathcal{D}_A, \mathcal{D}_B)$ is the similarity of their bounding box size, and the third term $\beta / |\{x : x \in \operatorname{Vox}(p_B), x \notin \operatorname{Vox}(p_A)\}|$ measures the number of extra voxels that B can add to A. Here α and β are the hyper-parameter

to control dependency on individual items.

$$\begin{aligned}\mathcal{H}(A,B) = \sum_{x \in P_A} \min_{y \in P_B} \|x - y\| - \alpha \operatorname{IoU}\left(\mathcal{D}_A, \mathcal{D}_B\right) \\ + \beta / \left|\{x : x \in \operatorname{Vox}\left(p_B\right), x \notin \operatorname{Vox}\left(p_A\right)\}\right|\end{aligned} \tag{3.1}$$

However, we noticed that this method can hardly reflect the shape of a real 2D mask when we project them to the image, especially when the object is very sparse. So here, the heuristics are modified as the follows: for every sample A, a best matching B will minimize the $\mathcal{G}(A,B)$ as shown in Equation 3.2. Here, we add a term to calculate the pixel-wise IOU between ground-truth and 2D mask obtained from the projection of completed objects. ϵ is an indicator function, when $n_i > 10$, ϵ equal to zero otherwise 1. The Γ is a surjection between space location and 2D location: $\Gamma : \mathbb{R}^{(n_i \times 3)} \rightarrow \mathbb{N}^{(n_i \times 2)}$, determined by the intrinsic matrix.

$$\mathcal{G}(A,B) = \mathcal{H}(A,B) + \epsilon IOU(c_A, \Gamma(p_B)) \tag{3.2}$$

$$p'_A = p_A + p_B \ \in \mathbb{R}^{(n_A + n_B) \times 3} \tag{3.3}$$

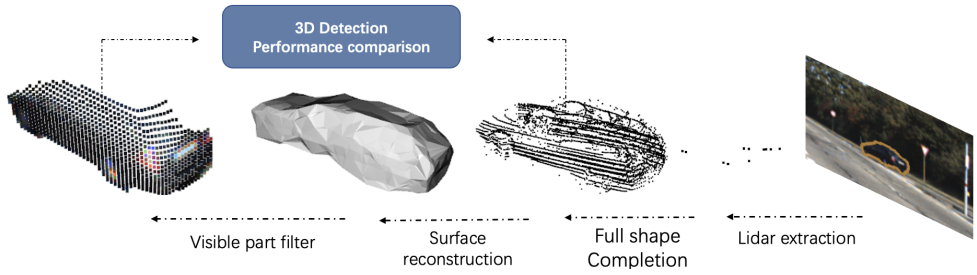


Figure 3.2: From sparse lidar to visible part ground truth. With the full shape completion and visible part completion groundtruth, we can compare the upper boundary of detection performance (see Table 1).

After the completed p'_A is obtained, the Poisson surface reconstruction [28] denoted as $\mathcal{F}$, is adopted to get the hull of objects, denoted by $\mathcal{M}_A$.

$$\mathcal{M}_A = \mathcal{F}(p'_A) \tag{3.4}$$

Placing the obtained hull(reconstructed surface) in 3D space, with the known camera intrinsic matrix, extrinsic matrix, and the pixel 2D coordinates of mask Ω_i, a ray casting model can be established. Note that since the rays from the pixel on the boundary sometimes misses the object, a volume expansion coefficient δ is used here to make sure there is

always a bijection from pixel set to depth set. For a specific pixel c in the mask Ω_i, when we project it to 3D space, all of its possible locations are distributed on a well-determined line, this line will intersect the camera plane at point $c_o : (0, y_c, z_c)$. The direction vector of this line denoted by $(\vec{x_c}, \vec{y_c}, \vec{z_c})$. Then the analytical equation of this line can be expressed as Equation 3.5, denoted by L_c.

$$L_c : \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 0 \\ y_c \\ z_c \end{pmatrix} + \lambda \begin{pmatrix} \vec{x_c} \\ \vec{y_c} \\ \vec{z_c} \end{pmatrix} \tag{3.5}$$

With the equation of line L_c and 3D mesh $\mathcal{M}_A$, use $\cap$ to represent the operation of taking the intersection point, the distance travelled by a ray of light from point c when it arrives on the mesh can be expressed as $|L_c \cap \delta\mathcal{M}_A - c_o|$. Considering the slope of LC, here the depth of the point on the mesh hit by the light to the camera plane is d_c. Here $\cdot$ means scalar multiplication operation.

$$d_c = |L_c \cap \delta\mathcal{M}_A - c_o| \cdot |\vec{x_c}| \tag{3.6}$$

By repeating the mentioned operation for every pixel, we can get the dense depth for the visible part, as shown in Figure 3.2. With the ground truth obtained, the foreground RMSE for 3 different models have been calculated as shown in Figure 3.1. The plain green line here demonstrates that among all candidate models, TWISE can achieve the best performance in foreground depth prediction. We thus showed that the key reason TWISE benefit downstream 3D detection is its high quality foreground depth prediction. In other words, the higher the foreground pseudo points quality, the better our 3D model performance. Pursuing with this idea, next we will present the result of using the generated ground truth to train a model directly. We will explore the upper boundary performance of this completion method and compare it with that obtained from the full completion.

3.1.2 Completion method comparasion

Intuitively, a more complete object should lead to better results. However, from our experiments, the visible part of ground truth provides a counter-intuitive answer.

Here we select 4 recent lidar detection models, which take the original lidar points and augmented points from different methods as input. As shown in Table 3.1, a wide performance increment can be observed especially in pedestrian detection. With only 11.3% (on average) of the points contained in full shape completion, in most cases, the

Table 3.1: The upper boundary performance (i.e. based on ground-truth shape completion)of different object completion forms a comparison on KITTI val: models mentioned in the table are typical point-based, voxel-based, and cross-aggregation-based methods. VP/FS: visible part completion or full shape completion. The bolded line is the method with the highest accuracy. Detection thresholds are 0.7/0.5/0.5 for Car and 0.5/0.5/0.5 for Pedestrian, The highest accuracy in the last 20 epochs are selected to list in the table

Methods	Object Category	Complete Category	3D AP_{R40}			BEV AP_{R40}		
			Easy	Mod.	Hard	Easy	Mod.	Hard
Point-RCNN [59]	Car	VP	**99.92**	**99.67**	**97.27**	**99.94**	**99.69**	**97.24**
		FS	97.27	97.26	97.19	97.29	97.26	97.19
	Ped	VP	**85.32**	**77.91**	68.25	**86.12**	**80.95**	**71.22**
		FS	77.86	72.84	**70.35**	80.34	74.89	71.10
IA-SSD [87]	Car	VP	**99.92**	**99.77**	99.40	**99.91**	**99.27**	99.32
		FS	99.71	99.47	**99.42**	99.65	99.22	**99.38**
	Ped	VP	**75.58**	**71.38**	66.47	**77.53**	**75.11**	70.37
		FS	71.92	69.79	**68.20**	74.89	74.37	**72.86**
Voxel-RCNN [8, 11]	Car	VP	99.62	**99.70**	**99.62**	99.63	**99.72**	**99.67**
		FS	**99.96**	99.47	99.47	**99.99**	99.49	99.49
	Ped	VP	**91.94**	85.73	82.28	**93.86**	**89.61**	83.97
		FS	91.74	**88.64**	**85.47**	92.94	89.58	**87.31**
PV-RCNN [58]	Car	VP	99.47	99.46	**99.46**	99.44	99.48	**99.47**
		FS	**99.98**	**99.99**	97.49	**99.98**	**99.99**	97.49
	Ped	VP	**88.74**	**84.09**	**79.20**	**92.64**	**88.00**	**83.15**
		FS	80.40	78.97	77.45	84.68	82.96	83.10

improvement of our proposed completion approach can be up to 8.5%. We also noticed that worse results happen in the hard category, which can be explained by the small number of pixels available.

This experiments proves that our proposed visual partial completion is in fact more suitable for 3D detection tasks. We believe this result can also be used to explain why the recent proposed depth completion model can outstand the vanilla shape completion-based methods: the former actually provides an overall higher upper boundary and needs less points to be predicted.

So far we have demonstrated the higher potential of the proposed visable part completion approach and the foreground depth do mpre important than the background one, which have answered the 2 questions raised at the beginning of this chapter. Our next question is if visable part foreground points matter, then how can we learn from these densified label and complete the objects? In the following section, we will introduce the proposed network which uses the points in frustum as input and generate pixel-wise visible part depth in a mesh-deformation manner. In Figure 3.3, we show the flowchart of the model proposed. In general, to get a foreground pixel-wise depth estimation, we need to first segment the foreground points with the 2D masks and then get a dense representation of these points.

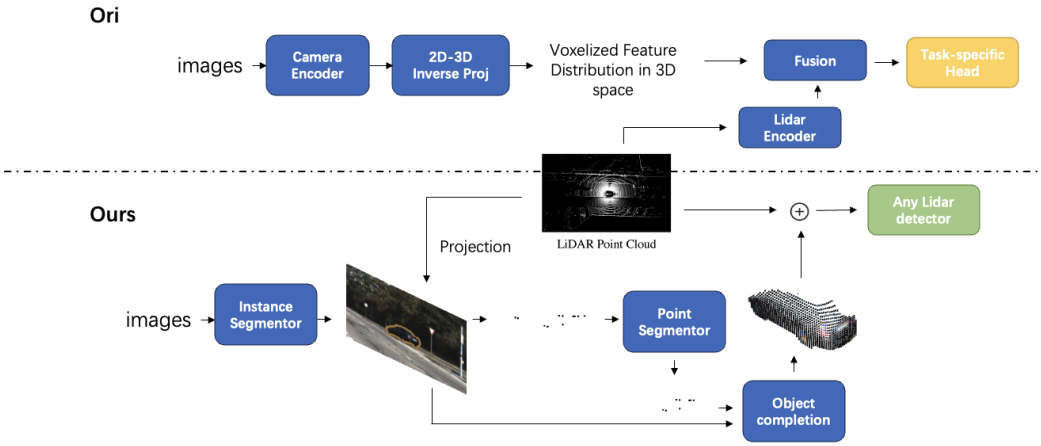


Figure 3.3: The comparision of pipeline: a new point segmentor and object completion module need to be proposed to get the densified objects. We call these 2 modules "Intra-Frustum Segmentation Transformer(IFST)" and "Mesh Depth completion Network(MDCNet)" in the following content.

