Toward Solving Occlusion and Sparsity in Deep Learning-Based 3D Object Detection through Collaborative Perception

Minh Quan Dao

COLLEGE DOCTORAL PAYS DE LA LOIRE
SCIENCES DE L'INGENIERIE ET DES SYSTEMES

CENTRALE NANTES

MEMOIRE DE DOCTORAT DE

L'ÉCOLE CENTRALE DE NANTES

ÉCOLE DOCTORALE N° 602
Sciences de l'Ingénierie et des Systèmes
Spécialité : *Robotique*

Par
Minh-Quan DAO

Toward Solving Occlusion and Sparsity in Deep Learning-Based 3D Object Detection through Collaborative Perception

Projet de recherche doctoral présenté et soutenu à l'École Centrale de Nantes le 24 novembre 2023
Unité de recherche : UMR 6004 – Laboratoire des Sciences du Numérique de Nantes (LS2N)

Rapporteurs avant soutenance :

Thierry CHATEAU	Professeur des universités, Université Clermont Auvergne, Clermont-Ferrand
Sergiu NEDEVSCHI	Professor, Universitatea Tehnica din Cluj-Napoca, Roumanie

Composition du Jury :

Président :	Samia AINOUZ	Professeure des universités, INSA Rouen Normandie
Examinateurs :	Thierry CHATEAU	Professeur des universités, Université Clermont Auvergne, Clermont-Ferrand
	Sergiu NEDEVSCHI	Professor, Universitatea Tehnica din Cluj-Napoca, Roumanie
	Ezio MALIS	Directeur de recherche, INRIA Côte d'Azur
	Holger CAESAR	Assistant professor, Technische Universiteit Delft, Pays-Bas
	Fawzi NASHASHIBI	Directeur de recherche, INRIA Rocquencourt
Directeur de recherches doctorales :	Vincent FRÉMONT	Professeur des universités, École Centrale de Nantes
Co-encadrant de recherches doctorales :	Elwan HÉRY	Maître de conférences, École Centrale de Nantes

Table of Contents

List of Acronyms **7**

List of Symbols **9**

1 Introduction **11**

1.1 Representing Point Cloud . . . 11
1.1.1 Voxel-based Methods . . . 12
1.1.2 Point-based Methods . . . 14
1.2 Representing Learning Target . . . 15
1.2.1 Anchor-based Methods . . . 15
1.2.2 Center-based Methods . . . 16
1.3 Two-stage Detection Pipelines . . . 17
1.4 The Transformer Craze . . . 20
1.4.1 In The Backbone . . . 20
1.4.2 In The Second Stage . . . 21
1.5 Remaining Challenges . . . 22
1.5.1 Densification in the 3D space . . . 23
1.5.2 Densification in Latent Space . . . 25
1.6 Contributions . . . 27

2 Improving Object Detectros using Point Cloud Sequences **33**

2.1 Introduction . . . 34
2.2 Related Works . . . 37
2.3 Aligning Point Cloud Sequences . . . 38
2.3.1 Ego Motion Compensation . . . 38
2.3.2 Rectifying EMC Point Clouds . . . 39
2.4 *Aligner*: Joint Objects Detection and Motion Estimation . . . 39
2.4.1 Object Motion Estimation . . . 39
2.4.2 BEV Image Rectification . . . 41
2.4.3 Region Proposal Network . . . 42

2.4.4 Loss Function . . . 42
2.5 *Aligner++* . . . 44
2.5.1 Incorporating HD Map . . . 44
2.5.2 Distilling Shadow-Effect-Free Model . . . 45
2.6 Experiments . . . 46
2.6.1 Dataset and Evaluation Setting . . . 46
2.6.2 Implementation Details . . . 48
2.6.3 Results . . . 49
2.7 Conclusion . . . 54

3 Improving Object Detectors using V2X Collaborative Perception 57
3.1 Introduction . . . 58
3.2 Related Works . . . 61
3.3 Methodology . . . 63
3.3.1 Problem Definition . . . 65
3.3.2 MoDAR for Object Detection on Point Cloud Sequences . . . 65
3.3.3 V2X Collaboration using MoDAR Points . . . 66
3.4 Experiments . . . 67
3.4.1 Dataset and Metric . . . 67
3.4.2 Implementation . . . 68
3.5 Conclusions . . . 75

4 Improving Object Detectors using Refinement Stage 77
4.1 Introduction . . . 78
4.2 APRO3d-Net for 3d object detection . . . 79
4.2.1 3D Backbone and Region Proposal Network . . . 80
4.2.2 ROI Feature Encoder . . . 81
4.2.3 Detection Heads and Learning Targets . . . 85
4.2.4 Loss Function . . . 86
4.3 Experiments . . . 88
4.3.1 Implementation Details . . . 88
4.3.2 Results On KITTI Dataset . . . 90
4.3.3 Results On NuScenes Dataset . . . 91
4.3.4 Qualitative Performance . . . 91
4.3.5 Ablation studies . . . 91

4.4 Conclusion 95

5 Conclusions and perspectives **97**
5.1 Conclusion 97
5.2 Perspective 99
5.2.1 Robustness to Localization Error 99
5.2.2 Auto Label V2X Dataset using Object Discovery 100

List of publications **107**

A1 A Two-Stage Data Association Approach to 3D Multi-Object Tracking 109
A1.1 Introduction 110
A1.2 Related work 112
A1.3 Method 113
A1.3.1 Problem Formulation 113
A1.3.2 Two-stage Data Association 114
A1.3.3 Motion Model and State Vector 118
A1.3.4 Complexity Analysis 119
A1.4 Experiments 120
A1.4.1 Tuning the hyper parameters 121
A1.4.2 Tracking Results 122
A1.4.3 Ablation Study 124
A1.5 Conclusion 125

Bibliography **125**

List of Acronyms

AD	Autonomous Driving
AMOTA	Average Multi-Object Tracking Accuracy
BCE	Binary Cross Entropy
BEV	Bird-Eye View
CAV	Connected automated vehicles
CNN	Convolutional Neural Network
CTRV	Constant Turning Rate and Velocity
CV	Constant Velocity
EMC	Ego Motion Compensation
FP	False Positives
FPS	Furthest Point Sampling
FN	False Negatives
FoV	Field of View
RFE	ROI Feature Encoder
IoU	Intersection Over Union
IRSU	Intelligent Road-Side Units
LAP	Linear Assignment Problem
MLP	Multi-Layer Perceptron
MOT	Multi-Object Tracking
RPN	Region Proposal Network
ROI	Region of Interest
ViT	Vision Transformer
V2X	Vehicle-to-Everything

List of Symbols

$\mathbf{p}$	A 3D point
$\mathcal{P}$	A point cloud which a set of 3D points
$\mathcal{V}$	A voxel which is cube-like volume
$\mathbf{f}$	the feature vector of a voxel or a pillar (a voxel with infinite height)
$\mathbf{G}$	A voxel grid
C	number of features (or channels) of a multi-dimensional tensor
$\mathbf{B}$	A Bird-Eye View image
$\mathcal{G}$	Global frame
$\mathcal{S}$	A point cloud sequence
$\mathcal{O}$	A foreground object
$\mathbf{o}$	Scene flow of a LiDAR-point
$\text{cat}(\cdot)$	Channel-wise concatenation operation
$\text{Max}(\cdot)$	Channel-wise max pooling
$\mathcal{A}_i$	An agent i of a V2X network
$\mathbf{b}$	An 3D bounding box representing a detected object
$\mathcal{B}_i$	The set of objects detected by an agent i of a V2X network
$\mathcal{E}_i$	The pose of agent i of a V2X network in the global frame
$\mathcal{E}_e$	The pose of the ego vehicle in the global frame
$\mathbf{m}$	a MoDAR point
mean	The operator that computes the arithmetic mean of a set of vectors
$[\cdot]_{x,y,z}$	The operator that extracts 3D coordinate of a *MoDAR* point
$\mathbf{F}$	A 4-dimension feature volume produced by the backbone of a detector
$\mathfrak{R}_r$	The 3D volume occupied an ROI r
$\mathbf{r}$	Feature of an ROI

$\mathcal{R}_{0:t}$	The set of all tracklets established from the first timestep to the present
$\mathcal{T}^i$	Tracklet i representing the trajectory of object i
$\mathcal{B}_t$	The set of detection at timestep t
$\mathbf{x}_k^i$	The state vector of tracklet i at a timestep k
$\mathbf{L}$	The cost matrix of local assignment
$\mathbf{G}$	The cost matrix of global assignment
$\mathbb{R}^{a \times b}$	the set of matrix of real numbers of size $a \times b$
O	Algorithmic complexity

Chapter 1

Introduction

Autonomous driving (AD) is an exciting technology that has the potential to transform our transportation to be safer and more efficient. Among many components of an AD software stack, perception plays a fundamental role as it provides inputs to safety-critical modules such as navigation and motion planning [54]. The perception module comprises several sub-module that handles various aspects of building an understanding of the environment in which the ego vehicle operates. These sub-modules include (i) localizing other road users at each time step, (ii) tracking them across time, and (iii) predicting their motion in the future so that the subsequent motion planning module can chart a safe course. Due to the sequential nature of these steps, localization of other road users, which is formally referred to as object detection, is paramount to the development of AD.

As autonomous vehicles (AV) conduct themselves in a 3D world, the detection of objects needs to be done in 3D. To achieve this, AVs are equipped with advanced sensing systems essentially made of cameras, LiDARs, and RADARs. While which type of sensors is sufficient for 3D object detection is still an open question, it is undeniable that LiDAR-based methods achieve superior performance on public benchmark [8, 36, 106], compared to other modalities especially cameras. This is because LiDARs can provide accurate depth, which is lacking in RGB images due to perspective projection [45], at a sufficient density, which can not yet be matched by RADAR. For this reason, this thesis regards LiDAR as the primary component for developing the object detection module for AD.

This chapter will walk through the state-of-the-art of LiDAR-based object detection methods and identify the remaining challenges that need to be addressed. These challenges are the premises of the following chapters of this thesis.

1.1 Representing Point Cloud

The central question of early LiDAR-based 3D object detection is how to represent data collected by LiDARs which is referred to as point clouds. A point cloud $\mathcal{P}$, as can

be seen in Fig.1.1, is a set of points with 3D coordinates, expressed in the frame of the LiDAR, and additional raw features such as reflectance.

$$\mathcal{P} = \left\{ \mathbf{p}_i = [x, y, z, r] \right\}_{i=1}^{|\mathcal{P}|} \tag{1.1}$$

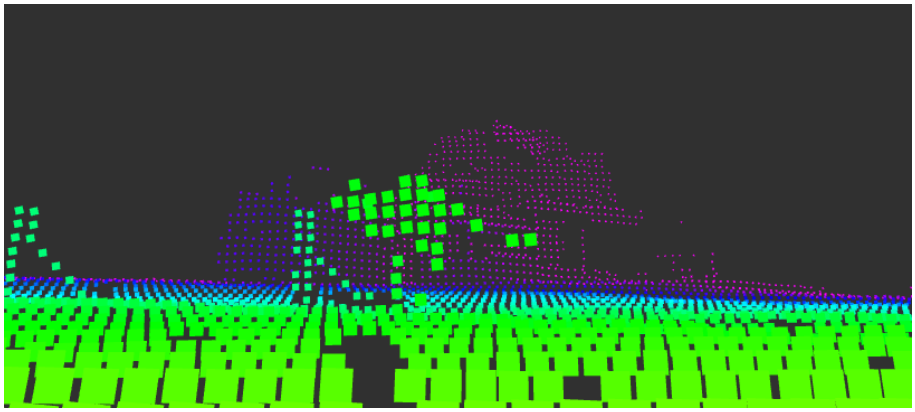


Figure 1.1 – A point cloud of a kangaroo.

1.1.1 Voxel-based Methods

Due to their unordered nature, point clouds cannot be processed straightforwardly by the Convolutional layer [64] which is the building block of the Deep Learning revolution. Since the convolution operation requires its input to be in the form of a regular grid (i.e., a tensor), the seminal work VoxelNet [150] converts point clouds to 3D voxel grids by voxelizing the 3D volume using cube-like voxel. After this voxelization step, each voxel $\mathcal{V}$ is assigned an initial feature vector $\mathbf{f}$ by applying the *Voxel Feature Encoding* (VFE) layer. This layer takes in the absolute 3D coordinate and reflectance of points inside a voxel, their relative coordinate with respect to the voxel's center $[v_x, v_y, v_z]$ as input, and performs a permutation invariant operation composed of a Multi-Layer Perceptron (MLP) and Max Pooling as following

$$\mathbf{f} = \text{MaxPool}\left(\left\{ \text{MLP}\left([x_i, y_i, z_i, r_i, x_i - v_x, y_i - v_y, z_i - z_i] \right) \right\}_{i=1}^{|\mathcal{V}|} \right) \tag{1.2}$$

After pre-processing the voxelized point cloud using a stack of VFE layers, the input point cloud becomes a 4D tensor $\mathbf{G}$ of the shape $C \times D \times H \times W$, with C denoting the number of features while D, H, W representing three dimensions of the voxel grid. This tensor

is subsequently processed by a stack of 3D convolution layers which finally reduces the D-dimension to one.

A noticeable feature of the resulting tensor $\mathbf{B} \in \mathbb{R}^{C' \times H' \times W'}$ is that its spatial dimension $H' \times W'$ is equivalent to an orthographic projection of the point cloud to the horizontal plane, and the feature dimension C' is similar to the 3 color channels of an RGB image. As a result, $\mathbf{B}$ can be regarded as an image of the point cloud taken by an orthographic camera from the top view. In the literature, $\mathbf{B}$ is often referred to as the (pseudo) **Bird-Eye View (BEV) image**.

The strength of the BEV image is that it offers a representation of the point cloud that preserves objects' shape and sizes and is compatible with image-based object detection methods which have a more mature literature. Indeed, after constructing the BEV image, VoxelNet uses the method of SSD[75] to detect objects in the BEV. The pipeline of VoxelNet, which is typical for voxel-based detection methods, is shown in Fig.1.2.

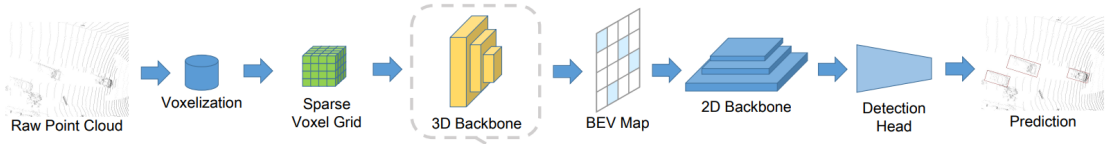


Figure 1.2 – Pipeline of voxel-based detection methods. This image is in custody of [84].

The success of VoxelNet is because it is the first to demonstrate end-to-end object detection in point clouds. Its major drawback is the large computational cost of its stack of 3D convolution layers, which hinders its deployment in real-world applications such as AD. This issue is resolved in SECOND [129] with the replacement of the dense convolution by the sparse [41] and submanifold [42] convolution. The resulting model can reach the inference speed of 40 frames per second. PointPillar [59] further boosts the inference speed by directly constructing the BEV image from raw point clouds thanks to the utilization of voxels having infinite size along the vertical direction which they refer to as pillars. Each pillar maps a cube-like volume directly to a pixel in the BEV image. Since the D-dimension of the voxel grid $\mathbf{G}$ starts at 1, the need for compressing this dimension using 3D convolution layers is dismissed, thus the lightning inference speed of up to 105 Hz.

Another family of voxel-based methods represented by PIXOR [131] implements a pure 2D convolution pipeline to ensure real-time speed by considering the C-dimension of the voxel grid $\mathbf{G}$ equal to 1. They achieve this by voxelizing point clouds using regular 3D voxels as VoxelNet but only consider the occupancy status (0 and 1) of each voxel as its

feature. Subsequently, the height dimension of the voxel grid $\mathbf{G}$ is treated as the channel dimension of the resulting BEV image $\mathbf{B}$.

1.1.2 Point-based Methods

Arguing that the voxelization causes loss of information, point-based methods utilize the PointNet [92], which is an equivalent of the convolution layer for point sets, to operate directly on raw point clouds. PointNet computes the feature $\mathbf{f}$ of each point $\mathbf{p}_i = [x, y, z, r]_i$ in a point cloud based on other points $\mathbf{p}_j$ residing inside its spherical neighborhood $\mathcal{N}$ using a permutation invariant operation as follows

$$\mathbf{f} = \text{MaxPool}\left(\left\{\text{MLP}\left(\mathbf{p}_j, \mathbf{p}_j - \mathbf{p}_i\right) | \|\mathbf{p}_j - \mathbf{p}_i\|_2 < r\right\}\right) \tag{1.3}$$

Comparing to Eq.(1.2), it can be seen that the computation of the point feature in Eq.(1.3) is based on the same principle used for computing the voxel feature that is to use to embed members' feature to a higher dimension using an MLP then aggregating their features to obtain the output.

The hierarchy of point features, which is to replicate the hierarchy of voxel features made by a stack of convolution layers, is made by a stack of Set Abstraction layers [92]. This layer samples its inputting point set using the Furthest Points Sampling (FPS) algorithm to obtain a subset of so-called keypoints whose features are computed based on regular neighbor points using Eq.(1.3).

PointRCNN [100] - the pioneer of point-based methods proposes a bottom-up approach that generates several object proposals for each keypoint, then refines the set of proposals by aggregating features of points residing inside the proposals' boundary to obtain the final output. 3DSSD [134] extends this seminal work by improving the keypoints sampling process to obtain a better coverage of foreground objects. Their extension is made by performing the FPS on both Euclidean space and points' latent space so that a better foreground-background ratio in the set of resulting keypoints is achieved.

While starting with the motivation of preserving information of point clouds, point-based methods have to downsample their inputs rather aggressively (e.g., keeping only 5K points of a 30K-point input) at the beginning of their pipeline to cope with the large number of points in outdoor point clouds and constraint memory. Therefore, it is hard to justify whether they can preserve more information than voxel-based methods. In fact, PointRCNN is outperformed by voxel-based methods such as Part-A^2 [101] and

VoxelRCNN [25] on the KITTI dataset. Another drawback of point-based methods is their low inference rate mostly caused by the FPS algorithm which involves memory-intensive operations to build spherical neighborhoods.

1.2 Representing Learning Target

1.2.1 Anchor-based Methods

Early 3D object detection methods [25, 101, 129] follow the anchor-based approach, which is popularized by image-based object detection literature (e.g., FasterRCNN [95], YOLO [94], SSD [75]), to encode the learning target. In the anchor-based framework, each pixel of the BEV image is assigned a set of boxes, referred to as anchor boxes or anchors, of predefined location, size, and orientation. These anchors play the role of the initial guess about the pose and size of ground truth objects on the scene. Given an input point cloud, a detection model strives to

— classify positive anchors (separating good guesses from bad guesses)

— regress to a vector that is used to adjust each positive anchor such that it fits better to its associated ground truth

For a set of ground truth objects representing their associated 3D bounding boxes, a positive anchor is defined as an anchor that has a sufficiently high Intersection-Over-Union (IoU) score with a ground truth bounding box. To ensure high coverage of ground truth, each pixel is assigned several anchors spanning different sets of sizes and orientations. This results in the predominant of negative anchors. Therefore, the loss function of the anchor classification task is usually the Focal Loss [73] due to its ability to handle severe class imbalance.

Given a positive anchor and its associated ground truth respectively parameterized by $[x^a, y^a, z^a, l^a, w^a, h^a, \theta^a]$ and $[x^t, y^t, z^t, l^t, w^t, h^t, \theta^t]$, the learning target of the regression task is

$$\begin{array}{rclrclrcl} x^* & = & \frac{x^t - x^a}{d}, & y^* & = & \frac{y^t - y^a}{d}, & z & = & \frac{z^g - z^a}{h^a} \\ l^* & = & \log \frac{l^t}{l^a}, & w^* & = & \log \frac{w^t}{w^a}, & h^* & = & \log \frac{h^t}{h^a} \\ \theta^* & = & \theta^g - \theta^a & & & & & & \end{array} \tag{1.4}$$

Here, $d = \sqrt{(x^a)^2 + (y^a)^2}$ is the length of the diagonal of the bottom face of the anchor.

The drawback of the anchor-based approach is two-fold. Firstly, it is memory intensive due to the number of anchors, which scales quickly with the size of the bird-eye view image,

that is assigned at each voxel to ensure high coverage. As a result, most anchor-based methods [25, 101, 102, 129, 150] have to settle for low-resolution BEV images (e.g., 0.8 meter-square voxel) to comply with GPU's memory constraint. A large pixel size makes it more challenging to detect small objects (e.g., pedestrians) as they occupy less than two pixels. Secondly, anchors require domain knowledge to be properly designed. This asks for access to statistics, which are not necessarily cheap to acquire, such as average object size, average height, or most likely heading direction.

1.2.2 Center-based Methods

Recently, the center-based approach [108, 137, 149] emerges as an alternative to the anchor-based approach. The central idea of this framework is that each object is encoded as 2D Gaussian, as shown in Fig.1.3, and the localization of objects (i.e., detecting their center) is equivalent to transforming the input image into a heat map that represents the probability of each pixel being the center of an object. This formulation opens up the possibility of directly applying the rich literature of image segmentation using fully convolutional neural networks to both 2D and 3D object detection, as point clouds can be converted to BEV images.

Figure 1.3 – Example heat maps of 2D images.

In addition to the paradigm-shifting idea, the center-based approach has two properties that make it more favorable to the anchor-based approach. First, it consumes less memory as it requires one matrix to represent objects of one class. Second, it dismisses the need for domain knowledge to design anchors.

An object of class c centering at the pixel coordinate $[c_x, c_y]$ is encoded as a Gaussian

as the following

$$Y_{x,y,c} = \exp\left(-\frac{(x-c_x)^2 + (y-c_y)^2}{2\sigma^2}\right) \tag{1.5}$$

Here, $[x, y]$ is the coordinate of a pixel and σ is calculated based on the size of the object based on the heuristic presented in [60]. The target probability computed in Eq.(1.5) is learned using the following variant of the Focal Loss

$$L = \frac{-1}{N}\sum_{x,y,c}\begin{cases}\left(1-\hat{Y}_{x,y,c}\right)^{\alpha}\log\left(\hat{Y}_{x,y,c}\right) & \text{if } Y_{x,y,c}=1\\ \left(1-Y_{x,y,c}\right)^{\beta}\left(\hat{Y}_{x,y,c}\right)^{\alpha}\log\left(1-\hat{Y}_{x,y,c}\right) & \text{otherwise}\end{cases} \tag{1.6}$$

Other attributes of the object's bounding box are learned in the same manner as the anchor-based method.

1.3 Two-stage Detection Pipelines

As methods for representing point clouds and learning targets get well established, the interest of the field is shifted toward improving detection accuracy. Under a large influence of image-based object detection literature which was dominated by two-stage methods (e.g., FasterRCNN [95], MaskRCNN [47], RetinaNet [73], FPN[72]), the following development of 3D object detection was about the second stage. The role of this stage is to refine object proposals so that true positive predictions are assigned high confident scores and fit better to their associated ground truth, while the score of false positive predictions is lowered, thus facilitating their removal in the post-processing step.

This trend is started by PointRCNN which pools features of keypoints inside each proposal to compute the proposal's feature, then decodes the result into new location, size, orientation, and score. The aggregating of pooled keypoints' features is done by a straightforward application Eq.(1.3). The aggregation method is further developed into the Region of Interest (ROI) Aware Pooling operation in Part-A^2 [101] to account for the different distribution, illustrated in Fig.1.4, of the same set of points in the local frame of different object proposals, which are referred to as ROI. The geometry of an ROI is accounted for by transforming 3D points from the LiDAR's frame to its canonical frame, defined in Fig.1.5. The ROI's feature is obtained by first voxelizing its volume, then computing an initial feature vector to each occupied voxel using PointNet (i.e., Eq.(1.3)), and finally employing a series of 3D convolution layers to compress the feature volume to

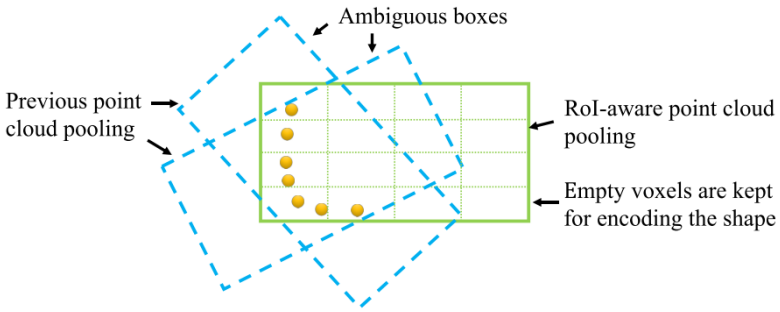


Figure 1.4 – ROI-aware pooling accounts for an ROI's geometry by transforming points from LiDAR's coordinate to ROI's canonical frame. This image is in the custody of [101].

a vector.

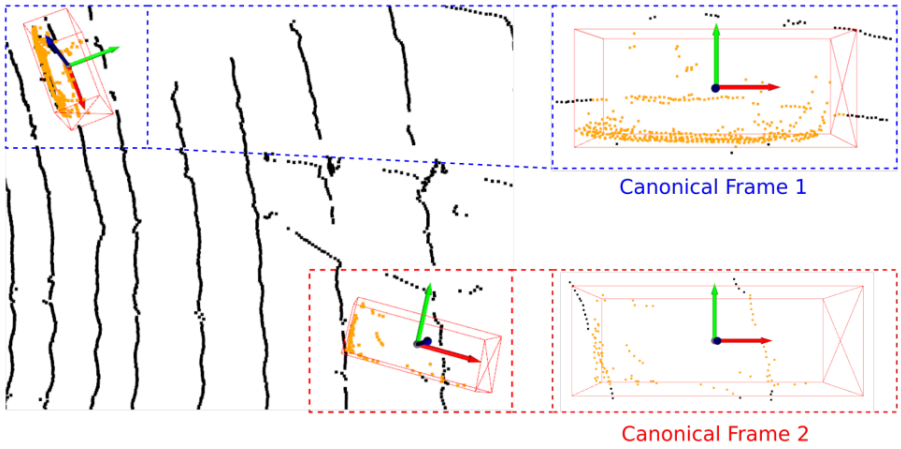


Figure 1.5 – A ROI's canonical frame: origin is at its center, the x-axis is parallel to its heading direction, the z-axis points upward, and the y-axis points to the lateral direction.

More modern works on the refinement stage are under the influence of the RoIAlign operation introduced in MaskRCNN [47]. The principle of RoIAlign, shown in Fig.1.6a, is that each ROI is divided into a grid of keypoints, denoted by red circles in Fig.1.6a. Each keypoint's feature is interpolated from its neighbor pixels, denoted by blue circles in Fig.1.6a. ROI feature is the aggregation (e.g., means or max pooling) of keypoints' features.

The idea of RoIAlgin is first implemented in 3D by PV-RCNN [102] in the form of the RoI-grid Pooling module. Here, each ROI is divided into a 3D grid whose points are assigned a feature vector by applying PointNet (Eq.(1.3)) to its neighbor 3D keypoints, as shown in Fig.1.6b. These keypoints are sampled from the input point cloud by the FPS algorithm. The aggregation of grid points' features to ROI's features is also done

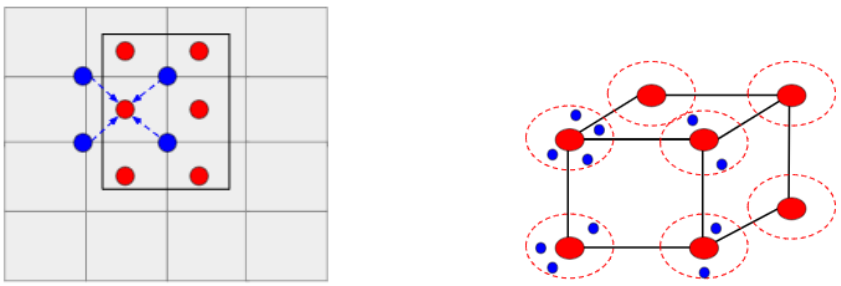


Figure 1.6 – Comparison between RoiAlign (a) and pooling operation of PV-RCNN [102] and VoxelRCNN [25] (b). Red circles denote points of the grid resulting from the discretization of the ROI. Blue circles represent pixels in sub-figure (a) and keypoints sampled from the input point cloud in sub-figure (b).

using PointNet. While achieving impressive performance, PV-RCNN has a major drawback which is the slow inference rate due to the computationally expensive FPS and Ball Query operator used respectively for sampling keypoints and building grid points' spherical neighborhood. VoxelRCNN [25] is built on the same idea of voxelizing object proposals as PV-RCNN. However, it offers a significantly better inference rate by leveraging the grid structure, illustrated in Fig.1.7, of voxelized input point cloud and its corresponding feature grids for fast neighbor query. The resulting method can reach a 21.4-Hz inference

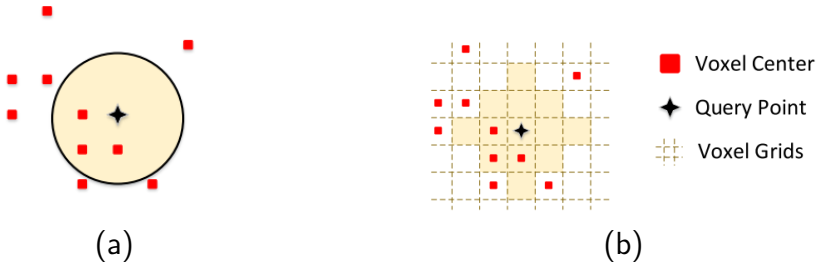


Figure 1.7 – Comparison between Ball Query used in PV-RCNN (a) and Voxel Query used in VoxelRCNN [25] (b). This image is in the custody of [25].

rate while outperforming PV-RCNN by a small margin on the KITTI dataset.

1.4 The Transformer Craze

1.4.1 In The Backbone

The shockwave made by the Vision Transformer (ViT) [28] did spread to the 3D object detection field, resulting in a large family of transformer-based architectures. PointFormer [87] - the elder of this family is based on PointRCNN but replacing PointNet with the Transformer [111] for point feature extraction. One source of strength of the Transformer is its capacity of modeling long-range dependencies as it computes the features of one token, which can be pixels (in image) or points (in point clouds), based on the features of every other token. As mentioned in Sec.1.1.2, PointRCNN computes keypoint features based on points residing in their spherical neighborhoods. Therefore, PointFormer limits the capacity of the Transformer to model the relationships among points near-by the others instead of the entire point cloud by following the formulation of PointRCNN. However, such limitation imposed on the Transformer is necessary to cope with its quadratic computation and memory complexity which can be infeasible even on high-end GPUs when coupled with the large size of outdoor point clouds.

Besides circumventing the quadratic complexity, there are other benefits to restricting the region of operation of the Transformer. As demonstrated in image-based methods [77, 152], introducing convolution-like priors such as translation equivariant, locality (i.e., the computation is done in a local region), and hierarchical features reduce the computation costs, and accelerates the training of transformer-based architectures while improving the performance. Voxel Transformer [84] is conceived under the influence of this observation. It replaces the sparse convolution layers in the backbone of SECOND with the self-attention module [111] operating on local regions. This effectively makes the self-attention module become convolution with dynamic weights as the weights are computed based on the points residing inside the field of view of the attention module.

Taking a different approach to dropping the Transformer to classic architectures, [31] use this module to develop a single-stride object detector to better detect small objects (e.g., pedestrians or cyclists). The main challenge to detect small objects using conventional convolution-based methods is that they occupy only one or two pixels in the final BEV images due to the large downsampling ratio of the convolution backbone. [31] overcomes this challenge by avoiding downsampling altogether, thus being single-stride. The interaction among voxels, illustrated in Fig.1.8, is provided using the combination of shifted windows and the cross-attention mechanism as in Swin Transformer [77].

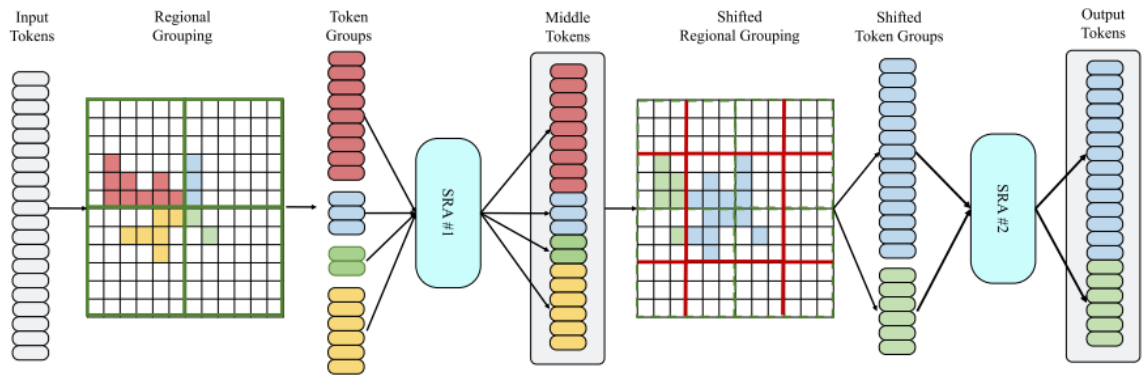


Figure 1.8 – Computation of voxel features in a single-stride detector using the Transformer and shifted windows mechanism. This image is in the custody of [31].

1.4.2 In The Second Stage

Following the success of the Transformer in the backbone, [99] develops a second stage for refining object proposals. Compared to the computation of BEV images using convolution layers, the ROI features extraction based on point features does not share the same set of inductive priors (e.g., translation equivariant or locality). This lack of priors makes most two-stage methods [84, 101, 102] resort to PointNet for ROI features extraction which is not necessarily the most effective. Since the Transformer in its unconstrained form is a powerful generic function approximator thanks to its ability to model long-range dependencies and information rewiring, it is an appealing solution. These abilities are leveraged by [99], using the pipeline shown in Fig.1.9, to further improve the performance of SECOND beyond the result of PV-RCNN and VoxelRCNN. In their framework, the initial value of an ROI's feature is a (trainable) parameter of the detector. This initial feature is updated by cross-attending to the features of points residing inside the ROI's volume.

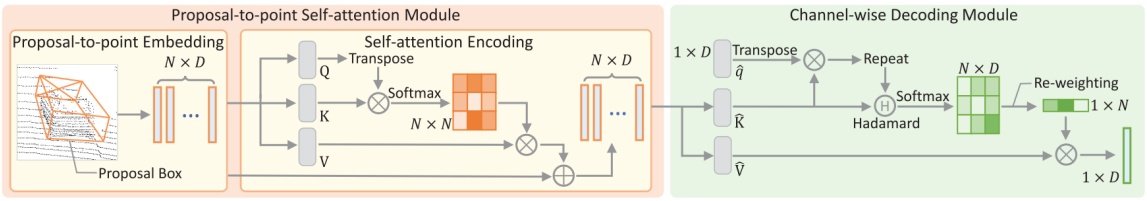


Figure 1.9 – Computation of ROI features from point features using the Transformer. This image is in the custody of [99].

1.5 Remaining Challenges

Since the first end-to-end model of VoxelNet, there have been a lot of advancements in the 3D object detection literature from the representation of point clouds, to the development of the second stage, to a new paradigm of representing learning targets, and to the current transformer craze. However, a challenge that remains troublesome for LiDAR-based methods is the low-fidelity measurement of foreground objects (i.e., objects have few LiDAR points), illustrated in Fig.1.10, which is caused by occlusion and sparsity. This challenge is inherent to the nature of this sensing modality because (i) the space behind each LiDAR point is unobservable, and (ii) the void between two laser beams increases with respect to the distance from the LiDAR. As LiDAR-based methods heavily rely on the presence of LiDAR points to produce detection [109], low-fidelity measurement severely affects the detection accuracy by reducing the number of LiDAR points (sometimes to zero).

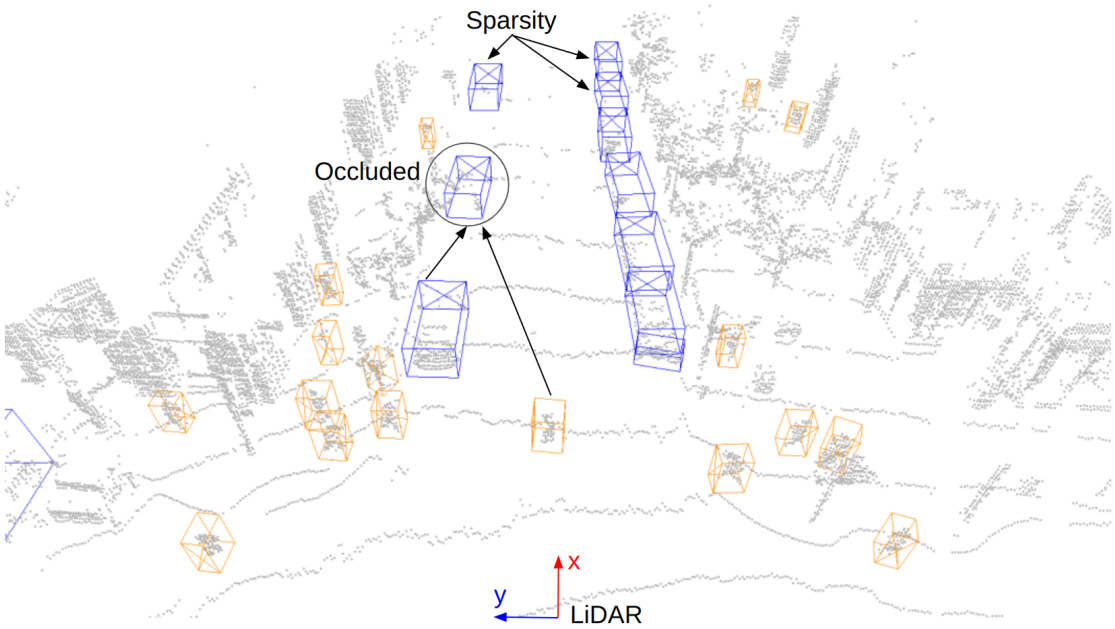


Figure 1.10 – Illustration of the low-fidelity measurement of foreground objects caused by occlusion and sparsity on point clouds. The car in the circle is occluded by a car and a pedestrian that is marked by back arrows, thus having few LiDAR points despite being close to the LiDAR. The three cars at the top of the image have few LiDAR points as they are far away from the LiDAR.

