Enhancing Point Cloud Through Object Completion Networks for the 3D Detection of Road Users

by

Zhang Zeping

Thesis submitted to the University of Ottawa
in partial fulfillment of the requirements for the degree of
Master of Applied Science, Electrical and Computer Engineering
with
Concentration in Applied Artificial Intelligence

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

With the advancement of autonomous driving research, 3D detection based on LiDAR point cloud has gradually become one of the top research topics in the field of artificial intelligence. Compared with RGB cameras, LiDAR point cloud can provide depth information, while RGB images can provide denser resolution. Features from LiDAR and cameras are considered to be complementary.

However, due to the sparsity of the LiDAR point clouds, a dense and accurate RGB/3D projective relationship is difficult to establish especially for distant scene points. Recent works try to solve this problem by designing a network that learns missing points or dense point density distribution to compensate for the sparsity of the LiDAR point cloud. During the master's exploration, we consider addressing this problem from two aspects. The first is to design a GAN(Generative Adversarial Network)-based module to reconstruct point clouds, and the second is to apply regional point cloud enhancement based on motion maps. In the first aspect, we propose to use an imagine-and-locate process, called UYI. The objective of this module is to improve the point cloud quality and is independent of the detection stage used for inference. We accomplish this task through a GAN-based cross-modality module that uses image as input to infer a dense LiDAR shape. In another aspect, inspired by the attention mechanism of human eyes, we use motion maps to perform random augmentation on point clouds in a targeted manner named motion map-assisted enhancement, MAE. Boosted by our UYI and MAE module, our experiments show a significant performance improvement in all tested baseline models. In fact, benefiting from the plug-and-play characteristics of our module, we were able to push the performance of the existing state-of-the-art model to a new height. Our method not only has made great progress in the detection performance of vehicle objects but also achieved an even bigger leap forward in the pedestrian category. In future research, we will continue to explore

the feasibility of spatio-temporal correlation methods in 3D detection, and 3D detection related to motion information extraction could be a promising direction.

Acknowledgements

Time flies, and the two-year master's study is coming to an end in a blink of eye. Looking back on my study during my master's, I have a feeling that I have learnt a lot. Like many, my master's study was not smooth sailing, and I encountered a lot of confusion during the period, but more of it was the joy of learning new knowledge and the desire to explore sciences. This experience will be a valuable asset in my life. Therefore, on the occasion of writing this thesis, I would like to express my heartfelt thanks to professor Laganiere, my colleagues, classmates and family members who have always helped me.

First of all, I would like to express my deepest gratitude to my mentor, Professor Dr. Laganiere, for his patient guidance and continuous encouragement throughout my study and research during my master's. I sincerely believe that any progress and gains in my studies are inseparable from the help of the professor. He has many years of expertise in the field of autonomous driving, and VIVA Research Lab is one of the few labs at the University of Ottawa in the field of AI with the ability to contribute research to top AI conferences. Professor Laganiere is knowledgeable and has brought us a broad horizon while pursuing a rigorous research attitude. He has been committed to providing a complete scientific research system and development platform for our research group, from academic cooperation in scientific research to technology landing in the industry. His amiable and serious attitude has greatly inspired us. Professor Laganiere will be my role model and the role model for lifelong study.

Then, I would also like to thank my colleagues in the research group, Tianran and Morteza also helped me a lot in my work. Especially for reaching out when I am stuck at some points, and their positive and enterprising attitudes have deeply affected me.

I would also like to thank my good friends, Boli Fang from Microsoft, and Miao Jiang

from the University of Texas at Austin. Whether it is scientific research or life, they have spared no effort to support me. Although we are each in the US, Canada, and Japan, we are always able to communicate and get work done most efficiently. During the master's, we worked together on a set of voice-activated Covid-19 detection models based on an audio transformer. It will be nice to start a business together with you in the future.

Also, I would like to thank my girlfriend. Although she was not in Canada during my research, she gave me great psychological support. She also helped with a lot of drawing and formatting during the process of reviewing the thesis. Thanks for caring and accompanying me all the time.

I would like to thank my parents. They have created a good study environment for me with their hard work and efforts. For more than 20 years, they have been working hard and giving me endless love and support. I will live up to their expectations and continue to explore the peak of the academic in my doctoral work.

Finally, I sincerely thank all the experts and professors who took time out of their busy schedules to participate in my thesis review and defense. Thank you. And to you who are reading the thesis, thank you for making my work worthwhile.

Unity, justice, and liberty.

Zhang Zeping

December 2022, in Ottawa, Canada.

Table of Contents

List of Tables

List of Figures

List of Acronyms

NMS	Non-maximum Suppression
AI	Artificial Intelligence
LSTM	Long short-term memory
AP	Average Precision
mAP	Mean Average Prevision
ACC	Accuracy
CNN	Convolutional Neural Network
DNN	Deep Neural Network
GAN	Generative Neural Network
MLP	Multi-Layer Preception
MLR	Multiple Linear Regression
FC	Fully Connected
DL	Deep Learning
SGD	Stochastic Gradient Descent
SVM	Support Vector Machine
BEV	Bird Eye View
LiDAR	Light Detection and Ranging
GAF	Grid-wise Attentive Fusion
ROI	Region of Interest
MAE	Mean Absolute Error
Conv	Convolution
TP	True Positive
TN	True Negative
FP	False Positive
FN	False Negative
CMS	Camera Monitor System
CAS	Collision Avoidance System
HUD	Head-up-display
LR	Learning Rate
ReLU	Rectified Linear Activation Unit
Tanh	Hyperbolic Tangent Activation Function
BN	Batch Normalization
LN	Layer Normalization
BS	Batch Size
RL	Reinforcement Learning
RF	Random Forest
VAE	Variational Autoenoder
YOLO	You Only Look Once(Object Detection model)
E2E	End to End Learning
Seq2Seq	Sequence to Sequence Learning
PCA	Principal Component Analysis
MLE	Maximum Likelihood Estimation
cGAN	Conditional Adversarial Network
Adam	Adaptive Moment Estimation

Chapter 1

Introduction

1.1 Motivations and Problem Overview

3D object detection plays a vital role in various computer vision applications such as autonomous driving, drones, robotic manipulation, and augmented reality [2,30,40,52,58].

In the process of autonomous driving, in addition to using the 2D camera to capture the surrounding environment, there is also a need to perceive the surrounding 3D space. 3D object detection as shown in Figure 1.1 is used to obtain the position and category information of the objects in the 3D space, which serves as the basis of the autonomous driving system and plays an important guiding role in the subsequent path planning, motion prediction, and collision avoidance steps. The main commercially available sensors used for generating point clouds include LiDAR and RADAR. With its wavelength, RADAR can detect objects at great distances, even through fog or clouds. But its lateral resolution is limited by the size of the antennas. Standard radars operate at ranges in excess of 100 meters and their resolution may be as high as several meters and that is the reason why

RADAR is usually used in aircraft collision avoidance, air traffic control, etc. LiDAR, on the other hand, is a compact solution capable of delivering a higher level of accuracy for 3D measurement, which makes it more suitable for autonomous driving systems. The current mainstream 3D detection algorithms can be divided into methods based on 2D images, LiDAR point clouds, and multimodal data fusion.

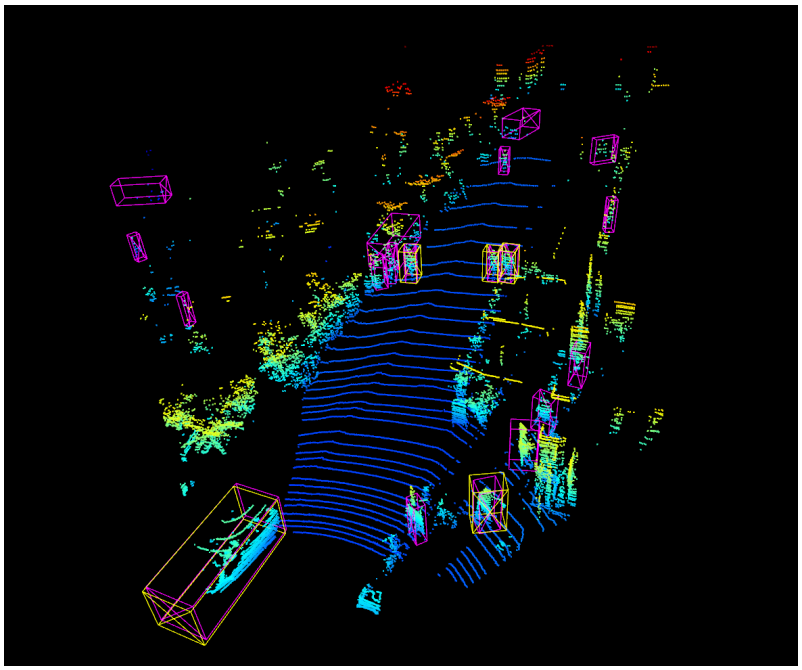

Figure 1.1: Detection in 3D space. The yellow boxes represent ground truth labels, and the pink boxes represent inferences. This figure is an example of the 3D detection results of PV-RCNN [44] on KITTI dataset [15, 16].

Many 3D detection solutions based on RGB cameras + LiDAR sensors [6,9,28,33,41,55] have been proposed in recent years and achieve good performance in 3D detection. Modern 64-beam LiDAR sensors are able to guarantee high-precision object positioning within a range of nearly 100 meters. However, compare with dense pixels information from RGB images, the distribution of cloud points is always sparser thus harder to analyze.

In this thesis we proposed a solution to the 3D object detection problem by improving 3D detection accuracy from the point clouds, which mainly starts from two aspects: first, we proposed the Use Your Imagination, UYI module to generate a new vehicle point cloud to compensate for the sparsity of the distant vehicle point cloud; second, use the motion map to make the target range of the point cloud denser.

1.1.1 Use Your Imagination Module

In our work, we consider how to directly improve the quality of the point cloud while making it independent to the detection part, making it a plug-and-play component. To the best of our knowledge, this work is the first to accomplish this task with a GAN [17]-based cross-modal module that uses images as input to describe the shape of objects in LiDAR format. In summary, we aim at answering the following three questions in following content.

1. How the sparsity of distant LiDAR point cloud affects LiDAR Pseudo-point cloud generation.

2. How to efficiently generate 3D object shape in LiDAR format using RGB information

3. How can we design this module to be plug-and-play on the widest possible range of models.

1.1.2 Motion Map Assisted Enhancement Module

Our idea based on motion maps is inspired by the attention mechanism of the human eye. Many previous works in the fields of physiology and biology research have shown that the human eye is more sensitive to moving objects than stationary objects [1, 13, 42]. For the human eye, the resolution of an object is divided into two types: dynamic resolution and static resolution. Dynamic resolution is the representation of an object moving over a period of time [50]. Consider a piece of chalk, when it is sitting still on a blackboard, we perceive it to have a high static resolution and a low dynamic resolution. And when the chalk is flying over the head of a hapless mischievous student, it will have a low static resolution but a high dynamic resolution. This process inspired our research on facilitating the hard cases of distant 3D object detection.

Distant detection objects usually have limited static resolution, and this resolution is constrained so it is difficult to improve it directly. Therefore, we consider introducing motion maps to improve the temporal resolution of moving objects. For a given 2D image, the motion map is a binary motion representation. Our motion map describes the parts of an image where moving elements are observed. A motion map is shown in Figure 1.2.

Figure 1.2: Motion map. The white area on the right is used to describe the motion area in the left image.

On this basis, considering that there is a fixed projection relationship between the 2D image of the autonomous driving dataset and its LiDAR point cloud, we can obtain the point cloud of the range that moves in the motion map. By densifying the point clouds corresponding to motion range, the quality of the point clouds tending to move can be densified in a targeted manner, where usually such point clouds are representations of road objects such as vehicles and pedestrians. The targeted point cloud enhancement based on motion maps has two main advantages. First, compared with the full-range point cloud completion method, our method based on motion maps can avoid the completion of irrelevant background point clouds, thus saving computing resources and improving the operating efficiency of the algorithm. The second is that the process of constructing the motion map itself actually provides a new source of inferencing reference for the traditional 2D image + 3D detection method for which the result of the motion map could serve as a reference for 2D and 3D detectors.

1.2 Research Contributions

1.2.1 Sparse Representation to Pseudo Point Cloud

3D detectors inevitably suffer from point cloud sparsity, especially the methods [8, 44, 61] that use LiDAR-only information. Therefore, Chan et al. [5] proposed to use a dense pseudo point cloud generated by depth completion to enhance the LiDAR point cloud. SFD [57] extracts 3D geometry in a pseudo point cloud by designing a new RoI feature fusion method named 3D-GAF (3D Grid-wise Attentive Fusion). However, the pseudo LiDAR data generated by depth completion modules generally suffers from the long tail problem. For these difficult scenarios where there exist very few LiDAR points, so most,

if not all, features the detector can acquire comes from pseudo points. These drawbacks limit the performance of SFDNet [57]. In this paper we propose an effective object-level LiDAR pseudo-point cloud generation process that increases the density and quality of a captured point cloud, thereby not only improving the performance of 3D detection, but also reducing the number of pseudo-points to be generated compared to global-level completion.

1.2.2 2D Point Cloud Colormap and Projection

To solve this problem, we transform the task of generating point clouds from RGB vehicle images into an image translation task. A GAN augmented with a self-attention module is introduced to convert the input RGB image into a 2D color point cloud image. We convert an input 3D vehicle point cloud box with depth into an RGB image by establishing a depth-color gradient correspondence as described in section 4.1. In previous works [14, 26] it has been proven that GANs are indeed capable of learning color information. GANs are widely used in applications such as color paintings and other color reconstruction task [21, 39, 47]. Here we want the GANs to learn depth information of point clouds which will provide us with an approach to process and generate 3D point cloud as an image.

1.2.3 Plug-and-Play Design

We note that for most of the previous models that use RGB information to complement the LiDAR information, their proposed enhancements were not readily adopted by subsequent models. We therefore opted for a design that could be used within as many models as possible. The modules proposed in this thesis have been designed to work in between the

input data and the detection model, and provide a higher-quality point cloud to the input side of the detection block to enhance detection.

1.2.4 Construct the Prior Knowledge of Motion

The motion map-assisted enhancement branch (hereinafter also referred to as MAE, Motion map Assisted Enhancement) aims to enhance the point cloud quality of autonomous driving datasets, such as the KITTI dataset. The essence of the motion map is to transfer the motion prior knowledge on the continuous autonomous driving dataset to the static dataset, so that the point cloud of the motion area in the static dataset can be enhanced. Therefore, in order to obtain this prior knowledge, we can obtain a motion map by processing the inter-frame motion information in the video captured by the RGB camera on continuous autonomous driving datasets. However, to capture this prior knowledge more easily, we built VIVA Motion. VIVA Motion is a high-quality motion video dataset collected on real streets in Ottawa, Canada. We will further describe VIVA Motion in Chapter 5. The method is to calculate the inter-frame gaps between each frame for each short-term video segmentation, and then aggregate them to form a motion map.

1.2.5 Learning the Prior Knowledge of Motion

After constructing the motion map, we need to train a generative model to learn the transform from a 2D image to its motion map. We found that this problem can be understood as an image translation process. In this method, the MAE will have the ability to infer a motion map from the input single-frame 2D image. At the same time, the selected point cloud area can be obtained by locking the target area through the fixed projection relationship of LiDAR to RGB plane. Thus we can apply densify operations on the targeted

point cloud.

1.3 Thesis Structure

In Chapter 1, we introduced the origin and inspiration points of the innovations in our research. Chapter 2 present the technology and development process related to this thesis. In Chapter 3, we introduce fundamental concepts in 3D detection, such as the sparse convolution process and performance metrics. In Chapter 4 and 5 we present the detailed design of Use Your Imagination Module and Motion Map Assisted Enhancement Module, we describe in detail the two innovative points of this thesis, UYI and MAE modules, and their experiments. In Chapter 6 and 7, we summarizes the strengths and limitations of our research in the thesis, the inspiration for the industry, and the insights for future research. Finally, in Chapter 8, we summarize the content of the entire thesis.

Chapter 2

Related Work

2.1 Object Detection

In recent years, with the vigorous development of autonomous driving, object detection based on deep learning has gradually become a research hotspot in the field of computer vision. Object detection is an important part of autonomous driving and intelligent transportation systems. In this section, we provide an overview of the two main types of goals involved in autonomous driving: vehicle and pedestrian detection. We will also sort out the key points, difficulties and development status of the detection task, and summarize the vehicle and pedestrian target detection models under deep learning with a timeline.

2.1.1 Key points and difficulties of vehicle and pedestrian detection tasks

The task of vehicle and pedestrian detection based on deep learning is to find the object of interest from a series of images or videos, and at the same time determine the size and position of the vehicle according to the feature information. However, in real road situations, there will always be many interference factors, such as lighting, occlusion, different scales and shapes, etc., which will cause missed and false alarms during the detection, affecting the accuracy of the model. Therefore, how to meet the accuracy requirements of the model is a key and difficult goal in object detection; at the same time, for the application of object detection in the field of autonomous driving, achieving real-time detection is also the focus of current research. A large number of researchers have made a series of outstanding contributions to these two tasks, especially the vigorous development of deep learning algorithms, which has brought better research prospects for 3D detection tasks.

At present, the huge challenge faced by the vehicle and pedestrian detection task is still the contradiction between the real-time performance of the detection model and the accuracy. How to better balance them has always been an important research direction of the object detection algorithm based on deep learning. Another research difficulty is that self-driving vehicles will encounter different environmental conditions during driving, so the collected images are often subject to different lighting conditions, the weather could be constantly changing, and the target object could be blocked by a large area. This will lead to insufficient feature extraction and missing objects during the detection and can further increase the difficulty of detection. Large-scale datasets for different weather scenarios have only begun to appear in recent years, such as the DENSE dataset [4]. However, the labeling standards for different environmental conditions have not yet been popularized on

the widely used mainstream detection datasets such as Waymo [11,51] and KITTI [15,16].

2.2 3D Detection and Autonomous Driving

2D target detection can only provide the positioning on the target plane, but the roads in the real world where the self-driving vehicle runs are three-dimensional, which means not only the position information of the target in 3D space, but also the depth information such as size and direction need to be obtained, thus object detection based on 2D images can no longer meet the needs of self-driving. Therefore researchers turn their attention to 3D object detection. 3D detection can obtain more detailed three-dimensional information about the target object space and improve the environmental perception ability of the automatic driving system. A brief introduction to some commonly used public datasets for 3D object detection is shown in Table 2.1.

Dataset	Classes	Frames	Description
KITTI	8	15k	The KITTI dataset is a classic computer vision evaluation dataset for autonomous driving scenarios, co-founded by Karlsruhe Institute of Technology (KIT) and Toyota Technological University Chicago (TTIC).
nuScenes	23	40k	The nuScenes dataset is a large-scale autonomous driving dataset established by the autonomous driving company nuTonomy. The dataset not only includes Camera and LiDAR, but also records radar data.
Waymo	4	200k	Waymo Open Dataset (WOD) is a data set released by Google's Waymo driverless company in 2020
DENSE	28	13.5k	The DENSE data set is an all-weather scene data set for autonomous driving, proposed by the European Union and Ulm University in Germany and Mercedes-Benz. It is designed to provide verification of the performance of detection that work reliably in all weather conditions.

Table 2.1: Comparison of widely used 3D detection datasets

2.3 Anchor Boxes

An anchor box refers to a set of predefined boxes used to identify detected objects during the target detection process, and its width and height match the width and height of objects in the data set. The size of these predefined anchor boxes contains the size combinations of all objects that may be detected in the dataset, eg. should include different ratios and scales. Usually 4-10 anchor boxes are predefined as candidate anchor boxes for each position in the image.