3.2 Intra-Frustum points segmentation

In order to identify the foreground points that need to be densified, we first project all lidar points onto a 2D mask which is obtained from an image instance segmentor. As we mentioned in section 2.2.3.2 as the drawback of the MVF-based points adding method, although this projection successfully removes most irrelevant background points, it is worth mentioning that there may still be some noisy lidar points within the frustum due to inaccuracies in 2D segmentation and potential occlusions in 3D space. So in this section, we propose a lightweight transformer network to identify the foreground lidar points.

Large-scale point cloud segmentation tasks have been well developed in recent years [19, 34, 55], however, few works focus on Intra-Frustum segmentation. From our experiments, we identify three essential characteristics that the performance of models can benefit from when conducting Intra-Frustum points segmentation.

- **Eliminate downsampling operation:** Different from the traditional full scene point segmentation scenario, in a typical frustum produced by an image mask, each point has the potential to bring in useful semantic information. Considering that the overall number of points in frustums is only of the order of 100-1000, there is no need to adopt any downsampling strategy in the design of this network.
- **Guidance from 2D location:** During the projection from 3D to 2D, the background points naturally have a higher probability of being located on pixels at the boundary of the mask. This is a priori assumption has to be considered in order to obtain a more accurate point segmentation.
- **Guidance from Perspective relationship and Points Density:** Another geometric property that is often overlooked is that the objects in the image are naturally larger when close to the camera. This perspective phenomenon allows us to easily filter out some of the noise points.

To optimize the utilization of the mentioned characteristics without compromising inference speed, we propose the Intra-Frustum Segmentation Transformer (IFST), as depicted in Figure 3.4. Here we use $p = \{p_0, p_1, ..., p_n\}$ to represent all points that can be projected to the 2D segment and $p_x = \{p_{x0}, p_{x1}, ..., p_{xn}\}$ to represent the depth of these points.

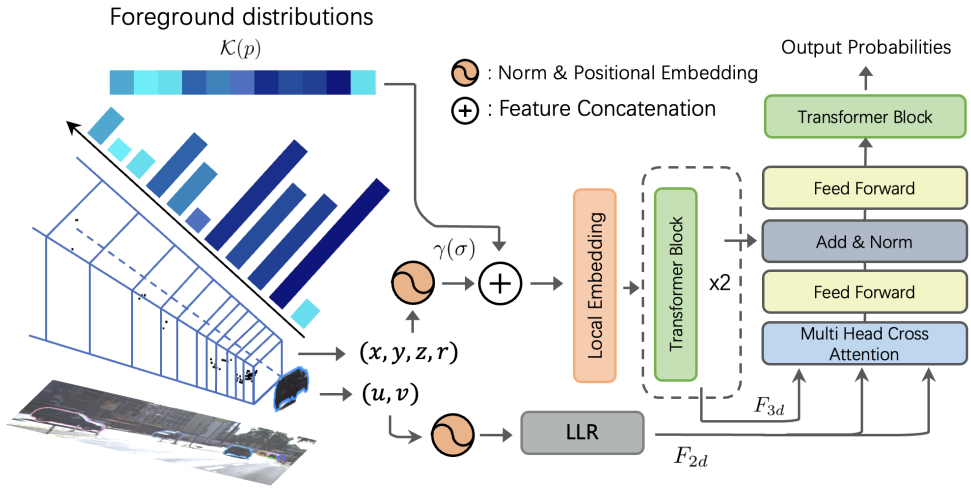


Figure 3.4: Illustration of the IFST design, a light weight Intra-Frustum Transformer. Both size of the mask and 2d relative location are used as prior knowledge to guide the 3D points segmentation.

As we mentioned the last item, the density and perspective relationship can be very useful for intra-frustum points segementation. To use these properties, we first divide the whole frustum by the density-adaptive splitting scheme shown in Algorithm 3.1.

Algorithm 3.1 Density-adaptive splitting frustum

Input: The depth values of the n points associated with one single foreground object are $p_x = \{p_{x0}, p_{x1}, ..., p_{xn}\}$. The elements of the set have been sorted from smallest to largest. Bandwidth h for density estimation and number of bins H.

$\forall p_{xi}, \hat{f}_n(p_{xi}) = \left| \frac{1}{nh} \sum_{j=0}^{n} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(p_{xi}-p_{xj})^2}{2h^2}\right) \right|$

$f_{score}(p_{xi}) = H * Softmax(\hat{f}_n(p_{xi})^{-\frac{1}{2}})$

$d_{bin} \leftarrow \varnothing, \quad j, k, d_{init} \leftarrow 0$

repeat

  repeat

    $j \leftarrow j + 1$

  until $\sum_{i=k}^{i=j} f_{score}(p_{xi}) = 1$

  $d_{bin} \leftarrow d_{bin} \cap \{d_{xj}\}; \quad k \leftarrow j$

until $|d_{bin}| = H$

Specifically, with this algorithm, we want to split the frustum into bins by the density of

points, and use this density infomation to preliminary estimate the foreground distribution of different bins as a supplymentary infomation for the subsequent network.

First of all, by combining Gaussian kernel density estimation and softmax function, we assign to each point a score proportional to the density of points around it. The second step in Algorithm 3.1 scales this score with a given number H and softmax function, this operation will scale the score and make sure the sum of $f_{score}(p_{xi}) = H$. This score will be used to split the frustum.

As we mentioned in the input part of Algorithm 3.1, the points set p_x have been sorted. Here by accumulating this value in the order of depth, we can split the original frustum with a finer granularity when the points are dense and a coarser when they are sparse. This process corresponds to lines 3 to 9 of Algorithm 3.1: We first initialise d_{bin} to an empty set, and j, k, d_{init} as zero. Then we keep count the sum of f_{score} from the first point(nearest one) to the j-th, and stop when equals to 1. The depth of j-th point will be used as the threshold to split the first frustum. By repeating this process, we can split different frustum based on the density, and we will give a foreground possibility to every bins(frustum) later.

After partitioning the frustum space, a pointnet-like structure, denoted by $\mathcal{PN}$, will process points in different sub-frustum with the 3D location, $f_{score}(p_{xi})$ and the size of 2D mask $\mathcal{S}$ as different channel of input. These information will be used to predicts the $\mathcal{K}(p)$ which shows the probability of each sub-frustum being foreground, as shown in Equation 3.7, $\oplus$ stand for concatenation operation. This probability distribution will later be concatenated with 3D location embedding, and then sent to the subsequent network.

$$\mathcal{K}(p) = MLP(\mathcal{PN}(p) \oplus \mathcal{S}) \tag{3.7}$$

To get a better representation, we adopted two tricks from [34, 51], in the design of IFST. First, we transform the 2D/3D points into the frequency domain using sinusoidal functions as shown in equation (3.8). This projection allows similar inputs under Euclidean space to be clearly recognized by the network. σ_i here represent the feature in the i-th dims and $\sum$ represent feature stack operation. Specifically, both p_i and its projection (u_i, v_i) are processed by this function separately, and the computed 3D embedding $\gamma(\sigma)$ will be concatenated with $\mathcal{K}(p)$ as shown in Fig 3.4. Secondly, to better describe the local features of the point cloud, an SA-Layer [55] is used to aggregate the features of local neighbors before the 3D features are processed by a transformer layer.

$$\gamma(\sigma) = \sum_{i=0}^{5} \left[\sin\left(2^0\pi\sigma_i\right), \cos\left(2^0\pi\sigma_i\right), \cdots, \sin\left(2^{L-1}\pi\sigma_i\right), \cos\left(2^{L-1}\pi\sigma_i\right)\right] \tag{3.8}$$

The 2D location will be processed by several stacks of Linear-LayerNorm-Relu layers, denoted by LLR in Figure 3.4, to get F_{2d} . F_{3d}, the feature from 3D stream will then be guided by F_{2d} in a cross-attention manner. More details on this attention pipeline will be introduced in the experiment chapter. The final output is the probability for each point to be part of foreground. The role of the IFST is therefore to filter out the noise/background points on the objects identified by the instance segmentation module.

3.3 Mesh deformation based foreground depth prediction

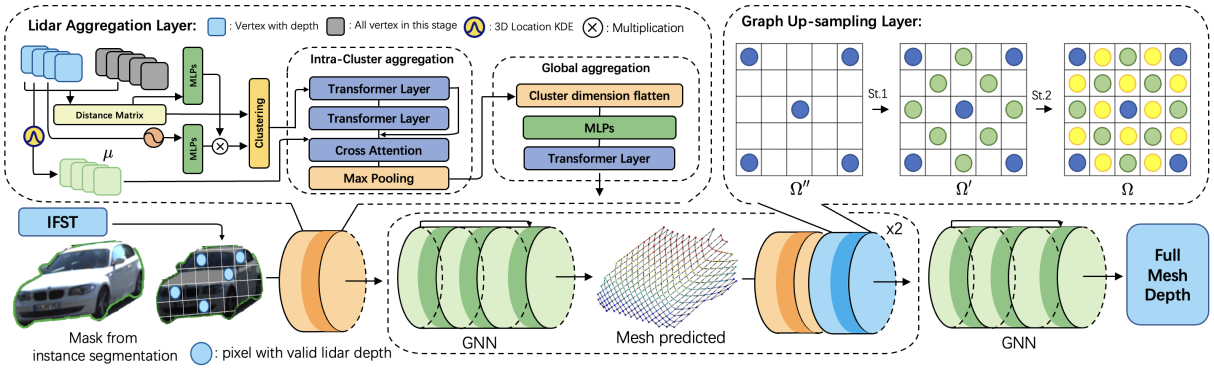


Figure 3.5: The design of MDCNet: After obtain foreground points from IFST, the relative distance between depth empty vertex(pixel without depth) and valid vertex used to weight the depth by direct mutiplocation after embedding. With the relative distance, an intra-cluster aggregation and global aggregation followed to estimate the depth representation of specific pixels better. The GNN next used to propagate the features among vertex in the specific stage. Every pixel obtain the estimated depth after 3 stage up-sampling.