The need for resolving this challenge is highlighted in an experiment made by [125]

where point clouds of the KITTI dataset are completed using the oracle information of object shapes. This experiment reveals a compelling finding that PV-RCNN can achieve perfect detection (99.95% average precision) in the absence of occlusion.

As the issue of low fidelity is essentially the low number of points covering foreground objects, methods that aim to address this challenge are about point cloud densification either in the 3D space or in the latent space represented by the BEV image.

1.5.1 Densification in the 3D space

A typical approach toward point cloud densification is to leverage RGB cameras due to their availability on autonomous vehicles and their ability to maintain dense measurements even at long ranges. An exemplar of this class of methods is MVP [138] which (i) samples pixels on the 2D instance mask (Fig.1.11.a) and (ii) interpolates their depth from the projection of LiDAR points that land on the same instance mask (Fig.1.11.b) to obtain virtual 3D points (Fig.1.11d). The densified point clouds result in a 14.5% improvement in precision compared to the same detector operating on original (not densified) point clouds.

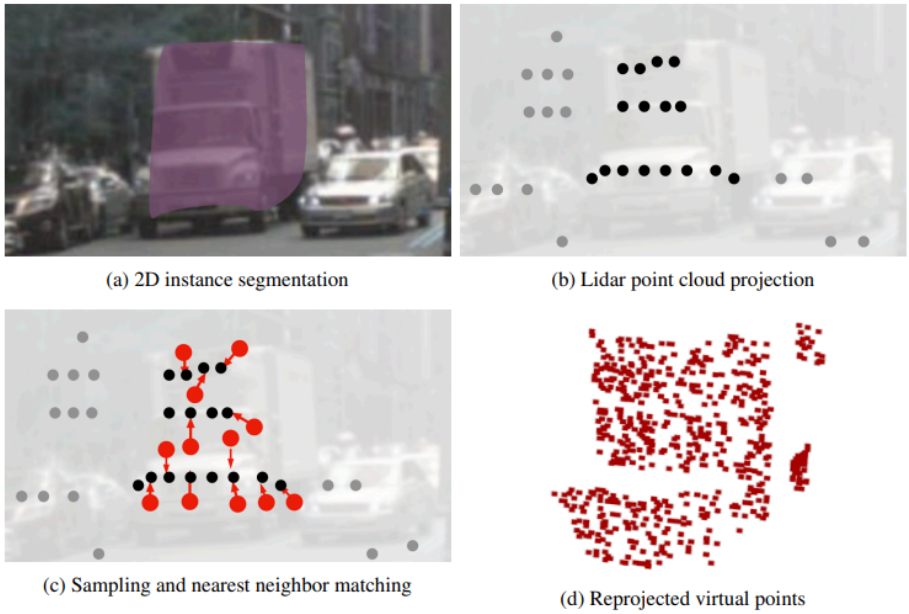


Figure 1.11 – Upsampling a point cloud by interpolating depth of pixels on 2D instance mask (a) using the projection of LiDAR points (b). This image is in the custody of [138].

This approach can be considered a late fusion between point clouds and images because

the processing of each modality, which is the instance segmentation on images in the case of [138], takes place independently. On the other hand, point clouds can be densified using early fusion as in [122] which upsamples point clouds by processing the concatenation of sparse depth maps, resulting from projecting point clouds to the image plane, and RGB images using a two-branch fully convolutional network. An example of upsampled point cloud is shown in Fig.1.12. While significantly increasing the point cloud density, point

Figure 1.12 – Comparison between point cloud (a) and upsampled point cloud (c) by fusion with the image from the forward-facing camera (b). This image is in the custody of [122].

clouds upsampled in this fashion suffer from two drawbacks namely local misalignment compared to LiDAR point cloud (Fig.1.13b) and depth distortion near objects' boundary which results in the long-tail effect (Fig.1.13c). In addition, the utilization of a sub-model for predicting depth from images adds significant computational overhead to the entire model. Moreover, point clouds densified by approaches that involve images are prone to poor precision at distances due to the inevitable LiDAR-camera calibration error. Therefore, the real-world deployment of this family of approaches remains challenging.

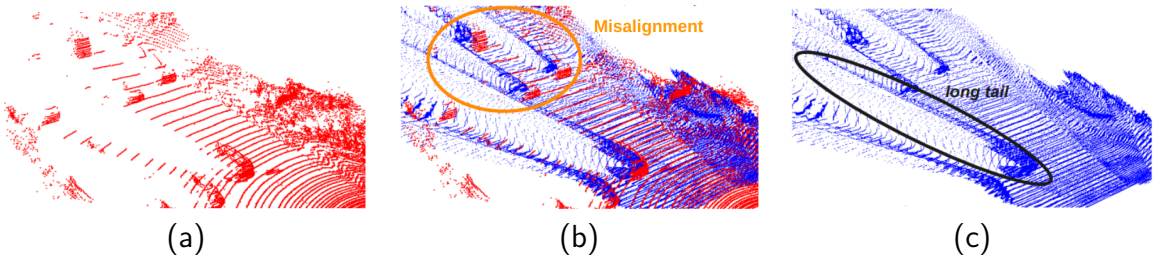


Figure 1.13 – The misalignment (b) and long-tail effect (c) in an upsampled point cloud, which is colored in blue, compared to the original (a), which is colored in red. This image is in the custody of [118].

To avoid the above defects of upsampled point clouds, [125] predicts the occupancy

probability of potentially occupied voxels (Fig.1.14a), thus enabling the detector to suppress spurious 'virtual' voxels, which are generated by the detector.

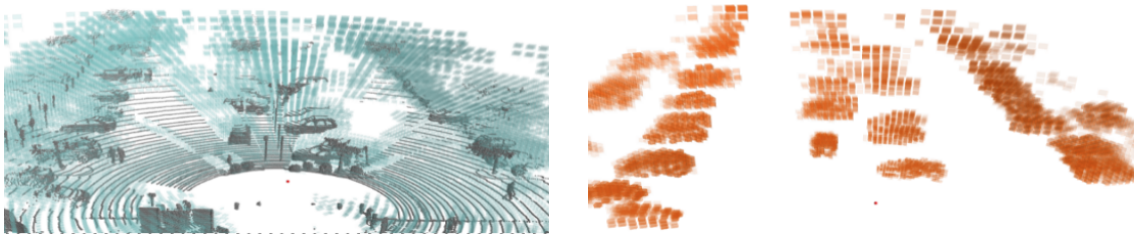


Figure 1.14 – Potential occupied voxels (a), which exhibit the long-tail effect similar to point clouds upsampled using RGB images, compared to voxels that have high predicted occupancy probability (b). This image is in the custody of [125].

1.5.2 Densification in Latent Space

Another approach toward point cloud densification is to emulate the BEV representation of occlusion-free point clouds, thus densification in the latent space. The common recipe for this approach is using knowledge distillation [49] which entails transferring the knowledge acquired by a large model (the teacher) to a smaller model (the student) by forcing the representation of the input data made by the student to be similar to the one made by the teacher. The similarity between the two representations is measured by a combination of Kullback–Leibler divergence, negative log-likelihood, and Euclidean distance (i.e., L_2 norm). In the early phase of distilling deep models, the representation is the output of the models (teacher and student) because the target application is the image classification where output is simply represented as a categorical distribution. As the knowledge distillation spreads to other fields, such as object detection, whose outputs are more complicated to represent, the representation chosen for distillation is the feature map produced in the middle of the architecture of the teacher and student.

The knowledge distillation framework is adapted to the context of point cloud densification in the latent space by training the teacher model on occlusion-free point clouds and transferring its knowledge to the student trained and tested on regular point clouds. The occlusion-free point clouds are synthesized from the regular point clouds by leveraging human-made annotations (e.g., 3D bounding boxes) that are available during training.

Assoc3DDet [30] is among the first in this class. It minimizes the impact of occlusion

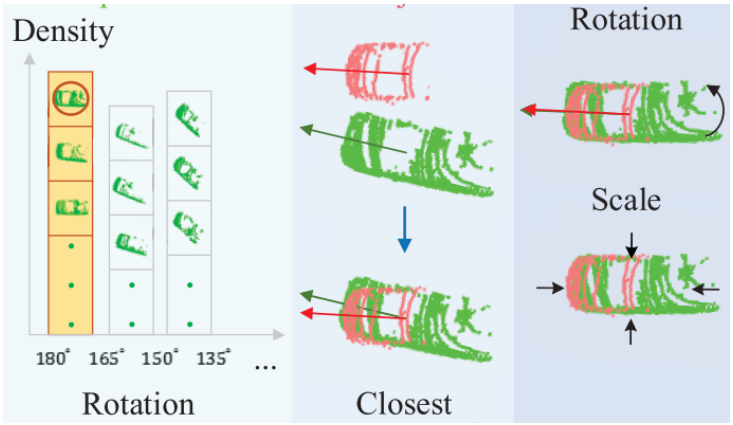


Figure 1.15 – The procedure for completing an object by searching for a denser object that has a similar orientation. This image is in the custody of [30].

on each foreground object by searching for the object that has the highest density in a database made of objects having similar orientations. LiDAR points of the object taken from the database are adjusted to match the orientation and scale of the object of interest, then merged with those of the foreground object. This procedure is illustrated in Fig.1.15. The transfer of knowledge from the teacher to the student, which has the same architecture, is done by applying the L_2 loss to the BEV images made by the backbone of the two models.

Sparse-to-Dense [115] takes a temporal approach, illustrated in Fig.1.16, toward generating dense object points by

(a) isolating point cluster of the object of interest collected over an entire driving sequence and sorting the extracted point clusters according to their size

(b) voxelizing the one with the highest density

(c) filling voxels in the voxel grid with points from all extracted point clouds

(d) mirroring the denser side of the object over its axial plane

Based on the reasoning that the student model that employs the submanifold sparse convolution layer [41] in its backbone is unable to generate features at empty voxels, thus cannot emulate the BEV images made by the teacher, [115] uses an additional module made of 2D convolution layers, which enables occupancy leaking, to transform the student's sparse BEV images to denser BEV images before distillation. A comparison between original BEV images and densified BEV images made by [115], shown in Fig.1.17, indicates dense feature populations in regions in regions associated with objects that are partially observed.

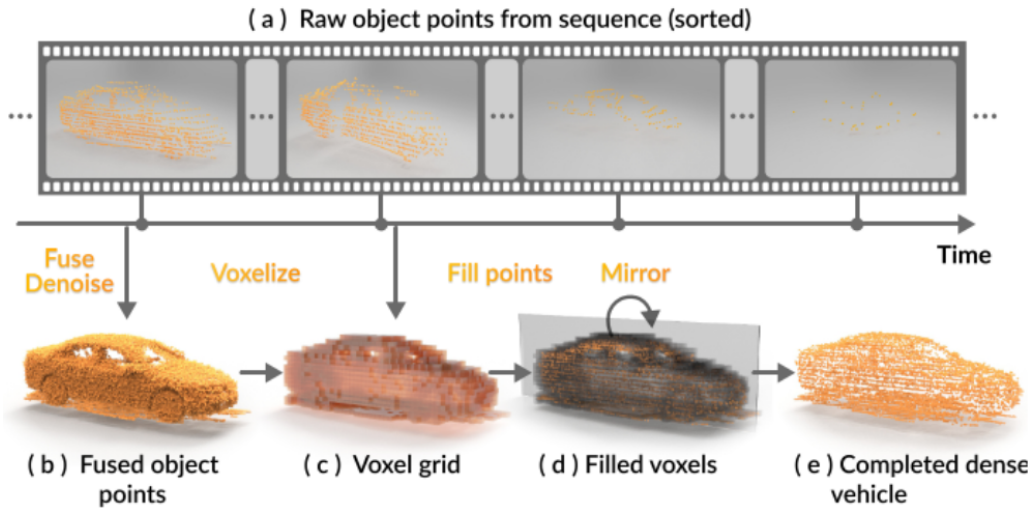


Figure 1.16 – The procedure for completing an object using its temporal point clouds. This image is in the custody of [115].

1.6 Contributions

The common numerator of the solutions to the low-fidelity measurement above is about generating more points using deep models. While achieving impressive performance, they haven't addressed the roots of the problem which are the occlusion and sparsity at long range. As a result, their performance is still limited by the availability of the original LiDAR points. For example, a failed case of MVP [138] is when there are few or even zero LiDAR points whose projections reside inside an instance mask, thus the lack of information for interpolating the depth of sampled pixels. Another example can be found in Fig.1.17b where a ground truth that has no LiDAR points is missed while another ground truth having a few LiDAR points is falsely located.

Aware of the aforementioned gap in the literature, this thesis addresses the low-fidelity measurements challenge based on the idea of physically densifying point clouds by leveraging

- point cloud sequences
- the availability of point clouds collected by different vehicles

The motivation for the utilization of point cloud sequences is that a straightforward concatenation of K point cloud in a sequence results in a point cloud that is on average K times denser than each individual. This increased number of points coupled with the fact that they are collected from different perspectives due to the motion of the ego vehicle and

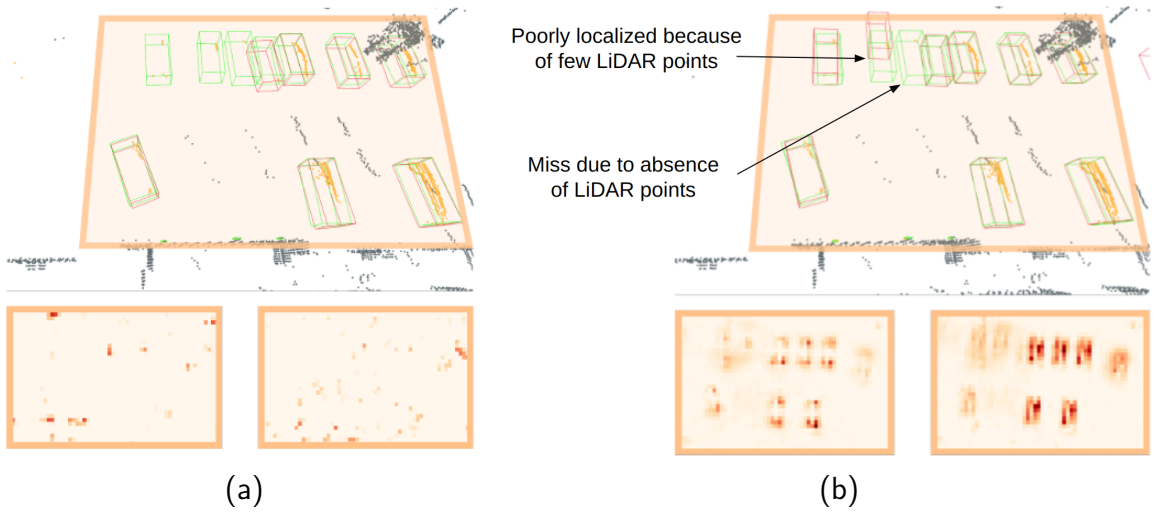


Figure 1.17 – Comparison between BEV images made by SECOND (a) and by SECOND enhanced by Sparse-to-Dense (b). The upper row shows the detected objects colored in red and ground truth colored in green. This image is in the custody of [115].

objects' independent motion often increases the coverage of the ego vehicle's measurement. As a result, detection models enjoy significant performance gains when operating on point cloud sequences instead of individual point clouds. These gains are particularly visible in low-resolution point clouds such as those collected by 32-beam LiDAR [8]. Since the concatenation of point clouds is mostly done by transforming temporal point clouds to a common frame using ego vehicle localization, it does not account for the motion of other objects. For this reason, the concatenated point cloud contains an artifact named *shadow effect* [81] that manifests as objects' points scattering along their trajectories as shown in Fig.1.18. This artifact, which is a misalignment in 3D between an object's location and its measurement - LiDAR points, results in a misalignment in the BEV representation that can eventually lead to false detection of medium and fast-moving objects [133]. A method for alleviating the impact of this artifact on detection models is developed in **Chapter 2**.

An increased number of points in the concatenated point cloud does not necessarily guarantee better coverage as the perspective of ego vehicle with respect to some objects can remain unchanged over the entire time span of a point cloud concatenation. For example, a vehicle waiting in an intersection, shown in Fig.1.19, will have little measurement of objects beyond the range of 10 meters because its field of view is severely occluded due to the presence of other road users. As a result, the problem of low-fidelity measurements cannot be reliably resolved solely by using point cloud sequences. However, the idea of

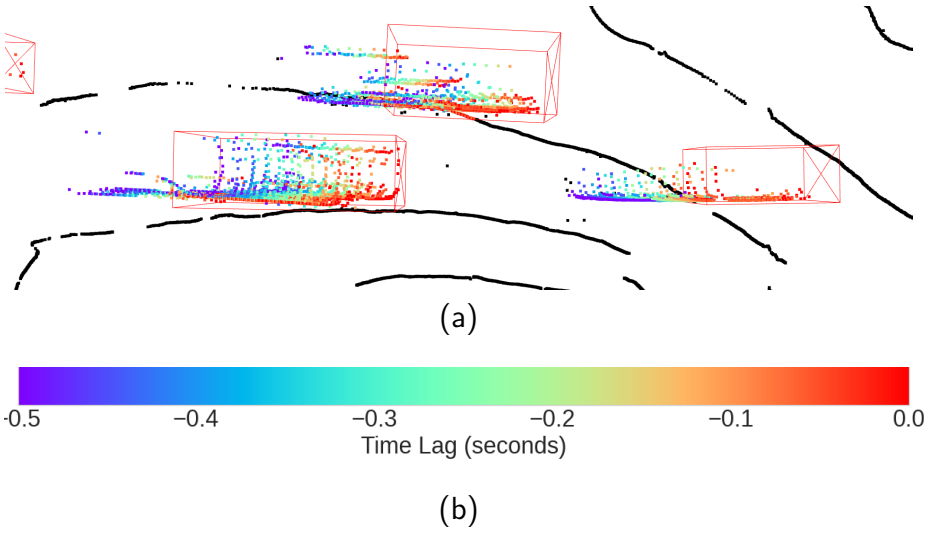


Figure 1.18 – Illustration of the shadow effect in the concatenation of a point cloud sequence spanning 0.5 seconds (a). This sequence contains three cars moving from left to right during the 0.5-second observation. Foreground points are color-coded according to their time lag with respect to the present. The color bar is shown in (b).

leveraging point clouds obtained from multiple perspectives does inspire another solution which is to create such a variety of perspectives using multiple vehicles presenting at different locations in a scene. This is the central idea of the emerging collaborative perception [11, 66, 116, 141, 142] via vehicle-to-everything (V2X) communication [82]. Similar to point cloud sequences, the concatenation of point clouds obtained by multiple vehicles also results in a significant performance boost [67, 128]. Unfortunately, this method is impractical in the V2X context due to the enormous bandwidth required to transmit raw point clouds. A less bandwidth-intensive method is to let each connected vehicle process its own measurement, and then fuse their detection for example by non-max suppression. This *late* fusion method immediately enables the ego vehicle to detect objects in unobservable regions conditioned on they are observable by other vehicles. However, a number of prior works [66, 116, 128] show that fusing the output of connected vehicles yields inferior performance boosts compared to fusing their BEV representations. While being attractive in terms of performance and bandwidth consumption, the fusion of connected vehicles' BEV representations requires (i) significant changes to be made to the architecture of single-vehicle detectors and (ii) a certain degree of synchronization among connected vehicles. These two drawbacks hinder the real-world deployment of the collaborative perception. **Chapter 3** addresses these issues by proposing a new collaborative

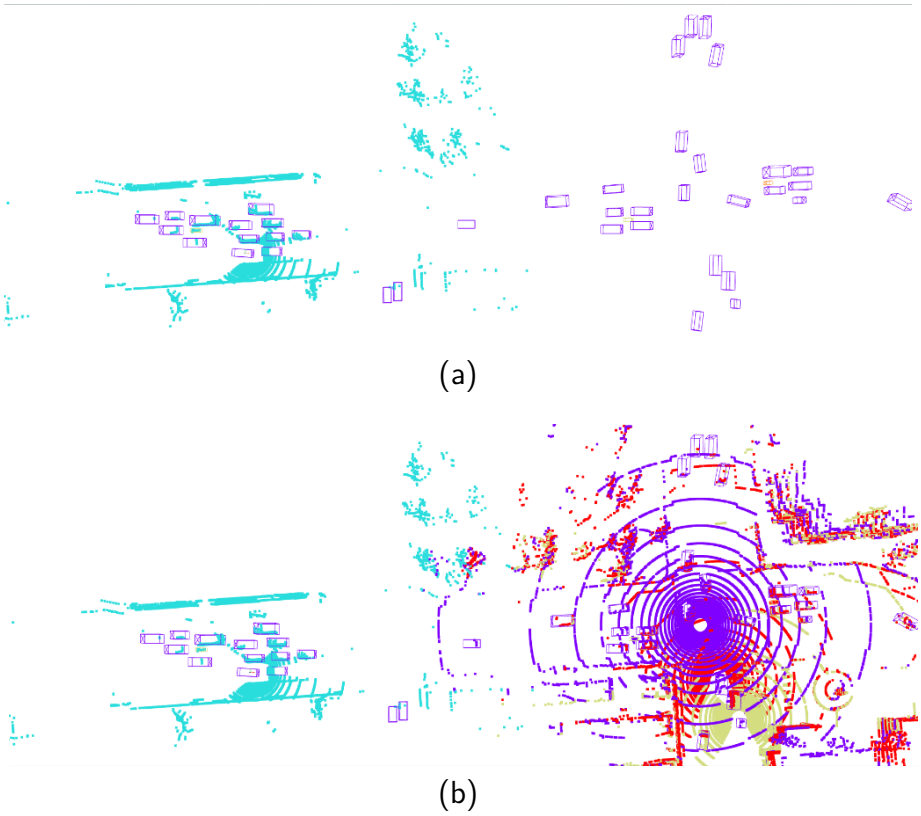


Figure 1.19 – An intersection in the V2X-Sim dataset [67] measured by a single vehicle (a) and by multiple vehicles (b). Ground truth objects are denoted by purple cuboids. The point cloud of each vehicle is assigned a unique color.

method, called *lately*, that enables an ear*ly* fusion of the ego vehicle's raw point cloud with other connected vehicles' prediction which is the signature of *late* fusion.

The combination of Chapter 2 and Chapter 3 constitutes a complete solution to the low-fidelity measurement challenge. In the framework of this solution, a connected autonomous vehicle detects objects based on the concatenation of point clouds and collaboration with other connected vehicles. The core of this solution is the single-vehicle detection model, which each connected vehicle uses to process its point cloud, as the *lately* collaboration method exchanges the output of connected vehicles. The important role that the single-vehicle detection model plays in our solution raises the necessity of improving its precision, particularly when popular models such as SECOND and PointPillar are designed to have a high recall rate at the risk of having a high number of false positives. To this end, **Chapter 4** develops a refinement stage that re-scores and re-adjusts the prediction made by these models. The general pipeline of this stage is to compute a fea-

ture for each ROI based on features of points (or voxels) contained by itself. The lack of an inductive bias for computing ROI features given points features requires a learnable operator that is as generic as possible. Such ability can be found in the Transformer [111]. However, the scalar weight that the vanilla Transformer assigns to each point falls short in modeling the fact that a point contributes differently to different attributes of an ROI. For example, a point near the center of a face has a small influence on the prediction of the ROI's size but is critical for predicting its center. Therefore, we based our refinement stage on a variant of the Transformer called vector attention [146] that computes different attention maps for different channels of point features, thus allowing each point to play different roles in the prediction of different ROI attributes.

Chapter 2

Improving Object Detectors using Point Cloud Sequences

A direct solution to the challenge of low-fidelity measurements is to increase the density of point clouds by concatenating a point cloud with its predecessor in a common coordinate frame. This method owes its effectiveness to a variety of perspectives that the ego vehicle takes over the time span when a point cloud sequence is collected. These different perspectives offer opportunities for occluded objects to become visible and far-away objects to get closer to the ego vehicle, thus increasing the number of LiDAR points on these challenging objects. This concatenation is done by transforming point clouds obtained at each time step to the common coordinate frame, which is referred to as the *target frame* hereon, using its localization. Since the transformation to the target frame only accounts for the motion of the ego vehicle, dynamic objects instantiate at different locations in the concatenated point cloud, resulting in the so-called shadow effect. This effect is essentially a misalignment between object points and object bounding boxes in 3D, as can be seen in Fig.2.1, which leads to a misalignment between object features in the BEV image and the projection of their bounding boxes to the BEV. As a result, the performance gain brought by replacing individual point clouds with concatenated point clouds is limited to stationary and slow-moving objects [133]. Scene flow allows aligning point clouds in 3D space, thus naturally resolving the misalignment in feature spaces. By observing that scene flow computation shares several components with 3D object detection pipelines, we develop a plug-in module that enables single-frame detectors to compute scene flow to rectify their Bird-Eye View representation. Experiments on the NuScenes dataset show that our module leads to a significant increase (up to 16%) in the Average Precision of large vehicles, which interestingly demonstrates the most severe shadow effect. The method and results presented in this chapter are published in "Aligning Bird-Eye View Representation of Point Cloud Sequences using Scene Flow", IEEE Intelligent Vehicles Symposium, 2023. The code of experiments used in this chapter is published at

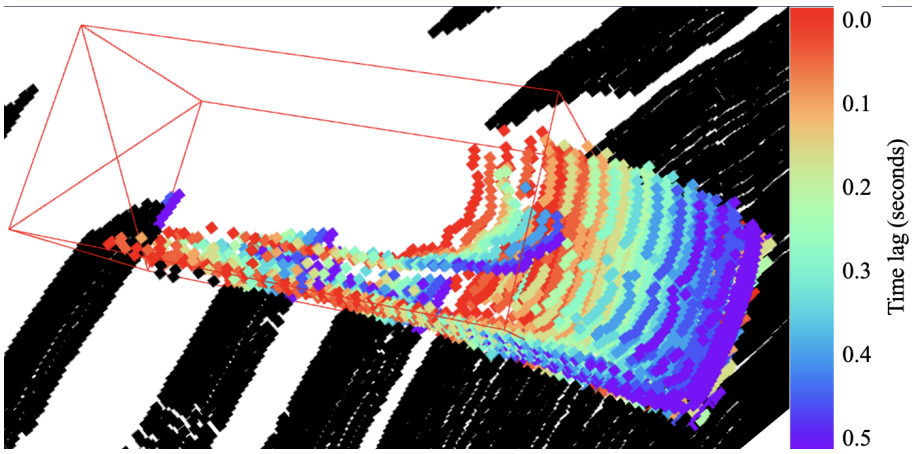


Figure 2.1 – A dynamic car's appearance on the concatenation of a 10-point-cloud sequence spanning 0.5 seconds. Points are color-coded according to their time lag with respect to the present.

`https://github.com/quan-dao/pc-corrector`.

2.1 Introduction

The concatenation of point cloud sequences is a straightforward yet highly effective method to enable any single-frame object detectors using multi-frame point clouds. This concatenation is done by transforming point clouds obtained at any time step during the observation period into a common reference frame - the *target frame* using the localization of the ego vehicle, thus its name Ego Motion Compensation (EMC). First introduced in [8], EMC has become the standard for object detection on low-resolution point clouds [4, 113, 134, 137, 151]. The most significant advantage of EMC is that it enables single-frame methods to enjoy a performance boost thanks to denser point clouds without changing their architecture. It is worth noticing that using EMC on any single-frame method effectively converts it to a multi-frame one.

On the other hand, the major drawback of EMC is the shadow effect [81] that manifests in dynamic objects' points scattering along their trajectories (Fig.2.2a). This misalignment in 3D space results in a misalignment in the feature space, shown in Fig.2.2b, which limits the performance gain brought by adding past point clouds using EMC to stationary and slow-moving objects only [133]. As a result, we seek to improve the performance of EMC-boosted single-frame methods by resolving the feature misalignment.

Prior object detection methods that explicitly address the feature misalignment issue

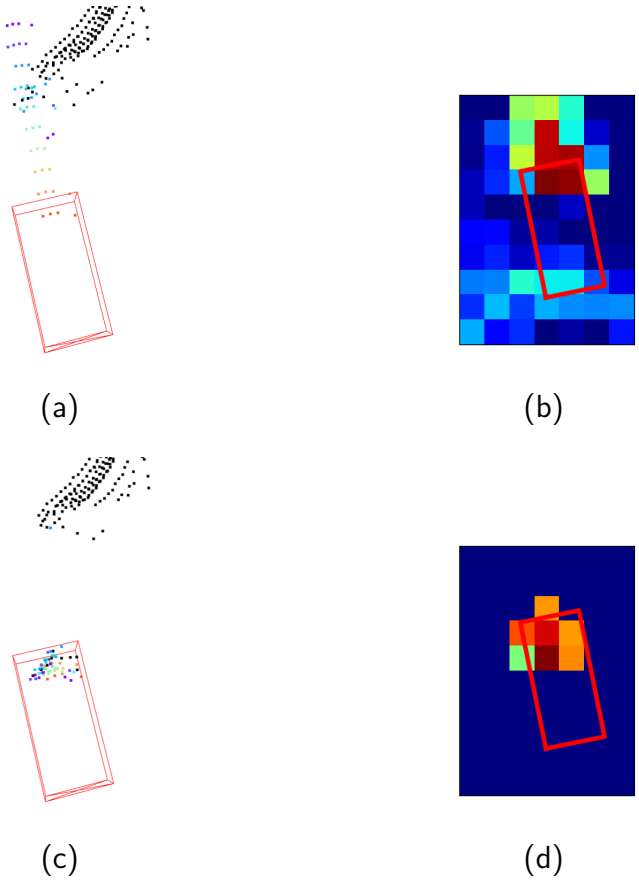


Figure 2.2 – Comparison between BEV representation of a dynamic object (b) affected by the shadow effect (a) and the representation (d) of rectified points (c). The ground truth object is denoted by the red cuboid in 3D and the red rectangle in the BEV plane. Foreground points are color-coded such that the hotter their color, the more recent they are.

can be divided into two categories that align either (i) BEV representation or (ii) object proposals' features. The former evolves from concatenating BEV feature maps [81] to sequentially mapping Range-view representation from one time step to another using a warp function made of the rigid transformation [58]. Its current state uses temporal layers such as Long Short-Term Memory (LSTM) [51] to fuse multi-frame features. 3D-MAN [133] is an exemplar of the latter category. It generates object proposals independently for each point cloud and stores them in a memory bank. Features of proposals at the target time step get refined by querying the memory bank.

A shortcoming of the methods mentioned above is the lack of explicit supervision of the alignment operation because the notion of "*well-aligned*" is challenging to establish

in the feature space. On the other hand, how well two point clouds align in 3D space can be straightforwardly measured using scene flow metrics. For this reason, we devise our feature alignment strategy based on rectifying EMC-concatenated point clouds using scene flow. In details,

1. Points in an EMC-concatenated point cloud are rectified according to their scene flow (Fig.2.2c).
2. The rectified point cloud is used to scatter points' features to the BEV plane to make a rectified BEV representation (Fig.2.2d).
3. The rectified BEV representation is fused with the BEV representation of the EMC-concatenated point cloud to obtain the fused BEV representation where feature misalignment and sparsity in the BEV representation are minimized (Fig.2.3).

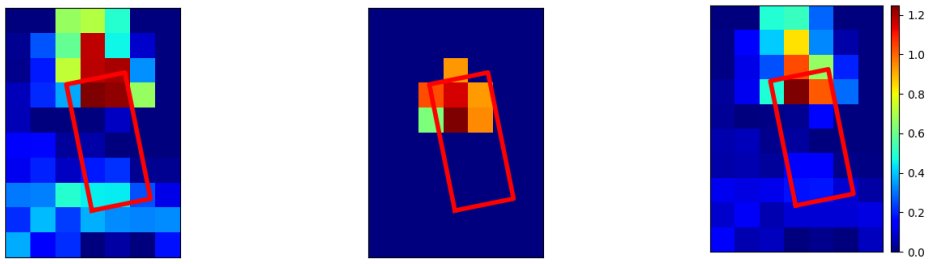


Figure 2.3 – Comparison between BEV representation of a dynamic object affected by the shadow effect (a), the representation after rectified points (b), and their fusion (c).

This chapter makes the following contributions:

- Developing a plug-in module that enables single-frame object detection methods to rectify their BEV representations of EMC-concatenated point clouds using scene flow.
- Conducting experiments on the NuScenes dataset, where it is standard to use EMC on single-frame methods due to its low-resolution LiDAR (32 beams). Adding our feature alignment method to PointPillars results in an improvement of up to 16% in Average Precision (AP). Despite being a multi-task method, our estimation of scene flow on the NuScenes dataset reaches 0.506 average End-Point Error (EPE), which is on par with strong scene flow baselines.

2.2 Related Works

The pioneering work [81] takes the mid-fusion approach to resolve the feature misalignment by processing the concatenation of BEV feature maps with a Convolutional Neural Network (CNN). On the other hand, the following works [9, 27, 71] take the early-fusion approach by concatenating voxelized point clouds along the height dimension, then feeding the result to a CNN. Their only difference compared to EMC is that the concatenation occurs after the voxelization instead of before. Moreover, the absence of modules dedicated to feature alignment in their architectures raises the question of how effectively the shadow effect is handled. MVFuseNet [58], a more recent work of this category, performs the alignment sequentially by mapping the Range-View representation at each time step to its successor using a warp function based on ego-motion, then refining the result with a CNN. Huang et al [51] take a similar approach by fusing features of two consecutive point clouds using the so-called "3D sparse conv LSTM".

3D-MAN [133] solves the feature misalignment issue using object proposals. It first generates object proposals and their associated features independently for each point cloud and stores them in a memory bank. Then, object proposals in the target point cloud (e.g., the most recent one) refine their features by querying the memory bank via the cross-attention mechanism [111]. While showing strong performance, [133] relies on a single-frame detector for per-frame proposals generation, thus requiring sufficiently dense point clouds.

Taking a different approach, we strive to align point clouds of a sequence in 3D space using computing scene flow of dynamic points, which naturally results in well-aligned feature maps. The motivation of our alignment strategy has two folds:

- Scene flow pipelines require points' features which can be computed by first converting input point clouds to BEV representation, then interpolating using points' projection on the BEV plane. The BEV representation is also needed for object detections, thus the possibility of combining them.
- Using scene flow enables explicitly supervising the feature alignment with a physically meaningful signal.

Our method for computation of scene flow takes inspiration from [52]. We share their method for obtaining point features by interpolating from the BEV representation of point clouds. However, we exploit the availability of ego vehicle localization to concatenate input point clouds before generating the BEV representation. As a result, we obtain the BEV

representation for the entire sequence in one shot.

Since motion only makes sense in the context of objects, many motion estimation and segmentation methods [13, 14, 40, 52] require instance segmentation by computationally expensive clustering. Therefore, we predict per-point flow directly from their features to achieve high inference speed. Since this flow is unconstrained, our prediction can contain physically infeasible motions (e.g., objects having different parts undergoing different motions). This risk is reduced by adding a simplified version of the "Object motion modeling" of [52], which we refer to as the Object Head, to our architecture. The role of this module is to predict a single rigid transformation for each instance. During training, we use the Object Head to guide the learning of per-point scene flow by enforcing consistency between their prediction (more details in Section.2.4.4). The correspondence between points and objects is available during training as ground truth, and the Object Head is deactivated during testing. As a result, instance segmentation is no longer necessary. This critical difference compared to [52] enables our short runtime shown in Tab.2.1.

2.3 Aligning Point Cloud Sequences

The first step toward aligning point clouds is removing the effect of the ego vehicle motion on LiDAR measurements using Ego Motion Compensation (EMC).