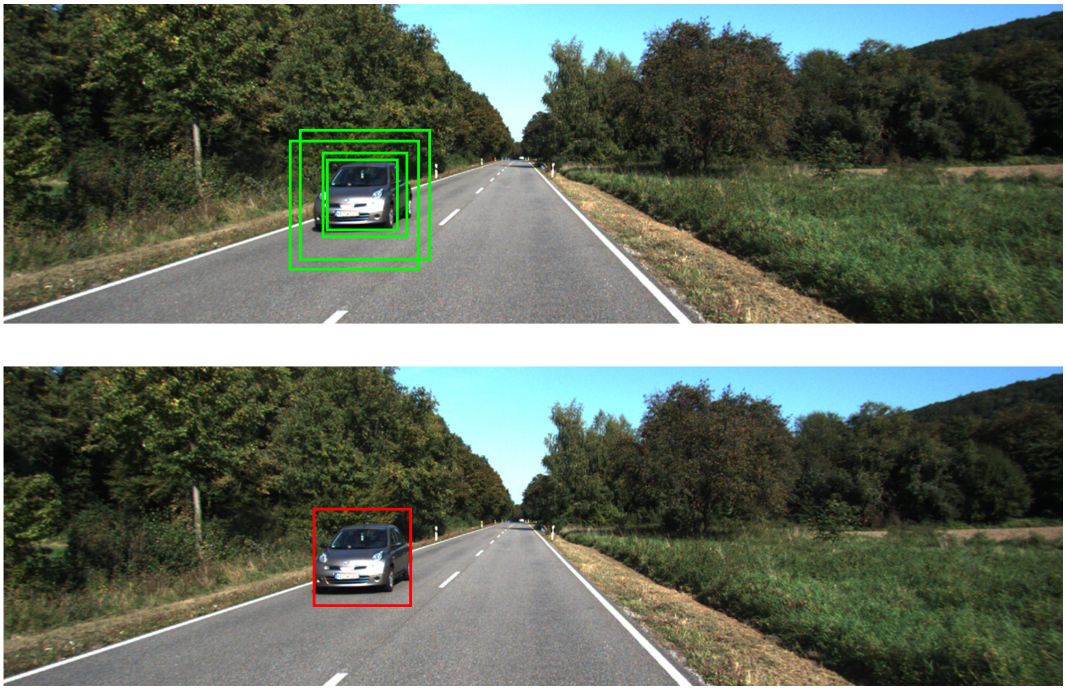

Figure 2.1: Non-maximum suppression demo: The above is the original output of multiple detection heads, and the bottom is the result after executing NMS process.

In the field of object detection, researchers initially used sliding window detectors to localize individual objects during propagation. A major problem with the sliding-window based object detection mechanism is that each box can only contain one object. Anchor

boxes allow us to detect multiple objects per window. Sliding window detectors were replaced by single-shot, two-stage detectors. These detectors are largely based on the concept of anchor boxes to optimize the speed and efficiency of sliding window detection algorithms. This is because sliding window detectors require multiple forward passes to process the image, but most of the forward passes do not process the main objects contained in the image. The anchor box that is considered to be successfully matched to the target object will generate a bounding box. However, since we pre-define anchor boxes of multiple sizes, we may get a series of outputs. As shown in Figure 2.1, in the process of object detection, a large number of candidate bounding boxes will be generated at the position of the same target, and these candidate boxes may overlap with each other. At this time, we will need to use non-maximum suppression to find the best target bounding box to eliminate redundant remaining bounding boxes. Non-maximum suppression is based on intersection-over-union(IoU), which indicates the intersection area of two bounding boxes divided by their union area. We will describe IoU and NMS(Non-maximum Suppression) in the next section.

2.4 Voxels in 3D Space

Voxel is the abbreviation of Volume Pixel, the concept is similar to the smallest unit of 2D image - pixel [66]. For the input point cloud, we usually use cubes of the same size to divide it, assuming we use a large cube with depth, height and width (D, H, W) to represent the input point cloud, the depth, height and width of each voxel v can be expressed as (v_D, v_H, v_W), then the number of voxel grids generated on each coordinate for the 3D voxelization result of the entire point cloud is $\left(\frac{D}{v_D}, \frac{H}{v_H}, \frac{W}{v_W}\right)$. A typical definition of 3D voxels and 2D pixels can be visualized in Figure 2.2.

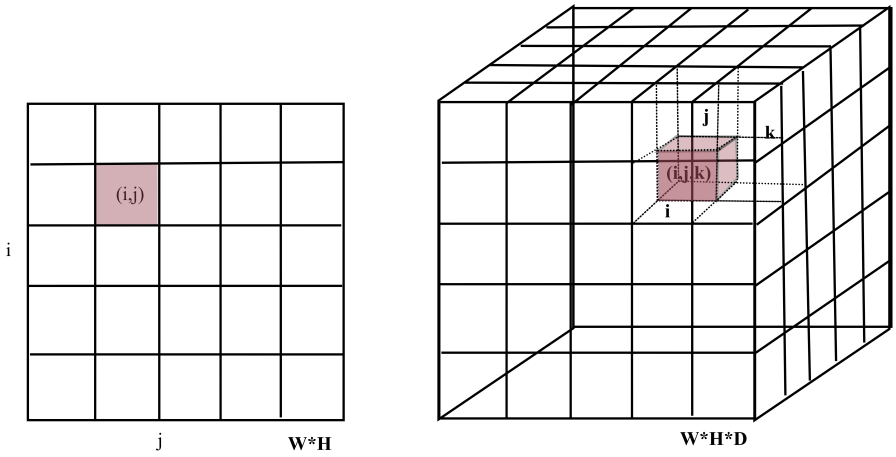

Figure 2.2: A typical definition of 3D voxels and 2D pixels. (i, j) in the figure can represent the coordinates of a pixel in the 2D image, and if a depth k is added, it can refer to a voxel in the 3D space.

After the voxel blocks are obtained, we need to perform voxel grouping as Figure 2.3. However, the density of the points scanned by LiDAR in different areas sometimes varies greatly. For example, the points on close-range objects tend to be denser than distant objects. This can be understood as a distribution imbalance problem. Moreover, high-density points will inevitably bring deviations to the calculation results of the neural network, and if all the points in the point cloud are processed, it will consume high computing power and video memory. Therefore, we usually need to perform random sampling operations before extracting features from voxels. For each voxel group, we randomly select the number T of the points. After that, we need to extract features of voxels by sparse convolution, which will be described in the next section.

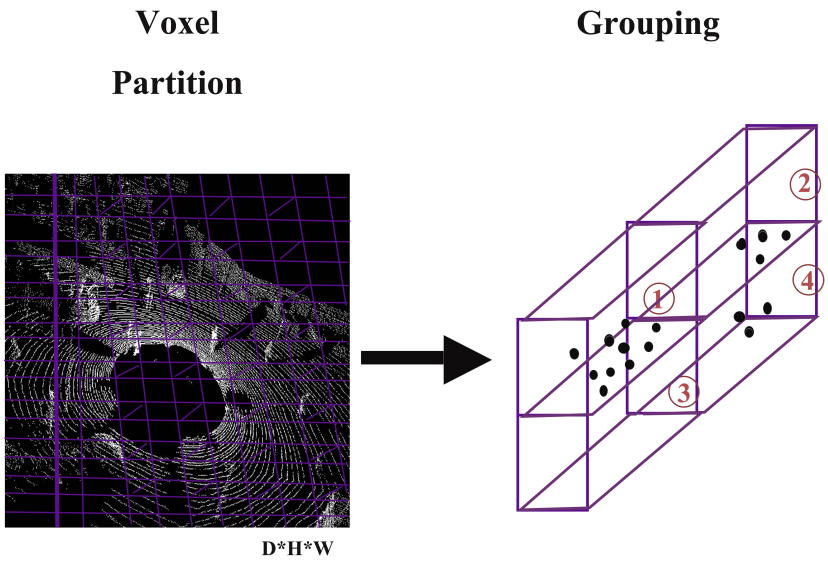

Figure 2.3: Demonstration of voxel partition and grouping

2.5 Sparse Convolution

The voxel feature encoding layer is a typical neural network layer used to encode voxel features, referred to as VFE layer(Voxel Feature Encoding layer). It accepts point-wise input and obtains a feature representation F within a voxel. The process is described as below.

For a point set V randomly sampled in a voxel grid, we can define that

$$V = \left\{ p_i = [x_i, y_i, z_i, r_i] \in \mathbb{R}^4 \right\}_{i=1....}, t < T$$

where $[x_i, y_i, z_i, r_i]$ are the X-axis, Y-axis and Z-axis coordinates of the point and the reflection intensity. We can calculate the average value (v_x, v_y, v_z) of all its points as the shape center of the voxel, and align all the points according to this center, thus we

can get:

$$V_{aligned} = \left\{ \hat{p}_i = [x_i, y_i, z_i, r_i, x_i - v_x, y_i - v_y, z_i - v_z]^T \in \mathbb{R}^7 \right\}_{i=1...t}$$

Then, each $\hat{p}_i$ will pass through a fully connected network and be mapped to a feature space $f_i \in \mathbb{R}^m$, here we assume that the output feature dimension will become m. The fully connected layer includes a linear mapping layer, a batch normalization layer and a layer that runs ReLU nonlinear operation, to obtain a point-wise feature representation.

Then we use MaxPooling to aggregate the feature representation obtained in the previous step element by element. This pooling operation is carried out between elements to obtain a locally aggregated feature f. Finally, these element-by-element features can be concatenated to obtain the output feature F, which is used as the feature representation of a single voxel.

$$F = \left\{ f_i^{\text{out}} \right\}_{i=1...t}$$

Through the processing of the above process, we can get a series of voxel features, which can be represented by a 4-dimensional sparse tensor:

$$F \times D' \times H' \times W'$$

In a real scene, LiDAR generates hundreds of thousands of points per scan. However, more than 90% of the voxels in the voxel space composed of these points are empty. Using sparse tensors to describe non-empty voxels can effectively reduce memory and computation waste during backpropagation.

2.6 LiDAR-only Methods

3D detection based only on LiDAR point clouds could be divided into two branches: point based and voxel based. Point-based approach uses the original point cloud data coordinates as the feature carrier and directly uses the LiDAR point cloud for processing. Voxel-based approach converts the point cloud data into regular data and uses convolution to achieve the task, in other words, this approach uses the voxel centres as CNN-aware feature carriers. A typical visualization of voxel in 3D space can be seen in Figure 2.4. In VoxelNet [66], the 3D point cloud is divided into a certain number of voxels, and after random sampling and normalization of points, several VFE layers are used for local feature extraction for each non-empty voxel, and then further feature abstraction is performed by an intermediate 3D convolutional layer to increase the perceptual field and learn geometric spatial features, finally the object could be classified using RPN detection and position regression. The method proposes an end-to-end, trainable deep network that can directly process sparse 3D point clouds, avoiding the information bottleneck problem introduced by manually designed features.

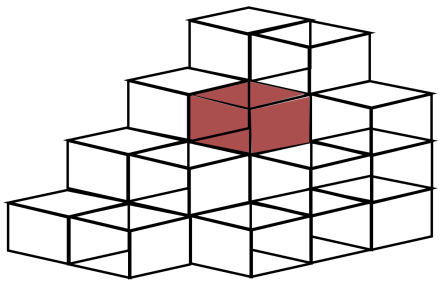

Figure 2.4: Visualization of voxels in 3D space.

However, 3D convolutions are computationally intensive and attempts have been made to improve the efficiency of 3D convolution. SECOND [61] is a one-stage target detection

method for 3D LiDAR point clouds that uses 3D sparse convolution to greatly improve the speed of 3D convolution. PointPillar [29] addresses this problem using a new encoder that uses the PointNet architecture to learn features of point clouds in a vertical column organization to complete end-to-end training of the 3D detection network; by setting all computations on the column to dense 2D convolution, a speedup of 2-4 times faster than other previous methods is achieved.

PointRCNN [45] is the first two-stage framework for 3D detection from raw point clouds and the first anchor-free proposal generation strategy based on point clouds, enabling 3D detection tasks to be performed purely using point cloud data, and solving the occlusion problem and the dependence on 2D detection results during detection. The framework consists of two parts: the first part enables the generation of 3D proposals from the original point cloud space by segmenting the foreground points; the second part adjusts the proposals by using canonical coordinates to obtain the final detection results. Part-A^2 [46] is the first application of sparse convolution to two-stage 3D detection, with the entire network divided into two modules, Part-Aware stage and Part-Aggregation stage. The Part-Aware stage rasterizes the entire space, then generates features for each lattice, and automatically extracts features from the point clouds within the raster using fully connected layers and the max-pooling method to obtain features for each raster. To combine the advantages of voxel-based and point-based methods, the team at the Chinese University of Hong Kong then proposed a novel high-performance 3D object detection framework, called PointVoxel-RCNN (PV-RCNN) [44], for accurate 3D detection from point clouds. The method is also a Two-Stage approach that abstracts 3D voxel convolutional neural networks and PointNet-based ensembles to learn more discriminative point cloud functions through deep integration. It takes advantage of the efficient learning and alternative

proposals of the 3D voxel CNN and the flexible reception range of the PointNet-based network. PV-RCNN [44] is not only a multi-scale and voxel fusion of feature information, but also a fusion of point and voxel. The approach based on points has the characteristics of a variable, multi-scale perceptual field, while the voxel approach is highly efficient, and PV-RCNN brings both of these to the fore. In fact, before SFDNet [57], there was a long period of time when methods based on LiDAR only achieved the highest performance. One possible reason is that the feature fusion efforts were not sufficiently correct or effective. More reviews on feature fusion will be discussed in the next section. In 2021, VOXEL-RCNN [8] was proposed and made the process of extracting 3D structure information more efficient. The authors believe that point-based methods are effective because they provide precise location information. The voxel-based method effectively aggregates the internal features of the grid through grid division, which give it significant advantages in feature extraction. While at the same time, the aggregation of points inside the grid loses the precise position information of each point. VOXEL-RCNN advocates that efficient point cloud feature extraction is not necessarily based on precise localization information. However, from an information-theoretic point of view, multiple sensors have more complementary information and the use of multimodal information should be able to further improve robustness and detection accuracy. We will talk about RGB and LiDAR fusion methods in the next section.

2.7 RGB and LiDAR Fusion Methods

In 3D target detection, the main methods for fusing LiDAR point cloud and image information are Early-Fusion, Deep-Fusion and Late-Fusion. Their brief processes are shown in Figure 2.5.

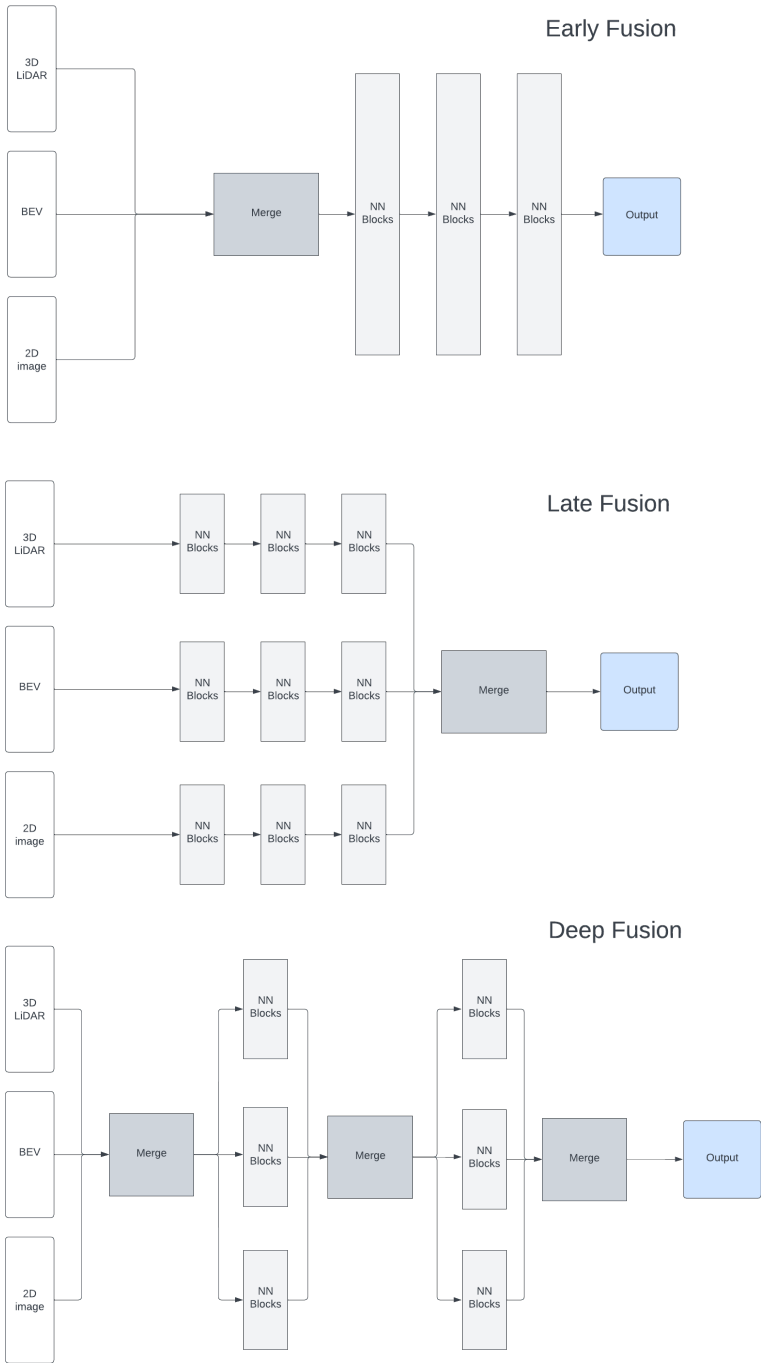

Figure 2.5: Three kinds of fusion. From up to down: early fusion, late fusion and deep fusion. In fact, Early-Fusion, Deep-Fusion and Late-Fusion are fusions at the input layer, feature layer and decision layer respectively.

Early fusion refers to the fusion of features prior to the task of extracting features from the raw sensor data. This is usually manifested by processing several separate datasets into a single feature vector, which is then fed into a classifier and then passed through a deep learning network to achieve border regression. In theory, this fusion method is the most effective one of the multimodal fusion methods, as the corresponding features here can share some indexing relationship and less feature abstraction in reality. However, Early fusion techniques usually do not make good use of the complementary properties between the different modal data, and the original data for early fusion often contains a very high amount of redundant information. Therefore, early fusion methods are often combined with feature extraction methods to remove redundant information, such as maximum correlation minimum redundancy algorithms (MRMR), autoencoders, principal component analysis (PCA), etc.

Deep fusion requires some interactions in the feature layer. Each branch of the LiDAR point cloud and image data would have its own feature extractor, and each branch of the network is fused at the feed-forward level to achieve semantic fusion of multi-scale information. The main feature is the flexibility to choose where to fuse. It is therefore the fusion method most likely to create new fusion methods.

Late fusion is the simplest fusion method. The core idea is that the features of the two modalities are not fused in the feature layer or at the beginning, because the data from the different sensors are inherently different, in the case of LiDAR and images, the biggest difference is in the view. In addition, the biggest difficulty in fusing point clouds and images to do feature layers is the indexing accuracy and domain differences between pixels and point cloud points. More common post-fusion methods include averaged fusion,

ensemble learning, and max-fusion. The errors in this fusion approach come from multiple classifiers, and the errors from different classifiers usually do not interfere with each other, thus not making the errors cumulative.