The existing depth completion models always suffer from boundary depth dispersion problem, which is actually due to the tendency of convolutional networks to smooth the

signal. Here we regard the pixel for which depth needs to be estimated as the vertices of a deformable mesh in 3D space.

The specific network structure of this module is described in Figure 3.5. Inspired by recently proposed depth completion networks [43, 88], the core of our Mesh Depth Completion Network (MDCNet) design is a geometric position-based hierarchical Transformer that allows the model to learn from points at different locations while still maintaining a strong prior: to estimate the depth of a specific pixel, neighboring lidar points in 2D space are more referential. It is the role of the aggregation layer to make the network learn from real lidar point distribution and densify the mesh in an iterative manner.

Based on the previous work [4,68], we also adopt a coarse-to-fine strategy to get the final shape. Specifically, it is difficult to do depth prediction for all pixels at once, considering that only a small fraction of pixels possess depth. Here we first predict a subset of all pixels and gradually recover the full shape: Given the 2D location of mask region $\Omega \in \mathbb{N}^{m\times 2}$, we first downsample the dense pixels to $\frac{1}{2}$ and then $\frac{1}{5}$ of the original by using the reverse process of the graph up-sampling layer presented in Figure 3.5, to get Ω' and Ω'', $|\Omega''| = 0.2m$. At the first stage, we try to use the pixel with valid depth from lidar sensor to predict depth for every pixels in Ω''. Then after going through another several layers of GNN to propagate the features, we use lidar aggregation layer to predict depth for Ω'. We now consider all the predicted depths in the previous stage as valid depths. After repeating the above process several times, we can obtain depth estimates for all pixel points.

Technically, with the points $p' \in \mathbb{N}^{t\times 3}$ filtered by IFST, their pixels location $\Omega_L \in \mathbb{N}^{t\times 2}$ can be calculated by the camera internal and external parameters. Ω^L should be a subset of Ω, with $\left|\Omega^L \cap \Omega''\right| \geq 0$. Let the depths of these points denoted by $p'_d \in \mathbb{R}^{+t\times 1}$, in this (first) stage, the depth of pixels $\Omega'' - \Omega^L$ need to be estimated by the obtained p'_d and Ω_L.

Here, we propose an explicit local-to-global feature aggregation strategy to estimate the depth of the pixels on a mask. In general, given a specific pixel using Ω''_i to represent its 2D location, we first sort the t pixels locations Ω_L by $\|\Omega''_i - \Omega_L\|_2$, i.e. the Euclidean distance. We then estimate the KDE embedding μ for $p\prime$, which will later used to guide the aggregation. Next, we calculate the distance matrix between different pixels with and without lidar depth and process them with a MLP. This embedded 2D distance will be multiplied by the lidar features to get the relative location-weighted 3D features . This process can be described by Equation (3.9). $\mathcal{F}_b \in \mathbb{R}^{m\times t\times c}$, c is features dimension. Further, using the distance matrix, we divide the features into η chunks, denoted by $\mathcal{F}_{bi} \in \mathbb{R}^{m\times \frac{1}{\eta}t\times c}$, to explore the intra-cluster relationship.

$$\mathcal{F}_b = MLPs(Embed(p'_d)) * MLPs(\|\Omega'' - \Omega_L\|_2) \tag{3.9}$$

For features in different chunks, we apply 2 layers of transformer encoder along the first dimension of $\mathcal{F}_b$ as shown in Equation (3.10). The stack operation denoted by $\sum$. This process allows the network to find a better representation for lidar in different distance bins. $\mathcal{F}_{bi}$ will then be used as a query for the attention matrix calculated by the KDE feature of corresponding lidar points.$\oplus$ in equation 8 stands for matrix multiplication.

$$\begin{aligned} \mathcal{F}_{bi} &= \sum_{j=0}^{s} SelfAtt(\mathcal{F}_{bij}) \quad \mathcal{F}_{bij} \in \mathbb{R}^{1\times\frac{1}{\eta}t\times c} \\ \mathcal{F}_{bi} &= sigmoid(MLPs(\mu_i) \oplus MLPs(\mu_i)) \oplus \mathcal{F}_{bi} \end{aligned} \tag{3.10}$$

At the end of cluster-wise aggregation, we dilute the intra-cluster feature for each vertex by max pooling to get new $\mathcal{F}_{bi} \in \mathbb{R}^{m\times\frac{1}{\eta}\times c} \rightarrow \mathcal{F}'_{bi} \in \mathbb{R}^{m\times 1\times c}$, so for all η cluster, the feature before global aggregation is $\mathcal{F}'_b \in \mathbb{R}^{m\times\eta\times c}$

For global aggregation, after the flattening in the last 2 dimensions of $\mathcal{F}'_b$, the dimension of representation for a different vertex is $|\eta * c|$, which corresponds to the feature learnt from all clusters. The final transformer block was added to allow the network to learn features from other vertex directly, instead of by multiple neighborhood propagation in later GNNs. Note that the hop of our network is quite larger than the classic scenario for GNNs, e.g. a social network or recommender systems, in which hop is around 6-10 [13,29]. In our scenario, the distance from vertex to vertex may need over 200 hops (proportional to the number of pixels in the mask), this design actually provides a short-cut for the vertex to exchange features.

The following GNNs aggregate and pass the features to the neighbor of every vertex. In our design, to maintain the stability of gradient flow, and for every 2 layers of GNNs, we add a residual connection. We leverage spectrum-free graph convolutions following [4]. Given the feature on vertex $\mathbf{f}$ and its neighbor $\mathcal{N}(i)$, the specific design is shown in Equation (3.11).

$$\mathbf{f}'_i = \frac{1}{1+|\mathcal{N}(i)|}\left[\mathbf{W}_0\mathbf{f}_i + \mathbf{b}_0 + \sum_{j\in\mathcal{N}(i)}(\mathbf{W}_1\mathbf{f}_j + \mathbf{b}_1)\right] \tag{3.11}$$

where $\mathbf{W_0}$ and $\mathbf{W_1}$ are learnable parameters for the vertex itself and its neighbours. After 6 GNN layers, a regression head is used to predict the depth of every vertex on the mesh, i.e. the Ω''. This process will be iteratively repeated 3 times with Ω'', then Ω' and Ω to obtain a dense depth for the object.

3.4 Loss function and summarize

The losses of the proposed modules can be divided into 2 parts, loss for segmentation and loss for mesh regression. Specifically, lidar segmentation is here a binary classification task, and a simple BCE Loss is adopted to provide guidance as shown in Equation (3.12), with y_i being denote the label of p_i.

$$L_{seg} = \frac{1}{n}\sum_{i=0}^{n} y_i \cdot \log \sigma\left(x_i\right) + \left(1 - y_i\right) \cdot \log\left(1 - \sigma\left(x_i\right)\right) \tag{3.12}$$

The mesh regression loss is composed of the location loss and mesh shape loss. We combine the MSE losses in all different stages as the location loss as shown in the first term of L_{mesh} in Equation (3.13). The λ_i in the early stage will be higher. For the shape loss, N represents the number of vertices in the mesh. The second and third items in L_{mesh} aim to control the length of the edges in the predicted mesh and provide consistency among the normals of adjacent faces. This approach effectively prevents the occurrence of a long tail problem in the estimated points. The *loc* denotes the predicted 3D depth of a specific vertex, and n_i represents the normal vector of the triangular plane on the mesh. To balance the different loss terms, we introduce ω_1, ω_2, and λ_m.

$$\begin{aligned}
L_{mesh} &= \sum_{i=1}^{3} \lambda_i MSE(loc, \hat{loc}) + \omega_1 L_{edge} + \omega_2 L_{con} \\
L_{con} &= \frac{1}{N}\sum_{i=0}^{N} 1 - cos(n_i, n_j), \quad j = Neighbour(i) \\
L_{edge} &= \frac{1}{N}\sum_{i=0}^{N} \left\| loc_i, loc_j \right\|_2, \quad j = Neighbour(i)
\end{aligned} \tag{3.13}$$

$$L = L_{seg} + \lambda_m L_{mesh} \tag{3.14}$$

In this section, we proposed 2 modules named IFST and MDCNet to help recover the foreground objects. The IFST will take the points which can be projected to the 2D mask as input, to remove the noise and keep the points only belong to the objects. The MDCNet followed will use this part of points as input, try to diffuse the sparse depth information to get a densified foreground objects cloud points. The goal of our proposed method is densify the objects to help the downstream detection tasks. In the next chapter, by combined with

our proposed method, we will show that most of lidar-based detector can stand on the same level of the SOTA cross-modality detector.

Chapter 4

Experiments

4.1 Datasets and Metrics

The KITTI [15] 3D object benchmark is one of the most famous datasets in autonomous driving perception. We follow the setting in previous works [63] that split the training part into 3712 and 3768 samples as training and validation sets. In the following content, most experiments will be reported on the KITTI validation set. A sample for KITTI are shown in the Figure 4.1.

Figure 4.1: Sample from KITTI dataset: All kitti data were collected during sunny daytime and the training set totaled 7480 frames of labeled data

For each frame, kitti provide a 24-bit RGB image from the front camera, which solution is 375 pixels in height and 1242 pixels in width and a calibrated laser radar file. The lidar signals were collected by a 64-beam Velodyne HDL-64 sensor operating at 10 HZ.

Compared to the former, nuScenes [5] is a benchmark dataset of a larger scale, which provides 10 times more training data than KITTI in the form of continuous frame labeling. Considering that information from all directions is important during vehicle operation, nuScenes provides a surround view from five cameras, also presented as 24bit RGB images. Further, the data provided by nuScenes also takes into account the different road conditions, while like KITTI it is collected during clear daylight hours. In the nuScenes dataset, 19.4% of the data was collected in rain and 11.6% in low lighting conditions. This makes the data distribution closer to that of real driving conditions and places higher demands on the detectors. Specifically, nuScenes includes more than up to 5.5 hours of labelled data collected in 1,000 scenarios. The granularity of these annotations is essentially at the object level and includes the 3D frame position, velocity, and category of the object. It is worth noting that the nuScenes data contains 23 categories of objects, far more than the 3 in kitti, which is more reflective of the complexity of real-world autopilot scenarios and also makes detection more difficult.