2.3.1 Ego Motion Compensation

Let $\mathcal{P}^t = \{\mathbf{p}_i^t = [x, y, z] \mid i = 1, \ldots, N\}$ denote a point cloud of size N collected at time t and having points expressed the ego vehicle frame $\mathcal{E}(t)$. A sequence of point clouds $\mathcal{S} = \left\{\mathcal{P}^{t-K}, \mathcal{P}^{t-K+1}, \ldots, \mathcal{P}^t\right\}$ spanning from the previous K steps to the current time t is concatenated according to EMC by transforming every point in each point cloud to a common global frame $\mathcal{G}$ using the ego vehicle pose ${}^{\mathcal{G}}\mathbf{T}_{\mathcal{E}(t-k)}$.

$$ {}^{\mathcal{G}}\mathbf{p}_i^{t-k} = {}^{\mathcal{G}}\mathbf{T}_{\mathcal{E}(t-k)} \cdot \mathbf{p}_i^{t-k} \tag{2.1} $$

Here, $k \in \{0, \ldots, K\}$. Notice that the global frame $\mathcal{G}$ can be the map frame or the ego vehicle frame at any time step (e.g., $\mathcal{E}^t$ as for NuScenes common practice).

2.3.2 Rectifying EMC Point Clouds

Since EMC does not account for objects' motions, points belonging to dynamic objects are scattered along their trajectories. Let $\mathcal{O}$ denote an object and ${}^{\mathcal{G}}\mathbf{T}_{\mathcal{O}(t)}$ represent $\mathcal{O}$'s pose in the global frame at time step t. At time step $t-k$, a laser beam emitted from the LiDAR hitting $\mathcal{O}$ produces a 3D point $\mathbf{p}^{t-k}$. The image of $\mathbf{p}^{t-k}$ computed by EMC (2.1) can be rectified by a two-step transformation as following

$$ {}^{\mathcal{G}}\hat{\mathbf{p}}^{t-k} = {}^{\mathcal{G}}\mathbf{T}_{\mathcal{O}(t)} \cdot {}^{\mathcal{O}(t-k)}\mathbf{T}_{\mathcal{G}} \cdot {}^{\mathcal{G}}\mathbf{p}^{t-k} \tag{2.2} $$

The first transformation in (2.2), ${}^{\mathcal{O}(t-k)}\mathbf{T}_{\mathcal{G}}$, returns the coordinate of $\mathbf{p}^{t-k}$ in $\mathcal{O}$'s body frame, which is constant under the rigid body assumption. The second transformation maps the point from the body frame to the global frame $\mathcal{G}$ using the object pose at time t, ${}^{\mathcal{G}}\mathbf{T}_{\mathcal{O}(t)}$. Hereon, we refer to ${}^{\mathcal{G}}\mathbf{T}_{\mathcal{O}(t)} \cdot {}^{\mathcal{O}(t-k)}\mathbf{T}_{\mathcal{G}}$ as the *rectification transformation*.

2.4 *Aligner*: Joint Objects Detection and Motion Estimation

The pipeline for estimating objects' motion shares several components with object detection models, specifically the computation of the BEV representation, which takes the form of a multi-channel image. Therefore, we propose to combine these two tasks in a unified architecture, which we name *Aligner*, shown in Fig.2.4.

In our pipeline, an EMC point cloud is voxelized and then processed by the CNN-based backbone to produce a BEV image $\mathbf{B}_0$. This image is consumed by the Object Motion Estimation branch, made of Point Head and Object Head, to estimate both scene flow and rectification transformation (2.2). To demonstrate our method, we choose two backbones: SECOND [129] and PointPillars [59].

2.4.1 Object Motion Estimation

Points' features are bilinearly interpolated from backbone-made BEV image $\mathbf{B}_0$ based on their projection to the BEV plane. Given points' features, the Point Head first segments the input point cloud into three classes: background, static foreground, and dynamic foreground. The definition of these classes is as follows:

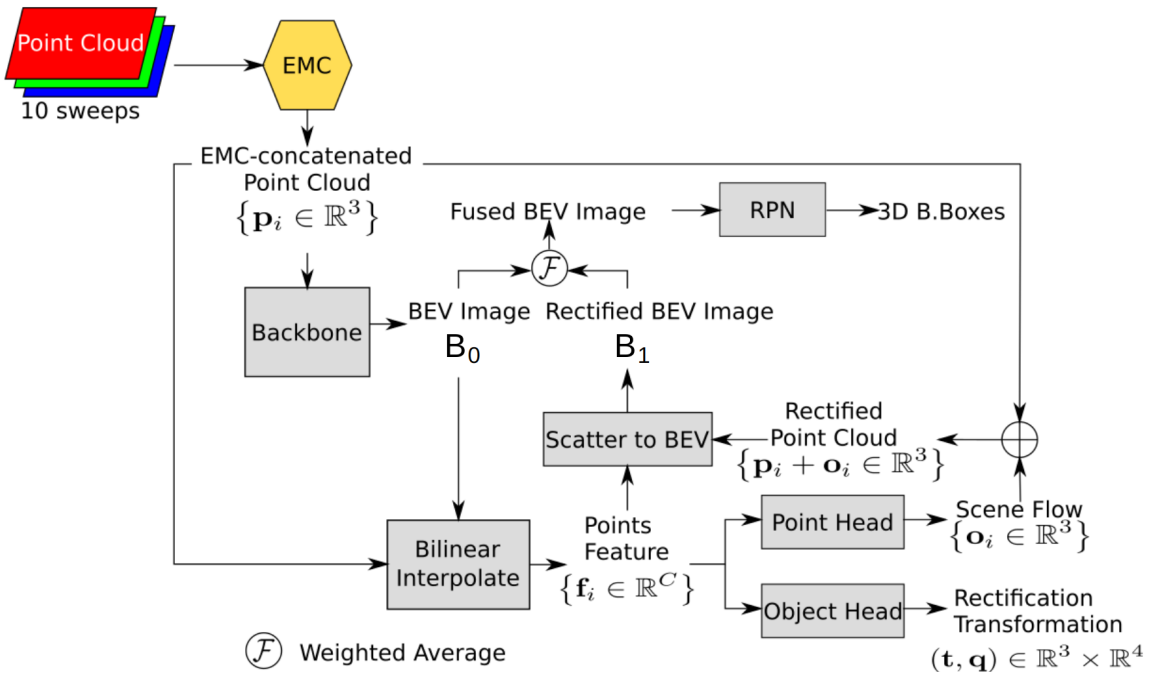


Figure 2.4 – Aligner's overview: A concatenated point cloud is first voxelized and converted to a BEV image $\mathbf{B}_0$ by a CNN backbone. Second, points' features $\{\mathbf{f}_i\}$ are obtained by bilinearly interpolating $\mathbf{B}_0$ using their projection on the BEV plane. Third, points' features are decoded into objects' rectification transformation and scene flow $\{\mathbf{o}_i\}$ respectively by Object Head and Point Head. The rectified BEV image $\mathbf{B}_1$ is the result of scattering $\{\mathbf{f}_i\}$ back to the BEV using the corrected point cloud $\{\mathbf{p}_i + \mathbf{o}_i\}$. Then, $\mathbf{B}_0$ and $\mathbf{B}_1$ are fused by a weighted sum before being consumed by the RPN to produce 3D bounding boxes.

- Background points are those on background objects (e.g., ground, building, traffic lights).
- Dynamic foreground points are those on foreground objects that exhibit a translation greater than 0.5 meters during the period of interest (e.g., 0.5 seconds).
- Static foreground points are neither background nor dynamic foreground.

For a dynamic foreground point $\mathbf{p}^{t-k}$, whose timestamp is $t-k$, the Point Head predicts a scene flow vector $\mathbf{o}^{t-k} \in \mathbb{R}^3$ which is defined as the difference between its location computed by EMC Eq.(2.1) and by using object trajectory Eq.(2.2).

$$\mathbf{o}^{t-k,*} =^{\mathcal{G}} \hat{\mathbf{p}}^{t-k} -^{\mathcal{G}} \mathbf{p}^{t-k} \tag{2.3}$$

Here, $*$ in the superscript denotes the ground truth.

In the Object Head, predicted foreground points are segmented into instances (i.e.,

objects) which we refer to as *global* groups. Each *global* group - $\mathcal{M}$ is further divided into *local* groups - $\mathcal{L}$ based on foreground points' timestamps. Let $\mathbf{f}_i \in \mathbb{R}^C$ denote the features of a foreground point $\mathbf{p}_i$. The features $\mathbf{f}_\mathcal{L}$ of a *local* group is computed as following

$$\mathbf{f}_\mathcal{L} = \text{Max}\left(\text{cat}\left(\mathbf{f}_i, \text{MLP}\left(\mathbf{p}_i - \bar{\mathbf{p}}_\mathcal{L}\right)\right) | \mathbf{p}_i \in \mathcal{L}\right) \tag{2.4}$$

Here, $\bar{\mathbf{p}}_\mathcal{L}$ is the mean coordinate of points in $\mathcal{L}$ and $\text{cat}(\cdot)$ refers to the channel-wise concatenation operation. The features $\mathbf{f}_\mathcal{M}$ of a global group $\mathcal{M}$ is the result of max pooling, $\text{Max}(\cdot)$, of its local groups' features.

$$\mathbf{f}_\mathcal{M} = \text{Max}\left(\mathbf{f}_{\mathcal{L}_j} | \mathcal{L}_j \in \mathcal{M}\right) \tag{2.5}$$

The rectification transformation in Eq.(2.2) is encoded as a 7-vector, which is the concatenation of the translation vector $\mathbf{t} \in \mathbb{R}^3$ and the quaternion $\mathbf{q} \in \mathbb{R}^4$ representing the rotation matrix. This transformation is predicted for each local group $\mathcal{L}_{t-k} (k = 0, \ldots, K)$ by an MLP shared among all local groups of all objects.

$$(\mathbf{t}, \mathbf{q})_{\mathcal{L}_{t-k}} = \text{MLP}\left[\text{cat}\left(\mathbf{f}_{\mathcal{L}_{t-k}}, \mathbf{f}_\mathcal{G}, \bar{\mathbf{p}}_{\mathcal{L}_{t-k}}, \bar{\mathbf{p}}_{\mathcal{L}_t}\right)\right] \tag{2.6}$$

At test time, the input EMC point cloud is rectified by translating dynamic foreground points $\mathbf{p}$ using their scene flow $\mathbf{o}$, instead of local groups' rectification transformation. As a result, the computationally expensive instance segmentation using DBSCAN can be bypassed, thus improving the model's inference speed.

2.4.2 BEV Image Rectification

The rectified point clo

MC according to their scene flow, is used to scatter points' features $\{\mathbf{f}_i\}$ back to the BEV, resulting in the rectified BEV image $\mathbf{B}_1$. Then, $\mathbf{B}_1$ is fused with backbone-made BEV image $\mathbf{B}_0$ via a weighted average. The weights are computed by stacking two BEV images along the channel dimension, then passing the result to a stack of two 2D Convolution layers with 3-by-3 kernels. The rationale of fusing $\mathbf{B}_1$ with $\mathbf{B}_0$ is as follows. $\mathbf{B}_1$ is fully sparse because it is created by scattering points' features in BEV. While not possessing the shadow effect, its sparsity harms the detection accuracy [112] as object centers, which is a strong indicator for localizing objects, are mostly in unoccupied regions. On the other hand, $\mathbf{B}_0$, which shows the shadow effect, is not sparse thanks to the

occupancy leaking caused by regular 2D convolution layers of the backbone. The fusion is to reduce the sparsity and prevent the shadow effect.

2.4.3 Region Proposal Network

The fused BEV image is consumed by a Region Proposal Network (RPN) to produce object detections as 3D bounding boxes. We use the anchor-based [129] and the center-based [137] RPN for SECOND and PointPillars, respectively.

The anchor-based RPN places two anchors in two orthogonal directions for each class of objects at each location of the BEV image and estimates the objectness of each anchor. For each positive anchor, the RPN also predicts a refinement vector to make the anchor fit tighter to its ground truth.

On the other hand, the center-based RPN encodes each ground truth object as a Gaussian on the BEV plane. The mean and covariance of each Gaussian are defined by its corresponding object's center and size, respectively. Then, it uses a series of Convolution layers to decode the input BEV image into pixel-wise center probability and box attributes (e.g., size and orientation).

2.4.4 Loss Function

Our model is trained end-to-end with a loss function L made of 3 terms corresponding to the loss of the two heads of the Object Motion Estimation branch and the RPN.

$$L = L_{\text{objects}} + L_{\text{points}} + L_{\text{RPN}} \tag{2.7}$$

Object Loss: L_{objects}

Let $(\mathbf{t}, \mathbf{q})$ and $(\mathbf{t}^*, \mathbf{q}^*)$ respectively be the prediction and ground truth of the rectification transformation of a local group $\mathcal{L}$. The object loss of this local group is

$$L_{\text{objects},\mathcal{L}} = \text{smooth}_{L_1}\left(\mathbf{t} - \mathbf{t}^*\right) + \left\|\text{Rot}\left(\mathbf{q}\right) - \text{Rot}\left(\mathbf{q}^*\right)\right\|_F + L_{\text{recon}} \tag{2.8}$$

Here, $\|\cdot\|_F$ denotes the Frobenius norm. $\text{Rot}\left(\cdot\right)$ represents the function that converts a quaternion to a rotation matrix. L_{recon} is the difference between points of $\mathcal{L}$ transformed

by the prediction and ground truth of the rectification transformation

$$L_{\text{recon}} = \frac{1}{N_{\mathbf{p}}^{\mathcal{L}}} \sum_{\mathbf{p} \in \mathcal{L}} \text{smooth}_{L_1} \left(\text{T} \left(\mathbf{t}, \mathbf{q} \right) \cdot \mathbf{p} - \text{T} \left(\mathbf{t}^*, \mathbf{q}^* \right) \cdot \mathbf{p} \right) \tag{2.9}$$

In Eq.(2.9), $N_{\mathbf{p}}^{\mathcal{L}}$ is the number of points in the local group $\mathcal{L}$. $\text{T}(\mathbf{t}, \mathbf{q})$ denotes the function that converts a translation vector $\mathbf{t}$ and a quaternion $\mathbf{q}$ to a homogenous transformation matrix.

L_{objects} in Eq.(2.7) is the sum of applying Eq.(2.8) to every local group $\mathcal{L}$ of every global group $\mathcal{G}$

$$L_{\text{objects}} = \frac{1}{N_{\mathcal{L}}} \sum_{\mathcal{G}} \sum_{\mathcal{L} \in \mathcal{G}} L_{\text{objects},\mathcal{L}} \tag{2.10}$$

where, $N_{\mathcal{L}}$ is the total number of local groups.

Point Loss: L_{points}

The loss of Object Motion Estimation's Point Head is made of classification loss, offset loss, and consistent loss.

$$L_{\text{points}} = L_{\text{cls}} + L_{\text{offset}} + L_{\text{consistent}} \tag{2.11}$$

Following [52], we use the sum of weighted binary cross entropy loss L_{bce} and Lovasz-Softmax loss L_{ls} [5] as the classification loss L_{cls}.

$$L_{\text{cls}} = \frac{1}{N_{\mathbf{p}}} \sum_{\mathbf{p}} \text{L}_{\text{bce}} \left(\mathbf{c}, \mathbf{c}^* \right) + \text{L}_{\text{ls}} \left(\mathbf{c}, \mathbf{c}^* \right) \tag{2.12}$$

In Eq.(2.12), $\mathbf{c}$ and $\mathbf{c}^*$ are the prediction and ground truth of the class probability of a point $\mathbf{p}$. $N_{\mathbf{p}}$ is the number of points in the inputted EMC point cloud.

L_{offset} measures the difference between dynamic foreground points $\mathbf{p}_+$ translated by predicted offset vectors $\mathbf{o}$ and their position after undergone the ground truth rectification transformation $(\mathbf{t}^*, \mathbf{q}^*)$.

$$L_{\text{offset}} = \frac{1}{N_{\mathbf{p}_+}} \text{smooth}_{L_1} \left(\mathbf{p}_+ + \mathbf{o} - \text{T} \left(\mathbf{t}^*, \mathbf{q}^* \right) \cdot \mathbf{p}_+ \right) \tag{2.13}$$

$N_{\mathbf{p}_+}$ is the number of dynamic foreground points in the input point cloud.

The Object Head groups points into local groups before predicting a rectification trans-

formation for each group, which is then applied to every point inside a group. For this reason, it enforces the rigid motion among dynamic objects which is a realistic assumption in the context of autonomous driving. On the other hand, the Point Head predicts an unconstrained offset vector for each dynamic point, thus risking rectified point clouds containing physically infeasible objects (e.g., deformed cars due to different parts undergoing different transformations). We reduce this risk by enforcing the consistency between predictions made by the two heads using $L_{\text{consistent}}$.

$$L_{\text{consistent}} = \frac{1}{N_{\mathbf{p}_+}} \text{smooth}_{L_1} \left(\mathbf{p}_+ + \mathbf{o} - \text{T} \left(\mathbf{t}, \mathbf{q} \right) \cdot \mathbf{p}_+ \right) \tag{2.14}$$

RPN: L_{RPN}

The loss function L_{RPN} of the anchor-based and center-based RPN are respectively defined according to Section.1.2.

2.5 *Aligner++*

To improve the accuracy of *Aligner* in estimating scene flow, thus ultimately improving object detection, we introduce two extensions (i) incorporating HD Map and (ii) distilling a model trained on the concatenation of point cloud sequences by the ground truth trajectories.

2.5.1 Incorporating HD Map

Previous works in integrating HD Maps into 3D object detection models [32, 130] opt for a mid-fusion approach that concatenates rasterized maps with backbone-made BEV representations. However, this approach is incompatible with copy-paste data augmentation [129], which randomly samples ground truth objects from a database and pastes them to each point cloud used for training, as pasted objects do not necessarily adhere to the semantics and geometry of the map (e.g., cars appear inside a building). Moreover, they only utilize the map's binary channels such as drivable areas, sidewalks, or car parks while omitting the lane direction which is potentially helpful for estimating scene flow and objects' heading direction.

Aware of these two limitations, we propose to extract the map feature for each 3D point and use it to augment the point's raw features prior to further processing (e.g.,

building a ground truth database or computing a BEV representation for object detection). The map feature extraction process is illustrated in Fig.2.5 where points' coordinate in the map's BEV is used for nearest neighbor interpolation. This attachment of the map features to points results in an early fusion between HD Maps and LiDARs, rendering the concatenation of maps' channels to backbone-made BEV representations unnecessary. As a result, HD Maps are made compatible with copy-paste data augmentation.

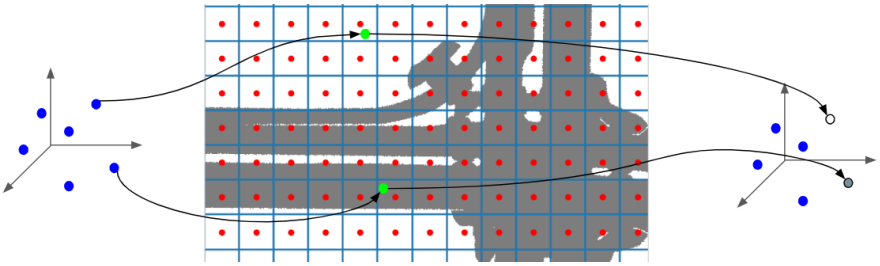


Figure 2.5 – Extraction of map features. LiDAR points are projected to the BEV. Their map feature is obtained by nearest-neighbor interpolation on the rasterized HD Map.

2.5.2 Distilling Shadow-Effect-Free Model

To improve the quality of the fused BEV representation, $\mathbf{B}^f$ in Fig.2.4, in terms of minimizing the feature misalignment caused by the shadow effect and reducing the sparsity, we use the teacher-student framework shown in Fig.2.6.

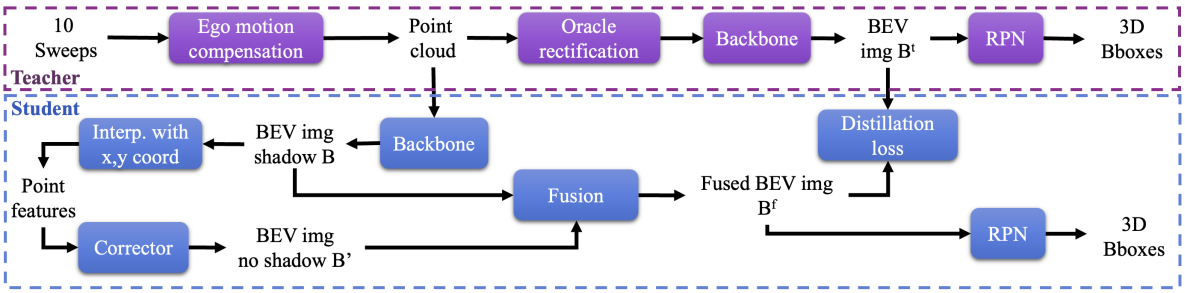


Figure 2.6 – *Alinger++* 's teacher-student framework: A model, pre-trained on point clouds concatenated using object ground truth trajectories, plays the role of the teacher. During training, the student model, which operates on point clouds concatenated using ego vehicle localization, strives to emulate the BEV images made by the teacher by minimizing a distillation loss.

In detail, we use ground truth objects' trajectories to rectify the concatenation of point cloud sequences and use the shadow-effect-free results, illustrated in Fig.2.7, to train a

teacher model. After the teacher converged, its BEV representation $\mathbf{B}^t$ is used to guide the student's fused BEV representation $\mathbf{B}^f$ by optimizing the students' weight so that the difference measured by the L_2 loss between these two representations is minimized.

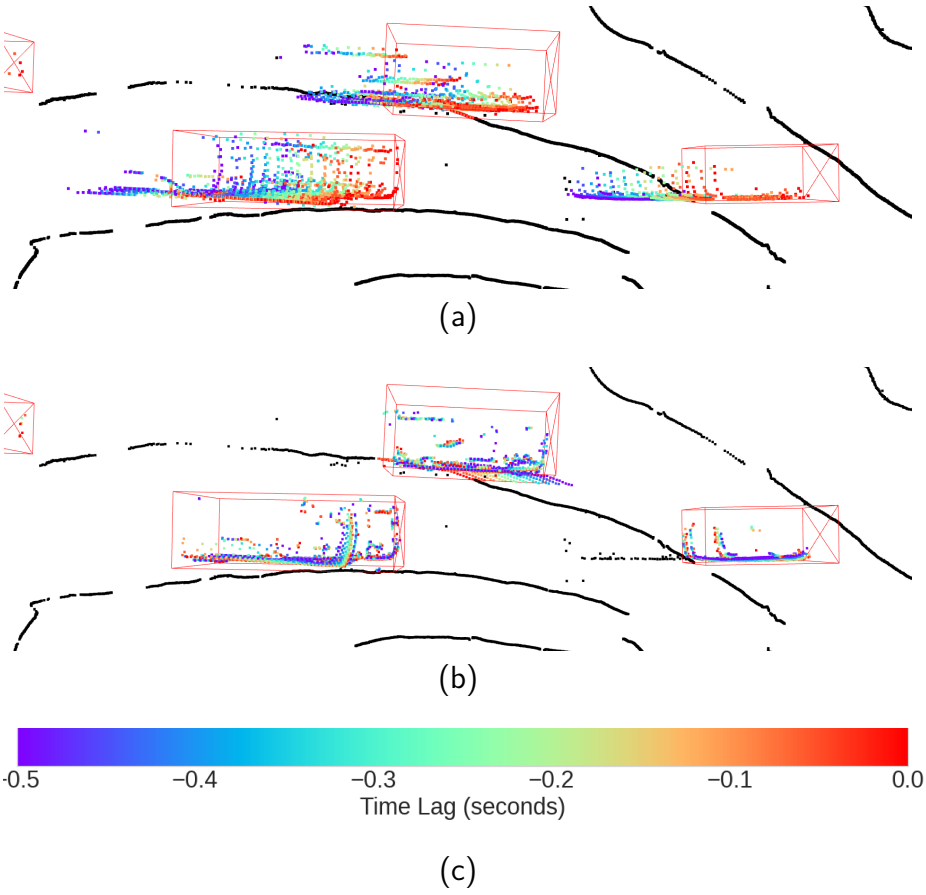


Figure 2.7 – Comparison between concatenation of a point cloud sequence (a) and its rectification using objects' ground truth trajectories (b). Points are color-coded according to their time lag with respect to the present. The color bar is shown in (c).

2.6 Experiments

2.6.1 Dataset and Evaluation Setting

As point clouds' resolution has a large impact on the performance of LiDAR-based models, we test our model on three resolutions: 16, 32, and 64. While 32-channel and 64-channel point clouds are readily available in NuScenes dataset [8] and KITTI dataset [36] respectively, to the best of our knowledge, there are no publicly available datasets

containing 16-channel point clouds. As a result, we synthesize a 16-channel dataset from the NuScenes dataset using the downsampling approach of [117].

The NuScenes dataset is made of 850 20-second scenes split into 700 scenes for training and 150 scenes for validation. Each scene comprises data samples collected by a multimodal sensor suite including a 32-beam LiDAR operating at 20 Hz. In *NuScenes*' convention, a keyframe is established once all sensors are in sync which happens every half a second. For each keyframe, objects are annotated as 3D bounding boxes. In addition, each object is assigned a unique ID that is kept consistent throughout a scene which enables us to generate the ground truth for the *rectification transformation* and scene flow. The downsampling of each point cloud in the *NuScenes* dataset to synthesize a 16-channel one is done by first identifying its points' beam index via K-mean (K is 32 in the case of *NuScenes*) clustering on their azimuth coordinate, then assigning an equivalent beam index with respect to 16-channel LiDAR. More details can be found in [117]. Hereon, we refer to this synthesized dataset as *NuScenes*-16.

The KITTI dataset 's 3D Object Detection partition contains 7481 and 7581 samples for training and testing. Each sample comprises sensory measurements collected by a 64-beam LiDAR operating at 10Hz and several cameras. A common practice when working with *KITTI* is to split the original training data into 3712 training samples and 3769 validation samples for experimental studies. A challenge we encounter when using *KITTI* is that its 3D Object Detection partition contains temporally disjointed samples, thus being not straightforward to obtain input (point cloud sequences) and ground truth (scene flow) for our model. We resolve this challenge by matching samples in the 3D Object Detection partition with *KITTI*'s raw sequences. Since there are a few raw sequences that do not have *tracklet* annotation, meaning objects are not tracked, we only retain data samples whose associated raw sequences have *tracklets* (i.e., trajectories of objects) in the training set and validation set.

Metrics We use mean Average Precision (mAP) to measure the performance of our model on the 3D object detection task. A prediction is matched with the closest ground truth, measured by an affinity. A match is considered valid if the affinity is below a predefined threshold. For each threshold, the average precision is obtained by integrating the recall-precision curve for recall and precision above 0.1. The mAP is the mean of the

average precision of the threshold set. For *NuScenes*, the affinity is the Euclidean distance on the ground plane between centers of predictions and ground truth. This distance has four thresholds: 0.5, 1.0, 2.0, and 4.0. For *KITTI*, the affinity is the Intersection-over-Union (IoU) in the BEV plane. Unlike *NuScenes*, *KITTI* uses only one affinity threshold for each class which is 0.7 for cars and 0.5 for pedestrians.

The evaluation of scene flow prediction is based on the set of standard metrics proposed by [76] which includes End-Point Error (EPE), strict/ relaxed accuracy (AccS/ AccR), and outlier (ROutliers). The EPE is the Euclidean distance between the predicted scene flow and their ground truth average over the total number of points. The AccS/ AccR is the percentage of points having either EPE < 0.05/ 0.10 meters or relative error < 0.05/ 0.10. The ROutliers is the percentage of points whose EPE > 0.30 meters and relative error > 0.30.

2.6.2 Implementation Details

We follow the common approach to handle the sparsity of NuScenes point clouds that concatenate a keyframe point cloud with all non-keyframe point clouds between itself and its predecessor using EMC [4, 8, 113, 134, 137, 151]. Let t denote the time step of the keyframe. The global frame $\mathcal{G}$ is set at the LiDAR frame at time step t. The point cloud of a non-keyframe collected at timestamp $t-k$ $(k = 1, \ldots, 9)$ are mapped from the LiDAR frame at this time step to the global frame using ego vehicle poses and LiDAR calibration.

The ground truth of rectification transformation in (2.2) requires knowing objects' poses in the global frame at the keyframe timestamp ${}^{\mathcal{G}}\mathbf{T}_{\mathcal{O}(t)}$ and non-keyframe timestamp ${}^{\mathcal{G}}\mathbf{T}_{\mathcal{O}(t-k)}$. Since NuScenes only provides objects' poses in keyframes, we obtain their poses in non-keyframes by linearly interpolating annotations of two keyframes that are respectively prior and successor to each non-keyframe.

To improve the generalization of our model, the following geometric transformations are applied to point clouds and ground truth: random flip along the x- and y-axis of the global frame, global scaling with a factor sampled from $\mathcal{U}_{[0.95,1.05]}$, and global rotation by an angle sampled from $\mathcal{U}_{[-\pi/8,\pi/8]}$ around the z-axis of the global frame. Here, $\mathcal{U}$ denotes the uniform distribution. Furthermore, we adopt the ground truth sampling strategy of [151] which randomly takes boxes and their points from a database and places them in the input point cloud. Notably, we introduce a modification to this sampling strategy by complementing each sampled ground truth with points in its trajectory spanning from the current time step to 10 steps in the past. This modification is similar to the "Sequence

GtAug" of [124]. Its motivation is to increase the number of moving objects in each point cloud, thus increasing the amount of supervision on the Object Motion Estimation. A comparison between the regular ground truth sampling and our modified version is shown in Fig.2.8.

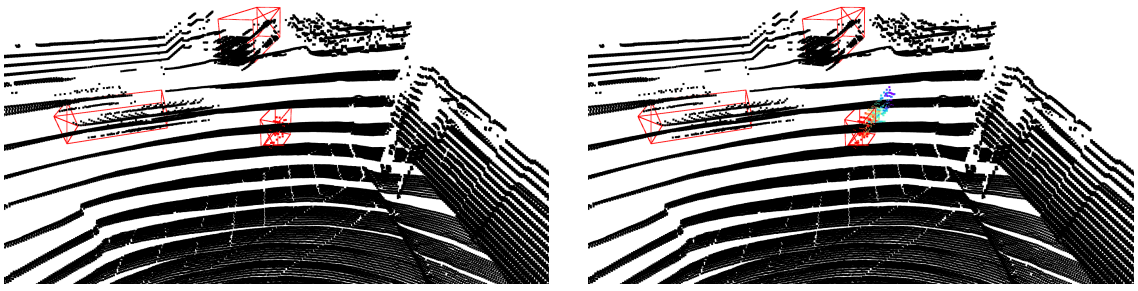


Figure 2.8 – The comparison between a scene augmented by the ground truth sampling strategy and by our strategy. Points belonging to the added object are colored-coded according to their timestamp. The hotter the color is, the more recent timestamp is.

In our experiments, input point clouds are limited to the range of $[-51.2, 51.2] \times [-51.2, 51.2] \times [-5.0, 3.0]$ meters along X-, Y-, and Z-axis of the world frame $\mathcal{W}$ and the voxel size is set to $(0.1, 0.1, 0.2)$. We use OpenPCDet [107] for our implementation. Further details on the model's hyperparameters can be found in our code release.

Due to the large size of the NuScenes dataset, we only train our model on a quarter of the training set. This mini partition of the training set is obtained by sorting keyframes by their point clouds' timestamps, then taking one every four keyframes. Our model is trained for 25 epochs with a total batch size of 16 distributed over 8 GPUs with sync batch norm. The optimizer is set to AdamW [80]. The learning rate is regulated by the One Cycle scheduler [103] with the maximum value of 0.003 for SECOND and 0.001 for PointPillars. It is worth noticing that the evaluation takes place on the entire validation set.

2.6.3 Results

Scene Flow Evaluation

The comparison of our model against methods specialized in estimating scene flow is shown in Tab.2.1. Since we only predict scene flow for dynamic foreground points, the

comparison in Tab.2.1 only accounts for these points. More importantly, we evaluate scene flow predicted by the Point Head of the Object Motion Estimation branch because the Object Head is deactivated at test time to avoid the computationally expensive instance segmentation using DBSCAN. In Tab.2.1 and the following tables, the best and second-best performances are marked by **bold** and underscore font, respectively.

Table 2.1 – Performance of Aligner and Aligner++ on scene flow metrics

Method	EPE ↓	AccS ↑	AccR ↑	ROutliers ↓	Runtime ↓ (seconds)
FLOT [90]	1.216	3.0	10.3	63.9	2.01
NSFPrior [65]	0.707	19.3	37.8	32.0	63.46
PPWC-Net [121]	0.661	7.6	24.2	31.9	0.99
WsRSF [40]	0.539	17.9	37.4	22.9	1.46
PCAccumulation [52]	**0.301**	26.6	53.4	12.1	0.25
PointPillar + Aligner	0.547	14.5	26.2	36.9	**0.06**
SECOND + Aligner	0.506	16.8	30.2	33.8	0.09
PointPillar + Aligner++	0.616	**46.4**	**66.6**	**6.8**	0.10

While being trained on fewer data (a quarter of NuScenes training set) and not having an architecture optimized solely for scene flow, our *Aligner* outperforms PPWC-Net and FLOT and are on par with WsRSF and NSFPrior. Notably, the integration of *Aligner* into SECOND is the second best in the metric EPE.

In addition, we report the runtime of our models measured in seconds on an NVIDIA A6000 GPU. The five baselines presented in Tab.2.1 have their runtime measured on an NVIDIA RTX 3090 GPU (as reported by [52]), which has a similar computing capability. Compared to them, our models achieve significantly better runtimes.

The last row of Tab.2.1 indicates a significant improvement compared to *Aligner* that reaches state-of-the-art on accuracy-related metrics namely AccS, AccR, and ROutliers while maintaining low inference time. The *Aligner++* has relatively high EPE this is because the evaluation is done for every point in the point cloud. This means the evaluation is carried out for both ground truth foreground points which have associated ground truth scene flow (can be nonzero or zero depending on dynamic) and ground truth background points which we assign the null vector as ground truth scene flow. Since the classification

module of the Object Head inevitably makes false positive / false negative foreground predictions, a number of background / foreground points are predicted to have nonzero / zero scene flow. Even though the portion of false predictions is small, as indicated by accuracy (AccS, AccR) and outlier metrics, the magnitude of their error is sufficiently large, due to the fact either the prediction or ground truth is zero, thus resulting in a large EPE.

Object Detection Evaluation

Effect of Aligner on object detectors: To verify the impact of aligning BEV representation using scene flow on the object detection performance, we modify SECOND and PointPillars to resemble the architecture shown in Fig.2.4 and train them end-to-end. Tab.2.2 shows an improvement brought by our module for most classes. Notably, trucks, construction vehicles, and buses, which exhibit severe shadow effects during motion due to their large size, enjoy significant performance gain (up to 7.8 AP or 16%). Furthermore, the detection improvement is higher on PointPillars since its BEV images have double the size of SECOND's, thus more severe feature misalignment. This highlights the importance of handling the misalignment between features and objects' locations. Interestingly, pedestrians also experience 0.8 AP improvement even though their motions violate the rigid body assumption made in Eq.(2.2).

Table 2.2 – Object detection results on NuScenes dataset evaluated by matching based on distance on BEV/ IoU

	Car	Truck	Const.	Bus	Trailer	Barrier	Motor.	Bicyc.	Pedes.	Traff.	mAP
SECOND	73.8/ 32.8	28.5/ 14.6	12.4/ 2.1	43.7/ 22.6	32.2/ 12.2	48.3/ 4.5	21.0/ 20.3	5.0/ 11.2	69.5/ 61.8	40.4/ 4.2	37.5/ 18.6
+ our module	**+0.8**/ **+1.1**	**+2.8**/ **+0.6**	**+1.6**/ **+0.4**	**+5.1**/ **0.6**	-1.2/ -2.0	**+2.1**/ **+1.0**	**+3.0**/ **+2.2**	-0.2/ -0.3	**+0.8**/ **+1.0**	**+0.6** **+0.3**	**+1.5**/ **+0.5**
PointPillars	78.9/ 27.0	37.9/ 13.2	4.2/ 0.3	48.9/ 20.8	21.4/ 4.0	48.4/ 3.8	28.1/24.9	7.3/ 15.8	73.3/ 65.4	41.5/ 2.5	39.0/ 17.8
+ our module	**+1.8**/ **+5.0**	**+7.3**/ **+3.7**	**+2.7**/ -0.2	**+7.8**/ **+6.1**	**+6.3**/ **+2.7**	-1.2/ **+0.3**	**+5.1**/ **+3.1**	**+0.3**/ -0.7	**+0.8**/ **+0.7**	**+3.5**/ **+1.0**	**+3.4**/ **+2.2**

Aligner compared to other multi-frame methods: The comparison of our models against other multi-frame methods on NuScenes is shown in Tab.2.3. Here, the matching between predictions and ground truth is based on the IoU. Our models exhibit strong performance in the class Pedestrians. As can be seen, our PointPillars $\frac{1}{2}$ exceeds the 66.1 AP of MultiXNet by 2.7 AP to be the second-best model despite being trained on only half of the data used for the baselines. We hypothesize that the removal of the shadow effect on pedestrians helps improve our models' generalization, thus getting high performance from fewer training data.

Table 2.3 – Object detection results on NuScenes dataset compared to other multi-frame methods. The numbers following the models' names denote the fraction of the NuScenes training set used for training.