Current multimodal fusion methods for 3D detection can be found back to MV3D [6], which projects point cloud into a 2D plane with a specific viewpoint and then fusing the data from different visual angles. This method loses accuracy when projecting a bird's eye view, and the final results of the experiments show that MV3D is only good for cars, and performs poorly for pedestrians.

Instead of fusing the LiDAR point clouds and images which are processed in parallel, F-PointNet [41] generates edges in the 2D target detector in a serial way and then projects them onto the 3D point cloud for further optimization work. This type of approach improves detection efficiency, enables dimension-by-dimension (2D-3D) localization, shortens the search time for the point cloud and has almost no loss of information in any dimension. However, its outstanding disadvantage is that the whole process is more dependent on 2D detection results and cannot solve the occlusion problem.

The main innovation of MMF [31] (CVPR19) is the first projection of image features into a bird's eye view (BEV map) for regression, followed by the solution of fusion of BEV view information and image information at the point-wise level.

The above approaches are excellent works in the direction of 3D target detection by fusion of LiDAR point cloud and image data in recent years, from which it can be seen. However, we still face the difficulties stated below about multimodal fusion.

1. Sensor perspective differences: cameras acquire information from the cone of view due to the small-aperture imaging principle, while LiDAR acquires information in the real 3D world.

2. Difficulty in information fusion: Image data has scale issues due to distance. In 2D detection, objects gradually become smaller in scale as the distance becomes farther, but this is not the case in 3D detection. In order to make better use of the depth information of point clouds and the denser characteristics of RGB images, the work in recent years has focused on integrating their advantages by doing depth completion.

2.8 Depth Completion

The aim of depth completion is to predict a dense depth map from a sparse depth map guided by RGB images [22, 36]. This process can be shown in Figure 2.6. The objectives of depth completion are divided into three main aspects.

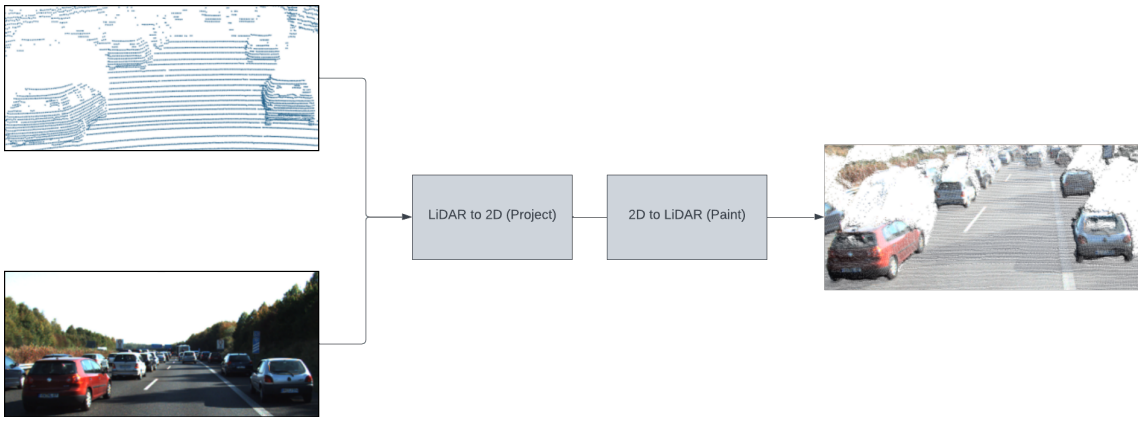

Figure 2.6: Demo of depth completion

1, the output point cloud should be able to retain the detailed structure of the input point cloud.

2, the model is imaginative enough to be able to reason about the complete shape from the mutilated shape.

3, the shape after reconstruction needs to be sharp and accurate enough.

Many efficient depth completion methods have been proposed in recent years [3,7,37,38], e.g. Hu et al. [19]; Imran et al [23]. This method is gradually being introduced into the field of 3D detection.

To make the previous depth completion more adaptable in 3D detection for autonomous driving, previous works [32, 59, 65] have considered how to use the high-resolution characteristics of RGB images to compensate for the drawbacks of LiDAR sensors. For example, early work like MV3D [6] and its successor AVOD [28] proposed methods that using RGB features as ancillary information for points. Later works [18, 24] have focused on enhancing the point clouds, ContFuse [34] proposes a new way to aggregate information between 2 modalities to enrich the original point cloud. BtcDet [60] uses operations such as shape prototypes and symmetric flips to densify the point cloud of a vehicle object. However, considering the extreme sparsity of some objects, the difficulty of point recovery will be high in some cases. Many works [53, 54, 62, 63] also attempt to make the network learn the missing points or the distribution of points to compensate for the sparsity of point cloud, but are confronted with the same issues. Their performance is greatly limited by the lack of accurate or sufficient original LiDAR point clouds. To make better use of the point cloud information for characterization in the depth reconstruction process, SFDNet [57] proposes color projection, i.e. introducing color information from RGB images into the point cloud. Through this contribution, SFDNet managed to become the first in the KITTI [16] ranking as of the first half of 2022. The best performing open source model at the time of writing is TED [56], which is also based on the work of point cloud complementation, where TED applies a sparse convolutional backbone to extract multi-channel transformed isovariant features of the complemented point cloud; these isovariant features are then aligned

and aggregated into a lightweight and compact representation for high-performance 3D detection

2.9 Previous Occlusion Processing Method

Occlusion poses an immediate performance problem for many computer vision tasks, such as detection [64], instance segmentation [12] and 3D detection for autonomous driving [60]. In the case of previous 3D detection, many approaches [60] have attempted to compensate for the occluded portion of the point cloud. For instance, Hu et al. [20] proposed a raycasting algorithm for efficiently computing object visibility and augmentation. Later BtcDet [60] was the first method to propose learning the occluded shapes in point cloud data by computing shape occupancy. However, both of them did not use RGB information for reconstruction. BtcDet complements the target using methods such as mirror symmetry and best-matching, which we designate as TFMM (Traditional Flip-Mirroring Method). We will compare our method with TFMM in the experiment section.

2.10 Image Translation using GAN

Image translation tasks can be used to convert a representation of a scene to a representation in another space. Conditional generative adversarial networks [35] are classic solutions to the image translation problem. Many works [27, 43, 67] complete the image translation task by learning the process of mapping from input images to output images. In this thesis, we extend this idea to 3D point cloud generation by introducing a self-attention layer and a texture-aware sentient module with distance color coding.

Chapter 3

Methodology

3.1 Intersection over Union and Non-maximum Suppression

Intersection over Union(IoU) is developed from the Jacquard index of set theory, and is used to calculate the overlapping degree of the real bounding box and the predicted bounding box. IoU is the ratio of the prediction box or the intersection 3.1 and union of the detection result and the ground truth.

Non-maximum suppression(NMS) is an IoU-based bounding box optimization algorithm. Normally a 2D detector tends to produce multiple detection bounding boxes for one object, NMS is used to select the most optimized bounding box and filter the rest as shown in figure 3.2. Its flow is as follows:

1. Sort according to the confidence score of the bounding box.

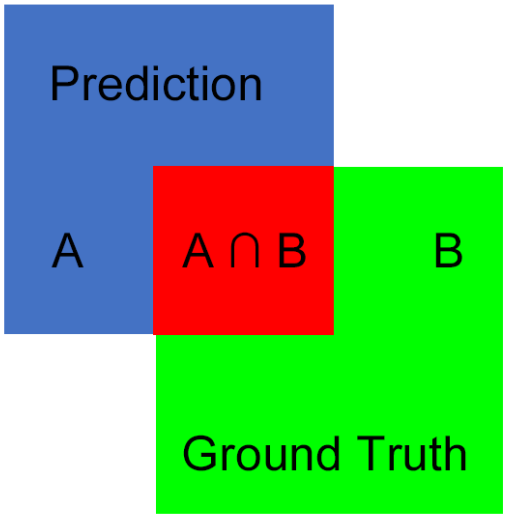

Figure 3.1: Definition of intersection area of IoU: The blue area represents the prediction area, the green area represents the ground truth area, and the red area represents the area of intersection.

2. Select the bounding box with the highest confidence to add to the final output list and delete it from the bounding box list.

3. Calculate the area of all bounding boxes.

4. Calculate the IoU of the bounding box with the highest confidence and other candidate boxes.

5. Remove bounding boxes with IoU greater than a threshold.

6. Repeat the above process until the list of bounding boxes is empty.

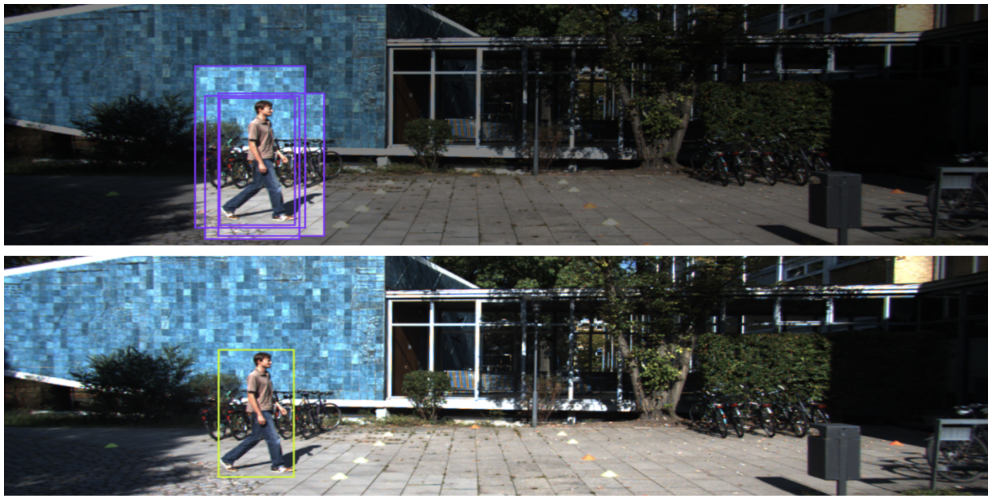

Figure 3.2: An example of NMS process: The upper sub-figure represents the bounding boxes before NMS, and the lower sub-figure represents the final bounding box after NMS.

IoU is also widely used as a metric for performance evaluation. For example, on the KITTI dataset, researchers usually focus on the detection performance when $IoU = 0.7$ to determine the performance of the model. For example, when the IoU of the predicted bounding box and the ground truth bounding box reaches 70% or more, it is regarded as a positive prediction, otherwise it is a wrong prediction.

3.2 Performance Metrics

In this section, we will introduce evaluation metrics commonly used in the field of 3D detection, including precision, accuracy, recall and f1-score. Before introducing each metric, we need to introduce the confusion matrix which is shown in Figure 3.3:

Confusion Matrix		Prediction	
		False	True
Ground Truth	False	TN	FP
	Truth	FN	TP

Figure 3.3: Definition of confusion matrix

1. TP (True Positive): A positive example that is correctly predicted. That is the bounding box of the detected object intersects the bounding box of a ground truth object with an IoU greater than a predefined threshold, like shown in Figure 3.4;

2. TN (True Negative): A negative example that is correctly predicted. That is, the real value of the data is a negative example, and the predicted value is also a negative example; TN cannot be estimated in object detection;

3. FP (False Positive): Positive examples that were incorrectly predicted. That is, the

real value of the data is a negative example, but it is wrongly predicted as a positive example;

4. FN (False Negative): A positive example that is not predicted. That is, the real value of the data is a positive example, but it is wrongly predicted as a negative example. The bounding box of the detected object intersects the bounding box of a ground truth object with an IoU smaller than a predefined threshold, like shown in Figure 3.5.

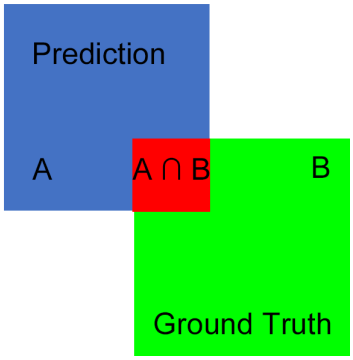

Figure 3.4: Visualization of False Negative

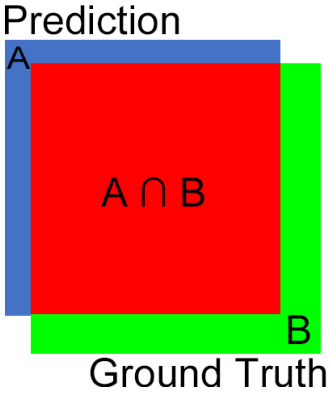

Figure 3.5: Visualization of True Positive

3.2.1 Precision

Precision indicates the proportion of samples that are actually positive in the predicted results, can be calculated as:

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

Precision is appropriate only when the cost of a negative example being mispredicted as a positive example (FP) is high. According to its formula, the higher the precision rate, the smaller the FP. For example, in the detection of particularly distant tiny vehicle scenes, false positives mean that the environment (actually negative) is wrongly predicted as a vehicle object (predicted to be positive). If the precision rate of a 3D detection model for tiny targets is not high, causing many environmental disturbances to be identified as motor vehicles, it may confuse the driving decision-making system and make wrong decisions.

During the evaluation of 3D detection models, we often used indicators such as $AP_{0.5}$ or $AP_{0.7}$, which refer to the performance of average precision when $IoU = 0.5$ and $IoU = 0.7$ respectively. In addition, BEV_AP represents the average precision in bird eye view. 3D_AP represents the average precision in 3D space.

3.2.2 Accuracy

Accuracy refers to the proportion of correctly classified samples to the total number of samples.

The correctly classified sample consists of two parts, namely, the case where the prediction is positive and the real positive is TP and the case where the prediction is negative and the real is also negative is TN.

For 3D detection, the total number of samples is the sum of TP, FP, and FN. Therefore, the calculation of accuracy is as follows:

$$\text{Accuracy } = \frac{TP + TN}{TP + FP + TN + FN}$$

Accuracy is the most straightforward indicator for measuring classification models, but it is only suitable for testing on datasets with uniform distribution of target categories. Assuming that if there are 1000 samples, 999 of which are positive samples, the classifier only needs to be predicted as positive all the time to get 99% accuracy, but it does not actually reflect the real performance of the classifier. When the proportion of samples of different categories is seriously unbalanced, the category with a large proportion will be the most important factor affecting the accuracy rate. In such cases, other metrics must be consulted to fully evaluate the performance of the model. In experiments, more commonly used metrics include AP (Average Precision) and mAP (mean Average Precision). AP refers to the weighted average of the accuracy rate under different thresholds, the weight is the increase in recall from the prior threshold. mAP is the average accuracy rate of the whole classes, which is obtained by weighting the average accuracy rate (AP) of all categories of detection.

3.2.3 Recall

The recall rate indicates the proportion of the actual number of positive samples in the predicted positive samples to the positive samples in the full sample. The recall rate can be calculated as:

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

Using recall as metrics will be suitable when positive examples are incorrectly predicted as negative examples (FN) and the cost is high. According to the formula, the higher the recall rate, the smaller the FN.

3.2.4 F1-score

F1 score is a weighted average of precision and recall, which can be calculated as:

$$F_1 = \alpha \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}},$$

where α represents the scaling factor. For example, in the vehicle model detection system, in order to ensure that the detected vehicles' bounding box area resolutions are as high as possible, the precision is more important at this time; while in the traffic flow statistical model, it is more hoped that there are as few missed vehicles as possible, this is the case when recall is more important. At this time, F-Measure can meet our specific emphasis requirements by adjusting the value of parameter α. Precision reflects the model's ability to distinguish negative samples. The higher the Precision, the stronger the model's ability to distinguish negative samples; Recall reflects the model's ability to identify positive samples. The higher the Recall, the stronger the model's ability to identify positive samples. The F1 score is a combination of the two, and the higher the F1 score, the more robust the model.

3.3 The KITTI Dataset

In this section, we provide information about the KITTI [15] [16] dataset and how we created the VIVAMotion dataset. The KITTI dataset is a dataset jointly sponsored by the Karlsruhe Institute of Technology in Germany and Toyota Technological University Chicago for research in the field of autonomous driving. It includes up to 6 hours of real traffic environments. The data set consists of corrected and synchronized images, radar scans, high-precision GPS information, IMU acceleration information and other modal information. The dataset consists of the following sensor configurations:

1. Two 1.4 megapixel PointGray Flea2 grayscale cameras;
2. Two PointGray Flea2 color cameras with 1.4 million pixels;
3. A 64-line Velodyne rotating LiDAR, 10Hz, angular resolution of 0.09 degrees, about 1.3 million points per second, horizontal field of view of 360°, vertical field of view of 26.8°, and a distance range of up to 120 meters; The laser scanner provide 10 FPS's scanning performace and is able to capture approximately 100k points per cycle, with a vertical resolution of 64.
4. One OXTS RT3003 integrated navigation system, 6 axes, 100Hz, resolution 0.02m, 0.1° as Inertial Navigation System (GPS/IMU).

The KITTI dataset can be divided into Road, City, Residential, Campus, and Person categories. The information contained in the dataset is presented in Table 3.1 below. The total size of the dataset is about 180G. Some samples from the KITTI dataset are shown in Figure 3.6.

Figure 3.6: Samples from the KITTI dataset

On the KITTI dataset, 2D evaluation mainly refers to 2D bounding box AP (bbox AP). In addition to 2D evaluation, we also have to focus on the performance of the model under BEV(Bird's Eye View) AP and 3D bounding box AP(3D_AP). The point cloud BEV (Bird's Eye View) view refers to the projection of the point cloud on a plane perpendicular to the height direction, a typical example of BEV is shown in Figure 3.7. BEV_AP refers to the average precision of the detection frame under the BEV view. The definition of 3D_AP(Average Precision) is similar to that of 2D, which refers to determining whether the prediction is correct or not according to different IoU space occupancy thresholds in 3D space. In addition, AP_R40 [48] is also introduced to KITTI, and use 40 recall positions instead of the 11 recall positions proposed in the original Pascal VOC benchmark which provides a more fair comparison of the results. For more information about performance evaluation, please refer to the previous section performance metrics.

3.4 The VIVA Motion dataset

The design process of the VIVA Motion dataset has been described above. Here we mainly introduce the design pattern and details of the dataset. The VIVA Motion data set uses the Sony Alpha 7 M III professional mirrorless interchangeable-lens camera with Sony G Master 24-70mm optical lens to record the format as 4K (3840×2160) 10bit color depth,

Data	**Path**	**Description**
images	/training/image_2 or /testing/image_2/	Store in png format with a resolution of about 1242*375
LiDAR data	/training/velodyne/	About 100,000 points per frame, stored in bin format
label	/training/label_2/	Including the labels for each detection category, stored in txt format
calib	/training/calib/	Including conversion information between camera and GPS/IMU, camera and Velodyne, etc. stored in txt format.
train/test/validation split file	/ImageSets/	The split files of the training set, test set and validation set.