Unlike KITTI's 64-beam lidar, in nus we use a lower 32-beam lidar. In other words, its only contains about half of the depth information in KITTI in each frame of data. For the splitting of the training and validation sets, we follow the dataset spliting in OpenPCDet [63] to decide the training and validation series of data.

Figure 4.2: A frame from nuScenes dataset: Each frame contains a circular view of the surroundings consisting of six images, (here the image from back camera is removed)

The performance of 3D detectors augmented with WYSIWYD generated pseudo points(to complete the objects) are tested on these 2 datasets.

For the generation of the visible part ground truth, note that the nuScenes only provide a 2D instance mask for nuImage, so we pretrained the segmentation model on nuImage, and performed inference on nuScenes. Since these masks are not accurate enough, our MDCNet has only been trained on the KITTI visible part ground truth. We then performed a zero-shot inference to complete the objects in nuScenes.

Evaluation metrics For the KITTI part we report results using average precision under 40 recall thresholds and 0.7, 0.5 IOU thresholds for cars and pedestrians respectively. Specifically, in KITTI, to define a positive and negative prediction, first the IOU between

the estimated 3D boxes and the ground truth and needed. The IOU can be calculated by the following equation, the G_B and $\hat{B}$ stand for ground truth and estimated boxes.

$$IOU = \frac{\hat{B} \cap G_B}{\hat{B} \cup G_B} \tag{4.1}$$

With the obtained IOU score, we use different threshold here define positive and negative prediction: for cars and pedestrians, we adopted 0.7 and 0.5 as threshold. Given the threshold and IOU, we can calculate the true positive(TP), false positive(FP) and false negative(FN) in the confusion matrix. Then the precision can be obtained by the equation 4.2.

$$Precision = \frac{TP}{TP + FP} \tag{4.2}$$

Finally, to get the Average Precision value, we need to introduce the confidence score and modify the mentioned principle a little bit. Here we first need categorize the confidence level of the 0-1 interval into 40 intervals, for each intervals, we use the upper boundary as the threshold to further define the positive samples. For an example, for the first interval, which start from 0 and end in 0.025, we use 0.025 as an threshold value. If the model think the object in the box is a car, only when its confidence score is above 0.025 and its IOU with ground truth is above 0.7, we categorize this estimation as a positive sample.

After calculate all predictions generated for different frame, we can get the precision as mention before. And after we calculate the precisions for all intervals, these values can be connected by a line, and the area enclosed by this line and the XY coordinate axis is considered as Average Precision(AP). Further, because we used 40 intervals, its also known as AP_{R40}, which will later be used in all tables in this chapter. In order to more accurately characterize the performance of the algorithms, the official evaluation script provided by KITTI has been changed from AP_{R11} to AP_{R40} several years ago.

For the nuScenes, we follow the official evaluation protocol to evaluate the accuracy: nuScenes detection score (NDS) which consists of average translation error (ATE), average scale error (ASE), average orientation error (AOE), average velocity error (AVE), and average attribute error (AAE). The NDS can be calculated as follows:

$$\text{NDS} = \frac{1}{10}\left[5\text{mAP} + \sum_{\text{mTP}\in\mathbb{TP}} (1 - \min(1, \text{mTP}))\right] \tag{4.3}$$

Here the mTP refers to the ATE, ASE, AOE, AVE and AAE and the method to obtained mAP in (4.3) is a little different from what mentioned in KITTI. Specifically,

the Euclidean distance between the center of the prediction box and the center of the ground truth in the BEV plane is used here to evaluate positive and negative samples. The definitions of ATE, ASE, AOE, AVE and AAE are listed below.

- ATE: Average Translation Error, Euclidean center distance in BEV plane. (meters)
- ASE: Average Scale Error, 3D intersection over union (IOU) after aligning orientation and translation (1 - IOU)
- AOE: Average Orientation Error, the smallest yaw angle difference between prediction and ground truth.
- AVE: Average Velocity Error, the absolute velocity error as L2 norm of the velocity differences in BEV (meters).
- AAE: Average Attribute Error, 1 minus attribute classification accuracy (1 - acc).

Compared to the 3D IOU we used in kitti, such an approach decouples the effect of the size and orientation of the object on the AP computation.

The accuracy for lidar segmentation are measured by mIOU, which is calcualted from TP, FP and FN as shown in Equation 4.4, c is the number of category. Here $c = 2$, since we only split foreground from background points.

$$meanIOU = \frac{1}{C}\sum_{c=1}^{C}\frac{\mathrm{TP}_c}{\mathrm{TP}_c + \mathrm{FP}_c + \mathrm{FN}_c} \tag{4.4}$$

4.2 Main Results

In this section, the experimental setup and related details are first introduced. Then we give a comparison between baseline models combined with WYSIWYD and previous SOTA solutions on both KITTI and nuScenes. Our code has been developed using the OpenPCDet toolbox [63].

4.2.1 Implementation detail

In this thesis thanks to the lightweight design of the module, all training is done by a single RTX3090. For the training of the baseline models, we used batches equal to 8 for 80

epochs training and other settings remain as in available implementation. For the training of MDCNet, we used a batch size of 4 and an Adam optimizer with a learning rate of 3e-5 for the first 15 epochs and 1e-5 for the remaining 25 epochs. The α in Equation 3 is set to 1.2 if the number of pixels is less than 2000, else 1.05. The ω_1, ω_2, and λ_m in the loss function are set to 2.0, 2.0, and 1.0 respectively. For the IFST, we trained it with 30 epochs before combining it with MDCNet to get a faster convergence in the early period of training and prevent gradient explosions.

For the 2D instance segmentation part, we used E2EC [84] for KITTI and HTC [7] for nuScenes to get the best balance between efficiency and accuracy. Considering that the number of other objects is quite limited and some categories of the objects in nuScenes are not available in nuImage, we only report the results of pedestrians and cars detection.

When we designed the network, we considered different settings in the self-attention Layer. We found that in our network, a sigmoid activation function outperforms the Softmax in the original design [65] and the combination of Conv-LayerNorm instead of Linear projection in the calculation of K, Q, V accelerates network convergence. So in IFST and MDCNet, the mentioned design is used to replace all self-attention operations. However, for the cross attention, the Softmax function is kept unchanged. Specific comparisons will be shown in the ablation study section.

In addition to the above-mentioned scheme of using 2D mask labels to generate visible part ground truth, we also use masks from E2EC [84] predictions. The input of IFST is the points filtered by 2D masks, however, if we only use label masks in the training process, the noisy points will be very sparse. In consequence, when we infer the model on the mask provided by e2ec in real cases, the noise points can hardly be identified. Note that in our model training. we partly use masks from e2ec as long as the IOU between e2ec's predicted mask and the true mask is greater than 0.7.

4.2.2 Performance Comparasion

In Table 4.1 and 4.2, we combine the proposed method with most of the available code baseline models and compare them with the SOTA solution. Here we didn't use any GT sampling when training the baseline model with WYSIWYD, however, for a more convincing comparison, we still remain in this step when retraining the baseline models themselves.

For all KITTI's results we have tested 3 times and in Table 4.1 the median values are mentioned. For the in nuScenes dataset, we obtained the results from only one run

considering the computational resources required. All results were obtained with fixed random seeds.

On the KITTI side, compared with the baseline detector, the proposed MDCNet and IFST provide improvements from 1.29% to 10.4% in 3D detection. For Voxel-RCNN, a 12.2% percent improvement in BEV detection is observed over the original performance. Furthermore, when combined with WYSIWYD, Voxel-RCNN achieves the SOTA pedestrian 3D detection model and surpasses all previous best models by 1.48%, 2.45%, 2.97%. For all other models, our method also brings significant

Table 4.1: Illustration of performance improvement brought by WYSIWYD on baseline models and the comparison with state-of-the-art solutions on **KITTI validation set**. The best results from augmented baseline models and SOTA methods are shown in bold. L: lidar, L+I: lidar and image. The best performance in different categories is marked in red. Here we use GT sampling in all baseline models but cancel this augmentation when combine them with WYSIWYD. we retrained all listed models if the code is available.

Methods	Reference	Modality	With WYSIWYD	Car 3D AP_{R40}			Car BEV AP_{R40}			Ped. 3D AP_{R40}			Ped. BEV AP_{R40}		
				Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard
CenterPoint [80]	CVPR21	L	✗	89.83	78.87	75.79	92.35	87.86	85.26	48.63	46.18	42.09	53.42	51.32	47.81
		L+I	✓	90.48	79.60	76.93	94.14	88.97	86.65	66.53	62.11	58.07	73.02	69.68	64.83
Point-RCNN [59]	CVPR19	L	✗	91.99	80.26	78.04	92.93	87.79	84.63	65.96	57.98	49.86	69.00	60.92	52.80
		L+I	✓	91.23	80.86	74.12	95.63	87.94	83.13	74.34	65.37	56.48	78.01	68.88	59.81
PV-RCNN [58]	CVPR20	L	✗	92.02	84.52	82.45	92.94	90.74	88.59	67.52	60.41	55.23	69.76	63.49	58.85
		L+I	✓	92.45	85.76	83.34	95.58	91.68	89.64	70.38	66.57	61.93	75.43	71.02	66.23
Voxel-RCNN [8]	AAAI21	L	✗	92.75	85.30	82.94	95.80	91.35	88.99	66.88	59.94	54.16	69.62	63.02	58.02
		L+I	✓	92.60	85.84	83.43	95.64	91.82	89.58	**75.56**	**69.38**	**64.56**	**80.10**	**75.25**	**68.70**
Part-A^2 [60]	TPAMI20	L	✗	91.72	83.08	80.45	94.51	90.41	88.23	67.32	60.32	54.49	69.80	63.10	58.15
		L+I	✓	92.64	83.76	81.27	**95.94**	89.86	89.22	68.61	64.25	59.16	75.13	70.54	65.38
CAT-Det [85]	CVPR22	L+I	N/A	90.12	81.46	79.15	-	-	-	74.08	66.35	58.92	-	-	-
SFDNet [72]	CVPR22	L+I	N/A	**94.99**	88.16	85.72	95.80	91.80	91.41	72.94	66.69	61.59	75.64	69.71	64.70
VFF-PVRCNN [38]	CVPR22	L+I	N/A	92.55	85.54	83.09	95.37	91.33	90.74	72.18	65.01	60.11	77.01	69.39	64.72
VirConv-T [71]	CVPR23	L+I	N/A	94.98	**89.82**	**88.01**	95.42	**93.82**	**91.60**	73.32	66.93	60.38	73.32	66.93	60.38
LoGoNet [37]	CVPR23	L+I	N/A	92.04	85.04	84.31	93.08	90.79	90.55	70.20	63.72	59.44	74.29	66.93	63.70

Table 4.2: Performance Comparsion on **nuScences validation set**. Only the accuracy of pedestrians and cars is accounted for here. Best result are shown in bold. ↑ higher is better, ↓ lower is better.