	Pedestrians	Cars	Bicyclists
IntentNet [9]	63.4	60.3	31.8
MultiXNet [27]	66.1	<u>60.6</u>	<u>32.6</u>
MVFuseNet [58]	**76.4**	**67.8**	**44.5**
Our SECOND $\frac{1}{4}$	62.8	33.9	10.9
Our PointPillars $\frac{1}{2}$	<u>68.8</u>	40.5	29.3

On the other hand, we explain the gap between our models and baselines in class Cars and Bicyclists by the small-size training set. Due to limited computational resources, we only use up to half of the NuScenes training set to keep our experiments affordable. Tab.2.4 shows that scaling from one-eight to half of NuScenes training leads to a 17.8 and 19.3 increase in the AP of Cars and Bicyclists. Therefore, we believe our performance can greatly improve if more resources are available. Last but not least, our models are trained with a smaller mini-batch size, which is 16 compared to 32 and 64 of MultiXNet and MVFuseNet. As pointed out by [88], a small mini-batch size can hurt detection performance by (i) failing to provide accurate statistics for the Batch Normalization layer and (ii) possessing an imbalance number of positive and negative examples (e.g., the dominant of negative proposals at the early stage).

Table 2.4 – Evolution of the performance of our PointPillars with respect to the portion of the NuScenes training set that is actually used for training.

Size of actual training set	Pedestrians	Cars	Bicyclists	Training Time (hours)
1/8	59.4	22.7	10.0	15
1/4	66.1	32.0	15.1	30
1/2	68.8	40.5	29.3	60

Comparison between *Aligner++* and *Aligner* The better scene flow estimation of the *Aligner++* results in a higher detection accuracy on the *NuScenes* dataset which can be seen in Tab.2.5. The integration of *Aligner++*, made of HD Map and distillation using the teacher-student framework, improves the mAP average over 10 classes of objects of

a plain PointPillar by 6 points which is almost double the gain brought by *Aligner* (3.4 points). Interestingly, the distillation is responsible for most (5.6 out of 6 points) of the success of the *Aligner++*.

Table 2.5 – Detection improvement due to our extension

PointPillar	Aligner	HD Map	Distilling Teacher	mAP (avg 10 classes)
✓				39.0
✓	✓			42.4
✓	✓	✓		42.8
✓	✓	✓	✓	45.0

Robustness against LiDAR resolutions The robustness of our *Aligner++* is demonstrated by experiments on the *KITTI* and the synthetic *NuScenes-16* dataset. As can be seen Tab.2.6 and Tab.2.7, the integration of *Aligner++* consistently improves the accuracy of detecting objects in point cloud sequences. A common point between these two tables is that performance gain amounts to the *Aligner++* is larger for vehicle-like classes (e.g., car, truck, or trailer).

Table 2.6 – Detection performance on kitti dataset

Class	Model	Sequence Length			
		1	2	3	4
Car	PointPillar	66.54	70.44	69.66	69.88
	+ Aligner++	/	+3.91	+7.45	+6.3
Pedes-trian	PointPillar	5.32	6.16	16.71	20.77
	+ Aligner++	/	+4.82	+4.07	+3.6

This is because the scene flow ground truth is generated using the *rectification transformation* which is established based on the rigid motion assumption. This assumption holds for vehicle-like classes while being an oversimplification for pedestrians. As a result, the estimation of scene flow for vehicle-like classes is more accurate, thus higher detection accuracy. Another important result can be drawn from Tab.2.6 and Tab.2.7 is that the effectiveness, in terms of detection accuracy, of using point cloud sequences over single point clouds persists on various LiDAR resolutions.

Table 2.7 – Detection performance on nuscenes-16

Model	PointPillar		PointPillar + Aligner++
Seq Length	1	10	10
Car	56.14	79.84	80.81
Truck	26.99	48.64	49.39
Const. Vehicle	0.61	8.21	9.45
Bus	38.06	59.91	59.99
Trailer	11.05	26.15	29.03
Barrier	38.02	59.07	59.26
Motorcycle	12.70	41.14	41.72
Bicycle	0.11	15.22	18.40
Pedestrian	45.57	76.13	76.64
Traffic Cone	33.04	54.66	54.51
mAP (avg. 10 classes)	26.23	46.90	47.92

2.7 Conclusion

The multiple perspectives taken by the ego vehicle during the time span of a point cloud sequence and the independent motion of objects result in different observation angles and relative distances between LiDAR and objects. Therefore, objects, which is hit by a few LiDAR beams in the presence due to occlusion or long distance, could be hit by more beams in the past. As a result, the concatenation of point cloud sequences drastically increases the fidelity of measurements made by the ego vehicle compared to using individual point clouds, thus having the potential of boosting the performance of object detection models. Such benefit of point cloud sequences comes with a caveat which is the shadow effect on dynamic objects manifesting as objects' points smearing along their trajectories. This effect imposes a negative impact on detectors' performance by inducing a misalignment between objects' features and their locations defined by the projection of their bounding boxes to the BEV.

To minimize such impact, this chapter develops a method to rectify the BEV representation using scene flow. Specifically, a plug-in module called *Aligner* first computes the scene flow for every foreground point based on their features interpolated from the BEV representation. Then, foreground points are translated according to their scene flow. The resulting point cloud, which is shadow-effect-free, is used to scatter point-wise features

back to the BEV to make a sparse shadow-effect-free BEV representation. Finally, this new BEV representation is fused with the old one to minimize the shadow effect and sparsity. Experiments on various datasets containing point clouds of different resolutions show a significant performance gain brought by the proposed *Aligner* and its extension *Aligner++*. This success in utilizing point cloud sequences paves our first step toward resolving the challenge of low-fidelity measurements in point clouds caused by occlusion and sparsity.

As effective as point cloud sequences can get, however, there are scenarios where the scene geometry is so complicated that the motion of the ego vehicle during a short time span cannot provide point cloud sequences having sufficient coverage. An example is intersections where the field of view of the ego vehicle is severely reduced due to a large number of road users along with man-made structures (e.g., buildings) and plantations presenting in its proximity. A solution to these scenarios is to leverage the presence of other vehicles and intelligent roadside units (IRSU), which are advanced sensing systems strategically statically positioned to have unobstructed views, to add to the variety of perspectives where the measurements of the scene are obtained. The central idea of this solution is that fusing the measurements obtained by multiple agents (vehicles and IRSU) can minimize or even eliminate occlusion. A realization of this solution is presented in the next chapter.

Chapter 3

Improving Object Detectors using V2X Collaborative Perception

The previous chapter has made our first step toward resolving the challenge of low-fidelity measurements by enabling single-frame detectors to benefit from the multiple perspectives offered by point cloud sequences. These multiple perspectives stem from the motion of the ego vehicle during a sequence's time span. As the ego vehicle moves across the scene, far-away objects can become closed and those occluded can become visible. Therefore, the concatenation of point cloud sequences provides measurements on objects that have few or no LiDAR points at present, thus reducing the impact of low-fidelity measurements on detectors' performance.

Despite the capacity of point cloud sequences, there are scenarios where the scene geometry is so complex that the concatenation of point cloud sequences can not offer more information than individual point clouds. Examples of such scenarios are intersections, Fig.3.1, where the field of view of the ego vehicle is severely obstructed by a large number of road users presenting in its proximity. In this situation, point clouds obtained by the ego vehicle at several consecutive time steps share the same appearance that is the contour of nearby objects. As a result, external information is needed to help the ego vehicle improve its perception in these scenarios.

Collaborative perception via V2X communication, which leverages the diverse perspective thanks to the presence at multiple locations of connected agents to form a complete scene representation, is an appealing solution. The major challenge of V2X collaboration is the performance-bandwidth tradeoff which presents two questions (i) which information should be exchanged over the V2X network, and (ii) how the exchanged information is fused. The current state-of-the-art resolves to the Mid Collaboration approach where the BEV images of point clouds are communicated to enable a deep interaction among connected agents while reducing bandwidth consumption. While achieving strong performance, the real-world deployment of most Mid Collaboration approaches is hindered by

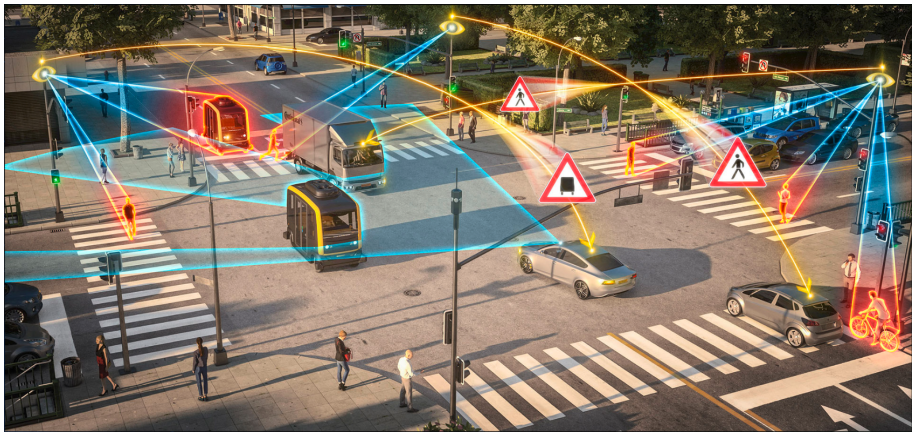


Figure 3.1 – Illustration of the impact of occlusion on the measurement made by autonomous vehicles at an intersection. Due to the presence of a large number of road users, individual vehicles inevitably have miss detection, some of which are safety critical.

their overly complicated architectures and unrealistic assumptions about inter-agent synchronization. This chapter devises a simple yet effective collaboration method based on exchanging agents' output that achieves a better bandwidth-performance tradeoff while keeping changes made to the single-vehicle detection models minimal. Moreover, the assumption about inter-agent synchronization is relaxed to the existence of a common time reference among connected agents, which can be achieved in practice using GPS time. Experiments on the V2X-Sim dataset [67] show that our collaboration method outperforms the early collaboration method while consuming as much bandwidth as the late collaboration. The method developed in this chapter is published in "Practical Collaborative Perception: A Framework for Asynchronous and Multi-Agent 3D Object Detection" which is under review at the time of writing. The code used for experiments in this chapter will be released in `https://github.com/quan-dao/practical-collab-perception`.

3.1 Introduction

Vehicle-to-Everything (V2X) collaborative perception is a promising solution to the challenge of low-fidelity measurements. Its core idea is to form a complete scene representation using measurements collected from multiple perspectives by leveraging the communication among multiple connected agents presenting at different locations. Connected agents can be either connected and automated vehicles (CAVs) or Intelligent Road-Side Units (IRSUs), which are advanced sensing systems strategically placed at elevated

locations to have maximal coverage of regions where complex traffic takes place. This enhanced perception capacity, thanks to V2X, comes with several new technical challenges; the most notorious among them being the performance-bandwidth tradeoff which presents two questions; (i) which information should be broadcast, and (ii) how the exchanged information should be fused.

This tradeoff establishes a spectrum of solutions ranging from Early to Late Collaboration. Raw measurements, which are point clouds in the context of this paper, are exchanged in the framework of Early Collaboration to reduce the impact of occlusion, thus achieving the highest performance at the expense of spending a very large amount of bandwidth. On the other extreme, Late Collaboration exchanges high-level outputs (e.g., object detection as 3D bounding boxes) to minimize bandwidth usage while limiting the performance gain thanks to collaboration. In an attempt to balance the two mutually excluding design targets, research on V2X collaboration frameworks [66, 116, 126, 128] are drawn toward the middle of this spectrum, thus the category's name of Mid Collaboration, where intermediate representations such as BEV images of agents' surrounding environment are chosen for broadcasting.

While the motivation is just, most Mid Collaboration methods require making substantial changes to the architecture of single-agent perception models to accommodate the fusion module where the combination of exchanged representations takes place. More importantly, these methods make strong assumptions about data synchronization among connected agents. For example, DiscoNet [66] considers a perfectly synced setting where agents share the same clock, collect and process point clouds at the same rate and the transmission/ receiving of BEV images experience zero latency. V2VNet [116] and ViT-V2X [128] account for latency by postulating a global misalignment between exchanged BEV images and those of the ego vehicle. The cause of such misalignment is the movement of the ego vehicle between the time step it queries the V2X network and the time when the exchanged BEV images are received. This implicitly assumes that agents in the V2X network obtain point clouds synchronously.

Another drawback of the current state-of-the-art of V2X collaborative perception is that only one point cloud per agent is used. Given that objects obscured in one frame may become visible in subsequent frames due to their movement or the motion of connected agents, and sparse regions might become dense as they draw nearer, harnessing sequences of point clouds is a compelling strategy to improve the performance of collaborative perception. In fact, the rich literature on multi-frame methods for single-vehicle object

detection [8, 27, 50, 58, 109, 136] has confirmed the effectiveness of point cloud sequences as a simple concatenation of point clouds in a common frame can boost detection accuracy by approximate 30% [8].

Finally, Mid Collaboration methods assume that connected agents share a common type of detector which is impractical for real-world deployment. This assumption is critical because of the domain gap between the BEV representation of point clouds made by different detectors which severely affects their fusion [126]. Prior work [126] resolves this challenge by introducing an additional module on top of those needed for the Mid Collaboration to account for differences in the BEV images made by SECOND [129] and PointPillar [81], thus further complicating the collaborative perception architecture.

Aware of the aforementioned drawbacks of previous works on V2X collaborative perception, we seek a practical collaboration framework that emphasizes:

- Minimal bandwidth consumption
- Minimal changes made to single-agent models
- Minimal inter-agent synchronization assumptions
- Support heterogeneous detectors networks

We aim our design at minimal bandwidth consumption, which can only be achieved by exchanging information about the objects detected by each agent. This design choice naturally satisfies the second design target as it dismisses the need for complex mid-representation fusion modules. We achieve the third target by only assuming that connected agents share a common time reference which is practically achievable using GPS time. To reach the last target, we decide to perform the fusion of exchanged information in the input of the ego vehicle's detector.

The challenge posed by our relaxed inter-agent synchronization assumption is that information (i.e., detected objects) broadcast by agents in the V2X network may never have the same timestamp as the query made by the ego vehicle. In other words, the detections made by other agents that are available on the V2X network always have an older timestamp compared to the current timestamp of the ego vehicle. This timestamp mismatch results in a misalignment between exchanged detected objects and their associated ground truths (if the detections are true positives), thus risking the overall performance. Our solution to this issue lies in the information that prior works have neglected - point cloud sequences. We reason that objects detected in the past can be propagated to the present if their velocities are available. The prediction of objects' velocities pertains to

motion prediction [27, 89, 110] or object tracking [119] which can negate our minimal architecture changes design target. Since we assume that agents produce predictions at least at their rate acquiring point clouds, we only need short-term (e.g., 0.1 seconds if point clouds are collected at 10Hz) velocity prediction, which can be computed using the scene flow, rather than long-term prediction (e.g., 3 seconds [89]) offered by motion prediction or tracking. As a result, we use the plug-in module for scene flow estimation developed in our previous work [21]. This choice effectively makes our V2X collaboration framework a multi-frame method, thus enabling each connected agent, as an individual, to enjoy a boost in detection accuracy as single-vehicle multi-frame methods do.

This chapter makes the following contributions:

- Deriving a practical framework for V2X collaborative perception that outperforms the Early Collaboration while consuming as much bandwidth as the Late Collaboration. In addition, our method does not make any assumptions about inter-agent synchronization except the existence of a common time reference, introduces minimal changes to the architecture of single-vehicle detectors, and supports heterogeneous detector networks.
- Demonstrate the benefit of point cloud sequences in V2X collaborative perception
- Performing extensive evaluations on V2X-Sim [67] datasets to verify our method

3.2 Related Works

As described in the previous section, the main challenge of V2X cooperative perception is the performance-bandwidth tradeoff which establishes a solution spectrum ranging from *early* to *late* collaboration. In the framework of Early Collaboration, as depicted in Fig.3.2, agents exchange their raw measurements - point clouds. At every timestep, the ego vehicle concatenates its own point cloud with those obtained by other agents to form the input for its perception model. This combined point cloud offers a comprehensive view of the scene with minimal occlusion and sparsity thanks to the diverse perspectives of connected agents, thus being regarded as the upper bound of the performance of the V2X collaborative perception [67, 140]. However, due to the significant amount of bandwidth required to transmit raw point clouds (the order of 10 MB), the early collaboration strategy is not feasible for real-world deployments.

On the other extreme of the performance-bandwidth tradeoff, Late Collaboration focuses on minimizing bandwidth usage by exchanging only the agents' output, precisely the

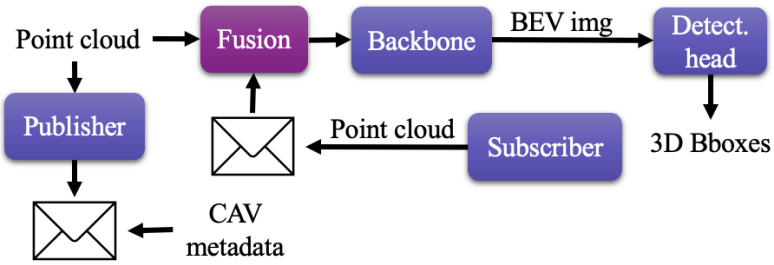


Figure 3.2 – Early Collaboration framework. Point clouds broadcast by connected agents are concatenated with those of the ego vehicle, forming the input to its detector.

detection results in the form of 3D bounding boxes. As illustrated in Fig.3.3, each agent independently detects objects using its point cloud. Subsequently, the agent merges its predictions with those made by others to generate the final output. While this strategy is more feasible for real-world deployments thanks to its minimal bandwidth consumption, it exhibits significantly lower performance gains compared to early fusion. The limited interaction among agents in the late collaboration approach contributes to this inferior performance. In noisy environments where latency is a factor, the late collaboration strategy even underperforms single-vehicle perception [128].

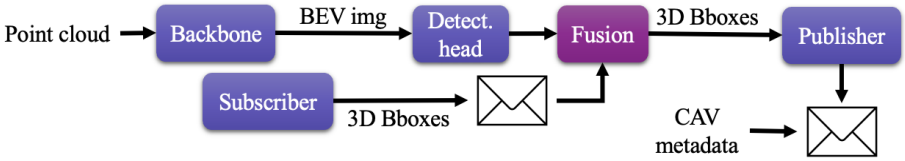


Figure 3.3 – Late Collaboration framework. Objects detected by other agents are fused with those detected by the ego vehicle to form its final output.

Mid Collaboration aims to find a balance between performance and bandwidth consumption by exchanging intermediate scene representations generated by the backbone of the agents' perception model. The motivation behind this approach is that the intermediate scene representation contains more contextual information compared to the final output (i.e., 3D bounding boxes), enabling deeper interaction among agents. Moreover, this representation is more compact than raw point clouds since it has been reduced in size through a series of convolution layers in the backbone and can be further compressed using an autoencoder to minimize bandwidth usage. While the idea is elegant, implementing the intermediate collaboration strategy requires a range of modules, shown in Fig.3.4, including compressor, decompressor, and representation fusion, among others, to match the performance of early collaboration. The fusion, in particular, is quite intricate as it

involves learnable collaboration graphs using techniques such as Graph Neural Networks [66, 116] or Transformers [128] to effectively fuse exchanged representations.

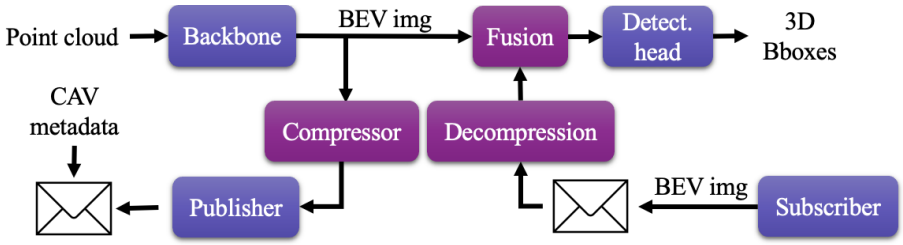


Figure 3.4 – Mid Collaboration framework. The ego vehicle fuses the BEV image of its point cloud with those broadcast by other agents to improve its performance.

In addition, dedicated modules are required to account for different practical challenges. For example, [116, 128] use the Spatial Transformer [53] to resolve the global misalignment between the ego vehicle's representation and others' caused by the ego vehicle's motion between when it makes the query and when it receives exchanged information. In [126], the Fused Axial Attention [127] is used to bridge the domain gap between representations made by different detection models (e.g., PointPillar [59] and VoxelNet[150]) used by different agents in the V2X network. Finally, most Mid Collaboration methods make strong assumptions about inter-agent synchronization which is either (i) perfect synchronization where exchanged representations always share the same timestamp [66] or (ii) synchronized point cloud acquisitions, meaning all agents obtain and process point clouds at the same rate and at the same time [116, 128]. Because of these complexities, the real-world deployment of intermediate fusion remains challenging.

3.3 Methodology

This chapter aims to resolve the aforementioned complexities of Mid Collaboration to obtain a practical framework for V2X collaborative perception. Our design is grounded in minimal bandwidth consumption as this is nonnegotiable in real-world deployment. To achieve this goal, we choose object detection, in the form of 3D bounding boxes, as the information to be exchanged, which is similar to the Late Collaboration strategy. We further relax the assumption on inter-agent synchronization to agents sharing a common time reference (e.g., GPS time) and acknowledge that agents produce detections at different rates. As a result, exchanged detections always have older timestamps compared

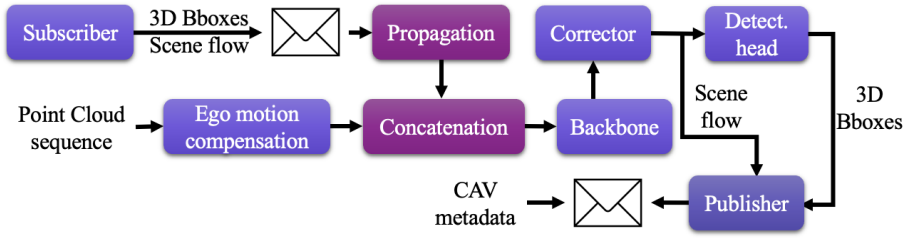


Figure 3.5 – Our *late-early* collaboration framework for V2X Cooperative Perception. In which, objects detected by each connected agent at a past time that is closest to the present are broadcast. Exchanged detected objects are propagated to the present using their velocity computed by pooling point-wise scene flow, then fused with the point cloud which the ego vehicle collected at the present to enhance its perception.

to the timestamp of the query made by the ego vehicle, thus risking a spatial misalignment between exchanged detections and their associated ground truth (if detections are true positive). We resolve this misalignment by predicting objects' velocity simultaneously with their locations by pooling point-wise scene flow which can be produced by integrating our *Aligner* module, developed in Sec.2.4, to any BEV-based object detectors. Finally, we avoid the inferior performance of Late Collaboration by devising a new collaboration strategy that fuses exchanged detections with the ego vehicle's raw point cloud for subsequent processing by its detection model.

The resulting framework is illustrated in Fig.3.5. We name our method *Late-Early* Collaboration as connected agents broadcast their outputs, which is the signature of Late Collaboration, while the ego vehicle fuse received information with its point cloud, which is the signature of Early Collaboration.

The innovation of our collaboration strategy lies in our recognition of the similarity between object detection using point cloud sequences and collaborative detection. In both cases, there is a need to fuse information obtained from diverse perspectives. Point cloud sequences involve capturing the motion of the ego vehicle, which results in varying viewpoints, while collaborative detection entails incorporating insights from other agents present in the environment. By drawing this parallel, we leverage the shared principle of fusing information from multiple perspectives to enhance the accuracy and robustness of both object detection approaches. Specifically, we utilize the latest advancement in the multi-frame object detection literature, called *MoDAR* [68], which interprets previously detected objects to 3D points with additional features made of object sizes, heading, confident score, and predicted class. These points are propagated to the present and merged with the point cloud obtained at the present to form the input of any off-the-shelf detec-

tors.

3.3.1 Problem Definition

The V2X setting that we target comprises multiple CAVs and IRSUs which are collectively referred to as agents. An agent $\mathcal{A}_i$ is equipped with a LiDAR to localize in a common global frame $\mathcal{G}$ and detect objects in its surrounding environment. The detection is based on the processing of point cloud sequences using a detection model made of the integration of the *Aligner* presented in the Sec.2.4 into an off-the-shelf single-frame object detector such as PointPillar [59]. At a time step t_i, agent $\mathcal{A}_i$ uses a K-point-cloud sequence $\mathcal{S}_i = \left\{\mathcal{P}_i^{t_i-K+1}, \ldots, \mathcal{P}_i^{t_i}\right\}$ as an input to its detection model. An object $\mathbf{b}_{i,j}$ detected by agent $\mathcal{A}_i$ is parameterized by a nine-vector $[x, y, z, w, l, h, \theta, s, c]$. The first seven numbers localize the object by its center location $[x, y, z]$, size $[w, l, h]$, heading direction θ. The last two numbers, s and c, respectively denote confidence score s and the predicted class c.

Upon receiving a query having timestamp t, agent $\mathcal{A}_i$ will communicate its detection $\mathcal{B}_i^{t_i} = \{\mathbf{b}_{i,j}\}_{j=1}^{M_i}$ and metadata produced timestamp t_i that is prior to and closest to t. The agent's metadata produced at a timestamp t_i is made of the timestamp itself and the agent's pose $\mathcal{E}_i(t_i)$ at this timestamp.

Given this setting, we aim to enhance the ego vehicle's capacity of detecting objects by fusing its point cloud sequence $\mathcal{S}_e = \left\{\mathcal{P}_e^{t-K+1}, \ldots, \mathcal{P}_e^{t}\right\}$ with the *MoDAR* interpretation of predictions $\mathcal{B}_i^{t_i}$ made by other agents. The following section provides an overview of the origin of *MoDAR* and presents in detail how we adapt this concept to collaborative perception via the V2X context.

3.3.2 MoDAR for Object Detection on Point Cloud Sequences

MoDAR is created to enable detecting objects in extremely long point cloud sequences (hundreds of frames). In the *MoDAR* framework, a sequence is divided into several short sequences where objects are detected by a single-frame detector, tracked by a simple multi-object tracktor (e.g., [119]). Then, the data-driven motion forecasting model MultiPath++[110] predicts objects' future poses based on objects' trajectories established by the tractor. Using prediction about objects' future poses, detected objects in each short subsequence are propagated to the desired time step (e.g., the present). Next, each propagated object, which is represented as an up-right 3D bounding box (parameterized by the

location of its center, size, and heading) with a confidence score and a class, is interpreted into a 3D point that takes the box's center as its coordinate and the box's size, heading, confidence score and class as features. These points are referred to as *MoDAR* points. They enable packing an entire subsequence into a small number of points, thus enabling an efficient fusion of extremely long point cloud sequences.

3.3.3 V2X Collaboration using MoDAR Points

We draw the following similarity between single-vehicle object detection on point cloud sequences and V2X collaborative detection: these two tasks share the common challenge of finding an effective method for fusing information obtained from different perspectives caused by the motion of the ego vehicle in the case of point cloud sequences and the presence of other agents in the case of collaboration via V2X. Based on this observation, we use *MoDAR* points as the medium for conveying information among agents in the V2X network. Specifically, we interpret an object detected by agent $\mathcal{A}_i$ as a 3D bounding box $\mathbf{b}_{i,j} = [x, y, z, w, l, h, \theta, s, c]$ to a *MoDAR* point $\mathbf{m}_{i,j}$ by assigning

- $[x, y, z]$ to the coordinate of $\mathbf{m}_{i,j}$
- $[w, l, h, \theta, s, c]$ to $\mathbf{m}_{i,j}$'s features

The challenge in our V2X setting is that different agents detect objects at different rates, thus forcing the ego vehicle to utilize *MoDAR* points made by other agents at passed time steps. This timestamp mismatch results in a spatial misalignment between exchanged *MoDAR* points and ground truth dynamic objects, which can diminish the benefit of collaboration or even decrease the ego vehicle's accuracy. This challenge is encountered in the context of single-vehicle detection on point cloud sequences as well because dynamic objects change their poses from one subsequence to another. Li et al [68] resolves this by predicting objects' pose using a multi-object tracktor and the motion forecasting model MultiPath++.

While this is feasible in the V2X context, the implementation of those modules for future pose prediction does not align with our design target of minimal architecture. Instead, we use the scene flow to propagate *MoDAR* points from a passed timestep to the timestep queried by the ego vehicle. Since *MoDAR* points are virtual, their scene flow is not estimated directly from a point cloud sequence but is aggregated from the scene flow of points residing in the box they represent. To be concrete, let $\mathbf{m}_{i,j}$ be a *MoDAR* point representing a 3D bounding box $\mathbf{b}_{i,j}$ detected by agent $\mathcal{A}_i$ at timestep t_i. $\mathbf{p}_{i,h}$ and

$\mathbf{o}_{i,h}$ respectively denote a real 3D point in the concatenation of agent $\mathcal{A}_i$'s point cloud sequence $\mathcal{S}_i$ and its predicted scene flow. The scene flow $\mathbf{o}^{\mathbf{m}}_{i,j}$ of $\mathbf{m}_{i,j}$ is computed by

$$\mathbf{o}^{\mathbf{m}}_{i,j} = \text{mean}\left\{\mathbf{o}_{i,h} \mid \mathbf{p}_{i,h} \in \mathbf{b}_{i,j}\right\} \tag{3.1}$$

Once its scene flow is obtained, the *MoDAR* point $\mathbf{m}_{i,j}$ is propagated to the timestep t queried by the ego vehicle as following

$$[\hat{\mathbf{m}}_{i,j}]_{x,y,z} = [\mathbf{m}_{i,j}]_{x,y,z} + \frac{t - t_i}{|\mathcal{S}_i|}\ \mathbf{o}^{\mathbf{m}}_{i,j} \tag{3.2}$$

Here, $|\mathcal{S}_i|$ denotes the length of the point cloud sequence $\mathcal{S}_i$ measured in seconds. $[\cdot]_{x,y,z}$ is the operator that extracts 3D coordinate of a *MoDAR* point.

Finally, propagated *MoDAR* points are transformed from the agent $\mathcal{A}_i$'s pose $\mathcal{E}_i(t_i)$ at time step t_i to the ego vehicle pose $\mathcal{E}_e(t)$ at time step t using the localization in the common global frame $\mathcal{G}$ of the two agents

$${}^{\mathcal{E}_e(t)}[\hat{\mathbf{m}}_{i,j}]_{x,y,z} = {}^{\mathcal{G}}\mathbf{T}^{-1}_{\mathcal{E}_e(t)}\ {}^{\mathcal{G}}\mathbf{T}_{\mathcal{E}_i(t_i)}\ [\hat{\mathbf{m}}_{i,j}]_{x,y,z} \tag{3.3}$$

The concatenation between the set of *MoDAR* points received from other agents and the ego vehicle's raw point cloud (resulting from concatenating its own point cloud sequence) is done straightforwardly by padding

- points in the ego vehicle's raw point cloud with null vectors representing features of *MoDAR* points, which are boxes' size, heading, score, and class
- *MoDAR* points with null vectors representing features of points in the ego vehicle's raw point cloud which are points' intensity and time-lag.

Once exchanged *MoDAR* points and the point cloud of the ego vehicle are merged, the result can be processed by the single-vehicle model with *Aligner* integrated.

3.4 Experiments

3.4.1 Dataset and Metric

To evaluate our collaboration framework, we use the V2X-Sim 2.0 [67] which is made using CARLA [29] and the traffic simulator SUMO [79]. This dataset is made of 100 100-frame sequences of traffic taking place at intersections of three towns of CARLA which are

Town 3, Town 4, and Town 5. Each sequence contains data samples recorded at 5 Hz. Each data sample comprises raw sensory measurements made by the ego vehicle, one to four CAVs, and an IRSU, which is placed at an elevated position that has a large minimally occluded field of view of the intersection. Every vehicle and the IRSU are equipped with a 32-channel LiDAR. All agents are in sync which results in the same timestamp of data that they collect.

The V2X-Sim 2.0 dataset provides object annotations for each data sample. The official training, validation, and testing split are made of temporally disjoint data samples from three towns chosen such that there is no overlap in terms of intersections. Since we need point cloud sequences as input to our models, we can't use the official splits. Instead, we use sequences in Town 4 and Town 5 as the training set, and those in Town 3 as the validation set, thus ensuring there is no intersection overlap. This choice results in an 8900-data-sample training set and an 1100-data-sample validation set. Since this dataset follows the format of the NuScenes, we use NuScenes' implementation of mean Average Precision (mAP) (details in Sec.2.6.1) to measure the performance of our framework and baselines.

3.4.2 Implementation

In the convention of the V2X-Sim dataset, the IRSU and the ego vehicle are respectively assigned the identity of 0 and 1 while other CAVs get identities ranging from 2 to 5. To test our collaboration method's ability to handle asynchronous exchanged information, we set the time lag between the timestamp t of the ego vehicle's query and the timestamp t_i of the detection $\mathcal{B}_i = \{\mathbf{b}_{i,j}\}$ made by agent $\mathcal{A}_i (i \in \{0, 2, 3, 4, 5\})$ to the time gap between two consecutive data sample of V2X-Sim which is 0.2 seconds.

In the implementation of our V2X collaboration framework, every agent uses the single-vehicle detection model that is developed in Sec.2.4. The architecture and hyper-parameters are kept unchanged as the experiments of the single-vehicle model in Sec.2.6.2. While previous works using the V2X-Sim dataset [66, 67] set the detection range to $[-32, 32]$ meters along the X and Y axis, centered on the ego vehicle, we extend this range to $[-51.2, 51.2]$ to better demonstrate the performance gain thanks to collaborative perception via V2X.

Enhance Detection of Visible Objects

We benchmark our approach to collaborative perception against two extremes of the performance -bandwidth spectrum which are Late and Early collaboration. Late Collaboration achieves the minimal bandwidth usage by fusing the detection $\mathcal{B}_1 = \{\mathbf{b}_{1,j}\}_{j=1}^{|\mathcal{B}_1|}$ made by the ego vehicle with the detection made by other agents $\{\mathcal{B}_i | i \in \{0, 2, 3, 4, 5\}\}$ using Non-Max Suppression. We evaluate this baseline under three settings including

- *asynchronous* exchange where the gap between t_i and t is 0.2 seconds as described above
- *asynchronous* exchange with $\mathcal{B}_i$ propagated from t_i to t using scene flow by the procedure described in section 3.3.3
- *synchronous* exchange where there is no gap between t and t_i

On the other hand, Early Collaboration reaches high performance by exchanging the entire raw point cloud sequences $\mathcal{S}_i = \left\{\mathcal{P}_i^{t_i-K+1}, \ldots, \mathcal{P}_i^{t_i}\right\}$ collected by each agent $\mathcal{A}_i$ is the second baseline.

To verify the ability to enhance the single-vehicle perception of V2X collaboration, we evaluate our approach and baselines in the setting where ground truths are made of objects visible to the ego vehicle. This means eligible ground truth must contain at least one point of the point cloud $\mathcal{P}_e^t$, which the ego vehicle obtained at the time step of query t. The result of this evaluation is summarized in Tab.3.1 which shows that late collaboration in all three settings is not beneficial, as the best late collaboration is only 93.3% (4.42 mAP behind) of the single-agent perception. This is because True Positive (TP) detections made by other agents are counted as false positives if they are not visible to the ego vehicle. In addition, ill-localized but overly confident detections made by other agents can suppress good detections made by the ego vehicle, thus further reducing the overall performance. Remarkably, our collaboration approach using *MoDAR* points outperforms the early collaboration by 4.7% (3.35 mAP). This reaffirms our design philosophy that *a good multi-agent collaborative perception framework can be made on the foundation of a good single-agent perception model and a simple collaboration method.*

Enhance Detection of Invisible Objects

Besides enabling the ego vehicle to detect more accurately objects that are visible to itself, another great benefit of collaborative perception is to help the ego vehicle see the invisible objects - those that do not contain any points of its point cloud $\mathcal{P}_e^t$, thus

Table 3.1 – Performance of Collaboration Methods on Ground Truth Visible to The Ego Vehicle

Collaboration Method	mAP
None	65.71
Late - *async*	54.77
Late - *async* prop.	59.34
Late - *sync*	61.29
Early	71.84
Ours	**75.19**

overcoming the challenge of occlusion and sparsity. To demonstrate this, we relax the criterion of eligibility of ground truth such that they only need to contain at least one point emitted by any agents in the V2X network. In this experiment, we add a strong Mid Collaboration method - DiscoNet [66] to the set of baselines to showcase the capability of our approach compared to the state-of-the-art. In the framework of DiscoNet, connected agents, which have identical detection architecture, exchange BEV images of their point cloud. The ego vehicle fuses the BEV image of its point cloud with the exchanged BEV via a fully connected collaboration graph to use as the input to the detection head.

In addition to the detection precision, we also measure the bandwidth consumption of each collaboration method. Our measurement is based on the raw uncompressed form of the exchanged data to facilitate a fair comparison.