Table 3.1: Data contained in KITTI dataset

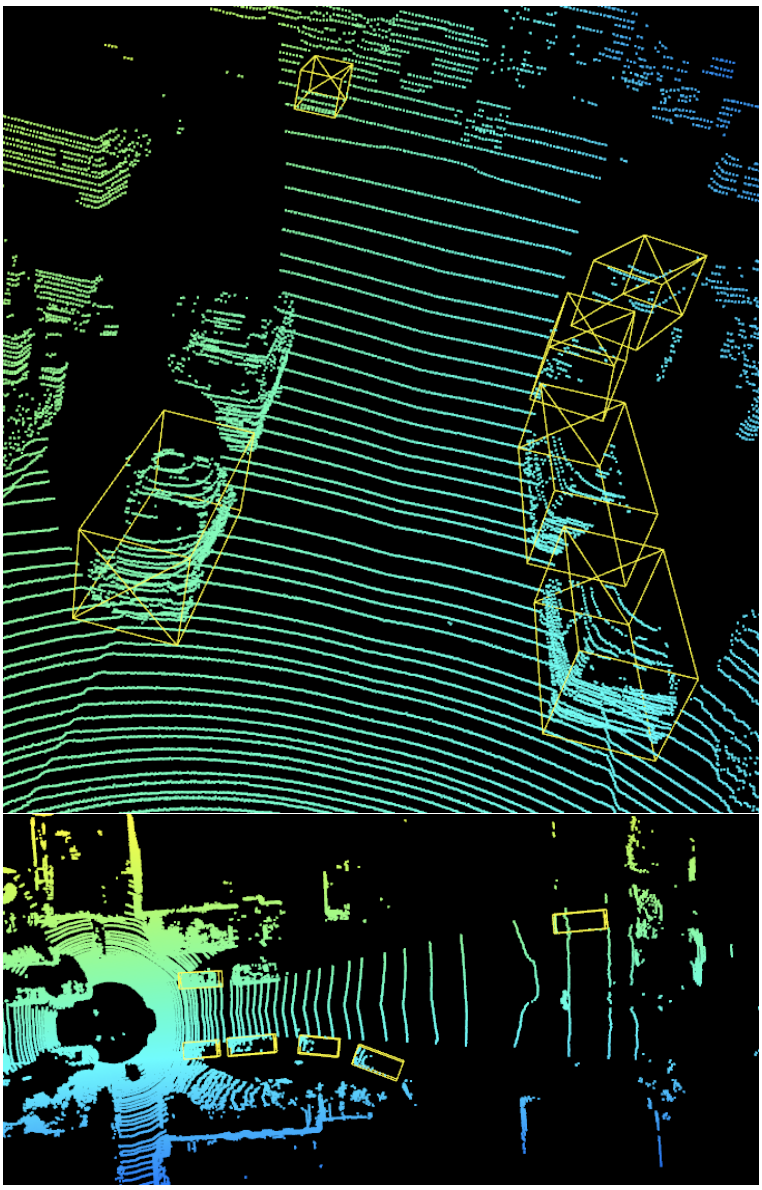

Figure 3.7: Visualization of samples in BEV. The upper subgraph represents the detection result of a 3D detector on the point cloud, and the lower subgraph is its representation on the BEV.

30FPS, 100Mbps to collect the original data and compress it to (1920×1080) 30 FPS to obtain an excellent and sharp video quality. The total number of samples in the dataset is 1798, generated from 423 raw video clips.

Our 2D image quality is significantly ahead of the previous nuScenes dataset and KITTI dataset, the resolutions of the latter two are about 1600×900 and 1242×375 respectively. Thanks to the original recording settings of 3840×2160 and 10bit color depth, our 1920×1080 resolution raw clips are in pretty sharp qualities. This picture quality undeniably surpasses most of the previous mainstream autonomous driving datasets collected with onboard pinhole cameras. Since our motion map is generated by computing the difference two adjacent between video frames, it is largely dependent on the picture quality of the input video. The high-quality raw videos of the dataset allows us to generate high-quality motion maps. Difference images are shown in Figure 3.8.

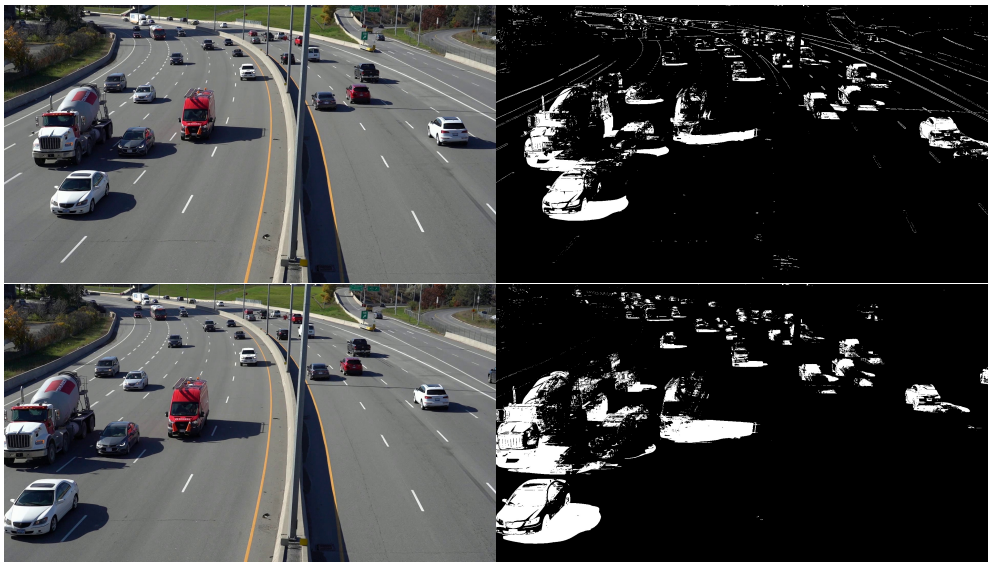

Figure 3.8: An example of a motion map generated using low-quality and high quality raw video. The resolution of the upper sub-figure is only 1280×720, recorded in a 5Mbps rate, while the resolution of the lower sub-figure is 1920×1080 with 30Mbps bit rate

Chapter 4

Use Your Imagination Module

4.1 From Depth Map to Color Space

To make object prediction from point clouds more effective, we propose a method to infer depth information from 2D images using color space mapping. We therefore propose to transform the depth information of a 3D vehicle point cloud box into color gradients frontal view projection. We do this by training a GAN model on the color mapping of the depth features. To this end, we created a 1280-level color/depth lookup table in which the color closer to purple will be considered to be farther from the camera while the color becomes more reddish when closer. This color depth mapping is used to construct the training set of our GAN. To be more specific, let $\{v_1, v_2, ..., v_n\} \in V_i$ denotes the points in i-th LiDAR point cloud frame from dataset, and let p_i be the projection of 3D point v_i on the camera image.

In general, for every v_i in V, its corresponding pixel p_i in the camera view can be found as [15] (4.1),(4.2):

$$\mathbf{p_i} = f(v_i) = \mathbf{P^0}_{\text{rect}}\, \mathbf{R^0}_{\text{rect}}\, \mathbf{T}^{cam}_{velo}\mathbf{v_i} \tag{4.1}$$

$$\mathbf{T}^{cam}_{\text{velo}} = \begin{pmatrix} \mathbf{R}^{\text{cam}}_{\text{velo}} & \mathbf{t}^{\text{cam}}_{\text{velo}} \\ 0 & 1 \end{pmatrix} \tag{4.2}$$

where $\mathbf{P_{0}}_{\text{rect}}$ is the projection matrix in camera coordinates, $\mathbf{R_{0}}_{\text{rect}}$ and $\mathbf{R}^{\text{cam}}_{\text{velo}}$ refer to the rectified rotation matrix and the LiDAR to camera rotation matrix, respectively. And $\mathbf{t}^{\text{cam}}_{\text{velo}}$ is the translator vector from LiDAR to camera.

Let us now consider a given object bounding box projection on the camera reference view. The point with the minimum z-value is defined as the point of depth x_0, then the x values of the other points can be re-corrected as $x_i - x_0$. Considering now the vehicle boxes in the KITTI dataset, we noted that, according to our statistics, the 95th percentile of vehicle box length in KITTI [15, 16] training set is about 4.52 meters. Assuming the length of most vehicles to be within 4.52 meters(we use [0, 4.52] as the standard depth range for generated vehicle point clouds). Therefore, for each point p_i in the bounding box, its corresponding corrected depth information d_i can be estimated as (4.3):

$$d_i = Depth(p_i) = (x_0 - x_i)/4.52 * 1279, (x_0 - x_i) \geq 0 \tag{4.3}$$

where x_i comes from 3D point v_i corresponding to p_i.

Now, to convert the corrected depth value into a color map, the RGB value for each pixel p_i in the bounding box of an object is converted according to the following transformation (4.4):

$$\begin{cases} (255 - d_i, 255, 255) & ,0 \leq d_i < 256 \\ (0, d_i - 255, 255) & ,256 \leq d_i < 512 \\ (0, 255, 255 - d_i + 512) & ,512 \leq d_i < 768 \\ (d_i - 768, 255, 0) & ,768 \leq d_i < 1024 \\ (255, 255 - d_i + 1024, 255) & ,1024 \leq d_i < 1280 \\ (0, 0, 0) & ,otherwise. \end{cases} \tag{4.4}$$

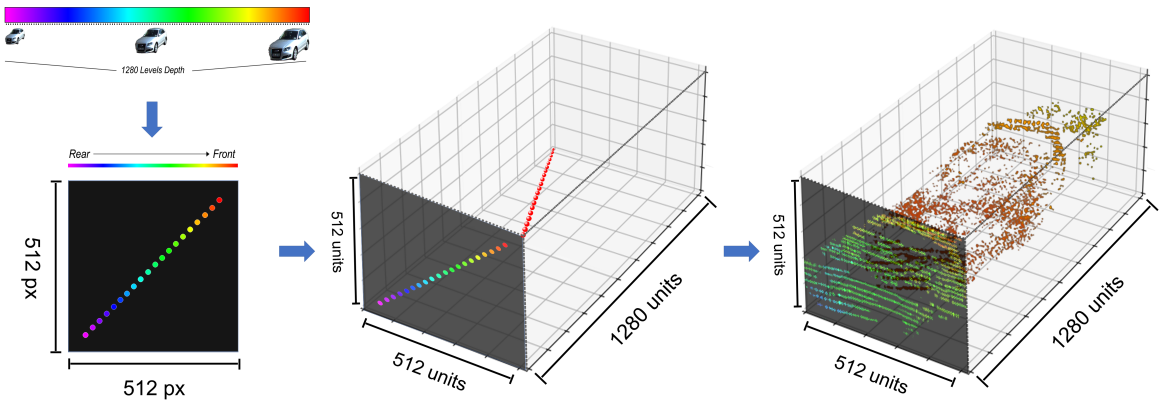

Figure 4.1: Depth-Color Projection: The depth-color map correspondence is shown on the upper left. The example image on the left is mapped to a diagonally distributed depth point cloud (shown in the middle). On the right is an example of 3D shape after depth restoration of the model.

Through the above transformation, for each given bounding box we can calculate its RGB depth map. The process is shown in Figure 4.1. Note that this depth-color projection process can simply be reversed for the process of projecting the RGB depth map back to the 3D vehicle box. During training, we manually select vehicle boxes (including relatively complete vehicle shapes) as ground truth to make GAN learn the semantic information of translation from 2D images to RGB depth map.

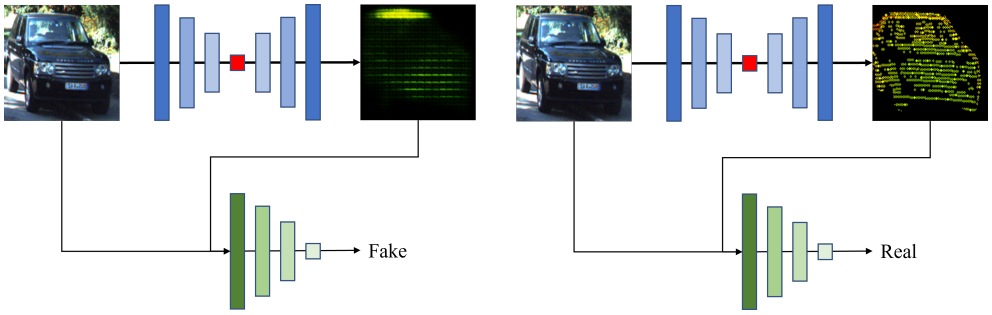

Figure 4.2: Adversarial Learning Process for Conditional GAN: On the left is shown an instance where the discriminator successfully identified the image from generator as fake. The right side shows an example where the generator successfully cheated the discriminator. The self-attention block is represented by the middle red block.

4.2 Sentient Generative Adversarial Network with Attention

4.2.1 Objective

In general, the generator of a conditional GAN in an image translation task learns a mapping from input image $real_a$ and a random noise vector z to output image $fake_b$, which would be compared with ground truth $real_b$ by the discriminator. The generator G tries to produce high quality images that discriminators can not distinguish. In the middle of the U-Net structure we use a self-attention block to better aggregate features. This procedure is described in Figure 4.2. Considering that the LiDAR information embedded in the RGB LiDAR map generated in the previous step behaves like texture features, we add a sentient loss $L_{sentient}$in our GAN. Based on a pretrained VGG19 [49] model as feature extractor, we map it to a pair of generated samples and real samples in the generator. This process is described in Algorithm 4.1, where N denotes the number of layers in the extractor, $E(real_b)$ and $feature_{real_b}$ denote features from $real_b$ generated by the extractor which is same for

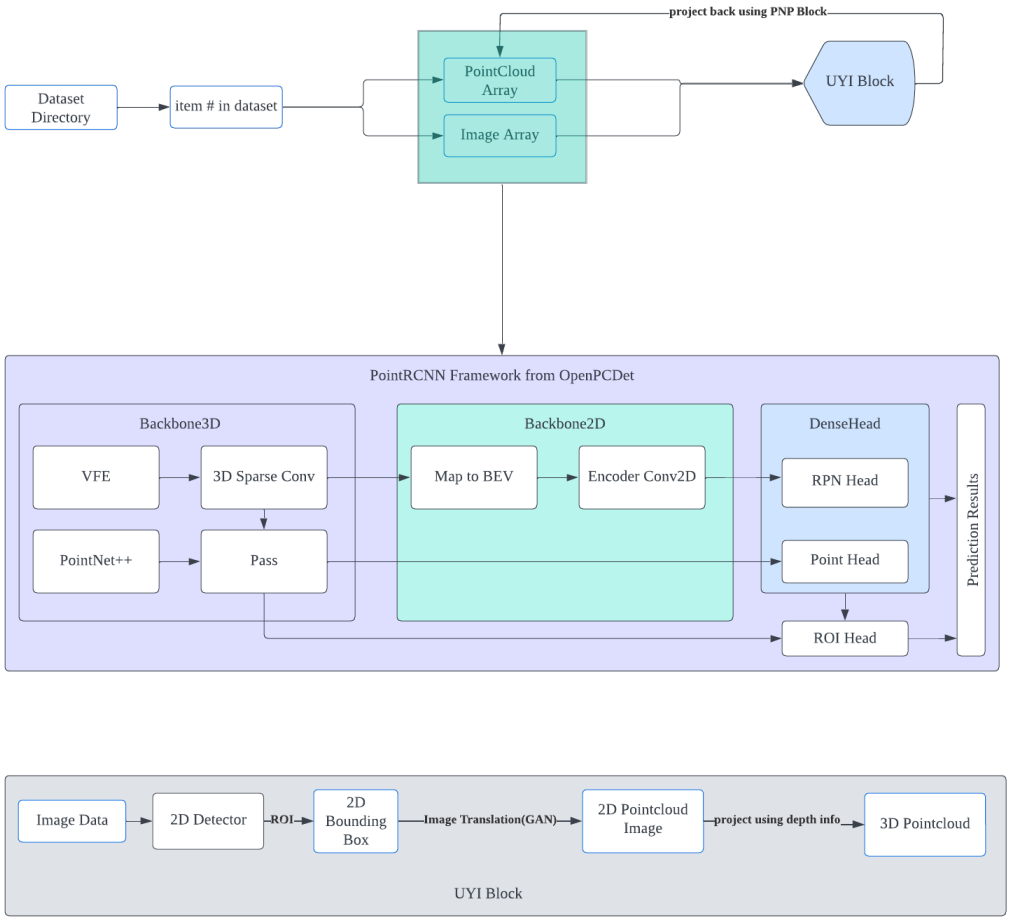

Figure 4.3: The Workflow of UYI Module: The UYI module takes the RGB image and point cloud objects from the Data Loader, then projects the generated targets back to the original point clouds to produce high-quality point clouds which serve as input of the subsequent detection network.

$fake_b$. $L_{sentient}$ denotes sentient loss, $feature_{real_b/fake_b}['relu_i']$ is the feature output of the i-th layer of the encoder, extracted from $real_b$ or $fake_b$.

Algorithm 4.1 Calculation of sentient loss in Generator

Input: $N \geq 0$, $real_a \neq \emptyset$, $real_b \neq \emptyset$
  Initialization:
  $L_{sentient} \leftarrow 0, E \leftarrow VGG19$
  $W \leftarrow [1.0/32, 1.0/16, 1.0/8, 1.0/4, 1.0]$
  $feature_{real_b} \leftarrow \mathrm{E}(real_b)$
  $feature_{fake_b} \leftarrow \mathrm{E}(fake_b)$
  while $i \leq N$ **do**
    $L_{sentient} \leftarrow L_{sentient} + W[j] * |feature_{real_b}['relu_i'] - feature_{fake_b}['relu_i']|$
  end while
  return $L_{sentient}$

The loss function of our generator can be described as (4.5):

$$\begin{aligned} L_{Generator} =& L_{L1}(real_b, fake_b) \\ &+ L_{GAN}(Discriminator(fake_b), True) \\ &+ L_{Sentient}(real_b, fake_b) \end{aligned} \tag{4.5}$$

,where L_{GAN} is the MSE loss from discriminator and L_{L1} denotes the L1 loss. L_{L1} measures the average absolute difference between the predicted image and the target image. It encourages the generator to produce images that are similar to the real images in terms of their overall structure and appearance. L_{GAN} measures the average squared difference between the predicted image and the target image. It penalizes larger errors more heavily than smaller ones, making it more sensitive to outliers. $L_{Sentient}$ can help improve the texture quality of the generated images by encouraging the generator to learn and reproduce the texture patterns from the real samples. $L_{Sentient}$ can enable GAN to learn the texture

patterns from the real samples and reproduce them in the generated images, resulting in outputs that are more visually appealing and realistic.

To test the importance of adding $L_{sentient}$ and self-attention block to generator, we also define the situation when $L_{Generator}$ does not contain $L_{sentient}$ as (4.6):

$$\begin{aligned} L_{Generator} =& L_{L1}(real_b, fake_b) \\ &+ L_{GAN}(Discriminator(fake_b), True) \end{aligned} \tag{4.6}$$

4.2.2 Model Design

As shown in Figure 4.3, our UYI block acts as a point cloud augmentation module before the detection phase. It accepts RGB and LiDAR input, integrates a YOLO detector into the RGB image, converts the content of the obtained detection area into a vehicle RGB depth map and then forwards it to the GAN to generate a complete vehicle RGB depth map. The RGB depth map will then be processed using color mapping as explained in 4.1 to generate vehicle point cloud box and project it back into the original point cloud. These procedures enables us to obtain a high-quality point cloud. The enhanced high-quality point cloud can effectively improve the detection performance of the model, which we will be presented in the next section of experiments.