Methods	Reference	Modality	With WYSIWYD	mAP↑	NDS↑	mATE↓	mASE↓	mAOE↓	mAVE↓	mAAE↓
CenterPoint [80]	CVPR2021	L	✗	0.860	0.834	0.155	0.214	0.266	0.192	0.138
		L+I	✓	0.875	0.842	0.149	0.211	0.269	**0.191**	**0.136**
SECOND [78]	SENSORS2018	L	✗	0.813	0.785	0.165	0.215	0.223	0.454	0.154
		L+I	✓	0.836	0.811	0.157	0.215	0.241	0.309	0.153
VoxelNext [9]	CVPR2023	L	✗	0.860	0.832	0.158	0.212	0.273	0.189	0.144
		L+I	✓	**0.892**	**0.852**	**0.140**	0.210	**0.231**	0.211	0.144
TransFusion [3]	CVPR2022	L+I	N/A	0.891	0.850	0.146	**0.208**	0.232	0.214	0.149
BEVFusion [47]	ICRA2023	L+I	N/A	0.885	0.849	0.146	0.216	0.223	0.205	0.144

improvements in pedestain detection. Under the 0.7 IOU threshold, we also observe a SOTA performance in Car BEV detection when testing Part-A^2. On the nuScenes side, a wide performance improvement is also witnessed on nuScenes performance. Specifically, when combine VoxelNext with the WYIWYD augmented points, we obtain 3.2% and 2% improvement in MAP and NDS, which makes this baseline model exceed the latest BEV perception methods.

In order to compare the performance improvement by WYSIWSYD more intuitively, we compare the performance of the current SOTA solution with that of the combined baseline+WYSIWSYD in Figure 4.3. Here we separate the accuracy of the model on the detection of pedestrians as the Y-axis and the effect on the detection of Cars as the X-axis. The accuracy of the model before and after combining wysiwyd is plotted in the left and right subplots of 4.3.

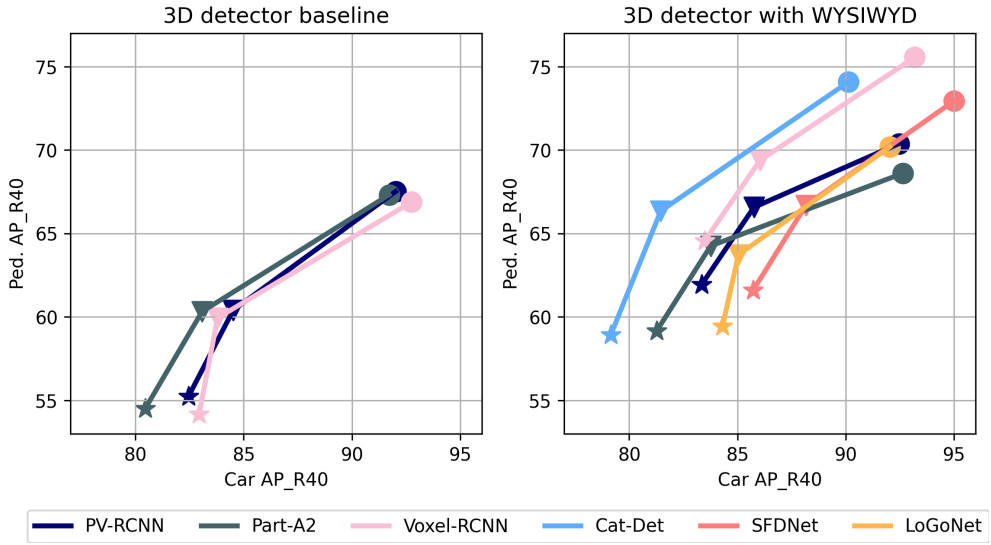


Figure 4.3: Performance improvements brought by our proposed modules on KITTI 3D val. We compare the performance of the baseline model after combining WYSIWYD with that of today's SOTA solution in the right figure. Without bells and whistles, by combining with our model, most baseline models can stand on the same level as today's SOTA solutions. Stars, triangles, and circles in the figure stand for detection under hard, moderate, and easy categories.

We also compare the proposed solution with the previous SOTA detector-independent lidar completion methods in Table 4.3 to demonstrate more salient properties of our method. Specifically, we directly refer to the data in SPG [77] and UYI [89], while for

BTC [76], we used the completed point cloud output from the completion network as the input to different baseline detector models. The BtcDet only released configuration on car detection training, therefore the pedestrian items are not reported in the Table. Our proposed method also provides the highest performance improvement in all 3 pedestrian detection categories and most car detection categories when compared to the evaluated methods.

Table 4.3: Comparison of the performance improvement by completion methods proposed recently with our solution. The IOU Thresholds are 0.7 and 0.5 for cars and pedestrians. The highest improvement marked in bold. $*$: SPG paper did not provide data on PartA2 and its code is not available.

Baselines	Completion Method	Car 3D AP_{R40}			Ped. 3D AP_{R40}		
		Easy	Mod.	Hard	Easy	Mod.	Hard
PV-RCNN [58]	Ori	92.02	84.52	82.45	67.52	60.41	55.23
	SPG [77]	+0.43	+0.95	+0.34	+2.35	+2.14	+2.47
	BTC [76]	+1.88	+0.77	+1.12	-	-	-
	UYI [89]	**+1.90**	+1.12	+0.68	+3.45	+0.91	+0.18
	Ours	+0.43	**+1.25**	**+0.89**	**+7.91**	**+6.16**	**+6.70**
Part-A^2 [60]	Ori	91.72	83.08	80.45	67.32	60.32	54.49
	SPG* [77]	-	-	-	-	-	-
	BTC [76]	+0.05	+1.02	+0.66	-	-	-
	UYI [89]	+0.23	**+1.98**	+0.77	+0.53	+0.71	+0.78
	Ours	**+0.92**	+0.68	**+0.82**	**+1.29**	**+3.93**	**+4.67**
PointPillars [35]	Ori	87.75	78.39	75.18	57.30	51.41	46.87
	SPG [77]	+2.02	**+2.97**	+3.67	+2.35	+2.14	+2.47
	BTC [76]	+1.66	+2.77	+1.03	-	-	-
	UYI [89]	+0.30	+0.40	+0.35	+0.30	+2.21	+2.26
	Ours	**+2.89**	+2.33	**+4.91**	**+4.62**	**+5.09**	**+8.77**

As mentioned before, to ensure the accuracy of the results, on the kitti dataset we repeated the training 3 times for all the results and listed the median values in Table 4.1, and here we show them with the data in Table 4.4, which reacts to fluctuations of the training results under the same random seeds. As we can see in the Table 4.4, most of the results are in fact quite stable. In contrast, given the small sample size of pedestrians, the fluctuations in their performance here are greater than those for car.

Table 4.4: Upper and lower ranges for multiple tests on the kitti dataset.

Baselines	Car 3D AP_{R40}		
	Easy	Mod.	Hard
PV-RCNN [58]	(-0.176, +0.051)	(-0.004, +0.122)	(-0.188, +0.121)
Part-A^2 [60]	(-0.116, +0.071)	(-0.021, +0.089)	(-0.002, +0.032)
PointPillars [35]	(-0.006, +0.043)	(-0.029, +0.091)	(-0.162, +0.033)
Voxel-RCNN [10]	(-0.077, +0.019)	(-0.106, +0.031)	(-0.101, +0.001)
Baselines	Ped 3D AP_{R40}		
	Easy	Mod.	Hard
PV-RCNN [58]	(-0.092, +0.111)	(-0.216, +0.221)	(-0.246, +0.351)
Part-A^2 [60]	(-0.145, +0.271)	(-0.122, +0.179)	(-0.322, +0.479)
PointPillars [35]	(-0.245, +0.351)	(-0.333, +0.219)	(-0.142, +0.259)
Voxel-RCNN [10]	(-0.077, +0.019)	(-0.281, +0.119)	(-0.312, +0.419)

4.3 Ablation study

Vehicle 3D Detection Analysis In Table 4.1, we noted the WYSIWYD brings less significant gains in car detection and there is even a decrease in some cases, compared with that of pedestrians. We attribute this to the detector independence of the proposed method. The added complementary point cloud is not perfect, as shown in Fig. 4.5, and from time to time the point cloud boundary exceeds the 3D GT box due to the inaccuracy of the 2D detection. Considering there is no specific design in the downstream detector to filter this noise, this seriously affects the performance of 3D detection under the 0.7/0.7/0.7 thresholds. However, if the thresholds are relaxed to 0.7/0.5/0.5, as shown in Table 4.5, the combination of Voxel-RCNN+WYSIWYD remains optimal in terms of performance. So in this way, the utilization of the proposed model can essentially mitigate the miss detection and low IOU detection problems.