The results shown in Tab.3.2 indicate that on a larger set of ground truths, Late Collaboration does improve performance, compared to single-vehicle (no collaboration). The improvement is significant (at least 8.35 mAP) even in the poorest setting where the set of exchanged detection $\{\mathcal{B}_i | i \in \{0, 2, 3, 4, 5\}\}$ is 0.2 seconds behind the time of query, thus resulting a spatial misalignment between detected objects that are exchanged and their associated ground truths if the underlying objects are dynamic. This can be explained by a significant number of static and slow-moving objects present in intersections whose past detections remain true positives at the present. When detections of dynamic objects are perfectly accounted for as in the *sync* setting where agents exchange their detections at the same timestamp as the query, the performance is largely improved by 9.29 mAP (15.2%), compared to the *async* setting. However, this *sync* setting is unrealistic because different agents have different detection rates. Interestingly, propagating detections using

scene flow as in the *async* prop. setting can reach 96.2% the performance of the *sync* setting (2.68 mAP behind). This implies the effectiveness of scene flow estimated by our *Aligner++*.

Table 3.2 – performance of collaboration methods, measured by mAP, on ground truth that are visible to at least one agent

Collaboration Method	Sync	Async	Bandwidth Usage (MB)
None	52.84	-	0
Late	70.48	61.19 67.80 [1]	**0.01**
Mid - DiscoNet [2]	<u>78.70</u>	73.10	25.16
Early	78.10	**77.32**	33.95
Ours	**79.20**	<u>76.72</u>	**0.01**

In the *sync* setting, DiscoNet narrowly exceeds Early Collaboration by 0.6 mAP (0.7%), thus showing the benefit of the deep interaction among agents powered by their dense collaboration graph. However, this benefit comes at the cost of relatively high bandwidth usage of 25 MB per broadcast BEV image on average which is only 25% less than the average bandwidth required by Early Collaboration. The operation of our method in this setting is as follows: *MoDAR* points broadcast by other agents have the same timestamp as the point cloud of the ego vehicle; therefore their propagation using scene flow is skipped. Our method slightly exceeds DiscoNet by 0.5 mAP (0.6%) to reach the highest performance while consuming the least bandwidth among collaborative methods. This confirms the effectiveness of *MoDAR* in facilitating collaboration among connected agents.

In the *async* setting, DiscoNet is severely affected by the misalignment among exchanged BEV images caused by timestamp mismatch as its performance dropped by 5.6 mAP (7.1%). Our method is affected because the misalignment between exchanged *MoDAR* points and their corresponding ground truth is explicitly taken into account by the propagation of *MoDAR* points using scene flow. As a result, the performance gap between ours and DiscoNet is extended to 3.62 mAP (5%). Early Collaboration shows the most robustness against timestamp mismatch with an 0.8 mAP (1%) drop. This can be explained by the detector's ability to implicitly handle to some extent the misalignment

among point clouds. This phenomenon is well known in the literature on multi-frame single-vehicle object detection [8, 133, 137, 151] where models trained on the concatenation of 0.5-second point cloud sequences enjoy a 30% mAP gain compared to models trained on individual point clouds.

Support Heterogeneous Detector Networks

To show that our collaboration method supports heterogeneous networks out of the box, we perform an experiment where we start with a network of agents uniformly using PointPillar as their detectors and gradually replace PointPillar with SECOND in one agent after another. The result of this experiment is shown in Fig.3.6. As the number

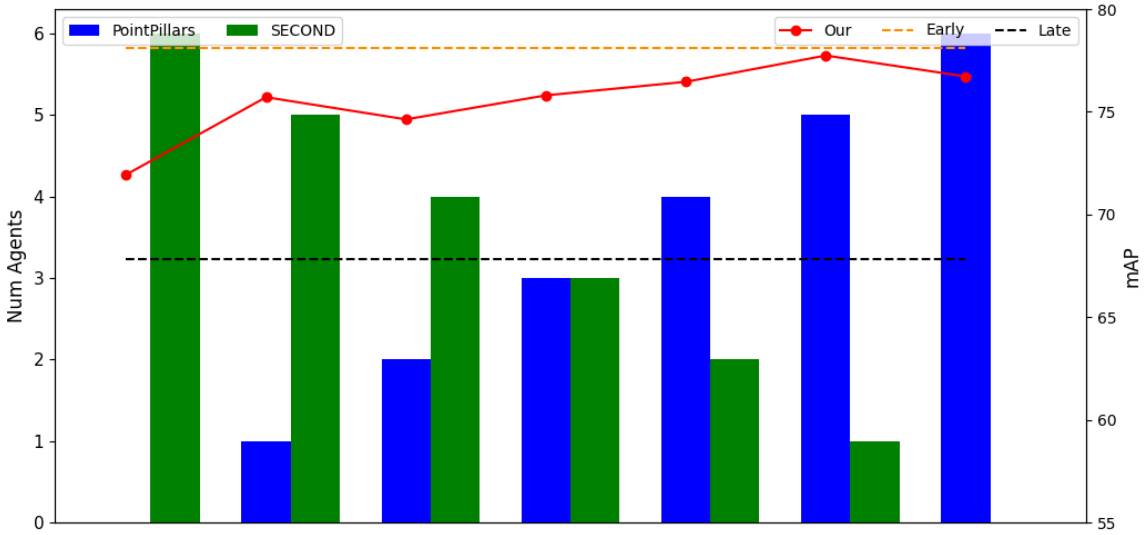


Figure 3.6 – Performance of the ego vehicle in heterogeneous collaboration networks made PointPillar and SECOND detectors. As the number of SECONDs increases from 0 to 6 - the size of the network, the ego vehicle's performance transits from its value in a full-PointPillar network to its value in a full-SECOND network without catastrophe dropping out of the lower bound of Late Collaboration.

of SECOND increases, the performance of the collaborative perception gradually transits in a downward trend from 76.72 mAP of a full-PointPillar network to 71.94 mAP of a full-SECOND network, without dropping below the lower bound of Late Collaboration, which is the case of Mid Collaboration method without dedicated modules for bridging

1. Exchanged detections are propagated using scene flow according to (3.2) before fusion by Non-Max Suppression.
2. Our implementation based on [66]

the domain gap [126]. This downward trend is because an individual PointPillar achieves higher AP than an individual SECOND on the V2X-Sim dataset when being trained under the same setting. More interesting, the combination made of the RSU using SECOND and all connected vehicles using PointPillar archives the best performance of 77.74 mAP, reaching 99.5% performance of Early Collaboration in *sync* setting.

Relation between Performance and Network Size

Finally, to study how the performance of collaboration perception evolves with respect to the number of agents in the V2X networks, we gradually increase the number of participants in the collaboration. As can be seen in Fig.3.7, the collaboration with only the IRSU brings 15.09 mAP (28.6%) improvement.

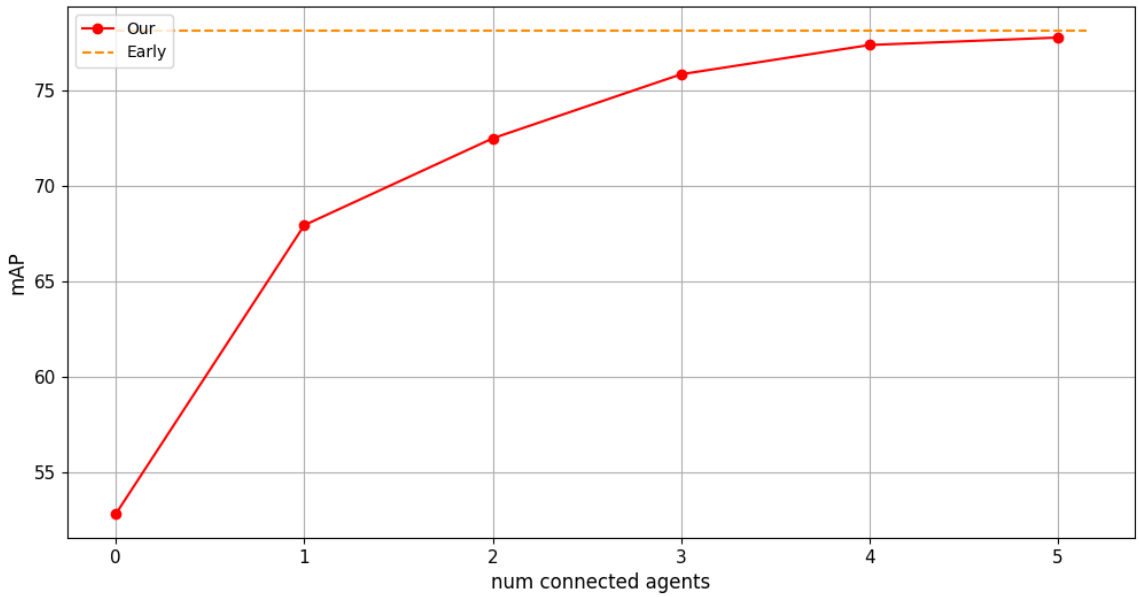


Figure 3.7 – The performance of ego vehicle's detection increases sharply in collaboration with only IRSU, then slows down and saturates at 98.2% of Early Collaboration in *sync* setting as the number of connected agents increases.

The magnitude of performance gain reduces and eventually saturated as the number of agents increases. An increased number of agents results in higher overlap in agents' field of view which makes objects in the overlapped regions well detected by many agents. However, this overlap also means the field of view of the collaboration network as a whole does not increase proportionally with the number of agents. Moreover, exchanging detected objects made by distant agents may not get used by the ego vehicle because they

are outside the predefined detection range.

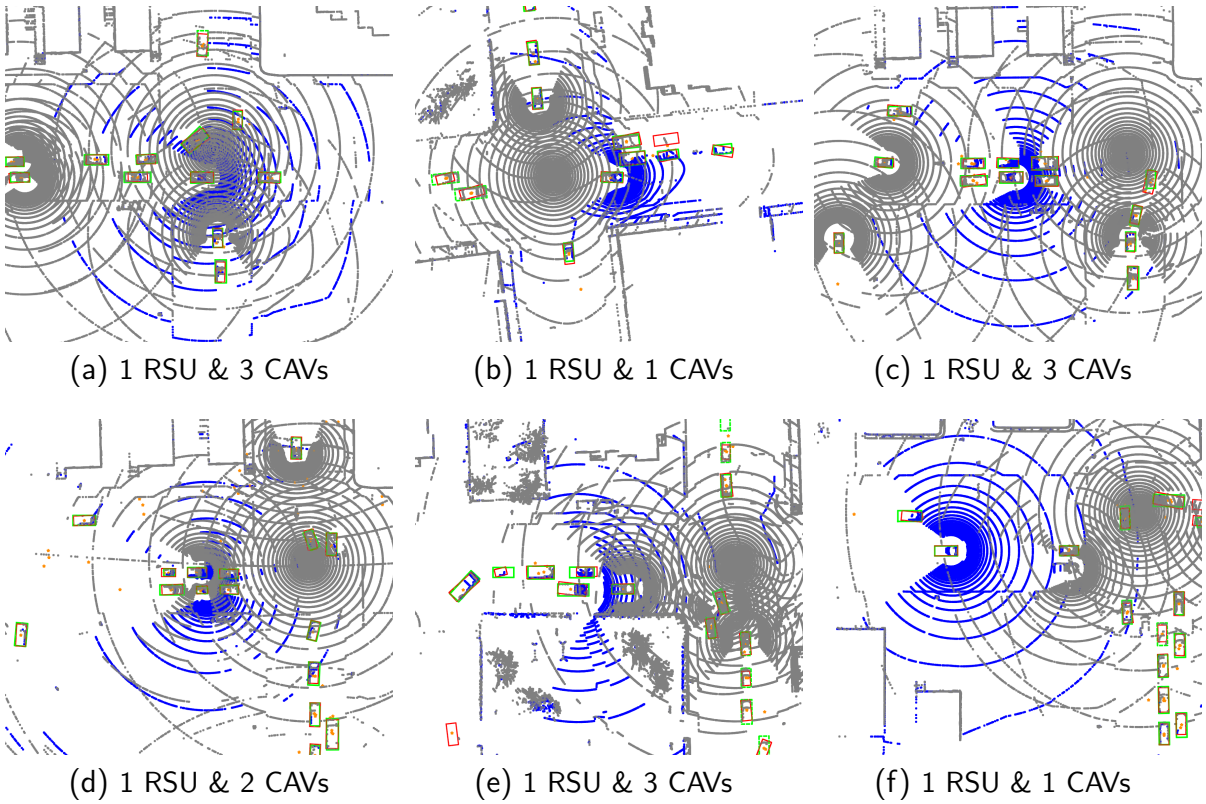


Figure 3.8 – Qualitative performance of our method on three samples of the V2X-Sim dataset. The number and type of connected agents are annotated in the title of each image. Blue points are LiDAR points collected by the ego vehicle. Gray points are LiDAR points collected by other agents which are displayed for visualization purposes only. Orange stars denote the MoDAR points broadcast by other connected agents. Green solid and dashed rectangles respectively represent ground truth visible and invisible to the ego vehicle. Red rectangles are the detections made by the ego vehicle using our method.

Qualitative Performance

The qualitative performance in Fig.3.8 shows that collaborative perception using *MoDAR* points enables the ego vehicle to detect objects that are occluded or have a few to zero LiDAR points due to long range. Particularly, the vehicle at the top of Fig.3.8a is occluded with respect to the ego vehicle due to the presence of a large vehicle. However, this occluded vehicle is successfully detected thanks to a single MoDAR point produced by the IRSU. Another example of occluded vehicles that are successfully detected is Fig.3.8e

where vehicles in the bottom right corner are occluded by a building-like structure. The illustration of successful detection at long range thanks to *MoDAR* points can be found in Fig.3.8b, Fig.3.8d, and Fig.3.8f where vehicles near their edges have a few or even zero LiDAR points of the ego vehicle.

3.5 Conclusions

This chapter brings together two solutions to the challenge of low-fidelity measurement in a unified framework that leverages the benefit of multiple perspectives, offered by the motion of the ego vehicles and by the presence of other connected agents, to the fullest. The result is a collaborative perception method that enables the ego vehicle to detect objects that are completely unobservable to its LiDAR due to occlusion or sparsity at long range. More importantly, the method developed here is highly feasible for real-world deployment because it (i) minimizes bandwidth usage, (ii) dismisses the need for inter-agent synchronization, (iii) makes minimal changes to single-agent object detectors, and (iv) supports networks of heterogeneous detectors.

Since our collaborative perception method is built on the foundation of single-agent detectors, its performance is inevitably capped by the performance of individual agents participating in the V2X network. An example failed case is where an object that is unobservable to the ego vehicle and is not detected by the agent to which is observable. In this case, there is no *MoDAR* point created for this false negative detection, thus making it remain unobservable to the ego vehicle after collaboration. Another case is regions, which are unobservable to the ego vehicle, are populated with high-confident false positive detections made by other connected agents. These high-confident false positive detections can induce false positive detections in the collaborative model. These failed cases highlight the need for improving single-agent detectors such that false positive and false negative detections are minimized.

The two most popular single-agent detectors are PointPillar and SECOND which are designed to have a theoretical recall rate of 100%. This design choice enables them to enjoy a significantly high recall rate in practice. For example, SECOND achieves a 0.95 recall rate on the KITTI dataset [24]. Such a high recall rate means a reduced amount of false negative detections but at the cost of an increased amount of false positive detections. The balance between false negative and false positive can be achieved by adding a second stage to the existing single-agent detectors to re-score their output so

that false positive detections have low confident scores. In addition, the second stage can re-adjust the prediction of single-agent detectors to make true positive detections fit better to their associated ground truth. The development of such a second stage is presented in the next chapter.

Chapter 4

Improving Object Detectors using Refinement Stage

As the challenge of low-fidelity measurement is addressed by Chapter.2 using point cloud sequences and by Chapter.3 using V2X collaborative perception, the perception capacity of the ego vehicle is tremendously increased such that it is able to detect objects that are invisible to its LiDAR. However, the detection result is not yet perfect due to the presence of high-confident detection in background regions and poorly localized true positive detection (i.e., detected boxes that have low overlap with their corresponding ground truth). These limitations are caused by the limited representation capacity of the BEV images which compress the 3D space into rather small size images (e.g., 128×128) using large pixel sizes (e.g., 0.8 meters). The low resolution is particularly challenging for detecting small objects such as pedestrians or cyclists as they often occupy a small number of pixels. A straightforward resolution increase can compromise a detector's accuracy on large objects if the receptive field of each pixel in BEV does not sufficiently cover the entire object it contains. As a result, the depth of the detector needs to be increased, thus increasing computational overhead and GPU memory consumption. Striking the balance in the accuracy of large and small objects is challenging.

Deng et al [24] make the observation that anchor-based detectors such as SECOND [129] achieve an exceptionally high recall rate (up to 95%) while having moderate precision. This is due to a large number of false positives and poorly localized detection. This observation makes a strong case for a two-stage approach toward 3D object detection which

1. recovers as much ground truth as possible
2. re-scores and re-adjusts detected objects to remove false positives and improve object localization, respectively

In order words, a two-stage detector first tries to achieve high recall, then picks only

good detections and makes them better using the second stage to achieve high precision. Following this two-stage approach, small objects can be well detected by detectors using low-resolution BEV images.

Aware of the reasoning above, this chapter develops a refinement stage to further improve the performance of LiDAR-based object detectors. The method developed in this chapter is published in "Attention-based Proposals Refinement for 3D Object Detection", IEEE Intelligent Vehicles Symposium, 2022. The code used for experiments in this chapter is available at `https://github.com/quan-dao/APRO3D-Net`.

4.1 Introduction

This stems from the observation that anchor-based methods such as SECOND [129] have exceptionally high recall rate (up to 95%) yet only achieves a moderate performance, e.g. SECOND's 78 Average Precision (AP) for Car class in KITTI. The role of a refinement stage is to unleash the full potential of anchor-based methods. The key to the refinement stage is how to compute ROI features effectively. Early works, e.g. PartA^2 [101], PV-RCNN [102] and VoxelRCNN [24], address this by first dividing ROI into a 3D grid then extracting feature at each grid location before feeding the concatenation of grid point features to a Multi-Layer Perceptron (MLP) to obtain the desired output. Their motivation is that such a grid can recover the 3D structure lost in the BEV representation used in the region proposal stage. Arguing that computing grid point features require several hand-crafted components, CT3D [99] devises a variant of the transformer [111] to compute ROI feature directly from points pooled from raw point clouds. Though having less inductive bias, CT3D achieves state-of-the-art performance, demonstrating the benefit of integrating the transformer into the 3D detection pipeline.

This chapter adds to the family of two-stage voxel-based 3D object detectors by making two main contributions. First, we develop a new ROI Feature Encoder (RFE) for computing per-proposal features based on Vector Attention [146]. RFE, together with a detection head, can serve as a refinement stage for voxel-based and point-based region proposal frameworks. Second, we observe that strong methods such as PV-RCNN [102] and CT3D [99] employ additional modules to learn pooled point features, thus increasing model size and reducing frame rate. Therefore, we propose to pool directly from feature maps generated by the backbone during the region proposal process. Inspired by [16], our pooling strategy effectively fuses multi-scale features, thus increasing the model's ability

to detect classes of different sizes.

Compared to PartA^2, PV-RCNN, and VoxelRCNN, we are similar in the interest of using inductive bias to compensate for the loss of 3D structure in the BEV representation used in the proposal generation stage. While their inductive bias is to impose a grid structure to ROI, ours takes place in position encoding of pooled points (Section 4.2.2). Specifically, pooled points' coordinates are mapped to ROI's canonical frame and then augmented with their displacement vector to ROI's eight vertices. The difference between pooled points' augmented coordinates and that of ROI's center is input to an MLP for computing position encoding.

Our pooling strategy uses the same source as VoxelRCNN which is the intermediate feature maps generated by the 3D backbone of the region proposal framework. Instead of concatenating features pooled across different scales like VoxelRCNN, we first pool from the highest one to compute initial ROI features, then update these initial ROI features using features pooled from another feature map at a lower scale. The reason is to condition ROI features obtained at lower scales on the higher ones, thus encouraging the consistency of learned features throughout the architecture.

CT3D [99] is the closest to our proposed approach since we share the method of computing ROI features via the attention mechanism. Compared to CT3D, we have two key differences. First, we use a different formulation of the attention mechanism, namely Vector Attention [146], to assign different attention weights to different channels of one point feature. The motivation will be explained in Section 4.2.2. Second, CT3D pools from the raw point cloud to enable its integration into virtually any detection framework. Such flexibility comes at the cost of ignoring the valuable intermediate results of the region proposal process. This forces CT3D to recompute features for pooled points before transforming them to ROI features using self-attention in which a pooled point feature is a weighted sum of others'. Our method pools from the backbone's intermediate feature maps. As a result, it is no longer necessary to recompute pooled features. Furthermore, by re-using backbone features, our pooling strategy maximizes the use of the information produced in the region proposal process.

4.2 APRO3d-Net for 3d object detection

Figure 4.1 shows the overview of APRO3D-Net made of integrating our RFE modules to SECOND [129]. Our framework first interprets feature maps created during the region

proposal process into point-wise features. ROIs pool these points based on their relative locations. Pooled point features and their position encoding that incorporate an ROI's geometry are transformed into the ROI's features by the Vector Attention. The ROI's features are mapped to its confidence score and refinement vector by two MLP-made heads.

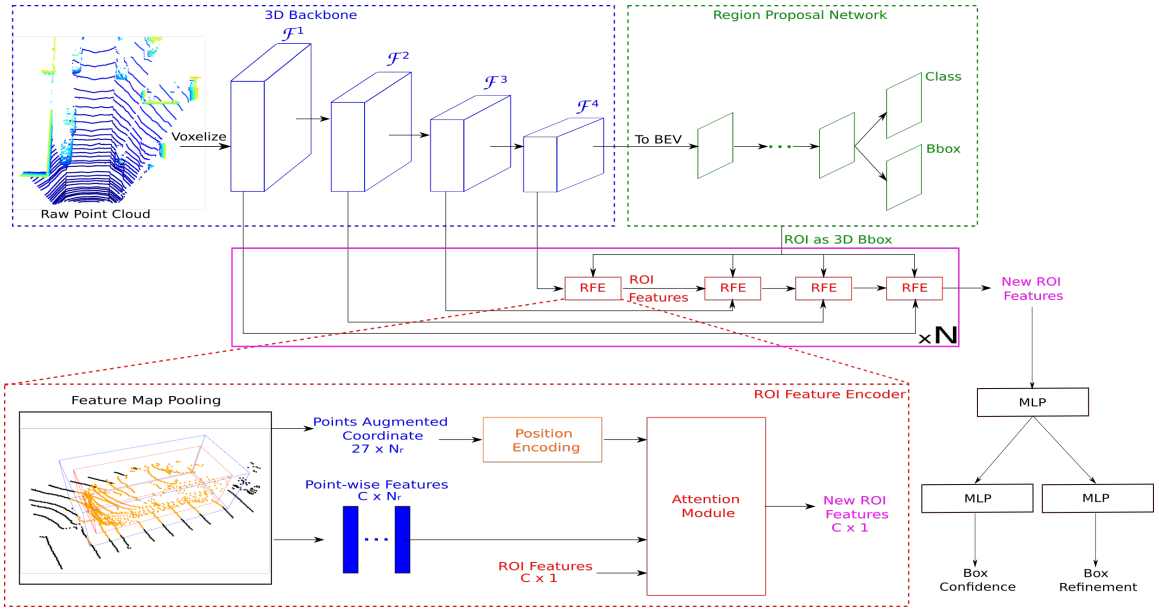


Figure 4.1 – The overall architecture of APRO3D-Net. The voxelized point cloud is fed to a 3D backbone for feature extraction. The backbone's output is then converted to a BEV representation on which an RPN is applied to generate ROI. Several ROI Feature Encoders (RFE) transform feature maps produced by backbone into ROI features by first pooling from inputted feature maps, then encoding pooled points position, and finally refining previous ROI features using pooled features and their position encoding via Attention Module. The refined ROI feature is mapped to confidence and refinement vector by two MLP-based detection heads. Here, blue cuboids and green parallelograms respectively denote feature maps computed by 3D and 2D convolution. Notice that the channel dimension is omitted for clarity.

4.2.1 3D Backbone and Region Proposal Network

The reason for choosing SECOND to demonstrate our method instead of a point-based method such as PointRCNN is twofold. First, point-based methods are not as computationally efficient because of their repetitive use of query operations (ball query and k-nearest neighbor query), which can take up to 80% computational time [78]. More

importantly, the model's final performance is strongly conditioned by how well ROIs produced by the RPN cover ground truth boxes. The indicator for such ability is the RPN's recall rate, which [101] shows is higher with SECOND-like RPN.

SECOND first uses a backbone made of Sparse Convolutions to learn a compact representation of the input point cloud. The backbone's final output is a C-channel feature volume $D \times H \times W$. It is then converted into a BEV representation of size $(C \times D) \times H \times W$ by flattening the channel and depth dimension. Each location of the resulting BEV image is associated with multiple anchors corresponding to different classes and orientations. Finally, an RPN made of a standard 2D CNN predicts class probability and offset vector w.r.t associated ground truth for each anchor. Anchors modified by predicted offset vectors become ROI. ROIs are post-processed by the non-max suppression procedure to remove redundant but low confident ROI to make the final output $\mathcal{R}$

$$\mathcal{R} = \{([x_r, y_r, z_r, dx_r, dy_r, dz_r, \theta_r], \mathrm{cls}_r)\}_{r=1}^{M} \tag{4.1}$$

where each ROI is parameterized by the location of its center $[x_r, y_r, z_r]$, its size $[dx_r, dy_r, dz_r]$, its heading direction (i.e. yaw angle) θ_r and its class cls_r.

4.2.2 ROI Feature Encoder

RFE has three sub-modules: Feature Map Pooling, Position Encoding, and Attention Module. The Feature Map Pooling interprets backbone-generated feature volumes into point-wise features and pools them according to their location relative to ROIs' bounding box. Points pooled by an ROI are assigned position encoding vectors to incorporate the ROI's geometry. The Attention Module transforms pooled points' features and their position encoding into the ROI's feature via the Vector Attention [146], which essentially is a weighted sum of pooled point features.

Feature Maps Pooling

Define the ego vehicle frame $\mathcal{E}$ as the following its origin is at the center of the rear axel of the ego vehicle, the X-axis coincides with the ego vehicle's heading direction, the Z-axis is the reversed gravity direction, and the Y-axis is the cross product of Z and X-axis. Without loss of generalization, assume that point clouds are expressed in this frame. An occupied voxel (d, h, w) of the feature map $\mathbf{F}^i (i = 1, \ldots, 4)$ (Figure 4.1) is interpreted into a 3D location (x, y, z) by

$$^{\mathcal{E}}\Big[x, y, z\Big] = \Big(\Big[w, h, d\Big] + 0.5\Big) \cdot V^i + {}^{\mathcal{E}}\Big[x, y, z\Big]_{\min} \tag{4.2}$$

Here, V^i is the size voxels in $\mathbf{F}^i$. d, h, w are respectively the voxel's *grid location* along the Z-axis, Y-axis, and X-axis. $^{\mathcal{E}}[x, y, z]_{\min}$ is the minimum metric coordinate of the detection region in the ego vehicle frame. Applying Eq.(4.2) to every occupied voxel of $\mathbf{F}^i$ results in a set of point-wise features $\mathcal{P}^i = \Big\{\Big({}^{\mathtt{L}}\mathbf{p}^i_j = \Big[x^i_j, y^i_j, z^i_j\Big], \mathbf{f}^i_j\Big)\Big\}_{j=1}^{N^i}$. Here, $\mathbf{f}^i_j$ is the feature at the grid location that gives rise to $\mathbf{p}^i_j$, while N^i is the number of occupied voxels in $\mathbf{F}^i$.

The pooling scheme, illustrated in the bottom-left corner of Figure 4.1, is performed based on the location of point-wise features $\mathcal{P}^i$ relative to the *enlarged* ROIs. The enlargement of ROIs is to incorporate missing foreground points due to the mismanagement between ROIs and their corresponding ground truth. Let $\mathfrak{R}_r$ denote the 3D volume occupied by an ROI r after being enlarged by $[\Delta_x, \Delta_y, \Delta_z]$. A point feature $\Big(\mathbf{p}^i_j, \mathbf{f}^i_j\Big)$ is pooled into ROI r if

$$\mathbf{p}^i_j \in \mathfrak{R}_r \tag{4.3}$$

Inspired by [100, 101], we transform pooled point-wise features to ROI's canonical coordinate system to reduce the variance during training, thus improving the model's generality. This coordinate system, shown in Fig.4.2, is defined as the following

- origin is at ROI's center
- the X-axis has the same direction as ROI's heading direction
- the Z-axis is vertical and points upward

From Eq.(4.1), a ROI is characterized by a seven-vector $[x_r, y_r, z_r, dx_r, dy_r, dz_r, \theta_r]$. A point $\mathbf{p}_j$ is transformed to ROI r's canonical frame by

$$^{r}\mathbf{p}_j = \begin{bmatrix} \cos\theta_r & \sin\theta_r & 0 \\ -\sin\theta_r & \cos\theta_r & 0 \\ 0 & 0 & 1 \end{bmatrix} \left({}^{\mathcal{E}}\mathbf{p}^T_j - \begin{bmatrix} x_r \\ y_r \\ z_r \end{bmatrix} \right) \tag{4.4}$$

Attention Module

Discussion Once pooled, point-wise features are used to compute a single feature vector representing the entire ROI. A straightforward method is to transform points individually (via an MLP) and then aggregate them using a permutation invariance operation (e.g., sum, mean, or max pooling). However, this approach disregards valuable information about an ROI's geometry, such as point distribution or the ROI's size. To remedy this,

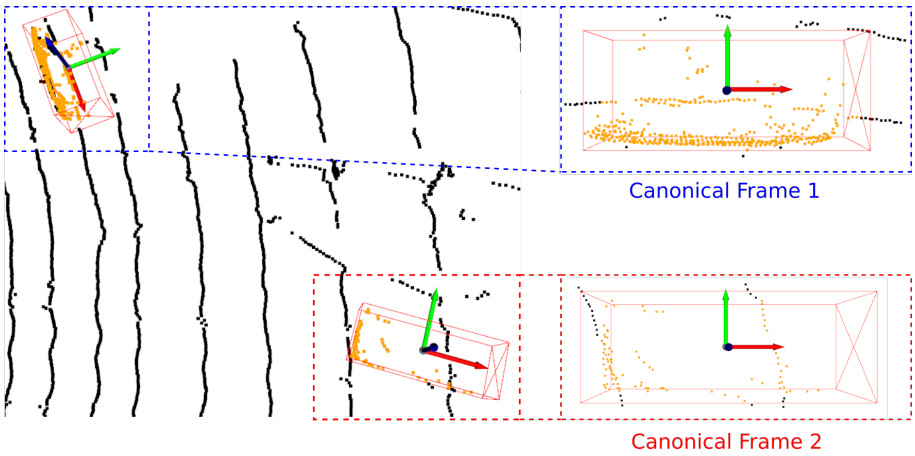


Figure 4.2 – ROIs and pooled points in LiDAR frame (Left) compared to them in ROIs' canonical frame (Right). Here, ROIs are denoted by red cuboids while pooled points are colored orange. Red, green, and blue arrows respectively represent the canonical frame's X, Y, and Z axis.

we propose to use the attention mechanism to compute the ROI feature given pooled point-wise features. The advantage of using the attention mechanism is two folds

- The model can dynamically define how much each point feature contributes to an ROI feature, thus naturally reducing the impact of background points while not suppressing them entirely. Such a balance can be helpful because background points, especially those on the ground, can provide context for estimating height.
- ROIs' geometry information (e.g., points location, ROI size) can be explicitly injected into the computation by position encoding.

While the original multi-head attention [111] used by ViT [28] and its variants have achieved remarkable successes in the realm of computer vision, it has a drawback of treating every channel equally. In other words, a single set of scalar weights is assigned to C-dimension point-wise features in the weighted sum for an ROI feature. Since we pool from feature maps generated by the backbone made of convolution layers, each channel of any feature map is a detector for a certain feature [143]. Therefore, using a single set of scalar weights can risk less important features overshadowing important ones. In addition, using multi-head attention can introduce inconsistency since ViT and CNNs learn significantly different features [93]. For the reasons above, we opt for Vector Attention [146] which is effective in 3D classification and segmentation tasks [147].

Vector Attention Essentially, the computation of an ROI feature is cross-attention, where the ROI feature queries the set of pooled point-wise features. Let $\mathbf{r}$ be the initial value of the ROI feature, and $\mathcal{P}_r = \{({}^r\mathbf{p}_j, \mathbf{f}_j)\}_{j=1}^{N_r}$ be the set of pooled point-wise features. The new ROI feature $\hat{\mathbf{r}}$ is computed by

$$\hat{\mathbf{r}} = \sum_{\mathcal{P}_r} \text{softmax}\left(\text{MLP}\left(\varphi\left(\mathbf{r}\right) - \psi\left(\mathbf{f}_j\right) + \boldsymbol{\zeta}\right)\right) \odot \left(\alpha\left(\mathbf{f}_j\right) + \boldsymbol{\zeta}\right) \tag{4.5}$$

Here, φ, ψ, α are linear projections. $\odot$ denotes the Hadamard product (i.e., element-wise multiplication). $\boldsymbol{\zeta}$ represents the position encoding, whose detail will be presented shortly. In Eq.(4.5), $\varphi\left(\mathbf{r}\right), \psi\left(\mathbf{f}_j\right), \alpha\left(\mathbf{f}_j\right)$ respectively take the role of query, key, and value.

Using a different set of weights for each channel requires storing $N_r \times C$ parameters for computing $\hat{\mathbf{r}}$. As a result, the space complexity of the Vector Attention is $O\left(MN_rC\right)$ with M as the number of ROI, thus making Vector Attention more expensive than multi-head attention. However, given that the number of ROIs in the refinement stage is relatively small (100 during testing), Vector Attention is still affordable on mid-end hardware.

Following [111], the Attention Module, shown in Fig.4.3, consists of Vector Attention, residual connection, normalization layers (BatchNorm by default), and MLP.

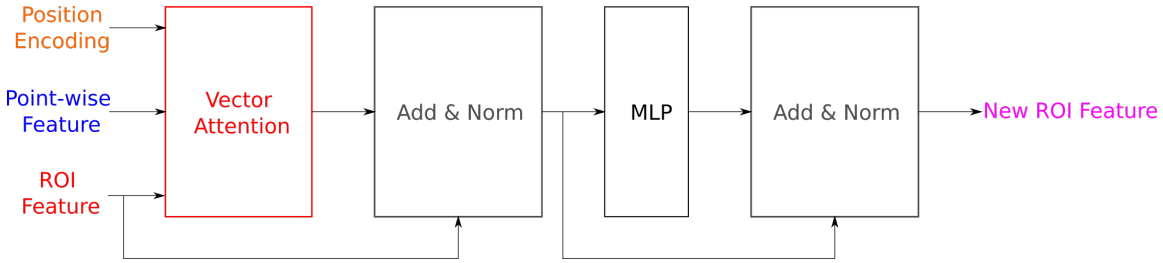


Figure 4.3 – Architecture of the Attention Module.

Position Encoding

We leverage position encoding to inject geometry information, including point location and ROI size, into the attention mechanism. A point j 's location is readily available in its coordinate ${}^r\mathbf{p}_j$ in the ROI r's canonical frame. To incorporate an ROI's size, we use the approach proposed by [99] in which a point's displacement relative to the ROI's eight vertices augments its coordinate.

$${}^r\tilde{\mathbf{p}}_j = \begin{bmatrix} {}^r\mathbf{p}_j & {}^r\mathbf{p}_{j,1} & \dots & {}^r\mathbf{p}_{j,8} \end{bmatrix} \in \mathbb{R}^{1\times 27} \tag{4.6}$$

In Eq.(4.6), ${}^r\mathbf{p}_{j,k}(k = \{1, \ldots, 8\})$ denotes the vector going from vertex k to ${}^r\mathbf{p}_j$.

Position encoding $\boldsymbol{\zeta}$ used in Eq.(4.5) is computed by

$$\boldsymbol{\zeta} = \text{MLP}\left({}^r\tilde{\mathbf{c}} - {}^r\,\tilde{\mathbf{p}}_j\right) \tag{4.7}$$

where ${}^r\tilde{\mathbf{c}}$ is the result of applying Eq.(4.6) to the ROI's center.

Handling Multi-scale Feature Maps

While prior works pool from one fixed source such as raw point clouds [99], a set of sampled points [102], or some feature maps [24, 101], we propose to pool from every feature map. Our motivation is that each feature map has a different scale thus helping detect objects of different sizes. For example, large-scale feature maps can help detect large objects such as cars thanks to the large receptive field at each location. On the other hand, their high sparsity makes pooling with small ROIs (e.g., ROI of class pedestrians or cyclists) returns a significantly low number of points or event empty, making extracting meaningful ROI features difficult.

Our pooling scheme is detailed by Alg.1. Inspired by [16], we sequentially pool feature maps from the largest to the smallest scale. Once a feature map is pooled, ROI features are computed from their associated point-wise features using Eq.(4.5). This process repeats N times with N different sets of RFEs to increase the model's depth.