4.3 Projection Awareness Positioning Module

In order to project the generated vehicle box back accurately into the original point cloud, we need the (x, y, z) coordinates of the target projection location. In practice, y can be

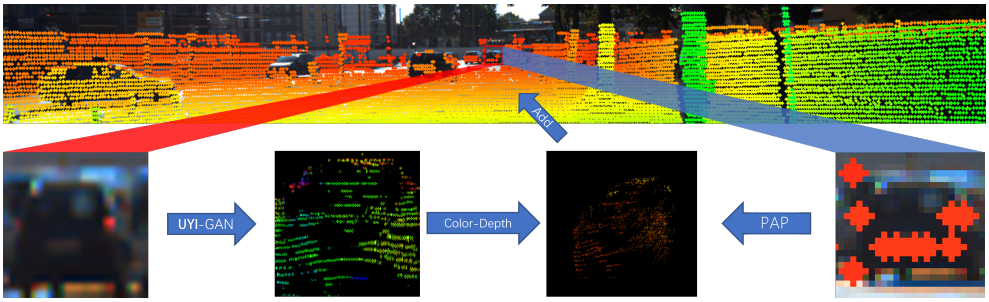

Figure 4.4: Projection Awareness Positioning (PAP) Module: The leftmost of the second row is the ROI area of a vehicle output by the 2D detector, we generate its RGB point cloud map through UYI-GAN, and project it back to 3D space through color depth conversion to generate a 3D vehicle box. Finally, the target projection position of the 3D vehicle box in the original point cloud can be calculated according to the position of the points projected to the ROI area on the RGB image.

obtained as a direct reflection from RGB to the point cloud to obtain a frustum range. The mean variance of the road height within a single sample is only 0.037m for the KITTI dataset, which enables us to simply interpret the lowest height value of the current point cloud as the reference value. The axis which needs more attention is the depth information, which is the value of the x-coordinate in the KITTI dataset. A typical approach is to use

Methods	**Average Error(distance $\geq$ 50m)**	**Mean Squared Error(distance $\geq$ 50m)**	**Average Error(Full)**	**Mean Squared Error(Full)**
Simple Linear Regression(MLR)	4.78 meters	45.71	2.03 meters	25.45
MLR+Polynomial Features	**3.06 meters**	**24.33**	**1.56 meters**	**21.91**
Previous Triangle Method	11.61 meters	321.00	4.09 meters	66.47

Table 4.1: Comparison of methods for estimating object depth from RGB: Considering that the error increases with the distance of the object, the experimental results are divided into the test of long-distance objects greater than 50 meters and the test of all cases. Kitti uses meter as the unit for 3D points' location, thus we are using meter as the unit to represent the error. It could be observed that our MLR with polynomial features achieves the best results in both categories. We achieved an average error of only 3.06 meters for objects over 50 meters away, and 1.56 meters for all objects.

the principle of similar triangles, we have:

$$\begin{aligned} Estimated\ Distance = height_{real} \\ * focal_v / height_{pixel} \\ + (length/2) \end{aligned} \tag{4.7}$$

However, after calculating the positions of all the vehicle boxes in the KITTI training set using similar triangle method we found that this approach could produce an average error of 11.61 meters. Therefore we came up with the idea to use the point cloud corresponding to the original point in the 2D detection box as the position information. First, for each 2D detector that returns an ROI region, we transform it to a complete vehicle RGB depth map. In this case, if the number of real LiDAR points inside the 4.52 meters bounding box, we calculate the geometric center of the resulting vehicle box. This process is shown in Figure 4.4. As it can be calculated that the average width of the vehicle boxes in the KITTI training set is 1.63 meters and 1.80 meters at 95th percentile. At the same time, considering the 95th percentile of vehicle box length in the training set is about 4.52 meters (mentioned in Section 3), thus the extent of the 3D area can be established as $O(x, y, z)$ using (4.8):

$$\begin{gathered} \{vi_x | O_x - 1.63/2 \leq vi_x \leq O_x + 1.63/2, vi \in V\}, while \\ \{vi_y | O_y - 4.52/2 \leq vi_y \leq O_y + 4.52/2, vi \in V\}\ and \\ \{vi_z | vi_z \in R, vi \in V\}. \end{gathered} \tag{4.8}$$

According to our statistics, 86% of the vehicle boxes can be localized using this method, with a localization accuracy ($IoU \geq 70\%$) of 94%. We found that 3 points is the minimum

required to achieve this average accuracy or it would be too sparse for positioning an object properly. For the situation where the original point cloud projection on the target object area is less than 3 points, we trained a multinomial machine learning model based on multinomial linear regression [10] to predict the target depth and use the objects in KITTI's training set as ground truth. For each training sample, the inputs were defined as 2D bounding box width, 2D bounding box height, P2[1][1], 2D bounding box center (x, y), with the true depth as the training target and normalization performed on all the variables to (0,1]. Table 4.1 shows a significant improvement for depth estimation after this process, especially for the distant vehicle box, where we succeed in reducing the mean error to 1.56m. It could be observed that our MLR with polynomial features achieves the best results in both categories. We achieved an average error of only 3.06 meters for objects over 50 meters away and 1.56 meters for all objects. In this case, we align the obtained depth value as the geometric center point in the vehicle point cloud box generated by the previous section, thus making it possible to estimate the projected position of the target when the points information for positioning are not rich.

4.4 Experiment for UYI

The work and the experiments presented in this chapter have been submitted to RA-L(IEEE Robotics and Automation Letters).

4.4.1 Environment Preparation

As the variance of toolkit and library versions used in 3D detection implementation can result in performance differences, we believe that it is necessary to describe the software

Component	Version
OS	Ubuntu Server 18.04 LTS
Python	3.7.6
Anaconda	2022.05
CUDA	11.3
CUDNN	8.4.0
PyTorch	1.12.1
Tensorflow 1.x	1.15.0
Tensorflow 2.x	2.8
Spconv	2.2.6
OpenCV-Python	4.6.0.66
Torchvision	0.13.1
Sklearn	0.0.post1
Pandas	1.4.2
MMCV	1.6.2
MKL-service	2.4.0

Table 4.2: Software and runtime environment

and hardware configuration of the system environment used in this thesis. Our code environment management is based on Torch 1.7 and spconv 1.5, using a Python 3.7 virtual environment running on Anaconda 3. As our hardware platform is based on the NVIDIA Ampere architecture, which only supports CUDA 11 and above, and as compatibility between CUDA 11 and various package versions could be complex, our environment configuration are provided in Table 4.2 and 4.3 and remain consistent throughout the thesis.

4.4.2 Hyper-parametery Settings

Before introducing the experimental results, it is also necessary to introduce the hyper parameters used for training UYI. These hyperparameters are tuned for optimal performance and are shown in tables 4.4. While adjusting the hyper parameters, we monitored

Component	**Model**
CPU	AMD Threadripper 2950x 16c 32t
GPU	2 × nVIDIA RTX 3090
Memory	DDR4 128GB
Storage	PCI-E SSD

Table 4.3: Hardware environment

Hyper-params	**Value**
Learning rate	0.0002 as initial learning rate for adam
Learning rate policy	linear
# of generation filters in the last conv layer	64
# of discriminator filters in the last conv layer	64
# of threads	8
Batch_size	1
Load_size	768
Crop_size	512
No_flip	True
No_dropout for the encoder	True

Table 4.4: Hyper params for UYI

multiple variables involves during training, including learning rate, batch size, number of epochs while monitoring the loss function to find the optimal values. We first started with default params available in public repos, most of them are good starting points because they are optimized for most general cases. These params are usually optimized for general use cases, and can be used as a baseline for further tuning. And while we are adjusting params, we also monitored the loss function throughout the training process. For an instance, if the loss function is not decreasing, it might be an indication that the params need to be adjusted.

4.4.3 Qualitative Results

In this section we present the results showing the performance improvements of typical baseline models by the high-quality point cloud enhanced with our UYI module. In part 1, we first describe the performance of our UYI module on VoxelRCNN [8] and PV-RCNN [44] by comparing our full UYI module with the UYI Module that replaces the point cloud generation block with the previous TFMM method. In part 2, we perform ablation studies on the performance improvement using UYI module on typical 3D models.

4.4.3.1 Comparision with traditional flipping and best matching method

In this section, we replace, in the UYI module, the point cloud generation block based on projection awareness positioning (PAP) with a simple TFMM point augmentation block as proposed in BtcDet (Section 3).

We followed the TFMM method proposed in BtcDet to perform the best matching of the vehicle point cloud completion. Other parameters include a random scaling of [0.95, 1.05], and using 4 as the Number of Point Features. The performance results of the point cloud enhanced for comparison with two designs of UYI module on Voxel-RCNN and PV-RCNN are as Table 4.5. The improvement of the original UYI module is greater than that of the UYI using TFMM compare to the PV-RCNN and Voxel-RCNN baselines. The lead is most significant in 3D_AP in the Hard category, reaching 2.03% and 2.11% on PV-RCNN and VOXEL-RCNN, respectively, followed by Medium and Easy.

4.4.3.2 Impact of attention and sentient module

In this section we assess the impact on vehicle detection results of using the attention and sentient modules in UYI. As in the previous section, we use PV-RCNN and VOXEL-RCNN

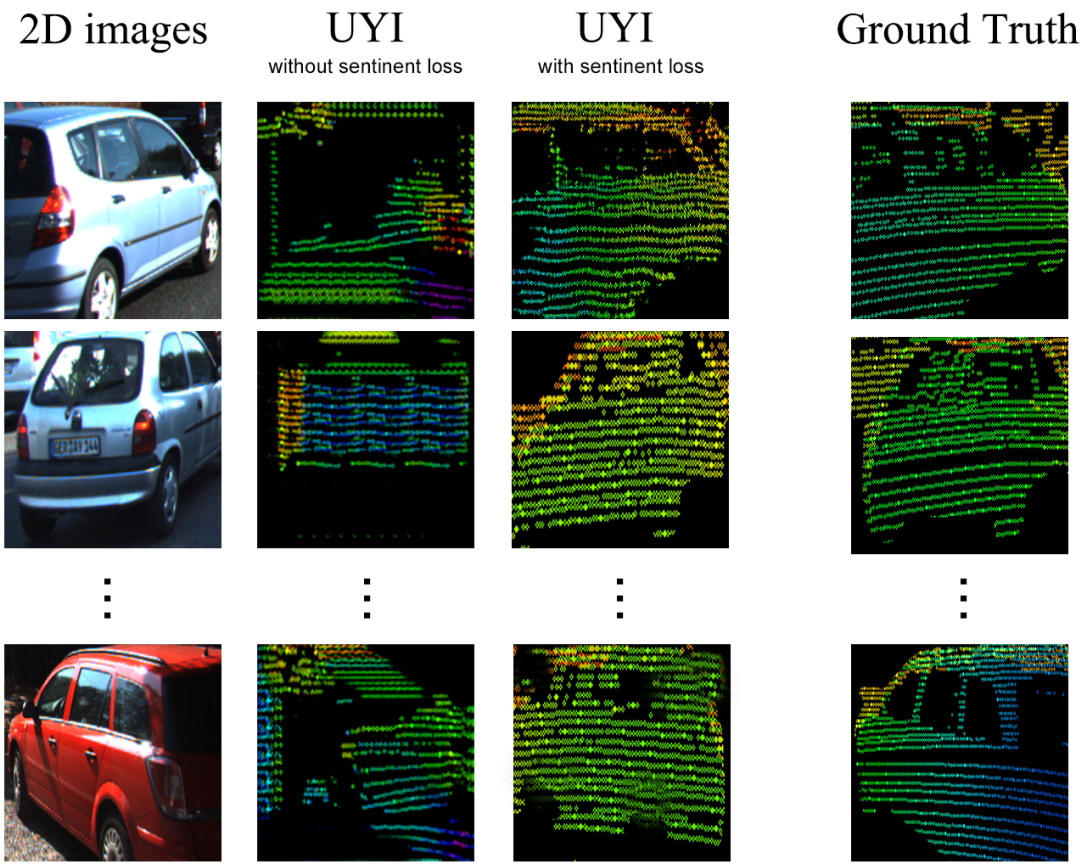

Figure 4.5: Comparison of generated 2D point cloud images between UYI with and without attention and sentient module on validation set: On the left is the input 2D image, the second from the left is the output of UYI without attention and sentient module, the third from the left is the output of UYI with attention and sentient module, and the right is groundtruth. We want to minimize the gap between the output of UYI and the groundtruth, so the criterion for optimization is how similar the output of UYI is to its corresponding groundtruth.

as well in this section because they are sufficiently strong baselines. The results are shown in Table 4.6 and Figure 4.5. The improvement of UYI with attention and sentient module is greater than that of the UYI without attention and sentient module. For PV-RCNN, in the category of 3D_AP we have a 2.16% increment for Hard objects, 0.29% in medium and 1.03% in easy. In VOXEL-RCNN we observe a more significant improvement for both BEV_AP and 3D_AP, with a 0.91% increase in 3D_AP for the hard case and with smaller boost in Medium and Easy. From the evaluations we can find that the attention and sentient modules can indeed generate better quality point cloud images, and these point cloud images of higher quality produce better 3D point clouds, therefore leading to more accurate detections. In fact, UYI without attention and sentient module produces unsatisfactory results on most samples in the validation set. And we can observe that UYI with attention and sentient modules can make UYI to pay more attention to the texture characteristics of the reconstructed objects, making the reconstruction process more accurate.

4.4.3.3 Comparison with State-of-the-art methods

To better illustrate the general improvements among previous models, we performed ablation experiments on several existing outstanding models, each of which was state-of-the-art when introduced. These are SECOND [61], PointPillar [29], PartA2 [46], PV-RCNN [44], Voxel-RCNN [8], BtcDet [60], SFD-Net [57]. The results are shown in Table 4.7 and 4.8. For Table 4.7, it can be seen that the high quality point clouds generated based on UYI module have led to significant improvements compared to baseline. More notable results were brought by PartA2, PV-RCNN and BtcDet, with also similar improvements in BEV_AP. Finally, we achieved 0.63% and 0.11% in Hard and Medium categories of SFD's 3D_AP. For Table 4.8, more notable results were brought by PartA2 and PV-RCNN, with

Methods	**Metrics**	**Easy**	**Medium**	**Hard**
PV-RCNN Baseline	BEV_AP	92.86	88.93	**88.74**
	3D_AP	91.99	82.86	80.40
UYI(TFMM)+PV-RCNN	BEV_AP	93.93	89.43	88.53
	3D_AP	92.04	82.91	80.35
UYI(Original)+PV-RCNN	BEV_AP	**94.63**	**90.28**	88.43
	3D_AP	**92.97**	**83.11**	**82.38**
VOXEL-RCNN Baseline	BEV_AP	95.35	90.83	88.64
	3D_AP	92.06	82.64	80.10
UYI(TFMM)+VOXEL-RCNN	BEV_AP	93.37	90.65	88.77
	3D_AP	92.10	82.99	80.46
UYI(Original)+VOXEL-RCNN	BEV_AP	**95.76**	**91.06**	**88.84**
	3D_AP	**92.31**	**83.09**	**82.57**

Table 4.5: Comparison of detection performance between using TFMM and UYI modules for pseudo point cloud generation on PV-RCNN and VOXEL-RCNN. In this table we replace our UYI module by a simple TFMM point cloud augmentation block.

Methods	**Metrics**	**Easy**	**Medium**	**Hard**
PV-RCNN Baseline	BEV_AP	92.86	88.93	**88.74**
	3D_AP	91.99	82.86	80.40
UYI(w/o attention and sentient module) + PV-RCNN	BEV_AP	93.40	**90.40**	88.53
	3D_AP	91.94	82.82	80.22
UYI(w/o attention and sentient module) + PV-RCNN	BEV_AP	**94.63**	90.28	88.43
	3D_AP	**92.97**	**83.11**	**82.38**
VOXEL-RCNN Baseline	BEV_AP	95.35	90.83	88.64
	3D_AP	92.06	82.64	80.10
UYI(without sentient loss) + VOXEL-RCNN	BEV_AP	94.88	91.05	88.76
	3D_AP	92.23	82.48	81.66
UYI(with sentient loss) + VOXEL-RCNN	BEV_AP	**95.76**	**91.06**	**88.84**
	3D_AP	**92.31**	**83.09**	**82.57**

Table 4.6: Comparison of detection results between adding attention and sentient module or not in UYI module for pseudo point cloud generation on PV-RCNN and VOXEL-RCNN.

Method	Reference	Metrics	UYI Easy	UYI Medium	UYI Hard	Baseline Easy	Baseline Medium	Baseline Hard	Improvement Easy	Improvement Medium	Improvement Hard
SECOND [61]	SENSORS 2018	BEV_AP	**92.27**	87.91	**86.63**	91.92	87.92	85.39	*+0.35*	*-0.01*	*+1.24*
		3D_AP	**88.88**	**79.15**	**76.07**	88.45	78.75	75.72	*+0.43*	*+0.4*	*+0.35*
PointPillar [29]	CVPR 2019	BEV_AP	92.04	**87.93**	85.39	92.04	87.74	86.65	*0*	*+0.19*	*-1.26*
		3D_AP	**88.51**	**78.78**	**75.72**	88.24	78.25	75.32	*+0.27*	*+0.53*	*+0.40*
PartA2 [46]	TPAMI 2020	BEV_AP	**92.96**	**88.21**	**87.56**	90.86	86.26	85.80	*+2.10*	*+1.95*	*+1.76*
		3D_AP	**91.83**	**82.06**	**79.57**	89.93	80.24	77.89	*+1.90*	*+1.82*	*+1.68*
PV-RCNN [44]	CVPR 2020	BEV_AP	**94.63**	**90.28**	88.43	92.86	88.93	88.74	*+1.77*	*+1.35*	*-0.31*
		3D_AP	**92.97**	**83.11**	**82.38**	91.99	82.86	80.40	*+0.98*	*+0.25*	*+1.98*
VOXEL-RCNN [8]	AAAI 2021	BEV_AP	**95.76**	**91.06**	**88.84**	95.35	90.83	88.64	*+0.41*	*+0.23*	*+0.20*
		3D_AP	**92.31**	**83.09**	**82.57**	92.06	82.64	80.10	*+0.25*	*+0.45*	*+2.47*
BtcDet [60]	AAAI 2022	BEV_AP	**93.89**	**91.79**	**89.28**	93.46	89.53	87.44	*+0.43*	*+2.26*	*+1.84*
		3D_AP	**92.81**	**85.80**	**83.08**	92.17	82.39	80.96	*+0.64*	*+3.41*	*+2.12*
SFD [57]	CVPR 2022	BEV_AP	**96.10**	91.68	91.08	95.85	91.92	91.41	*+0.25*	*-0.24*	*-0.33*
		3D_AP	**95.64**	**88.42**	**85.70**	95.01	88.31	85.69	*+0.63*	*+0.11*	*+0.01*

Table 4.7: Vehicle performance improvement on state-of-the-art methods on the KITTI validation set. From top to bottom: SECOND, PointPillar, PartA2, PV-RCNN, VOXEL-RCNN, BtcDet and SFD, sorted by publication time. In the middle of the table is the baseline result based on official codes been open sourced, at the left side is the result after adding the UYI Module.