Table 4.5: The Car 3D detection AP comparison under 0.7/0.5/0.5 thresholds. To provide a fair comparison, we report the best result in the Moderate categort of all models in the last 20 epochs

Models	Car 3D AP_{R40}		
	Easy	Mod.	Hard
PV-RCNN+WYSIWYD	98.89	97.49	95.45
Part-A^2+WYSIWYD	98.82	95.62	95.15
Voxel-RCNN+WYSIWYD	99.03	**97.92**	**95.68**
VFF+PV-RCNN	98.51	96.51	94.41
LoGoNet	98.48	96.50	94.44
SFDNet	**99.32**	97.04	95.01

As we mentioned in the previous section, since the cross-modality restrain, we cancel the GT-sampling in training process. However, this strategy in fact plays an important role in preventing overfitting. Here we compare the original baseline with the one without the GT sampling augmentation, to further illustrate the improvment bring by WYSIWYD. In Table 4.6, a more obvious improvements can be observed in both pedestain or car detection

Table 4.6: Performance comparsion among basline with/without GT-sampling and with the proposed methods. G: with GT-sampling, W: with WYSIWYD

Baselines	Category	Aug.	Car 3D AP_{R40}			BEV 3D AP_{R40}		
			Easy	Mod.	Hard	Easy	Mod.	Hard
VoxelRCNN	Car	✗	92.14	83.00	80.72	95.14	89.29	88.97
		G	92.75	85.30	82.94	95.80	91.35	88.99
		W	92.60	**85.84**	**83.43**	95.64	**91.82**	**89.58**
	Ped.	✗	65.21	57.87	54.23	69.62	63.77	58.23
		G	66.88	59.94	54.16	69.62	63.02	58.02
		W	**75.56**	**69.38**	**64.56**	**80.10**	**75.25**	**68.70**

Inference Speed comparison Another feature that deserves to be pointed out is the real-time nature of our algorithm. In Table 4.7 we compare its inference time with the

previous best-performing models on KITTI. In [71, 72], the time reported is not CUDA synchronized, which means the next frame can actually be processed when the GPU is available, as mentioned in [24]. When this is taken into account, the mentioned method will need more than 200 ms for single-frame inference. However, in our model, thanks to the fact that MDCNet is designed to only complete foreground points, compared to VirConv, the proposed method brings a 34.4 % efficiency improvement.

Table 4.7: Comparision of inference time in pseudo-points based completion. PENet [22] is used as completion network as mentioned in SFDNet and VirConv paper. The Voxel-RCNN is used as the baseline. The inference speed was tested on 1 single piece of RTX2080.

Modules	Inference Speed(ms)	Global Completion	Completion Time (if global)	Overall Time(ms)
SFDNet	66	✓	161	227
VirConv	60	✓	161	221
Ours	145	✗	N/A	145

Specifically, the reported 145ms in the Table is composed of the forward propagation time of Voxel-RCNN 44ms, the forward propagation time of E2EC 43ms and the 58ms of WYSIWYD.

Conditional Analysis In addition, in order to explore in which scenarios our proposed method brings greater improvements, we analyzed the performance gain on Voxel-RCNN and PV-RCNN using distance and occlusion degree as indicators. As shown in Table 4.8,

Table 4.8: 3D detection performance at different level of distance and different occlusion. The thresholds for IOU and other settings are same as mentioned in Table 2.

Baselines	Category	With WYSIWYD	Distance(m)			Occlusion		
			0-20	20-40	40-Inf	0	1	2
VoxelRCNN	Ped.	✗	62.11	34.36	1.04	60.14	19.39	4.59
		✓	68.26	51.42	7.46	70.82	34.01	10.39
	Improvement		+6.15	+17.06	+6.42	+10.68	+14.62	+5.80
	Car	✗	90.78	78.19	28.95	74.37	69.90	55.11
		✓	91.03	78.70	36.12	76.49	70.89	54.47
	Improvement		+0.25	+0.52	+7.17	+2.12	+0.99	-0.37
PV-RCNN	Ped.	✗	60.81	30.94	0.62	55.68	15.23	4.56
		✓	65.77	50.36	12.21	67.54	30.22	9.42
	Improvement		+4.96	+19.42	+11.59	+11.86	+14.99	+4.86
	Car	✗	90.43	77.63	29.69	73.79	69.04	53.22
		✓	90.48	78.85	38.32	77.65	70.21	54.07
	Improvement		+0.05	+1.22	+8.63	+3.64	+1.17	+0.85

To further explore the detail of the performance of our method, we split the detection results into 3 bins by distance and different occlusions as marked by KITTI. The results are shown in Table 4.8. In addition to this, we also compared the inference time between the proposed completion method and the previous pseudo-point-based solution in Table 4.7.

Table 4.9: Effect of different designs in WYSIWYD on KITTI val object mesh prediction. The downstream 3D detector selected is PV-RCNN in this experiment. The results of using 2D detection and 3D segmentation labels are also reported.

Modules	Cluster Aggregation	KDE Feature	Label	Mesh MSE Loss	Car 3D AP_{R40}			Car BEV AP_{R40}			Ped. 3D AP_{R40}			Ped. BEV AP_{R40}		
					Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard
WYSIWYD			N/A	489.3	91.12	82.00	82.61	92.01	88.75	89.09	63.02	61.78	56.72	72.40	68.80	63.02
WYSIWYD	✓		N/A	311.9	92.18	84.71	82.99	94.61	92.10	88.72	69.22	63.95	60.76	74.03	70.18	64.65
WYSIWYD	✓	✓	N/A	283.4	92.45	85.76	83.34	95.58	91.68	89.44	70.38	66.57	61.93	75.43	71.02	66.23
WYSIWYD	✓	✓	2D	277.3	93.24	89.24	86.54	95.79	92.98	90.53	74.36	69.99	65.50	78.01	73.66	70.61
WYSIWYD	✓	✓	2D+3D	275.9	94.69	90.06	87.74	95.98	93.06	92.93	78.11	72.98	68.82	80.51	75.99	73.31

As shown in this Table, our approach has a substantial improvement for the detection of distant objects: we can bring up to 19.42% performance improvement for objects in the range of 20-40 meters. Even for targets at more than 40 meters away from the camera, we achieve at least 6.42% performance improvement.

Table 4.10: Effect of different design in IFST on KITTI val lidar segmentation. For the IFST case, we adopt a vanilla Transformer with mentioned modification in projection layer and activation function

Modules	Mask Size	Lidar 2D location	Neighbour Embedding	Sinusoidal Embedding	mIOU(%)
PointNet++	N/A	N/A	N/A	N/A	78.44
IFST					83.87
IFST	✓				84.35
IFST	✓	✓			87.27
IFST	✓	✓	✓		88.37
IFST	✓	✓	✓	✓	89.02

IFST design verification Here we decomposed the module in IFTS and verified its effectiveness in Table 4.10. By adding of 2D mask size feature, lidar 2D location feature, Local embedding layer, and Sinusoidal Embedding, we get 0.48%, 2.92%, 1.10%, 0.65% improvement. In summary, compared to the vanilla PointNet++ [55] which is widely used in intra-frustum segmentation, IFST offers 10.58% performance improvement in terms of mIOU.

Ablation of MDCNet design In Table 4.9 In Table 4.9 we show the direct relationship between generated mesh quality and detection result. When adding local-global aggregation and KDE guide, we can obtain 3.76% and 4.79% performance increase in PV-RCNN, for car moderate detection and pedestrian moderate detection respectively. In this process, the MSE mesh loss continues to fall from 489.3 to 283.4.

Considering that our method requires both 2D instance segmentation and 3D points segmentation, we also report the performance of using the two labels directly in the last 2 lines of Table 4.9. As can be seen, our method still has great potential: a better 2D segmentation is enough to improve the performance by another 4.3% to 6.4% percent in terms of moderate 3D detection.

Transformer Layer design In the Figure 4.4, we show the role of normlized projection layer and sigmoid activation function in the convergence rate of IFST and MDCNet. As

shown, these design not only accelrate the training, but also improve the final accuracy from 2% to 9%.

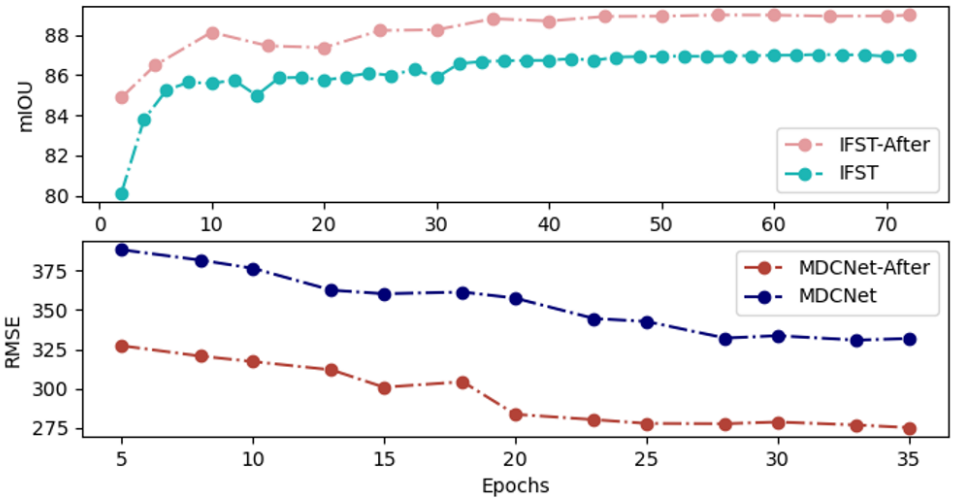
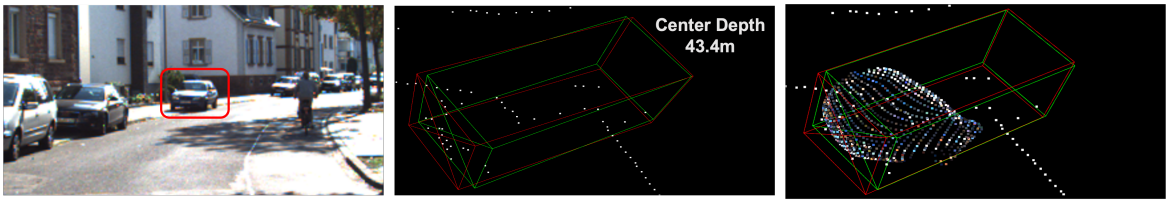


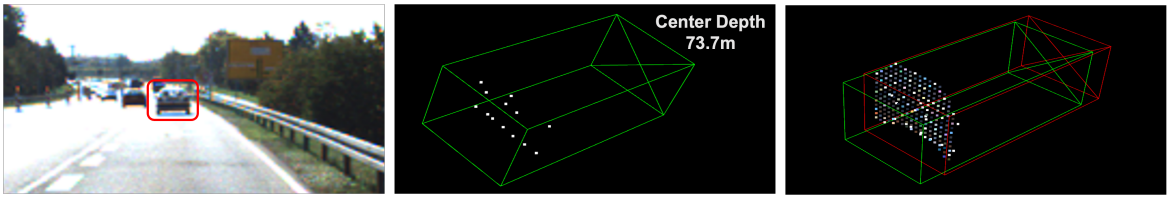
Figure 4.4: Comparsion of proposed module with different self-attention layer design and activation function

In order to illustrate the completion points brought by WYSIWYD more intuitively, we visualize some examples of obvious performance improvements brought by it in Figure 4.5.