4.2.3 Detection Heads and Learning Targets

An ROI's feature computed by a series of RFEs is mapped to a higher dimension space by a two-hidden layer MLP before being decoded into ROIs' confidence and refinement vector. Following [101], an ROI's confidence is set to the normalized IoU with its associated ground truth, thus making the refinement stage class-agnostic. Let $\texttt{IoU}$ denote ROI r's regular IoU, its normalized IoU is

$$c_r^* = \begin{cases} 1 & \text{if } \texttt{IoU} > \chi_H \\ 0 & \text{if } \texttt{IoU} < \chi_L \\ \frac{\texttt{IoU} - \chi_L}{\chi_H - \chi_L} & \text{otherwise} \end{cases} \tag{4.8}$$

where, χ_H and χ_L are foreground and background threshold.

Algorithm 1: Computing ROI features by pooling from multiple feature maps

Input:
$\mathcal{F}^i$ $(i = \{1, \ldots, 4\})$: feature maps generated by backbone
$\mathcal{R} = \{([x_r, y_r, z_r, dx_r, dy_r, dz_r, \theta_r], \text{cls}_r)\}_{r=1}^{M}$: set of ROI generated by RPN
Output: $\mathcal{RF} = \{\mathbf{r}_r\}$: set of ROI features
`// Initialize ROI features with model's learnable parameter` Θ
$\mathcal{RF} \leftarrow \emptyset$;
for r *in* $\{1, ..., |\mathcal{R}|\}$ **do**
  $\mathbf{r}_r \leftarrow \Theta$;
  $\mathcal{RF} \leftarrow \mathcal{RF} \cup \{\mathbf{r}_r\}$;
end
for N *times* **do**
  for i *in* $\{4, ..., 1\}$ **do**
    for r *in* $\{1, ..., |\mathcal{R}|\}$ **do**
      $\mathcal{P}_r = \{({}^r\mathbf{p}_j, \mathbf{f}_j)\} \leftarrow \texttt{Pool}\,(\mathcal{F}^i)$
      $\mathbf{r}_r \leftarrow \texttt{Attention}\,(\mathbf{r}_r, \mathcal{P}_r)$; `// Eq.`(4.5)
    end
  end
end

The target of the refinement head is the normalized residue of an ROI with respect to its associated ground truth. Given the parameters of ROI r defined by Eq.(4.1) and its associated ground truth, its normalized residue $\boldsymbol{\delta}_{\boldsymbol{r}}^* = [x^*, y^*, z^*, dx^*, dy^*, dz^*, \theta^*]$ is

$$\begin{aligned} x^* &= \tfrac{x_r^g - x_r}{d}, & y^* &= \tfrac{y_r^g - y_r}{d}, & z^* &= \tfrac{z_r^g - z_r}{dz_r} \\ dx^* &= \log \tfrac{dx_r^g}{dx_r}, & dy^* &= \log \tfrac{dy_r^g}{dy_r}, & dz^* &= \log \tfrac{dz_r^g}{dz_r} \\ \theta^* &= \theta_r^g - \theta_r \end{aligned} \tag{4.9}$$

In Eq.(4.9), the superscript g denotes the ground truth box's parameters, while the subscript r represents the ROI index. $d = \sqrt{x_r^2 + y_r^2}$ is the diagonal of the base of the ROI.

4.2.4 Loss Function

Our RFE module can be trained end-to-end with the RPN by optimizing the summation of the RPN loss, the refinement stage's loss, and an auxiliary loss.

$$\mathcal{L} = \mathcal{L}_{\text{RPN}} + \mathcal{L}_{\text{refine}} + \mathcal{L}_{\text{aux}} \tag{4.10}$$

RPN Loss

Since we adopt the backbone and the RPN of SECOND, the RPN loss is as in [129] which is the sum of classification and a regression loss

$$\mathcal{L}_{\text{RPN}} = \frac{1}{A_+} \sum_a \left[\mathcal{L}_{\text{cls}} \left(c_a, c_a^* \right) + \mathbb{1} \left(c_a^* \neq 0 \right) \mathcal{L}_{\text{reg}} \left(\delta_a, \delta_a^* \right) \right] \tag{4.11}$$

where, c_a and δ_a are the output of RPN's Class branch and Bbox branch, and A_+ is the number of positive anchors. The classification target of the RPN c_a^* is the class of ground truth box of anchor a. The regression target δ_a^* of anchor a is calculated according to Eq.(4.9). $\mathbb{1}\left(c_a^* \neq 0\right)$ is the indicator function which takes value of 1 for positive anchors whose $c_a^* \neq 0$ and 0 otherwise. The classification loss $\mathcal{L}_{\text{cls}}$ is the Focal Loss [73], while the regression loss $\mathcal{L}_{\text{reg}}$ is the Huber Loss (i.e. smooth-L1).

Refining Loss

Similar to the RPN loss, this loss makes of classification and a regression loss.

$$\mathcal{L}_{\text{refine}} = \frac{1}{M} \sum_r \left[\mathcal{L}_{\text{cls}} \left(c_r, c_r^* \right) + \mathbb{1} \left(c_r^* \geq \chi_{reg} \right) \mathcal{L}_{\text{reg}} \left(\delta_r, \delta_r^* \right) \right] \tag{4.12}$$

Here, the classification target is the normalized IoU (Eq.(4.8)), and $\mathcal{L}_{\text{refine}}$ is normalized by the total number of ROIs M. The regression threshold χ_{reg} used in Eq.(4.12) is different from the foreground threshold χ_H of Eq.(4.8).

Auxiliary Loss

Inspired by [46, 101], auxiliary supervision is applied to two backbone-generated feature maps, $\mathcal{F}^3$, and $\mathcal{F}^4$, to guide the feature extraction. To be specific, they are interpreted into point-wise features. Each point feature k is then fed into an MLP to predict its foreground probability f_k, offset toward the associated ground truth box's center o_k, and part probability p_k [101]. A point is labeled as foreground if it is inside a ground truth box. The label of the part probability of foreground points is essentially their coordinate in the canonical frame (Fig.4.2) of the associated ground truth box normalized by the box's sizes.

$$\mathcal{L}_{\text{aux}} = \frac{1}{P_+}\left[\sum_{k=1}^{P}\mathcal{L}_{\text{cls}}\left(f_k, f_k^*\right)\right] + \frac{1}{P_+}\left[\sum_{k=1}^{P_+}\mathcal{L}_{\text{reg}}\left(o_k, o_k^*\right) + \mathcal{L}_{\text{bce}}\left(p_k, p_k^*\right)\right] \tag{4.13}$$

In Eq.(4.13), $\mathcal{L}_{\text{bce}}$ is the Binary Cross Entropy (BCE) loss. The $*$ in the superscript denotes the label, while subscript k is the point index. P_+ is the number of foreground points. The regression loss and BCE loss are only calculated for foreground points.

4.3 Experiments

To demonstrate the effectiveness of our method, we evaluate it on KITTI [35] and NuScenes [8] datasets. The details of these datasets and the metric used for evaluation can be found in Sec.2.6.1. Furthermore, we carry out comprehensive ablation studies to understand the influence of each module on the overall performance.

4.3.1 Implementation Details

We build our method to work on top of SECOND (or other 3D proposal methods). We use the implementation of SECOND and other RPNs provided by the OpenPCDet [107] toolbox. To demonstrate the robustness of our choice of model's hyperparameters, we keep them constant for experiments on both KITTI and NuScenes. The three exceptions are the point cloud range, the initial voxel size, and the number of channels of the last feature map, $\mathcal{F}^4$.

RPN

Since we do not introduce any modification to SECOND, the following presents the parameters directly related to our method, while the rest can be found in OpenPCDet.

KITTI Point clouds are clipped by $[0\text{m}, 70.4\text{m}]$ in the X-axis, $[-40\text{m}, 40\text{m}]$ in the Y-axis, and $[-3\text{m}, 1\text{m}]$ in the Z-axis and voxelized with grid size of $[0.05\text{m}, 0.05\text{m}, 0.1\text{m}]$. SECOND's intermediate feature maps ($\mathcal{F}^i, i = 1, \ldots, 4$) have 16, 32, 64, and 64 channels.

Proposals are post-processed by non-max suppression with the overlapped threshold of 0.8 (0.7) to obtain 512 (100) ROIs during training (testing).

NuScenes point clouds' range $[-51.2\text{m}, 51.2\text{m}]$ along X-axis, Y-axis, and $[-5\text{m}, 3\text{m}]$ along Z-axis. The voxel size for discretizing the input point cloud is $[0.1\text{m}, 0.1\text{m}, 0.2\text{m}]$. The point cloud of a NuScenes keyframe (i.e., sample) contains about 40k points which are just one-third the size of a KITTI point cloud, thus making it highly difficult for any methods. We follow the common practice that maps point clouds in 10 previous non-keyframes to the timestamp of the keyframe using ego vehicle's odometry to increase the number of points by ten times. Regarding the 3D backbone, the number of channels in the last feature map $\mathcal{F}^4$ is 128, while the rest are similar to the KITTI configuration.

ROI Feature Encoder

128 ROIs are sampled from RPN output for the refinement stage during training. Each ROI is then enlarged by $0.5m$ along three dimensions for pooling. We empirically find that the pooling from the second feature map, $\mathcal{F}^2$, does not bring significant improvement to the final performance. Therefore, we opt for pooling 64, 128, and 256 points from $\mathcal{F}^4, \mathcal{F}^3, \mathcal{F}^1$ for each ROI.

The feature dimension is kept constant at d_a equal to 128 throughout the Attention Module. Features pooled from $\{\mathcal{F}^4, \mathcal{F}^3, \mathcal{F}^1\}$ are linearly mapped to d_a before going to the Vector Attention. The MLP of the Attention Module and Position Encoding has a 256-neuron hidden layer activated ReLU function. The sequential pooling from $\mathcal{F}^4$ to $\mathcal{F}^1$ for computing ROIs' feature is repeated 3 times with three different sets of RFEs.

Training

The entire architecture presented in Fig.4.1 is optimized end-to-end by the Adam optimizer. For KITTI, we train the model for 100 epochs with a total batch size of 24. The learning rate is controlled by the One-Cycle policy [104] with a maximum value of 0.01. For NuScenes, the training lasts 20 epochs while the same batch size and the learning rate policy remain unchanged. The maximum learning rate reduces to 0.03. In the detection head, the foreground threshold χ_H, background threshold χ_L, and regression IoU threshold χ_{reg} are $0.75, 0.25$ and 0.55. We use the same data augmentation strategy as [101, 102, 129].

4.3.2 Results On KITTI Dataset

The comparison of our method against the state-of-the-art on KITTI *test* set is presented in Tab.4.1. Among methods built on top SECOND, namely [24, 46, 99, 102], we achieve the best performance in class Cyclist and competitive performance in class Car, passing the 80 AP threshold while having the least number of parameters. Note that we train a single model for two classes instead of separate models as previously done by [59, 100, 129].

Table 4.1 – Performance comparison on KITTI *test* set with AP calculated with 40 recall positions

Method	Num Parameters (M)	Car - 3D Detection			Cyclist - 3D Detection		
		Easy	Mod.	Hard	Easy	Mod.	Hard
SECOND [129]	20	83.34	72.55	65.82	71.33	52.08	45.83
PointPillar [59]	18	82.58	74.31	68.99	77.10	58.65	51.92
PointRCNN [100]	16	86.96	75.64	70.70	74.96	58.82	52.53
SA-SSD [46]	226	88.75	79.79	74.16	-	-	-
Part A^2 [101]	40.8	87.81	78.49	73.51	-	-	-
PV-RCNN [102]	50	90.25	81.43	76.82	**78.60**	63.71	57.65
Voxel R-CNN [24]	28	**90.90**	81.62	77.06	-	-	-
CT3D [99]	30	87.83	**81.77**	**77.16**	-	-	-
APRO3D-Net (ours)	22.4	87.09	80.30	76.10	78.54	**64.55**	**57.78**

Our performance on KITTI *val* set with AP calculated at 40 recall positions is also reported in Tab.4.2, indicating that the gap between our method and top performers in class Car is shortened, with the highest difference being just 0.45 AP. Compared to CT3D which also computes ROIs' feature using the attention mechanism, we surpass their AP for class Pedestrian and Cyclist by 1.42 and 1.47.

Table 4.2 – Performance comparison on KITTI *val* set with AP calculated at 40 recall positions

Method	AP_{3D} - Moderate		
	Car	Cyclist	Pedestrian
PV-RCNN [78]	84.83	71.95	56.67
Voxel R-CNN [24]	**85.29**	-	-
Voxel R-CNN [3]	84.95	71.43	**58.24**
CT3D [99]	84.97	71.88	55.58
APRO3D-Net (ours)	84.85	**73.35**	57.41

4.3.3 Results On NuScenes Dataset

The performance on NuScenes *validation* set is shown in Table 4.3. In this table, SECOND and PointPillars are single-stage methods, while others, including ours, are two-stage. In this more challenging method, the benefit of integrating our RFE to SECOND is more prominent, indicated by almost 20 mAP improvements. In addition, ours outperforms 3DSSD [134] and InfoFocus [114], two recent two-stage methods, by a large margin except for class Car and Truck. This competitive performance on the NuScenes dataset shows our method's ability to handle object classes with high variance in scales.

Table 4.3 – AP on NuScenes dataset

Method	Car	Ped	Bus	Barrier	Traf. Cone	Truck	Trailer	Motor	Cons. Veh.	Bicycle	mAP
SECOND [129]	75.53	59.86	29.04	32.21	22.49	21.88	12.96	16.89	0.36	0	27.12
PointPillars [59]	70.5	59.9	34.4	33.2	29.6	25.0	20.0	16.7	4.5	1.6	29.5
3DSSD [134]	**81.20**	70.17	61.41	47.94	31.06	**47.15**	30.45	35.96	12.64	8.63	42.66
InfoFocus [114]	77.6	61.7	50.5	43.4	33.4	35.4	25.6	25.2	8.3	2.5	36.4
APRO3D-Net (ours)	77.75	**74.02**	**64.86**	**52.61**	**46.34**	43.99	**34.9**	**39.36**	**13.44**	**23.00**	**47.03**

4.3.4 Qualitative Performance

To show that different channels within the same point feature contribute differently to an ROI feature, we visualize pooled points' attention weight (the term on the left of $\odot$ in Eq.(4.5)) in Fig.4.4. As can be seen, the region of ROI, where attention is concentrated, varies across channels. For example, Box 1 respectively pays the most attention to its front and rear to compute two different channels of its feature.

In addition, a visual evaluation of our method's performance on the *test* split of KITTI and NuScenes dataset made by projecting predictions onto images, as in Fig.4.5, shows that our refinement module can avoid false positive and improve object localization.

4.3.5 Ablation studies

We perform extensive ablation studies to validate our design choices and understand the impact of each module on the overall performance. All models used in this section are trained on the KITTI *train* set and evaluated on the KITTI *val* set. Unless stated otherwise, evaluations are based on AP calculated at 40 recall precision.

3. Performance of model trained for 3 classes, reproduced from official code release

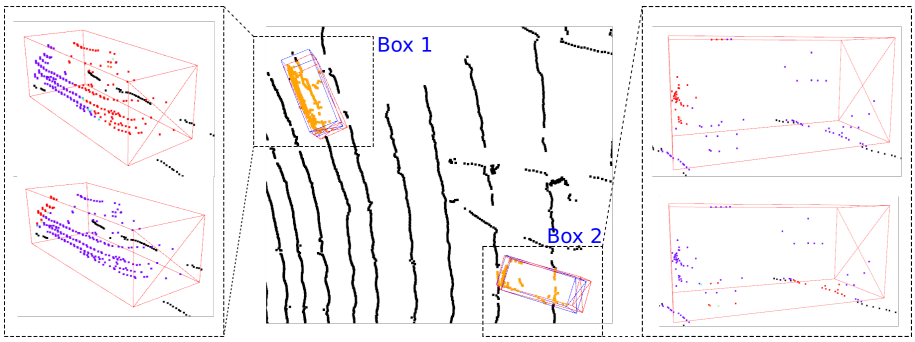


Figure 4.4 – Visualization of attention weights. Predictions and their associated ground truth boxes are respectively marked by red and blue. Orange denotes pooled points. In two zoomed windows, points are color-coded according to their attention weights. The hotter the color, the higher the attention weight.

First, we verify our motivation for choosing Vector Attention over multi-head attention by changing the attention formula in Eq.(4.5) while keeping the rest of the architecture unchanged. The result shown in Tab.4.4 confirms the superiority of Vector Attention with significant AP difference for class Car and Cyclist. Such difference is due to Vector Attention enabling the model to choose where to look (which points) and what to look for (which channels) when computing ROI features, as illustrated in Fig.4.4. On the other hand, multi-head attention can only choose where to look because of its scalar weight.

Table 4.4 – Performance on KITTI *val* set of Multi-Head Attention compared to Vector Attention

Method	AP_{3D} - Moderate		
	Car	Cyclist	Pedestrian
Multi-head Attention	82.50	70.35	57.58
Vector Attention	84.85	73.35	57.41
Improvement	**2.35**	**3.00**	-0.17

The second experiment is to analyze the impact of position encoding on the overall performance. The first row of Tab.4.5 shows the result of the model that does not use position encoding, meaning setting $\boldsymbol{\zeta}$ in Eq.(4.5) to 0. The second row is the performance of building position encoding from the point displacement relative to the ROI center only. In other words, ${}^{r}\mathbf{p}_{j,1}, \ldots, {}^{r}\mathbf{p}_{j,8}$ are removed from Eq.(4.6). Tab.4.5 shows that using position encoding increases performance for class Car, Cyclist, and Pedestrian to 8.19, 6.24, 4.03 AP. Moreover, position encoding contains the most information about ROI geometry (third row) and performs the best overall, especially for the most important

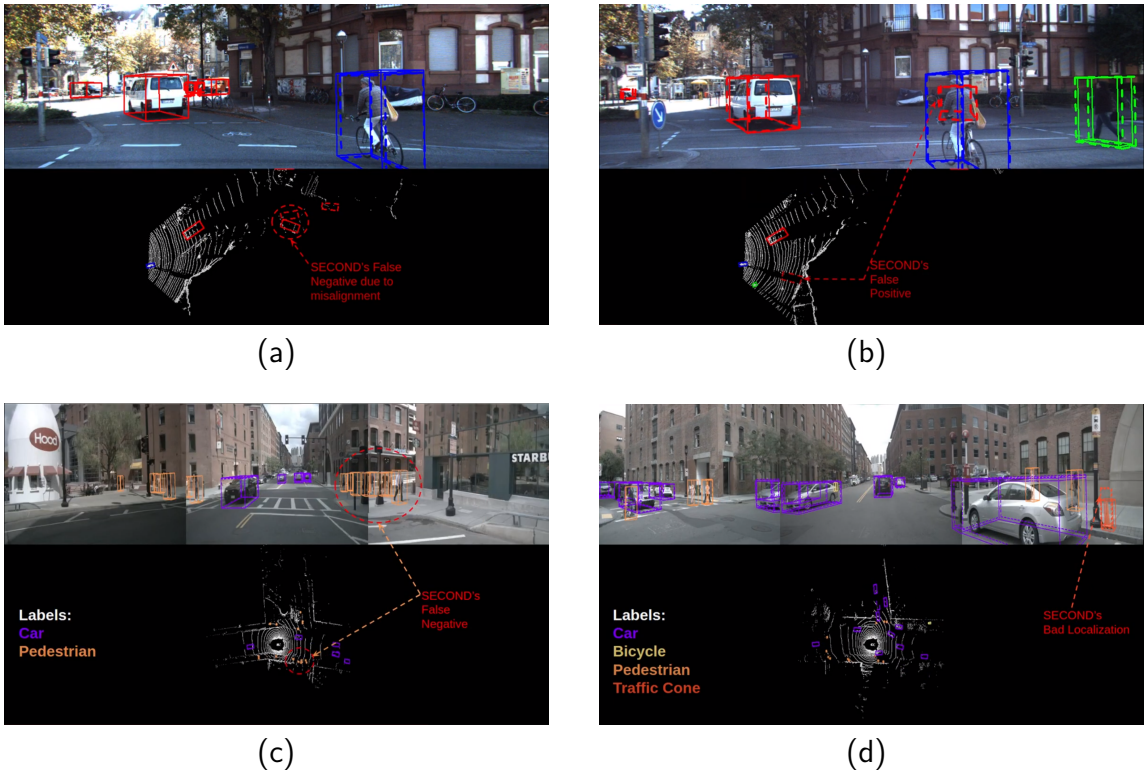


Figure 4.5 – Qualitative performance of APRO3D on KITTI (top row) and NuScenes (bottom row). Objects detected by SECOND and APRO3D are respectively represented by dashed and continuous cuboids.

class Car.

Table 4.5 – Performance on KITTI *val* set of different position encoding methods

Method	AP_{3D} - Moderate		
	Car	Cyclist	Pedestrian
None	76.66	69.65	53.38
Center	82.72	**75.89**	55.74
Center and Vertices	**84.85**	73.35	**57.41**

Next, three pooling strategies are compared. The performance shown in the first row of Tab.4.6 is obtained by equally pooling M points from each feature map $\mathbf{F}^i$ then concatenating their features before passing them to RFEs for ROIs feature computation. In the other rows, we sequentially pool from $\mathbf{F}^4$ to $\mathbf{F}^1$ while skipping $\mathbf{F}^2$. The difference between the second and third rows is the sequential pooling process takes place only once in the second row while it repeats three times in the third. In other words, N of Alg.1 is set to

1 and 3 in the row second and third, respectively. Even though pooling all at once (first row) does not show any significant performance drop for class Car while achieving the best performance in class Pedestrian, this pooling strategy is the most memory-intensive. The number of points to be processed, which linearly grows with the number of feature maps and ROIs, can quickly overflow GPUs' memory. Another aspect to be noticed is repeating the pooling process (with a different set of RFE) only brings marginal performance gain.

Table 4.6 – Performance on KITTI *val* set of different pooling methods

Method	AP_{3D} - Moderate		
	Car	Cyclist	Pedestrian
All at once	84.15	71.04	**57.55**
Sequential without repetition	84.66	**75.34**	56.15
Sequential with repetition	**84.85**	73.35	57.41

Finally, the versatility of our method is demonstrated by its integration with different RPNs: SECOND, PartA^2, PointRCNN. Since PartA^2 has a UNet-like backbone, we pool from feature maps produced by its up-sampling branch while keeping the rest of the architecture unchanged. In the case of PointRCNN, we pool from the final output of its backbone. Note that when not using the RFE, PointRCNN, and PartA^2 use their refinement stage. Tab.4.7 shows the performance gain at different difficulty levels of class Car, thus confirming the effectiveness of our method. The limited gain when integrating with PointRCNN can be explained by its lower recall rate compared to SECOND. This experiment validates our design choice regarding the RPN method.

Table 4.7 – Performance gain on KITTI *val* set brought by our RFE to different RPNs

Method	AP_{3D} (Car)		
	Easy	Moderate	Hard
SECOND	88.61	78.62	77.22
SECOND + RFE	**+0.75**	**+4.89**	**+1.56**
PartA^2	89.47	79.47	78.54
PartA^2 + RFE	-0.08	**+3.16**	**+0.36**
PointRCNN	88.88	78.63	77.38
PointRCNN + RFE	**+0.06**	**+0.38**	**+1.02**

4.4 Conclusion

As one-stage object detection models are designed to have a theoretical recall rate of 100%, thus translating to a high recall rate in practice, their precision is hurt by the presence of a large number of false positives. These false positives can deteriorate downstream modules such as object tracking or collaborative perception based on *MoDAR* as developed in the previous chapter. Therefore, the re-evaluation and re-adjustment of prediction made by one-stage detectors is necessary. This chapter addresses this need with an ROI refinement module that transforms features of points residing inside each ROI into its features using Vector Attention.

The rationale behind the design of this module is that the absence of an inductive bias for the computation of ROI features given points features requires the layer employed for this purpose to be as generic as possible. The Transformers [111] satisfies this condition; however, it is still restricted because it imposes the same scalar weight across different feature channels of the same point. This is not ideal as the prediction of different attributes can be influenced differently by different feature channels of each point. For example, ground points have little to do with the prediction of an object's center, but are instrumental to the prediction of its height. A scalar weight that suppresses a ground point's influence on the center prediction will also suppress the valuable information this point has to offer to the prediction of an object's height. Based on this reasoning, we opt for Vector Attention which offers a higher capacity for modeling the interaction among points and ROI features by enabling a vector weight for each point Such capacity comes at the expense of a higher memory complexity. However, this tradeoff is feasible due to the manageable number of ROI thanks to the preprocessing using Non-max Suppression.

An innovation of this chapter is that point features, which are inputted to Vector Attention, are obtained from intermediate feature volumes produced by the 3D backbone instead of being computed from raw point clouds, thus saving both inference time and training time. Moreover, the pooling of point features to compute ROI features takes place in multiple feature volumes. Since each of these has a different resolution, meaning a different level of occupancy, this pooling scheme ensures that ROIs of different sizes can pool a sufficient amount of point features. Experiments on KITTI and NuScenes datasets validate the effectiveness of our method.

The refinement stage presented in this chapter completes the set of solutions to 3D object detection that gradually boosts detection models using

1. Point cloud sequences instead of individual point clouds
2. V2X collaborative perception
3. A transformer-based refinement stage

Chapter 5

Conclusions and perspectives

5.1 Conclusion

The safety of autonomous vehicles critically relies on their ability to detect objects in 3D. While which sensing modality is sufficient to achieve this goal is still an open question, it is undeniable that LiDAR-based methods dominate every public benchmark. Their success is because LiDARs provide accurate depth measurement, which is unavailable when using RGB cameras, at a high density, which cannot be matched by RADARs. However, LiDAR measurements - point clouds are severely affected by occlusion and sparsity at long-range due to the operation principle of this sensor that relies on bounding laser beam off objects' surface.

The impact of these two issues on the safety of autonomous vehicles is highlighted in a safety report published by Waymo stating that 2 out 8 accidents involving their autonomous vehicles are due to occlusion at intersections [98]. As a result, a significant research effort has been devoted to finding their solutions. The common numerator of these prior works is to densify point clouds either in 3D space or in the latent space of the BEV representation. The former upsample the sparse depth map made by LiDAR by leveraging the fusion with RGB images [122, 138]. On the other hand, the latter transfers the dense BEV representation of a teacher model, which was pre-trained on point clouds densified using knowledge of object shape (available during training), to a student model that processes raw point clouds [30, 115]. While achieving competitive results on public benchmarks, these methods have not addressed the root cause of occlusion and sparsity which is the lack of the LiDAR points in some regions.

This thesis fills the above gap in the literature with a framework that leverages the availability of point clouds acquired from multiple perspectives to the fullest. The first component of this framework is an off-the-shelf single-vehicle detention model enhanced by the utilization of point cloud sequences as input, instead of individual point clouds. While a straightforward concatenation of point clouds directly leads to improved precision in the

detection of stationary and slow-moving objects, such improvement does not transfer to medium-to-fast-moving objects because their representations in the BEV are misaligned with their location (i.e., their bounding boxes). This representation misalignment is caused by the *shadow effect* in the concatenated point cloud which manifests as points of a moving object scattered along its trajectory. To extend the benefit of point cloud sequences, **Chapter 2** proposes a plug-in module, named *Aligner*, that estimates the scene flow and uses this to rectify such misalignment. The result is an improved precision over models using solely the concatenation of point cloud sequences. More importantly, the availability of scene flow allows computing the velocity of detected objects by averaging the scene flow of points that they contain. This velocity estimation is crucial for the second component of our framework.

The second component of our framework targets the scenarios where the field of view of the ego vehicle is heavily obstructed for an extended amount of time such that point cloud sequences cannot offer more information compared to individual point clouds. An example of such a scenario is when the ego vehicle waits at an intersection and is surrounded by a large number of other road users. In these scenarios, any strengths of single-vehicle models are effectively negated because most objects are unobservable and thus undetectable. Our framework addresses this challenge following the methodology of the emerging collaborative perception via V2X communication [66, 116, 126, 128]. In the presence of other connected agents, including other connected autonomous vehicles and intelligent roadside units, the perception capacity of the ego vehicle is enhanced by exchanging information with others. **Chapter 3** has pointed out the shortcomings of the current state-of-the-art collaborative perception, which centers on exchanging BEV representation among connected agents, including

— complicated architecture changes to accommodate the fusion of BEV representation

— strong assumption about inter-agent synchronization

— being unable to support a network of heterogeneous detection models

and derives a more practical method which we called *lately* fusion. This method is a combination of the *late* fusion method and *early* fusion method to strike a better performance-bandwidth tradeoff compared to prior works. Its operation is made of the exchange of connected agents' output, which is the signature of the *late* fusion, and the fusion of exchanged information to the raw point cloud of the ego vehicle, which is similar to the principal of the *early* fusion. The availability of detected objects' velocity thanks to the *Aligner* enables translating objects detected at a past timestep to the present. This dis-

misses the need for inter-agent synchronization as each agent can store the most recent set of detected objects in a database which the ego vehicle can query and compensate for the time difference using the velocity of detected objects. The *lately* fusion achieves a performance comparable to the *early* fusion, which is regarded as the performance ceiling of collaborative perception while consuming as little bandwidth as the *late* fusion, which is orders of magnitude smaller than the bandwidth needed for exchanging BEV representation.

5.2 Perspective

5.2.1 Robustness to Localization Error

An important factor in the function of our *lately* fusion framework is the precise localization of each connected agent. This is required to accurately transform an agent's output to the local frame of the ego vehicle. The positional and heading error of standard localization approaches [19, 69, 123] can be modeled by Gaussian distributions with a standard deviation of 0.2 meters and 0.2 degrees, respectively. Xu et al [128] have shown that this level of localization noise severely reduces (up to 50%) the precision of all collaboration frameworks including *early*, *late*, and *mid* collaborations. A study on the robustness of our method *lately* collaboration approach is required to strengthen its practicality.

To the best of our knowledge, [128] is the only work that addresses the impact of connected vehicles' localization errors. Their solution is based on data augmentation which randomly perturbs connected vehicles' poses with random variables sampled from Gaussian distributions. While being effective in preventing a sharp performance drop, their method remains unaware of other agents' localization uncertainty.

The flexibility of using *MoDAR* to communicate detection results among connected agents in our *lately* collaboration framework enables explicit incorporation of connected agents' localization uncertainty to their detection results. In detail, let Σ_i denote the covariance matrix of the pose $[x_i, y_i, z_i, \theta_i]$ in the common frame $\mathcal{W}$ of agent $\mathcal{A}_i$ and an object detected by this agent be represented by $\mathbf{b}_{i,j} = [x_{i,j}, y_{i,j}, z_{i,j}, w_{i,j}, l_{i,j}, h_{i,j}, \theta_{i,j}, s_{i,j}, c_{i,j}]$. Here, $[x_{i,j}, y_{i,j}, z_{i,j}, \theta_{i,j}]$ defines the pose of the object in the body frame of $\mathcal{A}_i$, $[w, l, h]$ is the object's size, s is the confident score, and c is the object's class. This object is

transformed into the common frame $\mathcal{W}$ as the following

$$
{}^{\mathcal{W}}\begin{bmatrix} x \\ y \\ z \\ \theta \end{bmatrix} = \begin{bmatrix} x_{i,j}\cos\theta_i - y_{i,j}\sin\theta_i + x_i \\ x_{i,j}\sin\theta_i + y_{i,j}\cos\theta_i + y_i \\ z_{i,j} + z_i \\ \theta_{i,j} + \theta_i \end{bmatrix} \tag{5.1}
$$

Let f represent the left-hand side of the equation above. The covariance matrix $\Sigma_{i,j}$ of the pose of $\mathbf{b}_{i,j}$ in the common frame $\mathcal{W}$ is

$$
\Sigma_{i,j} = \frac{\delta\ f}{\delta\ {}^{\mathcal{W}}\mathbf{q}_i}\ \Sigma_i\ \left(\frac{\delta\ f}{\delta\ {}^{\mathcal{W}}\mathbf{q}_i}\right)^T \tag{5.2}
$$

with ${}^{\mathcal{W}}\mathbf{q}_i = [x_i, y_i, z_i, \theta_i]^T$. The value of $\Sigma_{i,j}$ computed from Σ_i using the equation above can be flattened and concatenated to the feature of the *MoDAR* point that represents $\mathbf{b}_{i,j}$, to inform the detection model of the localization uncertainty of agent $\mathcal{A}_i$. How this explicit indication of connected agents' localization uncertainty improves the robustness of the collaborative perception performance, especially in comparison with data augmentation as in [128] is an interesting research prospect.

5.2.2 Auto Label V2X Dataset using Object Discovery

To demonstrate the capacity of our *lately* collaboration framework in a real-world setting, we set out to create our own V2X setup including one intelligent roadside unit and two connected autonomous vehicles. Because detection models in our framework are trained in a fully supervised manner, an annotated dataset is required. While the V2X hardware is accessible thanks to the maturity of the LiDAR technology, manually annotating thousands of point clouds is infeasible without a large budget. The recent development in unsupervised object detection called object discovery [139, 144] offers an interesting alternative to manual annotation. In the object discovery framework, an initial set of labels (i.e., 3D bounding boxes) $\mathcal{S}_0$ is created using clustering methods (e.g., DBSCAN or HDBSCAN) coupled with a set of heuristics. For example, [139] builds $\mathcal{S}_0$ by first assigning a dynamic score to each LiDAR point. The higher the dynamic score of a point, the higher its probability of belonging to a dynamic object. Then, each point cloud is clustered based on points' 3D coordinates and dynamic scores. Next, an upright bounding box is fit to each cluster using the algorithm developed in [132]. Finally,

the initial set of labels $\mathcal{S}_0$ is created by filtering improbable boxes according to a list of heuristics. Once $\mathcal{S}_0$ is obtained, a self-training process, Alg.2, is started to progressively increase the performance of detection models.

Algorithm 2: Self-training

Input:
Point clouds in the dataset $\{\mathcal{P}_i\}_{i=1}^{M}$
The initial set of labels $\mathcal{S}_0$
Number of self-training round N
Output: Trained detector D_N
$\mathcal{S} \leftarrow \mathcal{S}_0$;
for i *in* $\{1, ..., N\}$ **do**
  $\mathrm{D}_i \leftarrow$ randomly_initialize_weights()
  $\mathrm{D}_i \leftarrow$ train_detector $\left(\mathrm{D}_i, \{\mathcal{P}_i\}_{i=1}^{M}, \mathcal{S}\right)$
  $\mathcal{S} \leftarrow$ make_prediction $\left(\mathrm{D}_i, \{\mathcal{P}_i\}_{i=1}^{M}\right)$ `// inference mode`
end

A critical component of existing object discovery in point clouds is the unsupervised clustering algorithm, which is usually DBSCAN or its variant HDBSCAN. Due to its density-based nature, this algorithm requires a sufficiently high-fidelity measurement of an object to successfully group all its points into a single cluster. Such a requirement can only be satisfied at close ranges. For example, [144] limits its region of discovery to a square of size 64 meters centered at the 64-beam LiDAR of the ONCE dataset [83].

When adopting the above object discovery algorithms to the V2X context, we leverage the observation that a region at a long range with respect to one agent is at a close range with respect to another. Therefore, the early-fusion point clouds (i.e., the concatenation of point clouds obtained by every connected agent) offer a larger high-fidelity measurement window. Our object discovery algorithm, illustrated in Fig.5.1, is as follows

1. Concatenate point clouds obtained by every connected agent in the frame of the IRSU
2. Filter LiDAR points outside the discovery window and on the ground
3. Filter LiDAR points that do not belong to vehicle classes by projecting them to IRSU's images segmented by a semantic segmentation model
4. Fit an up-right bounding box to each cluster using the procedure proposed by [145]
5. Filter bounding boxes having excessive dimensions

6. Refine surviving small bounding boxes by anchoring each box at the corner that is closest to the LiDAR and expanding its width and height to match with precompute statistic of vehicle dimension

We apply the above procedure and the self-training algorithm Alg.2 to train one detector for processing point clouds of the IRSU and one for processing point clouds of Cars in the V2X-Sim dataset. In our experiments, the discovery window is set to a 64-meter-square square centered at the LiDAR.