Method	Reference	Metrics	UYI Easy	UYI Medium	UYI Hard	Baseline Easy	Baseline Medium	Baseline Hard	Improvement Easy	Improvement Medium	Improvement Hard
SECOND [61]	SENSORS 2018	BEV_AP	**60.86**	**57.03**	**51.54**	58.67	53.20	48.64	*+2.19*	*+3.83*	*+2.9*
		3D_AP	**53.94**	**49.66**	**45.00**	53.64	47.45	42.74	*+0.30*	*+2.21*	*+2.26*
PointPillar [29]	CVPR 2019	BEV_AP	**69.74**	**62.75**	**57.57**	59.50	54.37	50.13	*+10.24*	*+4.18*	*+3.71*
		3D_AP	**63.50**	**56.79**	**51.54**	55.11	49.57	44.78	*+8.39*	*+3.61*	*+3.38*
Point-RCNN [45]	CVPR 2019	BEV_AP	**73.82**	**64.62**	**55.72**	64.17	57.81	51.16	*+8.65*	*+3.40*	*+2.28*
		3D_AP	**69.41**	**58.79**	**50.04**	62.29	54.88	48.22	*+7.12*	*+1.95*	*+0.91*
PartA2 [46]	TPAMI 2020	BEV_AP	**71.97**	**65.65**	**60.53**	59.01	52.15	48.15	*+12.95*	*+6.75*	*+6.19*
		3D_AP	**66.85**	**59.61**	**54.28**	54.62	48.32	43.41	*+12.95*	*+5.65*	*+5.43*
PV-RCNN [44]	CVPR 2020	BEV_AP	**70.95**	**63.64**	**58.61**	58.91	51.43	47.67	*+12.05*	*+6.11*	*+5.47*
		3D_AP	**66.43**	**59.17**	**54.00**	55.78	48.42	43.87	*+10.65*	*+5.37*	*+5.06*

Table 4.8: Pedestrian performance improvement on state-of-the-art methods on the KITTI validation set. From top to bottom: SECOND, PointPillar, PointRCNN, PartA2, PV-RCNN, BtcDet, sorted by publication time. Since the authors of SFD and VOXEL-RCNN did not publish the training config for Pedestrian class, therefore we could not reproduce their pedestrian metrics accurately. For this reason in the comparison of pedestrians we have to exclude SFD. The definition of other parts of the table is the same as Table 4.7.

Metrics(in average)	UYI(Ours)	TWISE(Global level)	TWISE(Object level)	Original KITTI
Inference speed	$\sim 32ms$	$\sim 41ms$		-
LiDAR points generated per frame	14053	192071	56117	-
Total LiDAR points per frame	133278	311296	175342	119225

Table 4.9: Comparison of inference speed of UYI and TWISE. The speed is calculated as averages for inference over all samples on KITTI's validation set on a single 3090.

also similar improvements in BEV_AP. We achieved 5.43% ,5.65% and 12.95% leap forward in Hard, Medium and Easy categories of PartA2's 3D_AP. For the sake of rigor, all the data in the figures are rounded from the original four-digit decimal precision to two digits, and the numbers shown in Improvement column are also rounded up after calculation. Therefore, the values in the Improvement column may be different from the data calculated directly through the tables.

It can be seen that the UYI-enhanced PV-RCNN achieves a 1.97% 3D_AP performance improvement in the hard category, and a small improvement in Medium and Easy. The Hard category on Voxel-RCNN also achieved a 3D_AP improvement of 2.47%, and a similar improvement is also obtained in BtcDet and SFD-Net. The reason we have made more significant progress in the Hard category is that considering that KITTI's definition of hard catogory includes high occlusion or distant objects, and these objects could benefit more from our method while nearby objects are already obvious enough to be detected by the models. We also achieved similar improvements on BEV_AP. At the time we wrote this paper the SFD-Net which uses pseudo-point cloud presents the SOTA performance on KITTI's benchmark website. We successfully bring an improvement for 0.63% in the Easy category, with also 0.11% and 0.01% higher in Medium and Hard. Although there was a 0.33% drop in BEV's Hard category, it also gained 0.25% and 0.24% in Medium and Easy respectively. Given that pedestrian class are smaller objects than vehicles and suffer

more from LiDAR sparsity, it should have larger performance gains, which is verified in Table 4.7. At the same time, we can also observe in Table 4.8 that the improvement of the Pedestrian category is more significant than that of Vehicle category with an average of 7.88%, 3.76% and 3.41% increment on easy/medium and hard category of 3D_AP among the models we evaluated.

At the same time, according to our statistics in TABLE 4.9, the UYI module only adds a latency of about 32ms to the model in general while the latency of using TWISE global completion is about 41ms. Object level completion latency using TWISE is almost the same. At the same time, point cloud completion using object-level methods can greatly reduce the number of generated points.

Therefore our UYI Module could generally improve the point cloud quality and improving the quality of point clouds indeed leads to an improvement in the performance of 3D detection.

Chapter 5

Motion Map Assisted Enhancement Module

5.1 Motion Map Generation

The objective of motion map generation is to build an image translation model to learn the prior knowledge needed to infer the generation of motion map from 2D images, here referred as the motion prior in the following. We propose a 2D motion map dataset, called **VIVA Motion**, which is the first autonomous driving dataset to isolate motion feature maps from motion videos and can be used to train a variety of 2D motion prior knowledge-related models. To generate a usable motion map for learning, we first need a motion map dataset. More specifically, the dataset we need needs to meet the following specific conditions.

1. The dataset needs to be continuous, preferably a video dataset.

2. The dataset must keep the recording camera in a fixed and stable position.

3. The dataset must be in HD format.

As the data in KITTI dataset is non-continuous, we moved to continuous datasets such as Waymo and nuScenes. Initially, we tried to label the data directly on top of Waymo and nuScenes and form a derived dataset for motion map generation training that is relatively smaller. However, we found two problems: the nuScenes team did not record on a very diverse path and only included a limited number of scenes in the dataset, thus it could be difficult for the model to learn robust environment features. The second is that there are very few scenes in the Waymo and nuScenes datasets that satisfy condition 2, which further limits the usability of both datasets.

We have also tried using an automated approach to annotate our dataset with an algorithm that automatically determines if there is movement in the current scene. However, we found that it was difficult to define for a video whether only part of the environment was moving or whether the whole environment was moving. That is, it was difficult to distinguish between a situation where the camera was stationary or where the car carrying the camera was moving. So we decided to build a new motion dataset, called VIVA Motion. Our VIVA Motion dataset will be available to the public later in 2023.

To build the dataset, first we capture a number of video clips based on a variety of road conditions, usually around 1 to 2 seconds in length. The camera is stabilized during the recording process. For each video clip, a frame-to-frame difference weighted map is then calculated as our motion map. This process is shown in Figure 5.1.

Specifically, for each video clip, we initialize a two-dimensional matrix of all zeros as a motion map, calculate the difference pixels between all key frames in the video clip one by

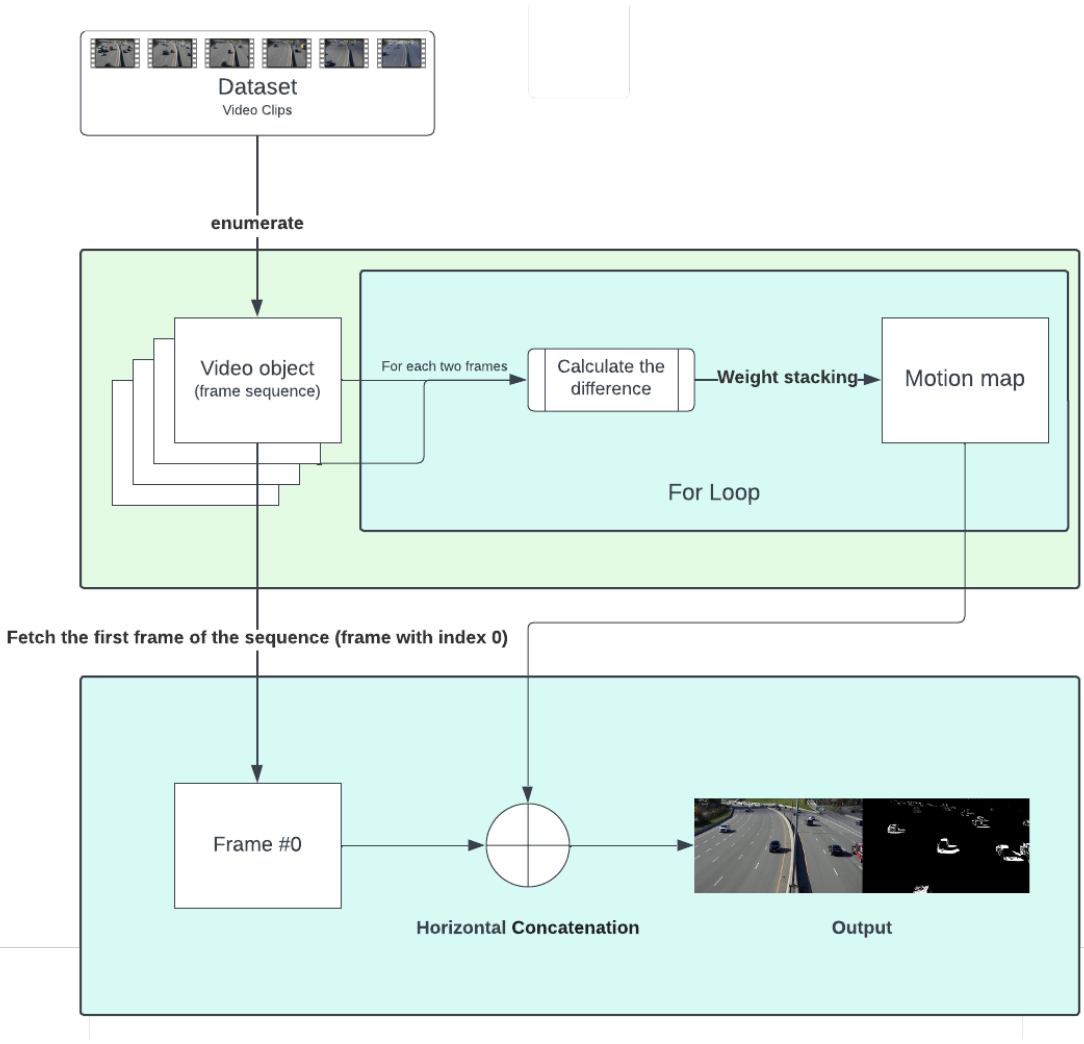

Figure 5.1: The process of motion map generation

one, record them with weights in the temporary motion map matrix, repeat the process and stack the temporary motion map matrix by point-wise addition. In more detail, the key frame is defined as the frame with index times of 5 starting from 0. The weight used is calculated by dividing 255 by the number of key frames in the clip. Then we save this motion map matrix and apply a binarization filter to it. Here we choose 128 as the threshold, which means that if we consider the motion map matrix as a 2D image, the pixels above 128 will appears to be white, otherwise they will be zeros.

Some keyframes motion maps from our dataset are shown in Figure 5.2. More samples can be found in Appendix .1.

5.2 Motion-based Enhancement

After obtaining a VIVA Motion dataset consisting of pairs of 2D images and motion maps, we attempted to learn the mapping relationships from the 2D images to their motion maps. This can be done by training an image translation module, which is the same as the one used in UYI and therefore will not be repeated here as it has already been described above. We can then obtain an image translation model that can generate a motion map from the 2D image it receives as input, this process can be visualized as Figure 5.3. After this, the next step is to enhance the original point cloud based on its motion map generated from 2D image. This process is named MAE for further reference.

For each input 2D image and point cloud pair, we feed the 2D image into our image translation model for motion estimation and obtain an inferred motion map, which is then overlaid with the projection of the point cloud on the RGB image to obtain the set of interest points to be densified.

Figure 5.2: Samples from VIVA Motion dataset

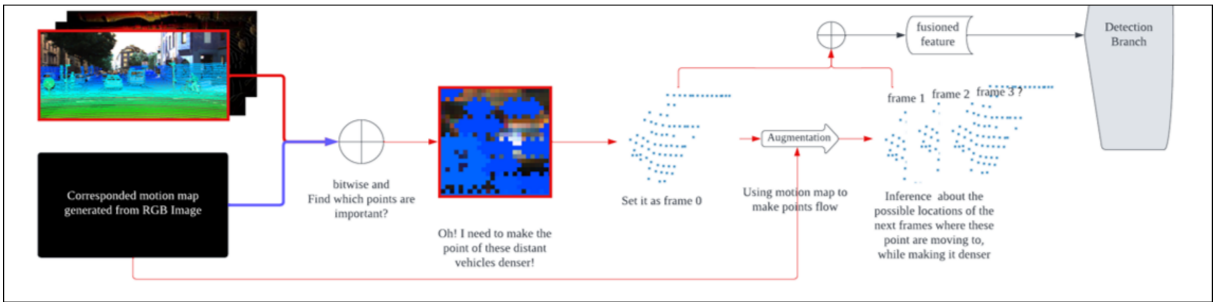

Figure 5.3: The workflow of MAE module

We use random flow as the density method. That is, for a selected point p in the set of points of interest P, we repeated n times to obtain a new set of points $\hat{P}$ with a random flow range of [-5,5] units in x,y coordinates. In the experiments in this thesis, we use the point cloud generated when n is 5 as the enhanced point cloud. For example, if a point of interest p has x,y,z coordinates $(10, 10, 10)$, then it will generate 5 new points. An example distribution could be $[(5, 11, 10), (6, 8, 10), (15, 12, 10), (9, 9, 10), (7, 6, 10)]$. Assuming that P originally contained n points, the resulting $\hat{P}$ will contain $n + 5$ points after pushing the resulting new points back into P. To better explain how this part works, please refer to the following pseudocode in Appendix .2.

Comparisons between the enhanced point clouds after the motion-assisted enhancement and the original point clouds can be visualized in Figure 5.4.

In the experimental section, we will demonstrate the performance improvement that the MAE module can bring by improving the quality of the point cloud.

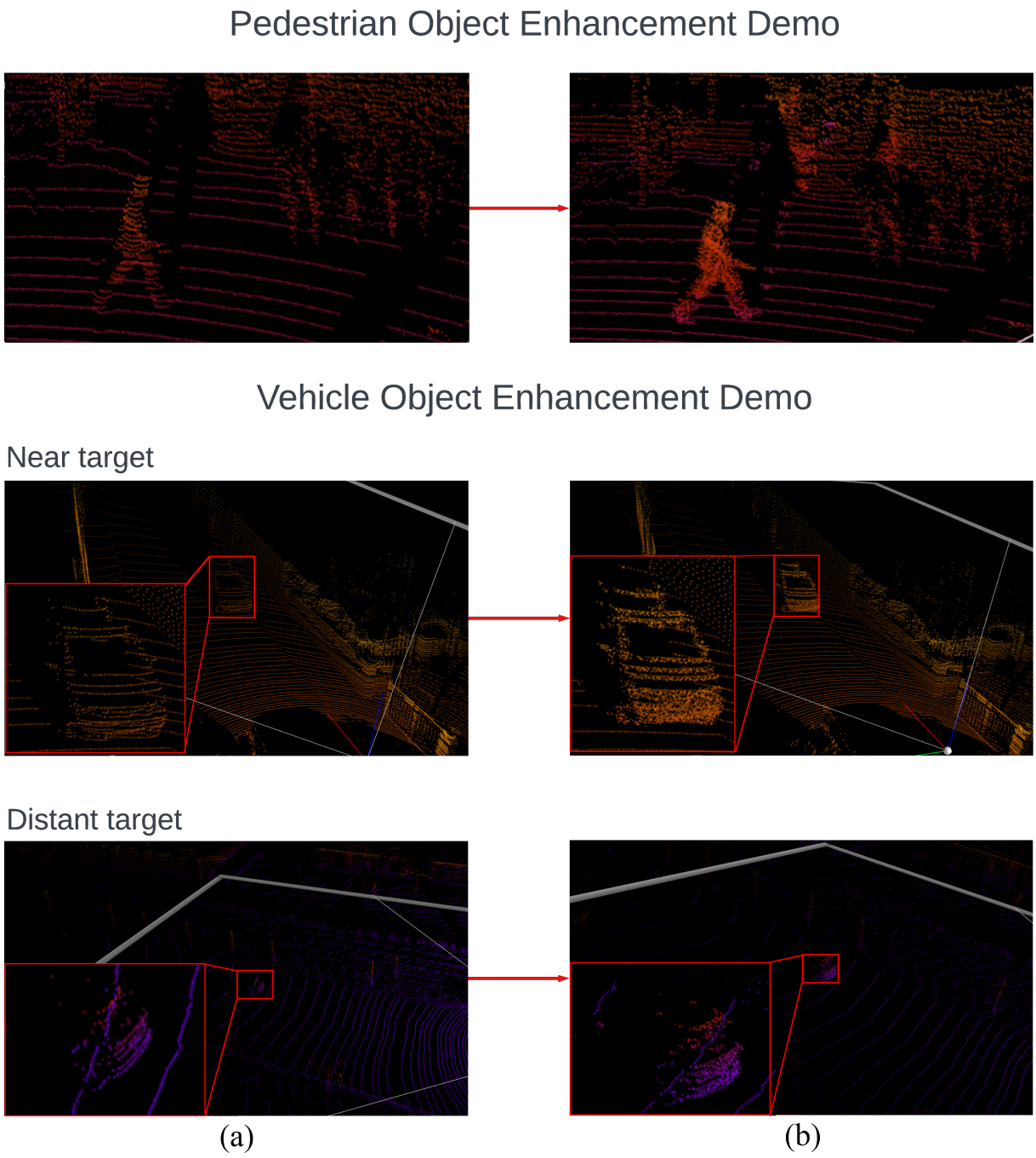

Figure 5.4: Motion-based enhancement demo: column (a) shows the visualization of the original point clouds from KITTI dataset. column (b) shows the enhanced point clouds using MAE.

Hyper-params	Value
Learning rate	0.0002 as initial learning rate for adam
Learning rate policy	linear
Key frame gap	5
Difference threshold for each color channel	5
Binarization threshold	128
# of generation filters in the last conv layer	64
# of discriminator filters in the last conv layer	96
# of threads	8
Batch_size	1
Load_size	1280
Crop_size	1024
No_flip	True
No_dropout for the encoder	True

Table 5.1: Hyper params for MAE

5.3 Experiments of MAE

5.3.1 Hyper-parametery Settings

The hyperparameters used by MAE are shown in Tables 4.4 and were tuned for optimal performance.

5.3.2 Results of MAE

Table5.2 shows the impact on performance when point clouds are augmented with MAE. Same as the experiment section from Chapter 4, we also completed experiments on SECOND [61], PointPillar [29], PartA2 [46], PV-RCNN [44], VOXEL_RCNN [8], Btcdet [60] and SFD [57]. The results of vehicle detection are shown in Table 5.2, and the results of pedestrian detection are shown in Table 5.3.

It can be seen that the MAE module does improve the performance of 3D detection, but the distribution of the improvement is different from that of UYI, which is very interesting. The performance gains from the UYI module varied considerably between models and were more often seen in the Medium and Hard categories. In contrast, the MAE boost is more uniform compared to the boost brought by the UYI module, with a smooth boost across different difficulty targets and different models. We consider that this may be due to the fact that the MAE module has a much smaller error in distance inference for target objects than the UYI module: the UYI module needs to use the PAP process to determine the target location of the point cloud in order to place a new dense point cloud object, whereas the MAE module operates on the existing point cloud and therefore achieves a much lower error. The same trend is also reflected in the pedestrian category, where MAE's enhanced pedestrian category also leads the vehicle category in terms of performance improvements.

The same trend is reflected in the pedestrian category, where the MAE-enhanced pedestrian category also leads the vehicle category in terms of performance gains. However, the improvement is not as pronounced as that brought about by the UYI module. We consider that this is also because the MAE module itself does not repair the target profile by adding new points from new sources to the point cloud, it simply densifies the original point cloud.

For the same reason, we place the MAE module before UYI in the next section of the experiments on the joint use of MAE and UYI.