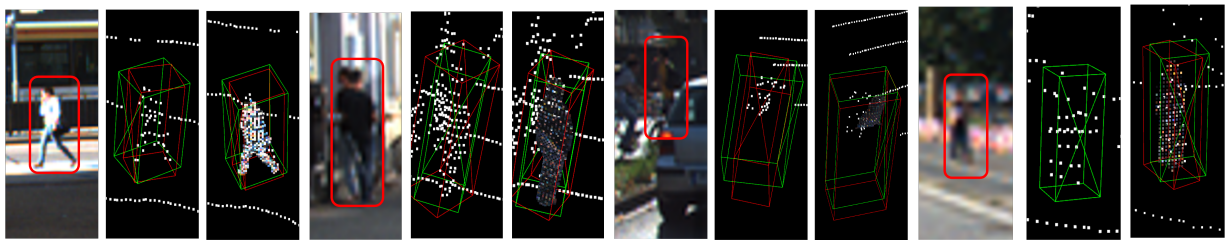
For Figure 4.5(a) and Figure 4.5(b), we report a better boundary box estimation in terms of IOU and missing detection. The proposed model recovered the missing details in the visible part in a mesh-deformation manner, which is especially important when the lidar data is extremely sparse, as shown in Figure 4.5(b). Figure 4.5(c) shows the impact of WYSIWYD added points on pedestrian detection. From left to right, in the first 3 sets of images, we observe an improvement in the estimation of the bearing angle. Furthermore, the last 3 images prove that the miss-detection issue can be eliminated by adding points as well.



(a) Result for WYSIWYD augmented car detection



(b) Improvement for distant car miss detection



(c) Result for pedestrian detection: Angle correction and Missing detection restoration

Figure 4.5: Qualitative verification for effectiveness of the proposed method. For each set of three images, the object positions in the RGB image, the detection results of the original method, and the WYSIWYD augmented point cloud detection results are shown from left to right. 3D Ground truth and predicted box are shown in green and red, respectively.

Chapter 5

Conclusion

In this work, we start with recent work on multimodal detection, and by analyzing their intrinsic connections, we point out the central factor that allowed them to succeed: object completion in homogeneous space. After this, we go on to point out the fact that the speed of models in the class of BEV perception is in fact limited by their depth estimation approach that does not distinguish between the foreground and the background. Based on this we argue that a good cross-modal perception model should focus only on foreground completion.

If a good cross-modal perception model should focus only on foreground completion, then how specifically should we complete the object? In section 3.1, we propose a new completion approach that requires fewer points to be added than the original approach, but has a higher performance upper bound. With this as our goal, we propose IFST and MCDNet to extract features from point clouds and predict dense foreground depths. These two networks, the former being a lightweight Transformer structure, are designed in such a way that we take into account the many distributional characteristics of the intra-frustum point clouds, allowing them to output the foreground point clouds with high accuracy under efficient operation. In the latter MCDNet, we generated a densified representation of the point cloud according to the filtered down viewpoint cloud. We treat this task as a mesh deformation process, where different pixels are considered as different vertices on the mesh. The original point cloud on the object is used as information to guide this deformation, and a lidar aggregation layer is thereby designed.

The advantages of our proposed method are unquestionable, almost any lidar-based baseline can match the performance of today's sota detectors by combining it with our method. At the same time our method doesn't need a long inference time, with a single-

frame latency of about 150ms in the NVIDIA 2080Ti graphics card environment. While 150ms is enough to make us faster than the vast majority of BEV-based approaches, we should still realize that this is still a long time for a real-time demanding task like autonomous driving. This is brought by the linear structure of the model, where our method needs to be added before the real 3D detector and cannot be end-to-end trained with it.

Another point to note is the fact that our proposed method relies heavily on the precision of the 2D segmentation: as can be seen in the fourth row of Table 4.9, when we use the ground truth to replace the 2D instance segmentor, we still get a further high performance gain. The reason for this is that our method is actually very sensitive to boundaries. This is because in fact for 2D segmentation, errors in the boundary occur from time to time, and this excess is also treated as part of the object to estimate the depth. This results in the prediction of a point cloud that exceeds the actual boundary of the object. At the same time, the inference speed of 2D segmentor is also a problem that cannot be ignored. In this thesis, we choose e2ec [84], which only needs about 40ms for instance segmentation of a single frame while maintaining the accuracy of SOTA, but for the whole system, considering that the whole process of the lidar-based 3D detector only needs 50ms, the time required is still too long.

In the next step, we will attempt to incorporate this approach into the model as a module, thus further reducing its inference time while ensuring the performance gains brought about by object completion.

References

[1] Abien Fred Agarap. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*, 2018.

[2] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[3] Xuyang Bai, Zeyu Hu, Xinge Zhu, Qingqiu Huang, Yilun Chen, Hongbo Fu, and Chiew-Lan Tai. Transfusion: Robust lidar-camera fusion for 3d object detection with transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1090–1099, 2022.

[4] Fabian Bongratz, Anne-Marie Rickmann, Sebastian Pölsterl, and Christian Wachinger. Vox2cortex: fast explicit reconstruction of cortical surfaces from 3d mri scans with geometric deep neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20773–20783, 2022.

[5] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11621–11631, 2020.

[6] Han Cai, Chuang Gan, and Song Han. Efficientvit: Enhanced linear attention for high-resolution low-computation visual recognition. *arXiv preprint arXiv:2205.14756*, 2022.

[7] Kai Chen, Jiangmiao Pang, Jiaqi Wang, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jianping Shi, Wanli Ouyang, et al. Hybrid task cascade for instance segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4974–4983, 2019.

[8] Yukang Chen, Yanwei Li, Xiangyu Zhang, Jian Sun, and Jiaya Jia. Focal sparse convolutional networks for 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5428–5437, 2022.

[9] Yukang Chen, Jianhui Liu, Xiangyu Zhang, Xiaojuan Qi, and Jiaya Jia. Voxelnext: Fully sparse voxelnet for 3d object detection and tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21674–21683, 2023.

[10] Jiajun Deng, Shaoshuai Shi, Peiwei Li, Wengang Zhou, Yanyong Zhang, and Houqiang Li. Voxel r-cnn: Towards high performance voxel-based 3d object detection. *arXiv:2012.15712*, 2020.

[11] Jiajun Deng, Shaoshuai Shi, Peiwei Li, Wengang Zhou, Yanyong Zhang, and Houqiang Li. Voxel r-cnn: Towards high performance voxel-based 3d object detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 1201–1209, 2021.

[12] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[13] Yingtong Dou, Zhiwei Liu, Li Sun, Yutong Deng, Hao Peng, and Philip S Yu. Enhancing graph neural network-based fraud detectors against camouflaged fraudsters. In *Proceedings of the 29th ACM international conference on information & knowledge management*, pages 315–324, 2020.

[14] Lue Fan, Feng Wang, Naiyan Wang, and Zhaoxiang Zhang. Fsd v2: Improving fully sparse 3d object detection with virtual voxels. *arXiv preprint arXiv:2308.03755*, 2023.

[15] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3354–3361. IEEE, 2012.

[16] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[17] Benjamin Graham, Martin Engelcke, and Laurens Van Der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9224–9232, 2018.

[18] Benjamin Graham and Laurens Van der Maaten. Submanifold sparse convolutional networks. *arXiv preprint arXiv:1706.01307*, 2017.

[19] Meng-Hao Guo, Jun-Xiong Cai, Zheng-Ning Liu, Tai-Jiang Mu, Ralph R Martin, and Shi-Min Hu. Pct: Point cloud transformer. *Computational Visual Media*, 7:187–199, 2021.

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[21] Jonas Heylen, Mark De Wolf, Bruno Dawagne, Marc Proesmans, Luc Van Gool, Wim Abbeloos, Hazem Abdelkawy, and Daniel Olmeda Reino. Monocinis: Camera independent monocular 3d object detection using instance segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 923–934, 2021.

[22] Mu Hu, Shuling Wang, Bin Li, Shiyu Ning, Li Fan, and Xiaojin Gong. Penet: Towards precise and efficient image guided depth completion. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 13656–13662. IEEE, 2021.

[23] Tengteng Huang, Zhe Liu, Xiwu Chen, and Xiang Bai. Epnet: Enhancing point features with image semantics for 3d object detection. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XV 16*, pages 35–52. Springer, 2020.

[24] Tengteng Huang, Zhe Liu, Xiwu Chen, and Xiang Bai. Penet, code implementation, 2022.

[25] Saif Imran, Xiaoming Liu, and Daniel Morris. Depth completion with twin surface extrapolation at occlusion boundaries. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2583–2592, 2021.

[26] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr, 2015.

[27] Yang Jiao, Zequn Jie, Shaoxiang Chen, Jingjing Chen, Lin Ma, and Yu-Gang Jiang. Msmdfusion: Fusing lidar and camera at multiple scales with multi-depth seeds for 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21643–21652, 2023.

[28] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, volume 7, 2006.

[29] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[30] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. *Advances in neural information processing systems*, 30, 2017.

[31] Inyong Koo, Inyoung Lee, Se-Ho Kim, Hee-Seon Kim, Woo-jin Jeon, and Changick Kim. Pg-rcnn: Semantic surface point generation for 3d object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 18142–18151, 2023.

[32] Alex Krizhevsky. The cifar-10 dataset, 2017.

[33] Jason Ku, Melissa Mozifian, Jungwook Lee, Ali Harakeh, and Steven L. Waslander. Joint 3d proposal generation and object detection from view aggregation. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–8, 2018.