Algorithm 3: Obtaining a long-range detector for the IRSU

Input:
Point clouds in the dataset $\{\mathcal{P}_i\}_{i=1}^{M}$
Number of self-training round N
short-range for initial object discovery r_0
targeted detection range r_1
Output: Trained long-range detector for the IRSU $\mathrm{D}_N^{\mathrm{I},r_1}$
`// for the IRSU`
$\mathcal{S}_0^{\mathrm{I}} \leftarrow$ discover_objects_in_irsu_frame $\left(\{\mathcal{P}_i\}_{i=1}^{M}, r_0\right)$
$\mathrm{D}_N^{\mathrm{I}} \leftarrow$ self_training $\left(\{\mathcal{P}_i\}_{i=1}^{M}, \mathcal{S}_0^{\mathrm{I}}, N\right)$
`// for Cars`
$\mathcal{S}_0^{\mathrm{C}} \leftarrow$ discover_objects_in_car_frame $\left(\{\mathcal{P}_i\}_{i=1}^{M}, r_0\right)$
$\mathrm{D}_N^{\mathrm{C}} \leftarrow$ self_training $\left(\{\mathcal{P}_i\}_{i=1}^{M}, \mathcal{S}_0^{\mathrm{C}}, N\right)$
`// to extend IRSU detection range from` r_0 `to` r_1
$\mathcal{S}_0^{\mathrm{I+C},r_1} \leftarrow$
for i *in* $\{1, ..., M\}$ **do**
  `// all detectors are in inference mode`
  $\mathcal{B}^{\mathrm{I},r_0} \leftarrow$ make_prediction $\left(\mathrm{D}_N^{\mathrm{I}}, \mathcal{P}_i, r_0\right)$ `//` $\mathcal{P}_i$ `limits in the range` r_0
  $\mathcal{B}^{\mathrm{C},r_0} \leftarrow$ make_prediction $\left(\mathrm{D}_N^{\mathrm{C}}, \mathcal{P}_i, r_0\right)$
  transform $\mathcal{B}^{\mathrm{C},r_0}$ to IRSU frame
  $\mathcal{B}^{\mathrm{I},r_1} \leftarrow$ non_max_suppression $\left(\mathcal{B}^{\mathrm{I},r_0}, \mathcal{B}^{\mathrm{C},r_0}\right)$
  $\mathcal{S}_0^{\mathrm{I+C},r_1} \leftarrow \mathcal{S}_0^{\mathrm{I},r_1} \cup \mathcal{B}^{\mathrm{I},r_1}$
end
$\mathrm{D}_N^{\mathrm{I},r_1} \leftarrow$ self_training $\left(\{\mathcal{P}_i\}_{i=1}^{M}, \mathcal{S}_0^{\mathrm{I+C},r_1}, N\right)$

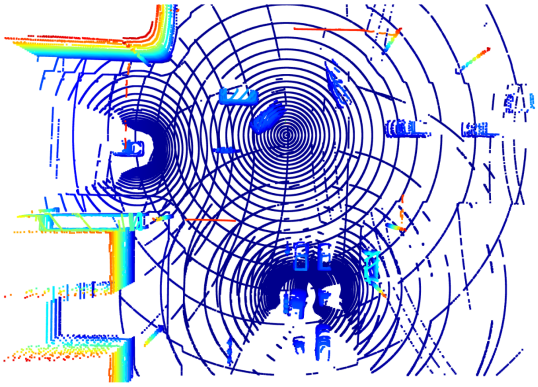
To extend the detection range beyond the discovery window, we use trained short-range detectors to generate detection for both Cars and IRSU in every sample of the V2X-Sim dataset. Notice that each sample of this dataset contains point clouds collected by a number of connected agents including 1 IRSU and at least 1 connected autonomous vehicles. Then, the detection made by all agents are transformed to the reference frame of the IRSU and merged using the Non-max Suppression procedure. The complete procedure for obtaining a long-range detector for the IRSU is presented in Alg.3

The result of experiments in the V2X-Sim dataset is shown in Tab.5.1. In these experiments, the architecture of every detector is PointPillar. Oracle models are trained with manually created bounding boxes instead of discovered bounding boxes. These experiments show that the model trained by labels from discovered objects and self-training can reach up to 94% performance of the oracle model when a concatenation of point cloud sequences obtained by the IRSU only is used as the input instead of the early fusion point cloud.

Table 5.1 – Performance of Unsupervised Object Detection by Object Discovery and Self-training

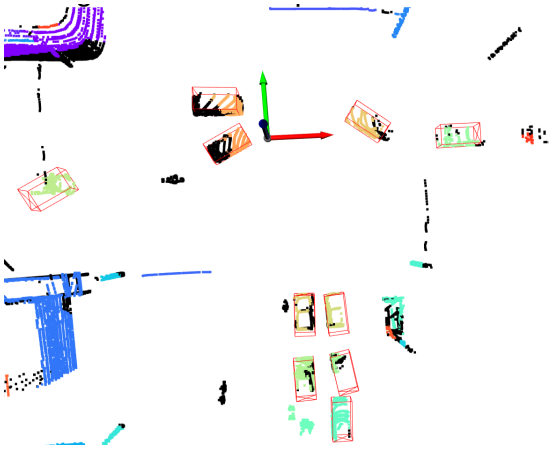
Detector	$r_0 = 32$m	$r_1 = 51.2$m	Early Fusion Point Cloud	mAP train	mAP val
$\mathcal{S}_0^{\mathrm{I}}$	✓		✓	43.42	
$\mathrm{D}_1^{\mathrm{I}}$	✓		✓		72.97
$\mathrm{Oracle}^{\mathrm{I}}$	✓		✓		87.0
$\mathcal{S}_0^{\mathrm{C}}$	✓		✓	22.52	
$\mathrm{D}_1^{\mathrm{C}}$	✓		✓		62.77
$\mathcal{S}_0^{\mathrm{I}}$		✓	✓	28.65	
$\mathcal{S}_0^{\mathrm{I+C}}$		✓	✓	36.54	
$\mathrm{D}_1^{\mathrm{I}}$		✓	✓		48.31
		✓			50.18
$\mathrm{Oracle}^{\mathrm{I}}$		✓	✓		60.19
		✓			52.94

During this preliminary study, we found that our auto-labeling method is heavily affected by false positives due to the imperfect initial labels as well as the false positive prediction used as labels during self-training. We hypothesize that false positives enable a



(a) Get early fusion point cloud

(b) Remove ground then fit an up-right bounding box to each cluster. Each cluster is represented by a unique color



(c) Refine boxes' size and heading

Figure 5.1 – Object discovery procedure

predecessor to pass on its mistake to its successor during the self-training. If a type of false positive occurs frequently enough (e.g., poles are falsely detected as pedestrians) in a label set, the model trained using this label set will make the same false positive detection with a high confidence score. As a result, these false positives survive the confidence threshold and are present in the label set of the next self-training round. The impact of false positives can be minimized by applying a high threshold on the confidence score of predictions used as labels at the early stage of self-training. However, high confidence prediction usually amounts to easy cases such as nearby unoccluded objects, a too-high confidence threshold can limit the final model to detect easy objects only. A study on how to minimize the presence of false positives in the labels set while reducing the model's ability to self-improve through self-training is necessary for the adaptation of auto-labeling. Moreover, it is interesting to see how collaborative detection models trained on auto-label data perform in comparison with those trained on manually labeled data.

List of publications

1. M.-Q. Dao, J. S. Berrio, V. Frémont, M. Shan, E. Héry, and S. Worrall, "Practical collaborative perception: A framework for asynchronous and multi-agent 3d object detection," arXiv preprint arXiv:2307.01462, 2023
2. M.-Q. Dao, V. Frémont, and E. Héry, "Aligning bird-eye view representation of point cloud sequences using scene flow," in 2023 IEEE Intelligent Vehicles Symposium (IV). IEEE, 2023.
3. M.-Q. Dao, E. Héry, and V. Frémont, "Attention-based proposals refinement for 3d object detection," in 2022 IEEE Intelligent Vehicles Symposium (IV). IEEE, 2022, pp. 197–205.
4. H. Sun, M.-Q. Dao, and V. Fremont, "3d-flownet: Event-based optical flow estimation with 3d representation," in 2022 IEEE Intelligent Vehicles Symposium (IV). IEEE, 2022, pp. 1845–1850.
5. M.-Q. Dao and V. Frémont, "A two-stage data association approach for 3d multi-object tracking," Sensors, vol. 21, no. 9, p. 2894, 2021.

Appendix A1

A Two-Stage Data Association Approach to 3D Multi-Object Tracking

The main chapters of this thesis have developed two solutions to the challenges posed by low-fidelity measurements of LiDARs to object detection models including (i) using point cloud sequences instead of individual point clouds as the input and (ii) collaborative perception via V2X. This set of solutions enables the ego vehicle to detect objects that are occluded and at long range. As the detection task is well addressed, the next step in the AV perception stack is to track objects. The purpose of this step is the establishment of objects' historical trajectories which is essential for the prediction of their future location.

The capacity of the detection developed in prior chapters enables tackling the tracking task using the track-by-detection paradigm which formulates the tracking as a data association problem. The core idea of this paradigm is that at each time step a set of trajectories, which are also referred to as tracklets, or trajectories, that was established in the previous time steps is grown by finding the correspondent of each tracklet in the set of detection made at the current time step. Due to the coupling of the tracking performance with the detection performance, a state-of-the-art detector usually produces state-of-the-art tracking accuracy even when being integrated with a rather straightforward data association algorithm. As a result, most 3D Multi-Object Tracking (MOT) comprises a strong detector and a simple Linear Assignment Problem (LAP) serving as the data association module.

This chapter contributes to the literature on 3D MOT by adopting a two-stage data association method which was successfully applied to image-based tracking in the 3D setting. Our method outperforms the baseline using one-stage bipartite matching for data association by achieving 0.587 Average Multi-Object Tracking Accuracy (AMOTA) on the validation set of the NuScenes dataset and 0.365 AMOTA (at level 2) on the test set

of the Waymo dataset. The code is released at `https://github.com/quan-dao/track_with_confidence`

A1.1 Introduction

Multi-object tracking has been a long-standing problem in the computer vision and robotics community since it is a crucial part of any autonomous system. From the early work of tracking with hand-craft features, the revolution of deep learning which results in highly accurate object detection models [75, 94, 96] has shifted the focus of the field to the track-by-detection paradigm [7, 97]. In the framework of this paradigm, tracking algorithms receive a set of object detection, usually in the form of bounding boxes, at each time step and they aim to link detection of the same object across time to form trajectories.

While image-based methods of this paradigm have reached a certain maturity, 3D tracking is still in its early phase where most of the published approaches originated from successful 2D exemplars. One popular method is [119] which extends [7] into 3D space. In these works, detections are linked to tracks by solving a bipartite matching with the Hungarian algorithm [57], then states of tracks are updated by a Kalman filter. Taking a similar approach to establish a detection-to-track correspondence, [71] trains a network for calculating the matching cost instead of using the 3D IoU. In [81, 136], objects' poses in the current and a number of future frames are predicted by deep neural networks. Thus, tracks can be formed by greedy closest-point matching.

Even though 3D tracking has progressed rapidly thanks to the availability of standardized large-scale benchmarks such as KITTI [38], NuScenes [8], Waymo Open Dataset [106], the focus of the field is placed on developing better object detection models rather than developing better tracking algorithm as evidenced in the Table.A1.1 which presents the performance measured by the AMOTA metric of tracking algorithms following the track-by-detection paradigm and the performance of their object detector measured by mAP.

AMOTA is a scalar value representing how well the algorithm does in limiting:

- ID switches (IDS): the number of times tracks are associated with wrong detections;
- False Positives (FP): the number of times real objects are missed detected;
- False Negatives (FN): the number of times the tracking algorithm reports tracks in places where there are no real objects present.

Table A1.1 – Summary of tracking methods which details are published in the leader board of NuScenes and Waymo Open Dataset

Dataset	Method Name	Tracking Method	AMOTA	Object Detector	mAP
NuScenes	CenterPoint [136]	Greedy closest-point matching	0.650	CenterPoint	0.603
	PMBM	Poisson Multi-Bernoulli Mixture filter [33]	0.626	CenterPoint	0.603
	StanfordIPRL-TRI [18]	Hungarian algorithm with Mahalanobis distance as cost function and Kalman Filter	0.550	MEGVII [**zhu2019megvii**]	0.519
	AB3DMOT [119]	Hungarian algorithm with 3D IoU as cost function and Kalman Filter	0.151	MEGVII	0.519
	CenterTrack	Greedy closest-point mathcing	0.108	CenterNet [149]	0.388
Waymo	HorizonMOT [26]	3-stage data associate, each stage is an assignment problem solved by Hungarian algorithm	0.6345	AFDet [34]	0.7711
	CenterPoint	Greedy closest-point matching	0.5867	CenterPoint	0.7193
	PV-RCNN-KF	Hungarian algorithm and Kalman Filter	0.5553	PV-RCNN [102]	0.7152
	PPBA AB3DMOT	Hungarian algorithm with 3D IoU as cost function and Kalman Filter	0.2914	PointPillars and PPBA[17]	0.3530

There are two trends that can be observed in this table. First, tracking performance experiences a boost when a better object detection model is introduced. Second, the method of AB3DMOT [119] which uses the Hungarian algorithm on some metrics (e.g. 3D IoU, Mahalanobis distance) to perform data association, Kalman Filters to update tracks' states once they have associated detections and set of heuristic rules to manage birth and death of tracks, is favored by most recent 3D tracking systems.

The reason for AB3DMOT's popularity is that it is simple yet achieves competitive results in challenging datasets at a significantly high frame rate (more than 200 FPS on KITTI). However, its simplicity comes at the cost of the MOT system being vulnerable to false associations due to occlusion or imperfect detections which is the case for objects in a clutter or far away from the ego vehicle.

Aware of the shortage of a generic 3D tracking algorithm that can better handle occlusion and imperfect detections, yet remains relatively simple, we adapt the image-based tracking method proposed by [3] to the 3D setting. Specifically, this method is a two-stage data association scheme. In this scheme, each tracklet is assigned a confidence score computed based on how well the associated detection matches with itself. The first association stage aims to establish the correspondence between high-confident tracklets and detection. The second stage matches the leftover detection with low-confident tracklets as well as links low-confident tracklets to high-confident ones if they represent broken trajectories.

This paper makes two contributions

- Our main contribution is the adaptation of an image-based tracking method to the 3D setting. In detail, we exploit a kinematically feasible motion model, which is unavailable in 2D, to facilitate the prediction of objects' poses. This motion model defines the minimal state vector needed to be tracked.
- Extensive experiment carried out in various datasets proves the effectiveness of our approach. In fact, our better performance, compared to AB3DMOT-style models,

shows that adding a certain degree of re-identification can improve the tracking performance while keeping the added complexity to the minimum.

A1.2 Related work

A multi-object tracking system in the track-by-detection paradigm consists of an object detection model, a data association algorithm, and a filtering method. While the last two components are domain agnostic, object detection models, especially learning-based methods, are tailored to their operation domain (e.g. images or point clouds). To ensure a fair comparison with existing tracking methods, we use the detection result provided by baseline models of benchmarks (e.g. PointPillars of NuScenes)

Data association via the Hungarian algorithm was early explored in [37] where a 2-stage tracking scheme was proposed for offline 2D tracking. Firstly, detections are linked frame-by-frame to form tracklets by associating detections to tracklets via solving a LAP with the Hungarian algorithm. The cost matrix of this LAP is computed based on geometric and appearance cues. While the geometric cue is the 2D IoU, the appearance cue is the correlation between two bounding boxes. Secondly, tracklets are associated with each other to reduce trajectory fragments and ID switches due to occlusion. Similar to the previous step, this association is also formulated as a LAP and solved by the Hungarian algorithm.

Due to its batch-processing nature, the method of [37] cannot be applied to online tracking. Bewley et al [7] overcome this by eliminating the second stage, which effectively sacrifices the ability to re-identifies objects after a period of occlusion. Despite its simplicity, the method proposed by [7], named SORT, achieves a competitive result in MOT15[62] with lightning-fast inference speed (260 Hz). The success of SORT inspired [119] to adapt it to the 3D setting by using 3D IoU as the affinity function. The performance of SORT in the 3D setting is later improved in [18] showing the superiority over 3D IoU of the Mahalanobis distance which is the magnitude of difference between the expected detection given the ego vehicle pose and the real detection while taking into account their uncertainty. Mauri et al [85] integrates the 3D version of SORT into a complete perception pipeline for autonomous vehicles.

The two-stage association scheme is adapted to online tracking in [3] which proposes a confidence score to quantify tracklets quality. Based on this score, tracklets are associated with detections or other tracklets, or terminated. The appearance model learned by ILDA

in [3] is improved by deep learning in the follow-up work [2]. Recently, this association scheme is revisited in the context of image-based pedestrian tracking by [132] which proposed to use the rank of the Hankel matrix as tracklets motion affinity. To be specific, this technique estimates a tracklet's dynamic by a linear regressor taking its previous states as input. In noise-free scenarios, the order of such a regressor (i.e. the number of past states needed to estimate the current state) is equal to the rank of the Hankel matrix which formula can be found in [132]. The intuition behind this technique is that if two tracklets belong to the same trajectory, explaining their merged trajectory would require a low-order regressor. This technique is popular in image-based tracking despite being prone to deterioration due to noise because of the absence of an accurate motion model in this space. However, objects' motion in 3D can be well explained by their kinematic models. Therefore, our approach employs two different kinematic models for two different categories of objects to have more computationally efficient and accurate motion affinity.

Differently from [3] and its related works, this paper applies the two-stage association scheme to online 3D tracking. In addition, we can achieve strong performance while relying solely on geometric cues to compute tracklet affinity by leveraging the Constant Turning Rate and Velocity (CTRV) motion model which can accurately predict objects' position in 3D space by exploiting their kinematics.

A1.3 Method

A1.3.1 Problem Formulation

Online MOT in the sense of track-by-detection aims to gradually grow the set of tracklets $\mathcal{R}_{0:t} = \{\mathcal{T}^i\}_{i=1}^{|\mathcal{R}_{0:t}|}$ by establishing correspondences with the set of detections received at every time step $\mathcal{B}_t = \{\mathbf{b}_t^j\}_{j=1}^{|\mathcal{B}_t|}$ and updating tracklets state accordingly. A detection $\mathbf{b}_t^j$ at time step t encapsulates information of an object as a 3D bounding box

$$\mathbf{b}_t^j = \begin{bmatrix} x & y & z & \theta & w & l & h \end{bmatrix}^T, \tag{A1.1}$$

here, $[x, y, z]$ is the position of the box's center, θ is its heading direction, and $[w, l, h]$ is its size. It is worth noticing that in the context of autonomous driving, objects are assumed to remain in contact with the ground; therefore, their detections are up-right bounding boxes whose orientation is described by a single number - the heading angle. A tracklet is a collection of state vectors corresponding to the same object $\mathcal{T}^i = \{\mathbf{x}_k^i | 0 \leq t_s^i \leq k \leq t_e^i \leq t\}$,

here t_s^i, t_e^i are respectively the starting- and ending time of the tracklet.

The correspondence between $\mathcal{R}_{0:t}$ and $\mathcal{B}_t$ can be formally defined as finding the set $\mathcal{R}_{0:t}^*$ that maximizes its likelihood given $\mathcal{B}_t$.

$$\mathcal{R}_{0:t}^* = \underset{\mathcal{R}_{0:t}}{\mathsf{argmax}}\ \mathsf{p}\left(\mathcal{R}_{0:t}|\mathcal{B}_t\right) \tag{A1.2}$$

Due to the exponential growth of possible associations between $\mathcal{R}_{0:t}$ and $\mathcal{B}_t$, Eq.(A1.2) is computationally intractable after a few time steps. Here, such a correspondence is approximated by the two-stage data association proposed by [3] as shown in the following.

A1.3.2 Two-stage Data Association

Tracklet Confidence Score

The reliability of a tracklet is quantified by a confidence score which is calculated based on how well-associated detections match with its states across its life span and how long its corresponding object was undetected.

$$\mathsf{conf}\left(\mathcal{T}^i\right) = \left(\frac{1}{L^i}\sum_{k\in[t_s^i,t_e^i]|v^i(k)=1}\Lambda\left(\mathcal{T}^i, d_k^j\right)\right) \times \exp\left(-\beta\frac{W}{L_i}\right) \tag{A1.3}$$

where $v^i(k)$ is a binary indicator that takes 1 if the tracklet has a detection $\mathbf{b}_k^j$ associated with it at time step k, and 0 otherwise. L^i is the number of time steps that the traklet gets associated with detection. $\Lambda(\cdot)$ is the affinity function that evaluates the similarity between a track and a detection. Its detail will be presented in the following subsection. β is a tuning parameter that takes a high value if the object detection model is accurate. $W = t - t_s^i - L^i + 1$ is the number of time steps that the tracklet was undetected (i.e. did not have associated detection) calculated from its birth to the current time step t.

Applying a threshold τ^c this confidence score divides the set $\mathcal{R}_{0:t}$ into a subset of high-confident tracklets $\mathcal{R}_{0:t}^h = \{\mathcal{T}^{i,h}|\mathsf{conf}\left(\mathcal{T}^i\right) > \tau^c\}$ and a subset of low-confident tracklets $\mathcal{R}_{0:t}^l = \{\mathcal{T}^{i,l}|\mathsf{conf}\left(\mathcal{T}^i\right) \leq \tau^c\}$. These two subsets are the fundamental elements of the two-stage association pipeline shown in Fig.A1.1

Affinity Function

Affinity function $\Lambda(\cdot)$ is to compute how similar a detection is to a tracklet or a tracklet is to another. As mentioned earlier, due to the lack of colorful texture in point clouds, the

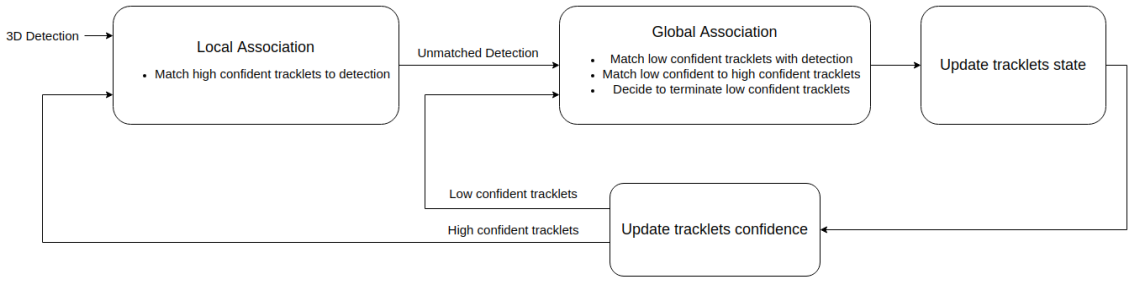
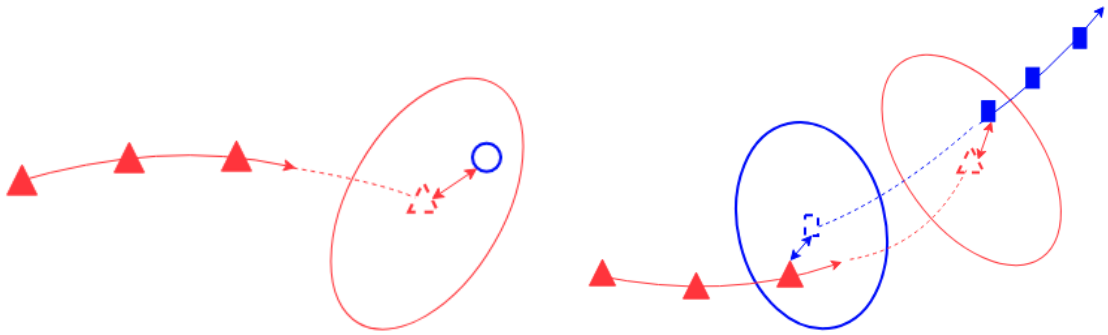


Figure A1.1 – The pipeline of two-stage data association. In the first stage - the local association establishes the correspondences between detections at this time step and high-confident tracklets. Then, the global association stage matches each low-confident tracklets with either a high-confident tracklet or a left-over detection or terminates it.

affinity function is just comprised of geometric cues. Specifically, it is the sum of position affinity and size affinity.

$$\Lambda(\mathcal{T}^i, d_t^j) = \Lambda(\mathcal{T}^i, d_t^j)^p + \Lambda(\mathcal{T}^i, d_t^j)^s \tag{A1.4}$$

The scheme for computing position affinity between a tracklet and detection or between two tracklets is shown in Fig.A1.2



(a) tracklet-to-detection position affinity (b) tracklet-to-tracklet position affinity

Figure A1.2 – The computational scheme of position affinity. The filled triangles (or rectangles) are subsequent states of a tracklet. The colored arrow represents the time order: the closer to the tip, the more recent the state. The triangle (or rectangle) in the dash line is the state propagated forward (or backward) in time. The covariance of these propagated states are denoted by ellipses with the same color. The two-headed arrows indicate the Mahalanobis distance. In subfigure (a), the blue circle denotes a detection.

As shown in Fig.A1.2a, the position affinity between a tracklet and a detection is defined as the Mahalanobis distance between tracklet's last state propagated to the current

time step t and the measurement vector $\mathbf{z}_t^j = [x, y, z, \theta]^T$ extracted from the detection $\mathbf{b}_t^j$

$$\Lambda(\mathcal{T}^i, d_t^j)^p = \left(\mathsf{h}\left(\bar{\mathbf{x}}_e^i\right) - \mathbf{z}_t^j\right)^T \cdot \mathbf{S}^{-1} \cdot \left(\mathsf{h}\left(\bar{\mathbf{x}}_e^i\right) - \mathbf{z}_t^j\right) \tag{A1.5}$$

where $\bar{\mathbf{x}}_e^i$ is the last state of tracklet $\mathcal{T}^i$ propagated to the current time step using the motion model which will be presented below. $\mathsf{h}(\cdot)$ is the measurement model computing the expected measurement using the inputted state.

The matrix $\mathbf{S}$ is the covariance matrix of the innovation which is the difference between the expected measurement and its true value

$$\mathbf{S} = \mathbf{H} \cdot \mathbf{P} \cdot \mathbf{H}^T + \mathbf{R} \tag{A1.6}$$

here, $\mathbf{H} = \delta\mathsf{h}/\delta\mathbf{x}$ is the Jacobian of the measurement model. $\mathbf{P}, \mathbf{R}$ are covariance matrix of $\bar{\mathbf{x}}_e^i$ and $\mathbf{z}_t^j$, respectively. These covariance matrices are calculated based on training data using the approach proposed by [18].

In the case of two tracklets $\mathcal{T}^i$ and $\mathcal{T}^j$, assuming $\mathcal{T}^j$ starts after $\mathcal{T}^i$ ended, their motion affinity is, according to Fig.A1.2b, is the sum of

- The Mahalanobis distance between the last state of $\mathcal{T}^i$ propagated forward in time and the first state of $\mathcal{T}^j$
- The Mahalanobis distance between the first state of $\mathcal{T}^j$ propagated backward in time and the last state of $\mathcal{T}^i$

$$\Lambda(\mathcal{T}^i, \mathcal{T}^j)^p = \Lambda(\mathcal{T}^j, \bar{\mathbf{x}}_e^i)^p + \Lambda(\mathcal{T}^i, \bar{\mathbf{x}}_s^j)^p \tag{A1.7}$$

here, $\bar{\mathbf{x}}_e^i$ is the last state of tracklet $\mathcal{T}^i$ propagated forward in time to the first time step of tracklet $\mathcal{T}^j$, while $\bar{\mathbf{x}}_s^j$ is the first state of tracklet $\mathcal{T}^j$ propagated backward in time to the last time step of tracklet $\mathcal{T}^i$.

The size affinity $\Lambda(\mathcal{T}^i, d_t^j)^s$ is computed as following

$$\Lambda\left(\mathcal{T}^i, d_t^j\right)^s = -\frac{|w_e^i - w_t^j|}{w_e^i + w_t^j} \cdot \frac{|l_e^i - l_t^j|}{l_e^i + l_t^j} \cdot \frac{|h_e^i - h_t^j|}{h_e^i + h_t^j} \tag{A1.8}$$

here, $[w_e^i, l_e^i, h_e^i]$ are the size of the last state of tracklet $\mathcal{T}^i$, while $[w_t^j, l_t^j, h_t^j]$ are the size of the detection d_t^j. In the case of two tracklets $\mathcal{T}^i$ and $\mathcal{T}^j$, assuming $\mathcal{T}^j$ starts after $\mathcal{T}^i$ ended, their size affinity is

$$\Lambda\left(\mathcal{T}^i, \mathcal{T}^j\right)^s = -\frac{|w_e^i - w_s^j|}{w_e^i + w_s^j} \cdot \frac{|l_e^i - l_s^j|}{l_e^i + l_s^j} \cdot \frac{|h_e^i - h_s^j|}{h_e^i + h_s^j} \tag{A1.9}$$

The subscript e, s in Equation (A1.9) respectively denote the ending and starting state of a tracklet.

To reduce the risk of false association, a threshold is applied to the affinity

$$\Lambda(\mathcal{T}^i, \mathbf{b}_t^j) = \begin{cases} \Lambda(\mathcal{T}^i, \mathbf{b}_t^j), & \text{if } \Lambda(\mathcal{T}^i, \mathbf{b}_t^j) < \sigma \\ \infty, & \text{otherwise} \end{cases} \tag{A1.10}$$

Local Association

In this association stage, high-confident tracklets ($\mathbb{T}_{0:t}^h$) are extended by their correspondence in the set of detections $\mathcal{B}_t$. This tracklet-to-detection is found by solving the LAP characterized by the cost matrix $\mathbf{L}$ as following

$$\mathbf{L} = \left[l_{i,j}\right] \in \mathbb{R}^{h \times d}, \text{ with } l_{i,j} = -\Lambda\left(\mathcal{T}^{i,h}, \mathbf{b}_t^j\right), \mathcal{T}^{i,h} \in \mathcal{R}_{0:t}^h \tag{A1.11}$$

where h, d are respectively the number of high-confident tracklets and the number of detections. The intuition of this association stage is that because tracklets with high-confident have been tracked accurately for a number of time steps, the affinity function can identify if a detection belongs to the same object as the tracklet with high accuracy, thus limiting the possibility of false correspondences. In addition, low-confident tracklets are usually resulted from fragment trajectories or noisy detections, excluding them from this association stage help reduces the ambiguity.

Global Association

As shown in Figure A1.1, the global association stage carries out the following tasks

- Matching low-confident tracklets with high-confident ones
- Matching low-confident tracklets with detections left over by the local association stage
- Deciding to terminate low-confident tracklets

These tasks are simultaneously solved as a LAP formulated by the following cost matrix

$$\mathbf{G}_{(l+d')\times(h+l)} = \begin{bmatrix} \mathbf{A}_{l\times h} & \mathbf{B}_{l\times l} \\ \infty_{d'\times h} & \mathbf{C}_{d'\times l} \end{bmatrix} \tag{A1.12}$$

here, l, d are respectively the number of low-confident tracklets and detections left over by the local association stage. $\infty_{d'\times h}$ is the matrix of size $d' \times h$ with every element is set to ∞. Recall h is the number of high-confident tracklets. Submatrix $\mathbf{A}$ is the cost matrix of the event where low-confident tracklets are matched with high-confident ones

$$\mathbf{A} = [a_{i,j}] \in \mathbb{R}^{l\times h}, \text{ with } a_{i,j} = -\Lambda\left(T^{i,l}, T^{j,h}\right) \tag{A1.13}$$

Submatrix $\mathbf{B}$ represents the event where low-confident tracklets are terminated.

$$\mathbf{B} = [b_{i,j}] \in \mathbb{R}^{l\times l}, \text{ with } b_{i,j} = \begin{cases} -\log\left(1 - \mathsf{conf}\left(\mathcal{T}^i\right)\right), \text{ if } i = j \\ \infty, \text{ otherwise} \end{cases} \tag{A1.14}$$

Finally, submatrix $\mathbf{C}$ is the cost of associating low-confident tracklets with detections left over by the local association stage.

$$\mathbf{C} = [c_{i,j}] \in \mathbb{R}^{d'\times l}, \text{ with } c_{i,j} = -\Lambda\left(\mathcal{T}^j, d_t^i\right) \tag{A1.15}$$

The solution to the LAP in this stage and in the Local Association stage is the association that *minimize* the cost and can be either found by the Hungarian algorithm for the optimal solution or by a greedy algorithm which iteratively picks and removes correspondence pair with the smallest cost until there is no pair has cost less than a threshold. The detail of this greedy algorithm can be found in Alg.4 in Sec.A1.3.4.

Once a detection is associated with a tracklet, its position, and heading are used to update the tracklet's state according to the equation of the Kalman Filter, while its sizes are averaged with the tracklet's sizes in the past few time steps to result in updated sizes. Detections do not get associated in the global association stage and are used to initialize new tracklets.

A1.3.3 Motion Model and State Vector

Leveraging the fact that objects are tracked in 3D space of a common static reference frame which can be referred to as the world frame, the motion of objects can be described by more kinematically accurate models, compared to the commonly used Constant Veloc-

ity (CV) model. Here, we use the CTRV model to predict the motion of car-like vehicles (e.g. cars, buses, trucks), while keeping the CV model for pedestrians.

For car-like vehicles, its state can be described by

$$\mathbf{x} = [x, y, z, \theta, \mathtt{v}, \dot{\theta}, \dot{z}]^T \tag{A1.16}$$

here, $[x, y, z]$ is the location in the world frame of the center of the bounding box represented by the state vector, θ is the heading angle, $\mathtt{v}$ is the longitudinal velocity (i.e. velocity along the heading direction), $\dot{\theta}, \dot{z}$ are respectively velocity of θ and z. The motion on the XY plane of car-like vehicles can be predicted using CTRV as following

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \begin{bmatrix} \frac{\mathtt{v}}{\dot{\theta}}\Big(\sin(\theta + \dot{\theta}\Delta t) - \sin(\theta)\Big) \\ \frac{\mathtt{v}}{\dot{\theta}}\Big(-\cos(\theta + \dot{\theta}\Delta t) + \cos(\theta)\Big) \\ \dot{z}\Delta t \\ \dot{\theta}\Delta t \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{A1.17}$$

where, Δt is the sampling time. Note that in Equation (A1.17), z is assumed to evolve with constant velocity. In the case of zero turning rate (i.e. $\dot{\theta} = 0$),

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \begin{bmatrix} \mathtt{v}\cos(\theta) & \mathtt{v}\sin(\theta) & \dot{z}\Delta t & \dot{\theta}\Delta t & 0 & 0 & 0 \end{bmatrix}^T \tag{A1.18}$$

The state vector of a pedestrian is

$$\mathbf{x} = \begin{bmatrix} x & y & z & \theta & \dot{x} & \dot{y} & \dot{z} & \dot{\theta} \end{bmatrix}^T \tag{A1.19}$$

The motion of pedestrians is predicted according to the CV model

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \begin{bmatrix} \dot{x} & \dot{y} & \dot{z} & \dot{\theta} & 0 & 0 & 0 & 0 \end{bmatrix}^T \cdot \Delta t \tag{A1.20}$$

A1.3.4 Complexity Analysis

As shown in Fig.A1.1, our data association pipeline is made of four components: Local Association, Global Association, Update Tracklets' States, and Update Tracklets' Confi-

dence. This section gives an analysis of the time complexity referred to as the complexity of these four components.

Let d and h be the number of detections and the number of high-confident tracklets, respectively. The time complexity of the Local Association step is the sum of the complexity of computing the cost matrix $\mathbf{L}$ in Eq.(A1.11) and solving the LAP represented by $\mathbf{L}$. Since $\mathbf{L}$ has the size of $h \times d$, the complexity of computing $\mathbf{L}$ is $\mathrm{O}(hd)$.

The LAP represented by $\mathbf{L}$ can be solved by either the Hungarian algorithm or a greedy algorithm [18]. The complexity of the Hungarian algorithm is $\mathrm{O}(hd^2)$. On the other hand, the greedy algorithm is made of two steps presented in Alg.4. The first step of sorting the flattened cost matrix $\mathbf{C} \in \mathbb{R}^{r \times c}$ has the complexity of $\mathrm{O}(rc \log(rc)) = \mathcal{O}(rc \log(c))$ assuming $c > r$. The complexity of the second step in the best-case scenario where the for loop is stopped at $k = 0$, meaning there is no valid association, is $\mathrm{O}(1)$. The worst case scenario happens when the For Loop proceeds till the last value of k, which means every possible association has an affinity less than the threshold σ. In this case, the complexity is $\mathcal{O}(|\mathrm{c}^{flat}|) = \mathcal{O}(rc)$. As a result, the complexity of the greedy algorithm is

$$\mathrm{O}(rc \log(c)) + \mathrm{O}(rc) = \mathrm{O}(rc \log(c)) \tag{A1.21}$$

Using Eq.(A1.21), the complexity of the Local and Global Association step solved by the greedy algorithm is $\mathrm{O}(hd \log(d))$ and $\mathcal{O}((l + d')(h + l)) \log(l + d')$, respectively. Recall l and d' are the number of low-confident tracklets and the number of detections left over by the Local Association step. The other steps, Update Tracklets' States and Update Tracklets' Confidence have linear complexity because they are made of one loop through all tracklets.

A1.4 Experiments

The effectiveness of our method is demonstrated by benchmarking against SORT-style baseline models on 3 large-scale datasets: KITTI, NuScenes, and Waymo. In addition, we perform ablation studies using the NuScenes dataset to better understand the impact of each component on our system's general performance.

Algorithm 4: Greedy algorithm for solving LAP

Input: Cost matrix $\mathbf{C} \in \mathbb{R}^{r \times c}$ and cost threshold σ
Result: List of pairs of indices. Each pair of (i, j) denotes the correspondence between a row i and a column j of the inputted cost matrix $\mathbf{C}$

```
Flatten C into an array c^flat and sort c^flat into the ascending order ;
P = ∅ ;
for k ∈ {0, 1, ..., |c^flat| − 1} do
    if c^flat > σ then
        break ;
    else
        j = k mod c ;
        i = (k − j)/c;
        if both i and j aren't in any pairs in P then
            add (i, j) to P
        end
    end
end
```

A1.4.1 Tuning the hyper parameters

There are 3 hyperparameters in our data association pipeline: the confidence threshold τ^c, the detection model accuracy β in Eq.(A1.3), and the affinity threshold σ.