5.3.3 Results of UYI+MAE

In this section, we show the analysis of detection performance when UYI+MAE are used together. Under this setting, the flow chart of the model operation is shown in Figure 5.5. The results of vehicle detection are shown in Table 5.4, and the results of pedes-

Method	Reference	Metrics	MAE Easy	MAE Medium	MAE Hard	Baseline Easy	Baseline Medium	Baseline Hard	Improvement Easy	Improvement Medium	Improvement Hard
SECOND [61]	SENSORS 2018	BEV_AP	**92.40**	**88.32**	**86.09**	91.92	87.92	85.39	*+0.48*	*+0.40*	*+0.70*
		3D_AP	**88.79**	**80.41**	75.59	88.45	78.75	75.72	*+0.34*	*+1.66*	*-0.13*
PointPillar [29]	CVPR 2019	BEV_AP	**92.88**	**88.97**	**87.52**	92.04	87.74	86.65	*+0.84*	*+1.23*	*+0.87*
		3D_AP	**88.62**	**79.46**	**76.57**	88.24	78.25	75.32	*+0.38*	*+1.21*	*+1.25*
PartA2 [46]	TPAMI 2020	BEV_AP	**92.31**	**87.67**	**85.71**	90.86	86.26	85.80	*+1.45*	*+1.41*	*-0.09*
		3D_AP	**90.07**	**81.77**	**79.44**	89.93	80.24	77.89	*+0.14*	*+1.53*	*+1.55*
PV-RCNN [44]	CVPR 2020	BEV_AP	**94.44**	**89.68**	**88.97**	92.86	88.93	88.74	*+1.58*	*+0.75*	*+0.23*
		3D_AP	**92.07**	**83.35**	**80.81**	91.99	82.86	80.40	*+0.08*	*+0.49*	*+0.41*
VOXEL-RCNN [8]	AAAI 2021	BEV_AP	**96.29**	90.83	**89.52**	95.35	90.83	88.64	*+0.94*	*0*	*+0.88*
		3D_AP	**93.94**	**84.20**	**80.42**	92.06	82.64	80.10	*+1.88*	*+1.56*	*+0.32*
BtcDet [60]	AAAI 2022	BEV_AP	**95.01**	**89.85**	**88.01**	93.46	89.53	87.44	*+1.55*	*+0.32*	*+0.57*
		3D_AP	**92.44**	**83.78**	**81.98**	92.17	82.39	80.96	*+0.27*	*+1.39*	*+1.02*
SFD [57]	CVPR 2022	BEV_AP	**96.91**	**93.08**	**92.62**	95.85	91.92	91.41	*+1.06*	*+1.16*	*+1.21*
		3D_AP	**95.40**	**88.77**	**86.87**	95.01	88.31	85.69	*+0.39*	*+0.46*	*+1.18*

Table 5.2: Vehicle performance improvement for MAE on state-of-the-art methods on the KITTI validation set. From top to bottom: SECOND, PointPillar, PartA2, PV-RCNN, VOXEL-RCNN, BtcDet and SFD, sorted by publication time. In the middle of the table is the baseline result based on official codes been open sourced, on the left side is the result after adding the MAE Module. It can be seen that the high-quality point clouds generated based on the MAE module also led to significant improvements in the models(at the right side).

Method	Reference	Metrics	MAE Easy	MAE Medium	MAE Hard	Baseline Easy	Baseline Medium	Baseline Hard	Improvement Easy	Improvement Medium	Improvement Hard
SECOND [61]	SENSORS 2018	BEV_AP	**91.97**	**88.37**	**86.89**	58.67	53.20	48.64	*-0.3*	*+0.46*	*+5.57*
		3D_AP	**61.59**	**51.19**	**45.97**	53.64	47.45	42.74	*+7.95*	*+3.74*	*+3.23*
PointPillar [29]	CVPR 2019	BEV_AP	**63.35**	**57.11**	**56.61**	59.50	54.37	50.13	*+3.85*	*+2.74*	*+6.48*
		3D_AP	**58.91**	**52.66**	**47.66**	55.11	49.57	44.78	*+3.80*	*+3.09*	*+2.88*
Point-RCNN [45]	CVPR 2019	BEV_AP	**69.75**	**64.84**	**55.73**	64.17	57.81	51.16	*+5.58*	*+7.03*	*+4.57*
		3D_AP	**66.03**	**59.21**	**54.62**	62.29	54.88	48.22	*+3.74*	*+4.33*	*+6.4*
PartA2 [46]	TPAMI 2020	BEV_AP	**62.48**	**57.63**	**51.24**	59.01	52.15	48.15	*+3.47*	*+5.48*	*+3.09*
		3D_AP	**61.71**	**53.24**	**45.97**	54.62	48.32	43.41	*+7.09*	*+4.92*	*+2.56*
PV-RCNN [44]	CVPR 2020	BEV_AP	**63.78**	**58.13**	**49.87**	58.91	51.43	47.67	*+4.87*	*+6.71*	*+2.20*
		3D_AP	**60.51**	**53.38**	**47.18**	55.78	48.42	43.87	*+4.73*	*+4.69*	*+3.31*

Table 5.3: Pedestrian performance improvement for MAE on state-of-the-art methods on the KITTI validation set. From top to bottom: SECOND, PointPillar, PointRCNN, PartA2, PV-RCNN, BtcDet, sorted by publication time. The definition of the the settings of the table is the same as Table 4.8. In summary, the model with the biggest boost is the Point-RCNN, with also similar improvements reflected in BEV_AP.

trian detection are shown in Table 5.5. For vehicles, more notable results were brought by SECOND, VOXEL-RCNN and SFD_Net, with also similar improvements reflected in BEV_AP. Finally, we achieved 0.17% and 0.11% in Hard and Medium categories of SFD_Net's 3D_AP, respectively. While on the pedestrian category, the largest improvement is on the PV-RCNN, which achieves a 0.61% improvement in the Hard category and 0.37% in Medium.

In the Table 5.5 5.4, we have directly replaced the Baseline column in the previous Table 4.7, 4.8, 5.2 and 5.3 with the scores after enhancement from UYI module to get a more direct comparison for improvements. We find that using both UYI and MAE gives a lower improvement than if the two were stacked separately. In fact, when the two are added together, their boost is there but very limited. The improvement in the line class is still slightly greater than the vehicle class. Taken together, the largest improvement in the vehicle class is in SFD_Net, which gets a 0.17% mAP improvement in the Medium category; the largest improvement in the pedestrian class is in PV-RCNN, which achieves a 0.6% improvement in the Hard category. As can be seen, although the overall improvement is relatively small, the effect is positive.

5.4 Summary

In summary, we found that the MAE module alone gave a smaller but more even improvement than the UYI module. The UYI module, with the addition of point clouds from new sources, gives a more significant performance improvement for the Medium and Hard categories. And when both modules were used, the model showed a large improvement compared to the baseline level, but the improvement was relatively insignificant when compared to UYI only. However, in terms of overall experimental results, we have successfully

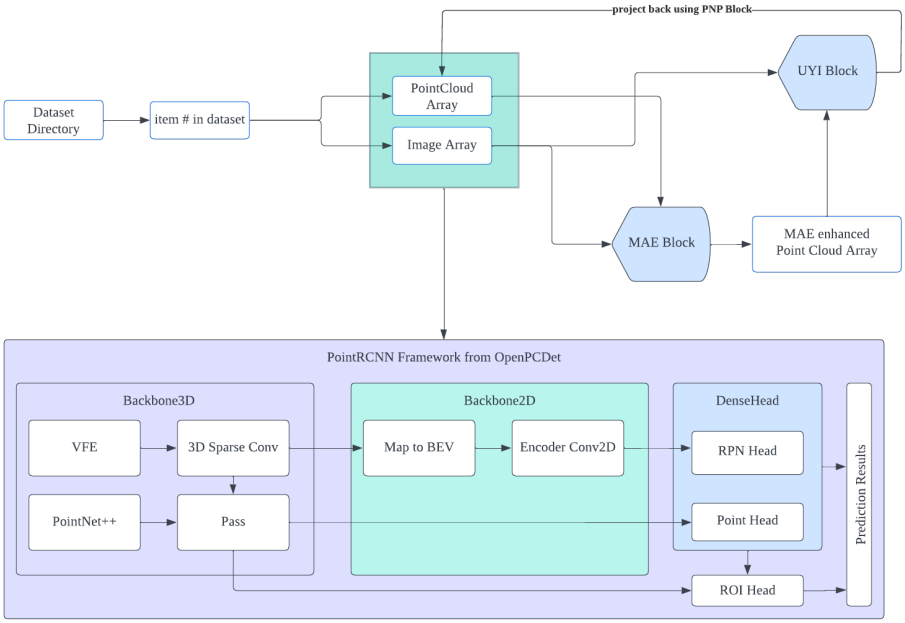

Figure 5.5: The flow chart of the model operation used by UYI and MAE at the same time. The 2D image and LiDAR point cloud from the data loader first pass through the MAE module and then through the UYI module. For the detection part we uses PointRCNN as an example here.

Method	Reference	Metrics	MAE+UYI Easy	MAE+UYI Medium	MAE+UYI Hard	Baseline Easy	Baseline Medium	Baseline Hard	Improvement Easy	Improvement Medium	Improvement Hard
SECOND [61]	SENSORS 2018	BEV_AP	91.97	**88.37**	**86.89**	92.27	87.91	86.63	-0.30	+0.46	+0.26
		3D_AP	88.60	**79.52**	**76.33**	88.88	79.15	76.07	-0.28	+0.37	+0.26
PointPillar [29]	CVPR 2019	BEV_AP	91.63	87.62	85.31	92.04	87.93	85.39	-0.41	-0.31	-0.08
		3D_AP	88.03	78.65	**76.04**	88.51	78.78	75.72	-0.48	-0.13	+0.32
PartA2 [46]	TPAMI 2020	BEV_AP	**93.37**	87.72	**87.69**	92.96	88.21	87.56	+0.41	-0.49	+0.13
		3D_AP	91.55	81.62	79.25	91.83	82.06	79.57	-0.28	-0.44	-0.32
PV-RCNN [44]	CVPR 2020	BEV_AP	94.56	89.88	**88.45**	94.63	90.28	88.43	-0.07	-0.40	+0.02
		3D_AP	**93.07**	82.92	81.97	92.97	83.11	82.38	+0.10	-0.19	-0.41
VOXEL-RCNN [8]	AAAI 2021	BEV_AP	**95.87**	**91.30**	**89.14**	95.76	91.06	88.84	+0.11	+0.24	+0.30
		3D_AP	92.03	**83.33**	**82.86**	92.31	83.09	82.57	-0.28	+0.24	+0.29
BtcDet [60]	AAAI 2022	BEV_AP	93.70	91.76	89.17	93.89	91.79	89.28	-0.19	-0.03	-0.11
		3D_AP	**93.22**	85.31	82.84	92.81	85.80	83.08	+0.41	-0.49	-0.24
SFD [57]	CVPR 2022	BEV_AP	**96.38**	**91.74**	91.07	96.10	91.68	91.08	+0.28	+0.06	-0.01
		3D_AP	**95.91**	**88.59**	85.67	95.64	88.42	85.70	+0.27	+0.17	-0.03

Table 5.4: Vehicle performance improvement for joint use of MAE and UYI on state-of-the-art methods on the KITTI validation set. From top to bottom: SECOND, PointPillar, PartA2, PV-RCNN, VOXEL-RCNN, BtcDet and SFD, sorted by publication time.

Method	Reference	Metrics	MAE+UYI Easy	MAE+UYI Medium	MAE+UYI Hard	Baseline Easy	Baseline Medium	Baseline Hard	Improvement Easy	Improvement Medium	Improvement Hard
SECOND [61]	SENSORS 2018	BEV_AP	60.85	56.86	**52.14**	60.86	57.03	51.54	-0.01	-0.17	+0.60
		3D_AP	**54.57**	**50.39**	**45.28**	53.94	49.66	45.00	+0.63	+0.73	+0.28
PointPillar [29]	CVPR 2019	BEV_AP	69.25	**62.99**	57.13	69.74	62.75	57.57	-0.49	+0.25	-0.44
		3D_AP	**63.76**	**56.96**	**51.79**	63.50	56.79	51.54	+0.26	+0.17	+0.26
Point-RCNN [45]	CVPR 2019	BEV_AP	73.70	**64.93**	55.5	73.82	64.62	55.72	-0.12	+0.31	-0.22
		3D_AP	69.01	**59.12**	**49.92**	69.41	58.79	50.04	-0.41	+0.33	-0.12
PartA2 [46]	TPAMI 2020	BEV_AP	**72.01**	65.45	**60.78**	71.97	65.65	60.53	+0.04	-0.20	+0.25
		3D_AP	**66.99**	**60.34**	53.91	66.85	59.61	54.28	+0.14	+0.73	-0.37
PV-RCNN [44]	CVPR 2020	BEV_AP	**71.08**	**64.05**	**59.29**	70.95	63.64	58.61	+0.13	+0.41	+0.68
		3D_AP	**66.61**	**59.54**	**54.60**	66.43	59.17	54.00	+0.18	+0.37	+0.61

Table 5.5: Pedestrian performance improvement for joint use of MAE and UYI on state-of-the-art methods on the KITTI validation set. From top to bottom: SECOND, PointPillar, PointRCNN, PartA2, PV-RCNN, BtcDet, sorted by publication time.

demonstrated the benefits of improving the quality of the point cloud for 3D detection. The MAE and UYI modules proposed in this thesis can be easily generalized for use on many existing 3D models.

Chapter 6

Discussions and Conclusions

6.1 Thesis Summary

In this thesis, we present a practical plug-and-play 3D point cloud quality booster for industry, including an MAE module based on motion information and a UYI module for repairing 3D target point clouds using GAN. For the design of UYI, we first discussed how the sparsity of distant LiDAR point cloud affects point cloud completion tasks to be dense and accurate. UYI is designed to improve the quality of point clouds, which uses a GAN-based cross-modal point cloud generator to generate LiDAR information by taking 2D images as input and trying to learn the perceptual connection to point clouds with color encoding, therefore improving the quality of point cloud by passing the generated high-density vehicle point cloud through the PAP process. And in MAE we discussed how the quality of point clouds can be enhanced using motion prior knowledge. To do this, we designed the 2D motion field dataset VIVA Motion, which allows us to learn the link between 2D images to motion maps as prior knowledge, and use this as a basis for locating

and enhancing the target point cloud for the KITTI dataset. Thanks to the plug-and-play feature of the modules, we can readily push the performance of existing state-of-the-art models to a new level and have a great potential for practical value in industry.

6.2 Why Not Use a Super-resolution Model to Improve Static Resolution

.

In the introduction section, we claimed that super-resolution models should not be used to enhance the resolution of remote targets. There are two reasons for doing so. The first is the lack of a super-resolution training dataset for remote vehicles, and none of the existing image resolution models tested are up to the task. This process introduces additional noise and unrealistic features that can drift the target semantics of the image. This effect is shown in Figure 6.1 below.

As can be seen in Figure 6.1, for more distant targets, all three typical super-resolution algorithms struggle to significantly improve feature resolution. In this way, it is difficult to assume that super-resolution-based algorithms can give a promising performance boost to the detector. In particular, the USM-based method produces a large number of cases of solid color overflow. This will clearly lead to a negative effect on the semantics of the original image. So we did not proceed with this approach and instead cast our eyes on multimodal point cloud quality enhancement.

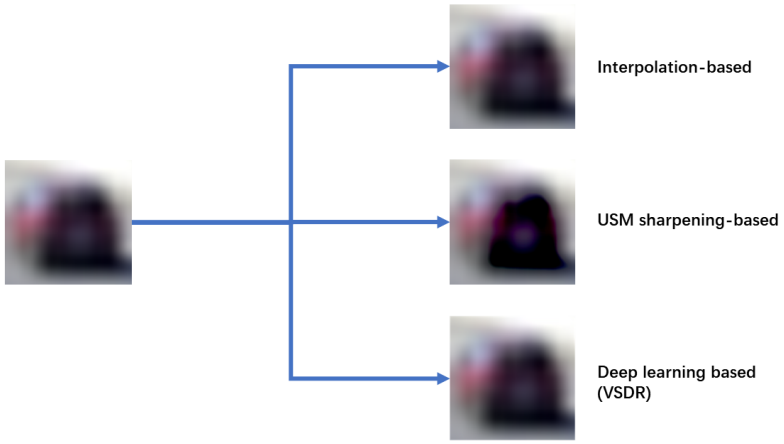

Figure 6.1: Comparison of super-resolution methods. Top right: algorithm based on interpolation; middle right: sharpening based on USM; bottom right: based on deep learning (VDSR model [25]). The VSDR model weights were trained on general super-resolution datasets.

6.3 Limitations and Insights

In this thesis, we attempt to improve 3D detection in a way that enhances the quality of the point cloud. However, after the research in this thesis, we find that there are still points that can be improved and investigated as follows.

1. Our motion map approach is primarily suited to improving performance on static datasets which frames are not continuous, and there could be other ways to capture the motion information of moving targets on datasets that are inherently continuous.

2. Since UYI's projection module PNP is still based on the number of original points in the target object point cloud, unacceptable errors may occur when the target object point cloud itself is extremely sparse.

For the first problem, one possible idea is that we merge motion information in the 3D detection process. For example by adding a motion information detection head and using the attention mechanism to find salient approximate motion regions to aid the original proposals. Alternatively, for a video dataset, we can use its continuous features to create a 3D model containing historical information using a number of consecutive frames. The possible trajectories of the moving target could be obtained by using feature encoders.

For the second problem, we can consider building a branch based on contrast learning that uses deep learning to learn the location of the target. This process could potentially lead to additional performance gains. We will share further research on this point on Arxiv, and hope the result could be available soon.

6.4 Contributions

Our point cloud quality booster can be easily plugged into a large number of 3D detection models for autonomous driving, which could serve as an almost free lunch as performance boosting for the industry. And our work has validated for the industry that improving the quality of point clouds can serve as an effective grip for improving 3D detection performance. Although our work successfully demonstrates the benefits of higher point cloud quality for 3D detection models, there is still room for model efficiency optimizations. As we previously mentioned in the section Discussion, designing models for motion information extraction and dynamic distance inference could be a promising direction in the future.

References

[1] Richard A Abrams and Shawn E Christ. Motion onset captures attention. *Psychological Science*, 14(5):427–432, 2003.

[2] Eduardo Arnold, Omar Y. Al-Jarrah, Mehrdad Dianati, Saber Fallah, David Oxtoby, and Alex Mouzakitis. A survey on 3d object detection methods for autonomous driving applications. *IEEE Transactions on Intelligent Transportation Systems*, 20(10):3782–3795, 2019.

[3] Amir Atapour-Abarghouei and Toby P. Breckon. To complete or to estimate, that is the question: A multi-task approach to depth completion and monocular depth estimation. In *2019 International Conference on 3D Vision (3DV)*, pages 183–193, 2019.

[4] Mario Bijelic, Tobias Gruber, Fahim Mannan, Florian Kraus, Werner Ritter, Klaus Dietmayer, and Felix Heide. Seeing through fog without seeing fog: Deep multimodal sensor fusion in unseen adverse weather. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[5] Derek Chan, Hylke Buisman, Christian Theobalt, and Sebastian Thrun. A Noise-Aware Filter for Real-Time Depth Upsampling. In *Workshop on Multi-camera and*

Multi-modal Sensor Fusion Algorithms and Applications - M2SFA2 2008, Marseille, France, October 2008. Andrea Cavallaro and Hamid Aghajan.

[6] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3d object detection network for autonomous driving. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1907–1915, 2017.

[7] Zhifeng Chen, Hantao Wang, Lijun Wu, Yanlin Zhou, and Dapeng Wu. Spatiotemporal guided self-supervised depth completion from lidar and monocular camera. In *2020 IEEE International Conference on Visual Communications and Image Processing (VCIP)*, pages 54–57, 2020.

[8] Jiajun Deng, Shaoshuai Shi, Peiwei Li, Wengang Zhou, Yanyong Zhang, and Houqiang Li. Voxel r-cnn: Towards high performance voxel-based 3d object detection. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(2):1201–1209, May 2021.