[34] Xin Lai, Jianhui Liu, Li Jiang, Liwei Wang, Hengshuang Zhao, Shu Liu, Xiaojuan Qi, and Jiaya Jia. Stratified transformer for 3d point cloud segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8500–8509, 2022.

[35] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12697–12705, 2019.

[36] Ruihui Li, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. Pugan: a point cloud upsampling adversarial network. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 7203–7212, 2019.

[37] Xin Li, Tao Ma, Yuenan Hou, Botian Shi, Yuchen Yang, Youquan Liu, Xingjiao Wu, Qin Chen, Yikang Li, Yu Qiao, et al. Logonet: Towards accurate 3d object detection with local-to-global cross-modal fusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17524–17534, 2023.

[38] Yanwei Li, Xiaojuan Qi, Yukang Chen, Liwei Wang, Zeming Li, Jian Sun, and Jiaya Jia. Voxel field fusion for 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1120–1129, 2022.

[39] Yinhao Li, Zheng Ge, Guanyi Yu, Jinrong Yang, Zengran Wang, Yukang Shi, Jianjian Sun, and Zeming Li. Bevdepth: Acquisition of reliable depth for multi-view 3d object detection, 2022.

[40] Ziyu Li, Yuncong Yao, Zhibin Quan, Wankou Yang, and Jin Xie. Sienet: Spatial information enhancement network for 3d object detection from point cloud. *arXiv preprint arXiv:2103.15396*, 2021.

[41] Ming Liang, Bin Yang, Yun Chen, Rui Hu, and Raquel Urtasun. Multi-task multi-sensor fusion for 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7345–7353, 2019.

[42] Tingting Liang, Hongwei Xie, Kaicheng Yu, Zhongyu Xia, Zhiwei Lin, Yongtao Wang, Tao Tang, Bing Wang, and Zhi Tang. Bevfusion: A simple and robust lidar-camera fusion framework, 2022.

[43] Yuankai Lin, Tao Cheng, Qi Zhong, Wending Zhou, and Hua Yang. Dynamic spatial propagation network for depth completion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 1638–1646, 2022.

[44] Tianran Liu, Zeping Zhang Morteza Mousa Pasandi, and Robert Laganiere. What you see is what you detect: Towards better object densification in 3d detection. *arXiv preprint arXiv:2310.17842*, 2023.

[45] Xinyu Liu, Houwen Peng, Ningxin Zheng, Yuqing Yang, Han Hu, and Yixuan Yuan. Efficientvit: Memory efficient vision transformer with cascaded group attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14420–14430, June 2023.

[46] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted

windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022, 2021.

[47] Zhijian Liu, Haotian Tang, Alexander Amini, Xinyu Yang, Huizi Mao, Daniela L Rus, and Song Han. Bevfusion: Multi-task multi-sensor fusion with unified bird's-eye view representation. In *2023 IEEE international conference on robotics and automation (ICRA)*, pages 2774–2781. IEEE, 2023.

[48] Anas Mahmoud, Jordan SK Hu, and Steven L Waslander. Dense voxel fusion for 3d object detection. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 663–672, 2023.

[49] Yunze Man, Liang-Yan Gui, and Yu-Xiong Wang. BEV-Guided Multi-Modality Fusion for Driving Perception. In *CVPR*, 2023.

[50] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[51] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

[52] Jonah Philion and Sanja Fidler. Lift, splat, shoot: Encoding images from arbitrary camera rigs by implicitly unprojecting to 3d. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIV 16*, pages 194–210. Springer, 2020.

[53] Charles R Qi, Or Litany, Kaiming He, and Leonidas J Guibas. Deep hough voting for 3d object detection in point clouds. In *proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9277–9286, 2019.

[54] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[55] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017.

[56] Rui Qian, Divyansh Garg, Yan Wang, Yurong You, Serge Belongie, Bharath Hariharan, Mark Campbell, Kilian Q Weinberger, and Wei-Lun Chao. End-to-end pseudo-lidar for image-based 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5881–5890, 2020.

[57] Zequn Qin, Jingyu Chen, Chao Chen, Xiaozhi Chen, and Xi Li. Uniformer: Unified multi-view fusion transformer for spatial-temporal representation in bird's-eye-view. *arXiv preprint arXiv:2207.08536*, 2022.

[58] Shaoshuai Shi, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10529–10538, 2020.

[59] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 770–779, 2019.

[60] Shaoshuai Shi, Zhe Wang, Xiaogang Wang, and Hongsheng Li. Part-aˆ 2 net: 3d part-aware and aggregation neural network for object detection from point cloud. *arXiv preprint arXiv:1907.03670*, 2(3), 2019.

[61] Vishwanath A Sindagi, Yin Zhou, and Oncel Tuzel. Mvx-net: Multimodal voxelnet for 3d object detection. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 7276–7282. IEEE, 2019.

[62] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *CoRR*, abs/1905.11946, 2019.

[63] OpenPCDet Development Team. Openpcdet: An open-source toolbox for 3d object detection from point clouds. https://github.com/open-mmlab/OpenPCDet, 2020.

[64] Tesla. Tesla ai day 2021, 2000.

[65] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[66] Sourabh Vora, Alex H Lang, Bassam Helou, and Oscar Beijbom. Pointpainting: Sequential fusion for 3d object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4604–4612, 2020.

[67] Haiyang Wang, Hao Tang, Shaoshuai Shi, Aoxue Li, Zhenguo Li, Bernt Schiele, and Liwei Wang. Unitr: A unified and efficient multi-modal transformer for bird's-eye-view representation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6792–6802, 2023.

[68] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2mesh: Generating 3d mesh models from single rgb images. In *Proceedings of the European conference on computer vision (ECCV)*, pages 52–67, 2018.

[69] Tianyu Wang, Xiaowei Hu, Zhengzhe Liu, and Chi-Wing Fu. Sparse2dense: Learning to densify 3d features for 3d object detection. *Advances in Neural Information Processing Systems*, 35:38533–38545, 2022.

[70] Zhoutao Wang, Qian Xie, Mingqiang Wei, Kun Long, and Jun Wang. Multi-feature fusion votenet for 3d object detection. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 18(1):1–17, 2022.

[71] Hai Wu, Chenglu Wen, Shaoshuai Shi, Xin Li, and Cheng Wang. Virtual sparse convolution for multimodal 3d object detection, 2023.

[72] Xiaopei Wu, Liang Peng, Honghui Yang, Liang Xie, Chenxi Huang, Chengqi Deng, Haifeng Liu, and Deng Cai. Sparse fuse dense: Towards high quality 3d detection with depth completion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5418–5427, June 2022.

[73] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017.

[74] Yichen Xie, Chenfeng Xu, Marie-Julie Rakotosaona, Patrick Rim, Federico Tombari, Kurt Keutzer, Masayoshi Tomizuka, and Wei Zhan. Sparsefusion: Fusing multi-modal sparse representations for multi-sensor 3d object detection. *arXiv preprint arXiv:2304.14340*, 2023.

[75] Danfei Xu, Dragomir Anguelov, and Ashesh Jain. Pointfusion: Deep sensor fusion for 3d bounding box estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 244–253, 2018.

[76] Qiangeng Xu, Yiqi Zhong, and Ulrich Neumann. Behind the curtain: Learning occluded shapes for 3d object detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 2893–2901, 2022.

[77] Qiangeng Xu, Yin Zhou, Weiyue Wang, Charles R Qi, and Dragomir Anguelov. Spg: Unsupervised domain adaptation for 3d object detection via semantic point generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15446–15456, 2021.

[78] Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. *Sensors*, 18(10):3337, 2018.

[79] Honghui Yang, Zili Liu, Xiaopei Wu, Wenxiao Wang, Wei Qian, Xiaofei He, and Deng Cai. Graph r-cnn: Towards accurate 3d object detection with semantic-decorated local graph. In *European Conference on Computer Vision*, pages 662–679. Springer, 2022.

[80] Tianwei Yin, Xingyi Zhou, and Philipp Krahenbuhl. Center-based 3d object detection and tracking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11784–11793, 2021.

[81] Tianwei Yin, Xingyi Zhou, and Philipp Krähenbühl. Multimodal virtual point 3d detection. *Advances in Neural Information Processing Systems*, 34:16494–16507, 2021.

[82] Wentao Yuan, Tejas Khot, David Held, Christoph Mertz, and Martial Hebert. Pcn: Point completion network. In *2018 international conference on 3D vision (3DV)*, pages 728–737. IEEE, 2018.

[83] Jinqing Zhang, Yanan Zhang, Qingjie Liu, and Yunhong Wang. Sa-bev: Generating semantic-aware bird's-eye-view feature for multi-view 3d object detection, 2023.

[84] Tao Zhang, Shiqing Wei, and Shunping Ji. E2ec: An end-to-end contour-based method for high-quality high-speed instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4443–4452, 2022.

[85] Yanan Zhang, Jiaxin Chen, and Di Huang. Cat-det: Contrastively augmented transformer for multi-modal 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 908–917, 2022.

[86] Yanan Zhang, Di Huang, and Yunhong Wang. Pc-rgnn: Point cloud completion and graph neural network for 3d object detection. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 3430–3437, 2021.

[87] Yifan Zhang, Qingyong Hu, Guoquan Xu, Yanxin Ma, Jianwei Wan, and Yulan Guo. Not all points are equal: Learning highly efficient point-based detectors for 3d lidar

point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18953–18962, 2022.

[88] Youmin Zhang, Xianda Guo, Matteo Poggi, Zheng Zhu, Guan Huang, and Stefano Mattoccia. Completionformer: Depth completion with convolutions and vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18527–18536, 2023.

[89] Zeping Zhang, Tianran Liu, and Robert Laganière. Use your imagination: A detector-independent approach for lidar quality booster. *IEEE Robotics and Automation Letters*, 2023.

[90] Hanqi Zhu, Jiajun Deng, Yu Zhang, Jianmin Ji, Qiuyu Mao, Houqiang Li, and Yanyong Zhang. Vpfnet: Improving 3d object detection with virtual point based lidar and stereo data fusion. *IEEE Transactions on Multimedia*, 2022.