The confidence threshold τ^c is set to 0.5 according to [3]. It is worth noticing that [3] suggests that this parameter does not have any significant effect on the tracking performance. The value of β is chosen empirically such that a high-confident tracklet becomes low-confident after being undetected for three frames.

As observed from experiments, the position affinity $\Lambda(\cdot, \cdot)^p$ is the dominant component in the tracklet-to-detection and tracklet-to-tracklet affinity. Since the position affinity, which is the Mahalanobis distance between expected detection and real detection, is χ^2 distributed, the affinity threshold σ in Eq.(A1.10) is chosen according to the percentile of χ^2 distribution where the position affinity resulted from a correct association is expected to fall into. Notice that the degree of freedom of the χ^2 distribution of our interest is 4 due to the dimension of the measurement vector.

Intuitively, the affinity threshold σ determines how conservative our tracking algorithm is. A small σ makes our algorithm be more skeptical by rejecting detections that are close, but not close enough to the prediction of tracks' states. This works well in the scenario where a large number of false-positive detections are present (e.g. Waymo dataset). How-

ever, too small σ can reject correct detections thus deteriorating the tracking performance. The method used for searching for a good value of σ is

- Performs a coarse grid search with the expected percentile of χ^2 distribution in the set $\{10\%, 50\%, 90\%, 95\%, 97.5\%, 99\%\}$ which means the value of σ is in the set $\{0.53, 1.67, 3.89, 4.75, 5.57, 6.64\}$ while keeping the rest of the hyperparameters unchanged. Note that here the value of the threshold σ is just half of the corresponding value in the χ^2 Distribution Table. This is because the motion affinity is scaled by half in our implementation to reduce its dominance over the size affinity.
- Once a performance peak is identified at $\hat{\sigma}$, a fine grid search is performed on the set $\{\hat{\sigma} - 0.2, \hat{\sigma} - 0.1, \hat{\sigma}, \hat{\sigma} + 0.1, \hat{\sigma} + 0.2\}$

The resulting value of σ on KITTI, NuScenes, and Waymo are respectively 6.5, 4.5, and 1.5.

A1.4.2 Tracking Results

Evaluation Metrics Classically, MOT systems are evaluated by the CLEAR MOT metrics [6] which compute tracking performances based on three cores quantities which are the number of False Positives, False Negatives, and ID Switches. Intuitively, this set of metrics aims at evaluating a tracker's precision in estimating tracks' states as well as its consistency (i.e. keeping an unique ID for each even in the presence of occlusion). As pointed out by [63] and later by [119], there is a linear relation between MOTA and object detectors' recall rate, as a result, MOTA does not provide a well-rounded evaluation performance of trackers. To remedy this, [119] proposes to average MOTA and MOTP over a range of recall rate, resulting in two integral metrics AMOTA and AMOTP which become the norm in recent benchmarks.

Datasets To verify the effectiveness of our method, we benchmark it on 3 popular autonomous driving datasets which offer 3D MOT benchmarks: KITTI, NuScenes, and Waymo. These datasets are collections of driving sequences collected in various environments using a multi-modal sensor suite including LiDAR. KITTI tracking benchmark interests in two classes of objects which are cars and pedestrians. Initially, KITTI tracking was designed for MOT in 2D images, and recently [119] adapts it to 3D MOT. NuScenes concerns a larger set of objects which comprises of cars, bicycles, buses, trucks, pedestrians, motorcycles, trailers. Waymo shares the same interest as NuScenes but groups car-like vehicles into a metaclass.

Public Detection As can be seen in Tab.A1.1, AMOTA highly depends on the precision of object detectors. Therefore, to have a fair comparison, the baseline detection results made publicly available by the benchmarks are used as input to our tracking system. Specifically, we use Point-RCNN detection for the KITTI dataset, MEGVII detection for NuScenes, and PointPillars with PPBA detection for Waymo.

The performance of our model compared to the SORT-style baseline model in 3 popular benchmarks is shown in Tab.A1.2. As can be seen, our model consistently outperforms

Table A1.2 – Quantitative performance of our model on KITTI validation set, NuScenes validation set, and Waymo test set. AMOTA is the primary metric of these benchmarks. FP, FN IDS and FRAG are absolute numbers in the case of KITTI and NuScenes, while they are divided by the total number of objects in Waymo. The performance on Waymo is calculated at the difficulty of LEVEL 2.

Dataset	Method	**AMOTA↑**	AMOTP↓	MT↑	ML↓	FP↓	FN↓	IDS↓	FRAG↓
KITTI (val)	Ours	**0.415**	0.691	NA	NA	766	3721	10	259
	AB3DMOT[119]	0.377	**0.648**	NA	NA	**696**	**3713**	**1**	**93**
NuScenes (val)	Ours	**0.583**	**0.748**	**3617**	1885	13439	**28119**	**512**	**511**
	StanfordIPRL-TRI[18]	0.561	0.800	3432	**1857**	**12140**	28387	679	606
Waymo (test @ L2)	Ours	**0.365**	**0.263**	NA	NA	**0.089**	**0.533**	0.014	NA
	PPBA-AB3DMOT	0.291	0.270	NA	NA	0.171	0.535	**0.003**	NA

the baseline model in terms of the primary metric AMOTA, thus proving the effectiveness of the 2-stage data association. Specifically, the improvement is $10.080\%, 3.922\%, 25.430\%$ for KITTI, NuScenes, and Waymo. It is worth noticing that, our approach has more track fragmentations, 259 compared to 93 of the baseline in KITTI. The reason for this is that at each time step tracklets have no matched detections are not reported by our approach, while the baseline predicts their pose using the CV model and reports this prediction.

The comparison runtime on the KITTI dataset of our tracking algorithm against AB3DMOT [119] is shown in Tab.A1.3. Despite the additional complexity added by the second stage of the data association, our approach can achieve a runtime that is close to AB3DMOT on KITTI and exceeds the real-time speed by a large margin. On more challenging datasets, the object detector generates a significantly larger number of detections per frame on average, 57.50 on NuScenes and 264.18 on Waymo, compared to 10.04 of KITTI. This large number of detections enlarges the cost matrix of the Local and Global Association step, thus making the LAPs represented by them more costly to solve. T herefore, the runtime of our approach is reduced to 1.44 frames-per-second (fps) on NuScenes and 0.35 fps on Waymo. This runtime can be greatly improved if our approach

is re-implemented in a compiling language such as C++.

Table A1.3 – Comparison of our approach's runtime on KITTI dataset against AB3DMOT's

Class of objects	Our runtime (fps)	AB3DMOT's runtime (fps)
Car	115	186
Pedestrian	497	424
Cyclist	1111	1189

A1.4.3 Ablation Study

In this ablation study, the default method has the following settings

- Two stages of data association (local and global). Each stage is formulated as a LAP and solved by a greedy matching algorithm [18].
- The affinity function is the sum of position affinity and size affinity (as in Equation (A1.4)).
- The motion model is CTRV for car-like objects (cars, buses, trucks, trailers, bicycles) and CV for pedestrians.
- The value of hyperparameters are set as following: $\beta = 1.35$ (in Eq.(A1.3)), tracklet confidence threshold $\tau^c = 0.45$, and the affinity threshold $\sigma = 4.5$ (in Eq.(A1.10))

To understand the effect of each component on the system's general performance, we modify or remove each of them and carry out experiments with the rest of the system being kept the same as the default method and the same hyperparameters. The changes and the resulting performance are shown in Tab.A1.4.

Table A1.4 – Ablation study using NuScenes dataset.

Method	**AMOTA**↑	AMOTP↓	MT↑	ML↓	FP↓	FN↓	IDS↓	FRAG↓
Default	0.583	0.748	**3617**	1885	13439	28119	512	511
Hungarian for LAP	**0.587**	**0.743**	3609	**1880**	13667	**28070**	596	573
No ReID	0.583	0.748	3616	1882	13429	28100	**504**	**510**
Global assoc only	0.327	0.924	2575	2244	26244	38315	4215	3038
Const Velocity only	0.567	0.781	3483	1966	12649	29427	718	606
No size affinity	0.581	0.748	3595	1904	13423	28448	512	508
3D IoU as affinity	0.535	0.898	3090	2075	**9168**	33041	550	528

It can be seen that solving the matching problem (formulated as a LAP) with the Hungarian algorithm instead of the greedy matching algorithm of [18] results in a marginal increase of AMOTA; however, this increased performance comes at the cost of increased execution time since the Hungarian algorithm has higher time complexity (cubic time compared to quadratic time). In addition, using the CV model only reduces the AMOTA by 2.744% compared to the default setting which shows the effectiveness of the CTRV model in predicting the motion of car-like vehicles. Finally, performing global association only deteriorates the tracking performance and confirms the importance of the local association step which significantly reduces the association ambiguity for the second stage.

A1.5 Conclusion

In conclusion, this chapter has successfully adapted an image-based tracking method to the 3D space. Particularly, extensive experiments carried out in various datasets show that our two-stage data association pipeline can result in significant improvement in the tracking accuracy by adding a certain degree of re-identification while keeping the added complexity to the minimum. Nevertheless, medium and long-term occlusion remains challenging for our approach due to the fact that the affinity function relies mostly on tracklets' position whose prediction's reliability reduces with the length of the prediction horizon. In the domain of image-based MOT, this problem is offend solved by exploiting trackets' appearance with Siamese networks [10, 61]. However, the extension of this method to 3D space is not straightforward due to the lack of color and texture in point clouds. A possibility to resolve this issue is to associate 3D tracklets with 2D object detections, then carry out re-identification in images. Taking a different approach, a recent work in graph neural networks [120] proposes to jointly learn affinity functions from point clouds and images.

BIBLIOGRAPHY

[1] Eduardo Arnold et al., « A survey on 3d object detection methods for autonomous driving applications », *in*: *IEEE Transactions on Intelligent Transportation Systems* 20.*10* (2019), pp. 3782–3795.

[2] Seung-Hwan Bae and Kuk-Jin Yoon, « Confidence-based data association and discriminative deep appearance learning for robust online multi-object tracking », *in*: *IEEE transactions on pattern analysis and machine intelligence* 40.*3* (2017), pp. 595–610.

[3] Seung-Hwan Bae and Kuk-Jin Yoon, « Robust online multi-object tracking based on tracklet confidence and online discriminative appearance learning », *in*: *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 1218–1225.

[4] Xuyang Bai et al., « Transfusion: Robust lidar-camera fusion for 3d object detection with transformers », *in*: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 1090–1099.

[5] Maxim Berman, Amal Rannen Triki, and Matthew B Blaschko, « The lovász-softmax loss: A tractable surrogate for the optimization of the intersection-over-union measure in neural networks », *in*: *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4413–4421.

[6] Keni Bernardin and Rainer Stiefelhagen, « Evaluating multiple object tracking performance: the CLEAR MOT metrics », *in*: *EURASIP Journal on Image and Video Processing* 2008 (2008), pp. 1–10.

[7] Alex Bewley et al., « Simple online and realtime tracking », *in*: *2016 IEEE International Conference on Image Processing (ICIP)*, IEEE, 2016, pp. 3464–3468.

[8] Holger Caesar et al., « nuscenes: A multimodal dataset for autonomous driving », *in*: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11621–11631.

[9] Sergio Casas, Wenjie Luo, and Raquel Urtasun, « Intentnet: Learning to predict intention from raw sensor data », *in*: *Conference on Robot Learning*, PMLR, 2018, pp. 947–956.

[10] Shuo Chang et al., « Online siamese network for visual object tracking », *in*: *Sensors* 19.*8* (2019), p. 1858.

[11] Qi Chen et al., « Cooper: Cooperative perception for connected autonomous vehicles based on 3d point clouds », *in*: *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, IEEE, 2019, pp. 514–524.

[12] Xiaozhi Chen et al., « Multi-view 3d object detection network for autonomous driving », *in*: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1907–1915.

[13] Xieyuanli Chen et al., « Automatic labeling to generate training data for online LiDAR-based moving object segmentation », *in*: *IEEE Robotics and Automation Letters* 7.*3* (2022), pp. 6107–6114.

[14] Xieyuanli Chen et al., « Moving object segmentation in 3D LiDAR data: A learning-based approach exploiting sequential data », *in*: *IEEE Robotics and Automation Letters* 6.*4* (2021), pp. 6529–6536.

[15] Xuesong Chen et al., « Mppnet: Multi-frame feature intertwining with proxy points for 3d temporal object detection », *in*: *European Conference on Computer Vision*, Springer, 2022, pp. 680–697.

[16] Bowen Cheng et al., « Masked-attention Mask Transformer for Universal Image Segmentation », *in*: *arXiv preprint arXiv:2112.01527* (2021).

[17] Shuyang Cheng et al., « Improving 3D Object Detection through Progressive Population Based Augmentation », *in*: *arXiv preprint arXiv:2004.00831* (2020).

[18] Hsu-kuang Chiu et al., *Probabilistic 3D Multi-Object Tracking for Autonomous Driving*, 2020, arXiv: `2001.05673 [cs.CV]`.

[19] Zhuang Jie Chong et al., « Synthetic 2D LIDAR for precise vehicle localization in 3D urban environment », *in*: *2013 IEEE International Conference on Robotics and Automation*, IEEE, 2013, pp. 1554–1559.

[20] Minh-Quan Dao and Vincent Frémont, « A two-stage data association approach for 3D Multi-object Tracking », *in*: *Sensors* 21.*9* (2021), p. 2894.

[21] Minh-Quan Dao, Vincent Frémont, and Elwan Héry, « Aligning Bird-Eye View Representation of Point Cloud Sequences using Scene Flow », *in*: *arXiv preprint arXiv:2305.02909* (2023).

[22] Minh-Quan Dao, Elwan Héry, and Vincent Frémont, « Attention-based proposals refinement for 3D object detection », *in*: *2022 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, 2022, pp. 197–205.

[23] Minh-Quan Dao et al., « Practical Collaborative Perception: A Framework for Asynchronous and Multi-Agent 3D Object Detection », *in*: *arXiv preprint arXiv:2307.01462* (2023).

[24] Jiajun Deng et al., « Voxel R-CNN: Towards High Performance Voxel-based 3D Object Detection », *in*: *arXiv preprint arXiv:2012.15712* (2020).

[25] Jiajun Deng et al., « Voxel r-cnn: Towards high performance voxel-based 3d object detection », *in*: *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, 2, 2021, pp. 1201–1209.

[26] Zhuangzhuang Ding et al., *1st Place Solution for Waymo Open Dataset Challenge – 3D Detection and Domain Adaptation*, 2020, arXiv: `2006.15505 [cs.CV]`.

[27] Nemanja Djuric et al., « Multixnet: Multiclass multistage multimodal motion prediction », *in*: *2021 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, 2021, pp. 435–442.

[28] Alexey Dosovitskiy et al., « An image is worth 16x16 words: Transformers for image recognition at scale », *in*: *arXiv preprint arXiv:2010.11929* (2020).

[29] Alexey Dosovitskiy et al., « CARLA: An Open Urban Driving Simulator », *in*: *Proceedings of the 1st Annual Conference on Robot Learning*, 2017, pp. 1–16.

[30] Liang Du et al., « Associate-3Ddet: Perceptual-to-conceptual association for 3D point cloud object detection », *in*: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 13329–13338.

[31] Lue Fan et al., « Embracing single stride 3d object detector with sparse transformer », *in*: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 8458–8468.

[32] Jin Fang et al., « Mapfusion: A general framework for 3d object detection with hdmaps », *in*: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2021, pp. 3406–3413.

[33] Angel F. Garcia-Fernandez et al., « Poisson Multi-Bernoulli Mixture Filter: Direct Derivation and Implementation », *in*: *IEEE Transactions on Aerospace and Electronic Systems* 54.*4* (Aug. 2018), pp. 1883–1901, ISSN: 2371-9877, DOI: 10.1109/taes.2018.2805153, URL: http://dx.doi.org/10.1109/TAES.2018.2805153.

[34] Runzhou Ge et al., *AFDet: Anchor Free One Stage 3D Object Detection*, 2020, arXiv: 2006.12671 [cs.CV].

[35] 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.

[36] Andreas Geiger, Philip Lenz, and Raquel Urtasun, « Are we ready for autonomous driving? the kitti vision benchmark suite », *in*: *2012 IEEE conference on computer vision and pattern recognition*, IEEE, 2012, pp. 3354–3361.

[37] Andreas Geiger et al., « 3d traffic scene understanding from movable platforms », *in*: *IEEE transactions on pattern analysis and machine intelligence* 36.*5* (2013), pp. 1012–1025.

[38] Andreas Geiger et al., « Vision meets robotics: The kitti dataset », *in*: *The International Journal of Robotics Research* 32.*11* (2013), pp. 1231–1237.

[39] Ross Girshick, « Fast r-cnn », *in*: *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[40] Zan Gojcic et al., « Weakly supervised learning of rigid 3D scene flow », *in*: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 5692–5703.

[41] Benjamin Graham, Martin Engelcke, and Laurens van der Maaten, « 3D Semantic Segmentation with Submanifold Sparse Convolutional Networks », *in*: *CVPR* (2018).

[42] Benjamin Graham and Laurens van der Maaten, « Submanifold Sparse Convolutional Networks », *in*: *arXiv preprint arXiv:1706.01307* (2017).

[43] Guy Hadash et al., « Estimate and Replace: A Novel Approach to Integrating Deep Neural Networks with Existing Applications », *in*: *arXiv preprint arXiv:1804.09028* (2018).

[44] Martin Hahner et al., « Quantifying data augmentation for lidar based 3d object detection », *in*: *arXiv preprint arXiv:2004.01643* (2020).

[45] Richard Hartley and Andrew Zisserman, *Multiple view geometry in computer vision*, Cambridge university press, 2003.

[46] Chenhang He et al., « Structure aware single-stage 3d object detection from point cloud », *in*: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11873–11882.

[47] Kaiming He et al., « Mask r-cnn », *in*: *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[48] Dan Hendrycks and Kevin Gimpel, « Gaussian error linear units (gelus) », *in*: *arXiv preprint arXiv:1606.08415* (2016).

[49] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean, « Distilling the knowledge in a neural network », *in*: *arXiv preprint arXiv:1503.02531* (2015).

[50] Yihan Hu et al., « Afdetv2: Rethinking the necessity of the second stage for object detection from point clouds », *in*: *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, 1, 2022, pp. 969–979.

[51] Rui Huang et al., « An lstm approach to temporal 3d object detection in lidar point clouds », *in*: *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVIII 16*, Springer, 2020, pp. 266–282.

[52] Shengyu Huang et al., « Dynamic 3D Scene Analysis by Point Cloud Accumulation », *in*: *European Conference on Computer Vision*, Springer, 2022, pp. 674–690.

[53] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al., « Spatial transformer networks », *in*: *Advances in neural information processing systems* 28 (2015).

[54] Shinpei Kato et al., « Autoware on Board: Enabling Autonomous Vehicles with Embedded Systems », *in*: *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)* (2018), pp. 287–296.

[55] George Kour and Raid Saabne, « Fast classification of handwritten on-line Arabic characters », *in*: *Soft Computing and Pattern Recognition (SoCPaR), 2014 6th International Conference of*, IEEE, 2014, pp. 312–318.

[56] George Kour and Raid Saabne, « Real-time segmentation of on-line handwritten arabic script », *in*: *Frontiers in Handwriting Recognition (ICFHR), 2014 14th International Conference on*, IEEE, 2014, pp. 417–422.

[57] Harold W Kuhn, « The Hungarian method for the assignment problem », *in*: *Naval research logistics quarterly* 2.*1-2* (1955), pp. 83–97.

[58] Ankit Laddha et al., « Mvfusenet: Improving end-to-end object detection and motion forecasting through multi-view fusion of lidar data », *in*: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 2865–2874.

[59] Alex H Lang et al., « Pointpillars: Fast encoders for object detection from point clouds », *in*: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 12697–12705.

[60] Hei Law and Jia Deng, « Cornernet: Detecting objects as paired keypoints », *in*: *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 734–750.

[61] Laura Leal-Taixé, Cristian Canton-Ferrer, and Konrad Schindler, « Learning by tracking: Siamese CNN for robust target association », *in*: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2016, pp. 33–40.

[62] Laura Leal-Taixé et al., « Motchallenge 2015: Towards a benchmark for multi-target tracking », *in*: *arXiv preprint arXiv:1504.01942* (2015).

[63] Laura Leal-Taixé et al., « Tracking the trackers: an analysis of the state of the art in multiple object tracking », *in*: *arXiv preprint arXiv:1704.02781* (2017).

[64] Yann LeCun et al., « LeNet-5, convolutional neural networks », *in*: *URL: http://yann. lecun. com/exdb/lenet* 20.*5* (2015), p. 14.

[65] Xueqian Li, Jhony Kaesemodel Pontes, and Simon Lucey, « Neural scene flow prior », *in*: *Advances in Neural Information Processing Systems* 34 (2021), pp. 7838–7851.

[66] Yiming Li et al., « Learning distilled collaboration graph for multi-agent perception », *in*: *Advances in Neural Information Processing Systems* 34 (2021), pp. 29541–29552.

[67] Yiming Li et al., « V2X-Sim: Multi-agent collaborative perception dataset and benchmark for autonomous driving », *in*: *IEEE Robotics and Automation Letters* 7.*4* (2022), pp. 10914–10921.

[68] Yingwei Li et al., « MoDAR: Using Motion Forecasting for 3D Object Detection in Point Cloud Sequences », *in*: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 9329–9339.

[69] You Li et al., « Toward location-enabled IoT (LE-IoT): IoT positioning techniques, error sources, and error mitigation », *in*: *IEEE Internet of Things Journal* 8.*6* (2020), pp. 4035–4062.

[70] Ming Liang et al., « Deep continuous fusion for multi-sensor 3d object detection », *in*: *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 641–656.

[71] Ming Liang et al., « PnPNet: End-to-End Perception and Prediction with Tracking in the Loop », *in*: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11553–11562.

[72] Tsung-Yi Lin et al., « Feature pyramid networks for object detection », *in*: *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2117–2125.

[73] Tsung-Yi Lin et al., « Focal loss for dense object detection », *in*: *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2980–2988.

[74] Jiang-Jiang Liu et al., « Improving convolutional networks with self-calibrated convolutions », *in*: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 10096–10105.

[75] Wei Liu et al., « Ssd: Single shot multibox detector », *in*: *European conference on computer vision*, Springer, 2016, pp. 21–37.

[76] Xingyu Liu, Charles R Qi, and Leonidas J Guibas, « Flownet3d: Learning scene flow in 3d point clouds », *in*: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 529–537.

[77] Ze Liu et al., « Swin transformer: Hierarchical vision transformer using shifted windows », *in*: *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 10012–10022.

[78] Zhijian Liu et al., « Point-Voxel CNN for Efficient 3D Deep Learning », *in*: *Advances in Neural Information Processing Systems*, 2019.

[79] Pablo Alvarez Lopez et al., « Microscopic Traffic Simulation using SUMO », *in*: *The 21st IEEE International Conference on Intelligent Transportation Systems*, IEEE, 2018, URL: https://elib.dlr.de/124092/.

[80] Ilya Loshchilov and Frank Hutter, *Fixing Weight Decay Regularization in Adam*, 2018.

[81] Wenjie Luo, Bin Yang, and Raquel Urtasun, « Fast and furious: Real time end-to-end 3d detection, tracking and motion forecasting with a single convolutional net », *in*: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2018, pp. 3569–3577.

[82] Zachary MacHardy et al., « V2X access technologies: Regulation, research, and remaining challenges », *in*: *IEEE Communications Surveys & Tutorials* 20.*3* (2018), pp. 1858–1877.

[83] Jiageng Mao et al., « One million scenes for autonomous driving: Once dataset », *in*: *arXiv preprint arXiv:2106.11037* (2021).

[84] Jiageng Mao et al., « Voxel transformer for 3d object detection », *in*: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 3164–3173.

[85] Antoine Mauri et al., « Deep Learning for Real-Time 3D Multi-Object Detection, Localisation, and Tracking: Application to Smart Mobility », *in*: *Sensors* 20.*2* (2020), p. 532.

[86] Ishan Misra, Rohit Girdhar, and Armand Joulin, « An End-to-End Transformer Model for 3D Object Detection », *in*: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 2906–2917.

[87] Xuran Pan et al., « 3d object detection with pointformer », *in*: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 7463–7472.

[88] Chao Peng et al., « Megdet: A large mini-batch object detector », *in*: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2018, pp. 6181–6189.

[89] Tung Phan-Minh et al., « Covernet: Multimodal behavior prediction using trajectory sets », *in*: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 14074–14083.

[90] Gilles Puy, Alexandre Boulch, and Renaud Marlet, « Flot: Scene flow on point clouds guided by optimal transport », *in*: *European conference on computer vision*, Springer, 2020, pp. 527–544.

[91] Charles R Qi et al., « Frustum pointnets for 3d object detection from rgb-d data », *in*: *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 918–927.

[92] Charles Ruizhongtai Qi et al., « Pointnet++: Deep hierarchical feature learning on point sets in a metric space », *in*: *Advances in neural information processing systems* 30 (2017).

[93] Maithra Raghu et al., « Do Vision Transformers See Like Convolutional Neural Networks? », *in*: *Advances in Neural Information Processing Systems* 34 (2021).

[94] Joseph Redmon et al., « You only look once: Unified, real-time object detection », *in*: *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[95] Shaoqing Ren et al., « Faster r-cnn: Towards real-time object detection with region proposal networks », *in*: *Advances in neural information processing systems* 28 (2015).

[96] Shaoqing Ren et al., « Faster r-cnn: Towards real-time object detection with region proposal networks », *in*: *IEEE transactions on pattern analysis and machine intelligence* 39.*6* (2016), pp. 1137–1149.

[97] Samuel Scheidegger et al., « Mono-camera 3d multi-object tracking using deep learning detections and pmbm filtering », *in*: *2018 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, 2018, pp. 433–440.

[98] Matthew Schwall et al., « Waymo public road safety performance data », *in*: *arXiv preprint arXiv:2011.00038* (2020).

[99] Hualian Sheng et al., « Improving 3d object detection with channel-wise transformer », *in*: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 2743–2752.

[100] 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*, 2019, pp. 770–779.

[101] Shaoshuai Shi et al., « From points to parts: 3d object detection from point cloud with part-aware and part-aggregation network », *in*: *IEEE transactions on pattern analysis and machine intelligence* 43.*8* (2020), pp. 2647–2664.

[102] Shaoshuai Shi et al., « Pv-rcnn: Point-voxel feature set abstraction for 3d object detection », *in*: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 10529–10538.

[103] Leslie N Smith, « A disciplined approach to neural network hyper-parameters: Part 1–learning rate, batch size, momentum, and weight decay », *in*: *arXiv preprint arXiv:1803.09820* (2018).

[104] Leslie N Smith and Nicholay Topin, « Super-convergence: Very fast training of neural networks using large learning rates », *in*: *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, vol. 11006, International Society for Optics and Photonics, 2019, p. 1100612.

[105] Haixin Sun, Minh-Quan Dao, and Vincent Fremont, « 3D-FlowNet: Event-based optical flow estimation with 3D representation », *in*: *2022 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, 2022, pp. 1845–1850.

[106] Pei Sun et al., « Scalability in perception for autonomous driving: Waymo open dataset », *in*: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 2446–2454.

[107] OpenPCDet Development Team, *OpenPCDet: An Open-source Toolbox for 3D Object Detection from Point Clouds*, `https://github.com/open-mmlab/OpenPCDet`, 2020.

[108] Zhi Tian et al., « Fcos: Fully convolutional one-stage object detection », *in*: *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 9627–9636.

[109] Darren Tsai et al., « MS3D: Leveraging Multiple Detectors for Unsupervised Domain Adaptation in 3D Object Detection », *in*: *arXiv preprint arXiv:2304.02431* (2023).

[110] Balakrishnan Varadarajan et al., « Multipath++: Efficient information fusion and trajectory aggregation for behavior prediction », *in*: *2022 International Conference on Robotics and Automation (ICRA)*, IEEE, 2022, pp. 7814–7821.

[111] Ashish Vaswani et al., « Attention is all you need », *in*: *Advances in neural information processing systems* 30 (2017).

[112] Kyle Vedder and Eric Eaton, « Sparse PointPillars: Maintaining and Exploiting Input Sparsity to Improve Runtime on Embedded Systems », *in*: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2022, pp. 2025–2031.

[113] Chunwei Wang et al., « Pointaugmenting: Cross-modal augmentation for 3d object detection », *in*: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 11794–11803.

[114] Jun Wang et al., « Infofocus: 3d object detection for autonomous driving with dynamic information modeling », *in*: *European Conference on Computer Vision*, Springer, 2020, pp. 405–420.

[115] Tianyu Wang et al., « Sparse2Dense: Learning to densify 3d features for 3d object detection », *in*: *Advances in Neural Information Processing Systems* 35 (2022), pp. 38533–38545.

[116] Tsun-Hsuan Wang et al., « V2vnet: Vehicle-to-vehicle communication for joint perception and prediction », *in*: *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, Springer, 2020, pp. 605–621.

[117] Yi Wei et al., « LiDAR distillation: bridging the beam-induced domain Gap for 3D object detection », *in*: *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXIX*, Springer, 2022, pp. 179–195.

[118] Xinshuo Weng and Kris Kitani, « Monocular 3d object detection with pseudo-lidar point cloud », *in*: *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, 2019, pp. 0–0.

[119] Xinshuo Weng et al., « AB3DMOT: A Baseline for 3D Multi-Object Tracking and New Evaluation Metrics », *in*: *arXiv preprint arXiv:2008.08063* (2020).

[120] Xinshuo Weng et al., « Gnn3dmot: Graph neural network for 3d multi-object tracking with 2d-3d multi-feature learning », *in*: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 6499–6508.

[121] Wenxuan Wu et al., « Pointpwc-net: Cost volume on point clouds for (self-) supervised scene flow estimation », *in*: *European conference on computer vision*, Springer, 2020, pp. 88–107.

[122] Xiaopei Wu et al., « Sparse fuse dense: Towards high quality 3d detection with depth completion », *in*: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 5418–5427.

[123] Xin Xia et al., « Advancing estimation accuracy of sideslip angle by fusing vehicle kinematics and dynamics information with fuzzy logic », *in*: *IEEE Transactions on Vehicular Technology* 70.*7* (2021), pp. 6577–6590.

[124] Jianyun Xu et al., « INT: Towards Infinite-Frames 3D Detection with an Efficient Framework », *in*: *European Conference on Computer Vision*, Springer, 2022, pp. 193–209.

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

[126] Runsheng Xu et al., « Bridging the domain gap for multi-agent perception », *in*: *arXiv preprint arXiv:2210.08451* (2022).

[127] Runsheng Xu et al., « CoBEVT: Cooperative bird's eye view semantic segmentation with sparse transformers », *in*: *arXiv preprint arXiv:2207.02202* (2022).

[128] Runsheng Xu et al., « V2X-ViT: Vehicle-to-everything cooperative perception with vision transformer », *in*: *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXIX*, Springer, 2022, pp. 107–124.

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

[130] Bin Yang, Ming Liang, and Raquel Urtasun, « Hdnet: Exploiting hd maps for 3d object detection », *in*: *Conference on Robot Learning*, PMLR, 2018, pp. 146–155.

[131] Bin Yang, Wenjie Luo, and Raquel Urtasun, « Pixor: Real-time 3d object detection from point clouds », *in*: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2018, pp. 7652–7660.

[132] Honghong Yang et al., « An efficient edge artificial intelligence multipedestrian tracking method with rank constraint », *in*: *IEEE Transactions on Industrial Informatics* 15.7 (2019), pp. 4178–4188.

[133] Zetong Yang et al., « 3d-man: 3d multi-frame attention network for object detection », *in*: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 1863–1872.

[134] Zetong Yang et al., « 3dssd: Point-based 3d single stage object detector », *in*: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 11040–11048.

[135] Zetong Yang et al., « Std: Sparse-to-dense 3d object detector for point cloud », *in*: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1951–1960.

[136] Tianwei Yin, Xingyi Zhou, and Philipp Krähenbühl, « Center-based 3d object detection and tracking », *in*: *arXiv preprint arXiv:2006.11275* (2020).

[137] Tianwei Yin, Xingyi Zhou, and Philipp Krähenbühl, « Center-based 3d object detection and tracking », *in*: *arXiv preprint arXiv:2006.11275* (2020).

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

[139] Yurong You et al., « Learning to detect mobile objects from lidar scans without labels », *in*: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 1130–1140.

[140] Haibao Yu et al., « Dair-v2x: A large-scale dataset for vehicle-infrastructure cooperative 3d object detection », *in*: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 21361–21370.

[141] Y Yuan and M Sester, « COMAP: A synthetic dataset for collective multi-agent perception of autonomous driving », *in*: *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 43 (2021), pp. 255–263.

[142] Yunshuang Yuan, Hao Cheng, and Monika Sester, « Keypoints-based deep feature fusion for cooperative vehicle detection of autonomous driving », *in*: *IEEE Robotics and Automation Letters* 7.2 (2022), pp. 3054–3061.

[143] Matthew D Zeiler and Rob Fergus, « Visualizing and understanding convolutional networks », *in*: *European conference on computer vision*, Springer, 2014, pp. 818–833.

[144] Lunjun Zhang et al., « Towards Unsupervised Object Detection From LiDAR Point Clouds », *in*: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 9317–9328.

[145] Xiao Zhang et al., « Efficient L-shape fitting for vehicle detection using laser scanners », *in*: *2017 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, 2017, pp. 54–59.

[146] Hengshuang Zhao, Jiaya Jia, and Vladlen Koltun, « Exploring self-attention for image recognition », *in*: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 10076–10085.

[147] Hengshuang Zhao et al., « Point transformer », *in*: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 16259–16268.

[148] Dingfu Zhou et al., « Iou loss for 2d/3d object detection », *in*: *2019 International Conference on 3D Vision (3DV)*, IEEE, 2019, pp. 85–94.

[149] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl, *Objects as Points*, 2019, arXiv: `1904.07850 [cs.CV]`.

[150] 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*, 2018, pp. 4490–4499.

[151] Benjin Zhu et al., « Class-balanced grouping and sampling for point cloud 3d object detection », *in*: *arXiv preprint arXiv:1908.09492* (2019).

[152] Xizhou Zhu et al., « Deformable detr: Deformable transformers for end-to-end object detection », *in*: *arXiv preprint arXiv:2010.04159* (2020).

COLLEGE SCIENCES
DOCTORAL DE L'INGENIERIE
PAYS DE LA LOIRE ET DES SYSTEMES

CENTRALE NANTES

Titre : Résoudre l'occlusion et la faible densité à distance des nuages de points pour la détection d'objets 3D par perception collaborative

Mots-clés : perception collaborative, V2X, détection objets en 3D, apprentissage profond

Résumé : La détection précise d'objets 3D est un enjeu majeur pour l'intégration sécurisée des véhicules autonomes dans le trafic routier. Le LiDAR, offrant des mesures de profondeur précises et relativement denses, est très présent dans les bases de données de référence. Cependant, ses nuages de points sont clairsemés à longue distance et soumis aux occlusions. L'état de l'art propose alors des techniques de suréchantillonnage, par fusion avec des caméras ou par distillation des connaissances, afin d'obtenir de bonnes détections. La première méthode permet de reconstruire la profondeur des pixels afin de générer des points supplémentaires tandis que la seconde vise à obtenir des nuages de points imitant ceux sans occlusion ni dispersion. Comme ces approches utilisent des mesures obtenues par le véhicule ego à chaque pas de temps, la détection est inévitablement affectée par ces régions inobservables. Conscient des limites des méthodes à perspective unique, ces travaux de thèse s'efforcent à résoudre les problèmes d'occlusion et de rareté en exploitant des perspectives multiples. Notre approche exploite d'une part les mesures du véhicule ego par séquences : au cours de son déplacement dans le temps. D'autre part, nous proposons une perception collaborative basée sur la fusion des informations obtenues par de multiples agents connectés.

Title: Toward Solving Occlusion and Sparsity in Deep Learning-Based 3D Object Detection through Collaborative Perception

Keywords: collaborative perception, V2X, 3D object detection, deep learning, LiDAR

Abstract: Detecting objects at a high precision in 3D is critical for the safety of autonomous vehicles. LiDAR measurements, presented as point clouds frequently suffer from occlusion and sparsity. Prior works address these challenges by upsampling (i) point clouds via fusion with RGB cameras or (ii) their representations via knowledge distillation. Because these approaches are designed to use point clouds obtained only by the ego vehicle at a single timestep, they are inevitably affected by unobservable regions caused by occlusion and sparsity. Aware of the limitations of single-perspective methods, this thesis strives to resolve occlusion and sparsity by leveraging point clouds obtained from multiple perspectives to the fullest. The core of our approach is made of two components that are respectively built on the utilization of point cloud sequences and collaborative perception via V2X communication. While the former is about using information obtained by the ego vehicle during its motion through time, the latter is based on the fusion of information obtained by multiple connected agents scattering over space.