[9] Jian Deng and Krzysztof Czarnecki. Mlod: A multi-view 3d object detection based on robust feature fusion method. In *2019 IEEE intelligent transportation systems conference (ITSC)*, pages 279–284. IEEE, 2019.

[10] J. Engel. Polytomous logistic regression. *Statistica Neerlandica*, 42:233 – 252, 04 2008.

[11] Scott Ettinger, Shuyang Cheng, Benjamin Caine, Chenxi Liu, Hang Zhao, Sabeek Pradhan, Yuning Chai, Ben Sapp, Charles R. Qi, Yin Zhou, Zoey Yang, Aur'elien Chouard, Pei Sun, Jiquan Ngiam, Vijay Vasudevan, Alexander McCauley, Jonathon Shlens, and Dragomir Anguelov. Large scale interactive motion forecasting for autonomous driving: The waymo open motion dataset. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9710–9719, October 2021.

[12] Patrick Follmann, Rebecca König, Philipp Härtinger, Michael Klostermann, and Tobias Böttger. Learning to see the invisible: End-to-end trainable amodal instance segmentation. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1328–1336, 2019.

[13] Steven L Franconeri and Daniel J Simons. Moving and looming stimuli capture attention. *Perception & psychophysics*, 65(7):999–1010, 2003.

[14] Xiang Gao, Yingjie Tian, and Zhiquan Qi. Rpd-gan: Learning to draw realistic paintings with generative adversarial network. *IEEE Transactions on Image Processing*, PP:1–1, 08 2020.

[15] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.

[16] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[17] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[18] Jiaqi Gu, Zhiyu Xiang, Yuwen Ye, and Lingxuan Wang. Denselidar: A real-time pseudo dense depth guided depth completion network. *IEEE Robotics and Automation Letters*, 6(2):1808–1815, 2021.

[19] 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.

[20] Peiyun Hu, Jason Ziglar, David Held, and Deva Ramanan. What you see is what you get: Exploiting visibility for 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[21] Xun Huang, Arun Mallya, Ting-Chun Wang, and Ming-Yu Liu. Multimodal conditional image synthesis with product-of-experts gans. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XVI*, pages 91–109. Springer, 2022.

[22] Yung-Lin Huang, Tang-Wei Hsu, and Shao-Yi Chien. Edge-aware depth completion for point-cloud 3d scene visualization on an rgb-d camera. In *2014 IEEE Visual Communications and Image Processing Conference*, pages 422–425, 2014.

[23] 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.

[24] 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 (CVPR)*, pages 2583–2592, June 2021.

[25] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Accurate image super-resolution using very deep convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1646–1654, 2016.

[26] Kyeong Seon Kim, Dohyun Kim, and Joongheon Kim. Hardness on style transfer deep learning for rococo painting masterpieces. In *2019 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, pages 452–454, 2019.

[27] Soohyun Kim, Jongbeom Baek, Jihye Park, Gyeongnyeon Kim, and Seungryong Kim. Instaformer: Instance-aware image-to-image translation with transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18321–18331, 2022.

[28] 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. IEEE, 2018.

[29] 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.

[30] Mengxin Li, Dai Zheng, Rui Zhang, Jiadi Yin, and Xiangqian Tian. Overview of 3d reconstruction methods based on multi-view. In *2015 7th International Conference on Intelligent Human-Machine Systems and Cybernetics*, volume 2, pages 145–148, 2015.

[31] 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 (CVPR)*, June 2019.

[32] 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 (CVPR)*, June 2019.

[33] Ming Liang, Bin Yang, Shenlong Wang, and Raquel Urtasun. Deep continuous fusion for multi-sensor 3d object detection. In *Proceedings of the European conference on computer vision (ECCV)*, pages 641–656, 2018.

[34] Ming Liang, Bin Yang, Shenlong Wang, and Raquel Urtasun. Deep continuous fusion for multi-sensor 3d object detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[35] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *CoRR*, abs/1411.1784, 2014.

[36] Danish Nazir, Alain Pagani, Marcus Liwicki, Didier Stricker, and Muhammad Zeshan Afzal. Semattnet: Toward attention-based semantic aware guided depth completion. *IEEE Access*, 10:120781–120791, 2022.

[37] Tri Nguyen and Myungsik Yoo. Dense-depth-net: a spatial-temporal approach on depth completion task. In *2021 IEEE Region 10 Symposium (TENSYMP)*, pages 1–3, 2021.

[38] Jaesik Park, Hyeongwoo Kim, Yu-Wing Tai, Michael S. Brown, and In So Kweon. High-quality depth map upsampling and completion for rgb-d cameras. *IEEE Transactions on Image Processing*, 23(12):5559–5572, 2014.

[39] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Gaugan: semantic image synthesis with spatially adaptive normalization. In *ACM SIGGRAPH 2019 Real-Time Live!*, pages 1–1. 2019.

[40] Abhipray Paturkar, Gourab Sen Gupta, and Donald Bailey. Overview of image-based 3d vision systems for agricultural applications. In *2017 International Conference on Image and Vision Computing New Zealand (IVCNZ)*, pages 1–6, 2017.

[41] Charles R. Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J. Guibas. Frustum pointnets for 3d object detection from rgb-d data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[42] Simon K Rushton, Mark F Bradshaw, and Paul A Warren. The pop out of scene-relative object movement against retinal motion due to self-movement. *Cognition*, 105(1):237–245, 2007.

[43] Xuning Shao and Weidong Zhang. Spatchgan: A statistical feature based discriminator for unsupervised image-to-image translation. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6526–6535, 2021.

[44] 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 (CVPR)*, June 2020.

[45] 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.

[46] Shaoshuai Shi, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. From points to parts: 3d object detection from point cloud with part-aware and part-aggregation network. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(8):2647–2664, 2021.

[47] Yugo Shimizu, Ryosuke Furuta, Delong Ouyang, Yukinobu Taniguchi, Ryota Hinami, and Shonosuke Ishiwatari. Painting style-aware manga colorization based on generative adversarial networks. In *2021 IEEE International Conference on Image Processing (ICIP)*, pages 1739–1743, 2021.

[48] Andrea Simonelli, Samuel Rota Bulo, Lorenzo Porzi, Manuel Lopez-Antequera, and Peter Kontschieder. Disentangling monocular 3d object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[49] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[50] Lew B Stelmach, Wa James Tam, and Paul J Hearty. Static and dynamic spatial resolution in image coding: An investigation of eye movements. In *human vision, visual processing, and digital display II*, volume 1453, pages 147–152. SPIE, 1991.

[51] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[52] Pascal Vivet, Gilles Sicard, Laurent Millet, Stephane Chevobbe, Karim Ben Chehida, Luis Angel Cubero, Monte Alegre, Maxence Bouvier, Alexandre Valentian, Maria Lep-

ecq, Thomas Dombek, Olivier Bichler, Sebastien Thuriès, Didier Lattard, Cheramy Séverine, Perrine Batude, and Fabien Clermidy. Advanced 3d technologies and architectures for 3d smart image sensors. In *2019 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 674–679, 2019.

[53] Yan Wang, Wei-Lun Chao, Divyansh Garg, Bharath Hariharan, Mark Campbell, and Kilian Q. Weinberger. Pseudo-lidar from visual depth estimation: Bridging the gap in 3d object detection for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[54] Yan Wang, Wei-Lun Chao, Divyansh Garg, Bharath Hariharan, Mark Campbell, and Kilian Q Weinberger. Pseudo-lidar from visual depth estimation: Bridging the gap in 3d object detection for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8445–8453, 2019.

[55] Li-Hua Wen and Kang-Hyun Jo. Fast and accurate 3d object detection for lidar-camera-based autonomous vehicles using one shared voxel-based backbone. *IEEE Access*, 9:22080–22089, 2021.

[56] Hai Wu, Chenglu Wen, Wei Li, Ruigang Yang, and Cheng Wang. Transformation-equivariant 3d object detection for autonomous driving. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2023.

[57] 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.

[58] Bao Xiaomin, Ni Xiaoqing, Wang Yaming, and Zhu Hanyu. Overview of 3d textile dynamic simulation research. In *2011 IEEE International Conference on Computer Science and Automation Engineering*, volume 1, pages 558–562, 2011.

[59] Liang Xie, Chao Xiang, Zhengxu Yu, Guodong Xu, Zheng Yang, Deng Cai, and Xiaofei He. Pi-rcnn: An efficient multi-sensor 3d object detector with point-based attentive cont-conv fusion module. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 12460–12467, 2020.

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

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

[62] Yurong You, Yan Wang, Wei-Lun Chao, Divyansh Garg, Geoff Pleiss, Bharath Hariharan, Mark Campbell, and Kilian Q Weinberger. Pseudo-lidar++: Accurate depth for 3d object detection in autonomous driving. *arXiv preprint arXiv:1906.06310*, 2019.

[63] Yurong You, Yan Wang, Wei-Lun Chao, Divyansh Garg, Geoff Pleiss, Bharath Hariharan, Mark Campbell, and Kilian Q. Weinberger. Pseudo-lidar++: Accurate depth for 3d object detection in autonomous driving. In *International Conference on Learning Representations*, 2020.

[64] Shifeng Zhang, Longyin Wen, Xiao Bian, Zhen Lei, and Stan Z. Li. Occlusion-aware r-cnn: Detecting pedestrians in a crowd. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[65] Zehan Zhang, Yuxi Shen, Hao Li, Xian Zhao, Ming Yang, Wenming Tan, ShiLiang Pu, and Hui Mao. Maff-net: Filter false positive for 3d vehicle detection with multi-modal adaptive feature fusion. In *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*, pages 369–376. IEEE, 2022.

[66] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4490–4499, 2018.

[67] Long Zhuo, Guangcong Wang, Shikai Li, Wanye Wu, and Ziwei Liu. Fast-vid2vid: Spatial-temporal compression for video-to-video synthesis. In *European Conference on Computer Vision (ECCV)*, 2022.

APPENDICES

.1 Appendix: VIVA Motion Preview

Please find more samples from the VIVA Motion dataset in Figure 1 and 2 below.

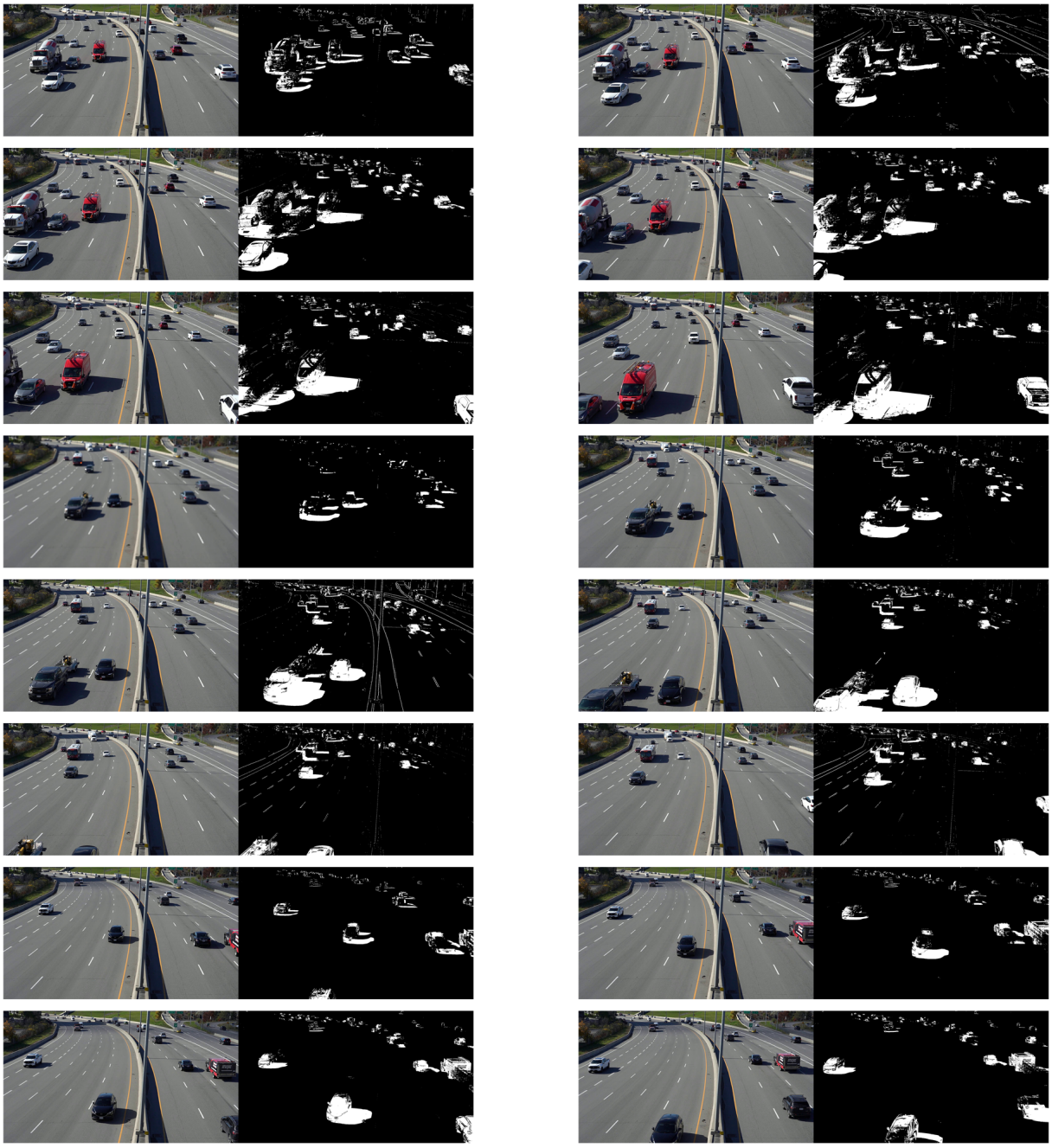

Figure 1: Additional sample preview from VIVA Motion-a

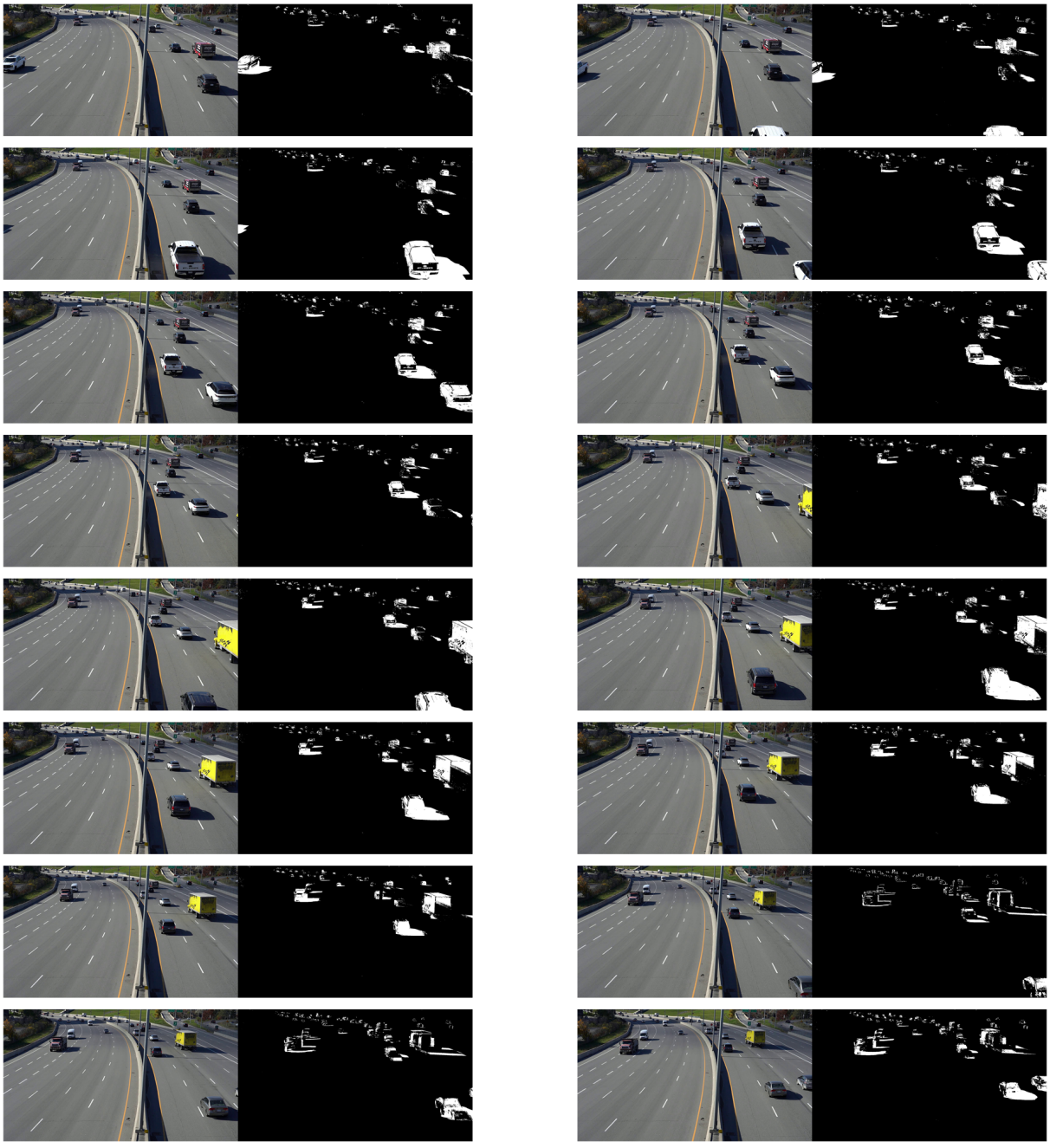

Figure 2: Additional sample preview from VIVA Motion-b

.2 Appendix: Pseudo Code for Motion Map Generation

The pseudo code for the motion map generation process is as below.

```
def lidar_on_motion(LiDAR_points, calibration_matrix, \\
    input_img_width, input img_height,locations,motion_map):

    ''' Project LiDAR points to image '''
    pts_2d = get_lidar_in_image_fov(LiDAR_points, calib, 0,
                                0, img_width, img_height, True)

    imgfov_pts_2d = pts_2d
    interested_points = []
    count = 0

    ''' For each LiDAR points '''
    for i in range(imgfov_pts_2d.shape[0]):
        # the x position of that point on the input image
        ptx = np.round(imgfov_pts_2d[i,0])

        # the y position of that point on the input image
        pty = np.round(imgfov_pts_2d[i,1])


        # safety concerns
        if np.isnan(ptx) or np.isnan(pty):
            count+=1
```

```
24              continue
25
26          # safety concerns
27          if ptx<0 or pty<0 or ptx>=img_width or pty>=img_height:
28              count+=1
29              continue
30
31          # If the point is what we want.
32          if motion_map[pty,ptx]==255:
33              # Record its index so we can fetch it in the future.
34              interested_points.append(count)
35
36          count+=1
37
38      return interested_points # array of interested points
39
40  def pointdenser(interested_points,pointarray):
41      points = {fetch the point from pointarray according to
42                      indexes recorded in interested_points}
43
44      for each point in points:
45          for i in range(0,5): # We do it 5 times.
46              x_shift = random.randint(1,6)
47              y_shift = random.randint(1,6)
48              z_shift = random.randint(1,6)
49
50              pointarray.append(newpoint with index(x_shift, y_shift,
```

```
                                                                z_shift))

    return pointarray # This will be the enhanced pointcloud.

if __name__ = __main__:
    pointarray = get LiDAR points from dataloader.

    interested_points = lidar_on_motion(params we don't care here)
    enhanced_pointarray = pointdenser(interested_points, pointarray)
```