Technische Universität München
TUM School of Computation, Information and Technology

# Temporal Data Processing for 3D Object Detection in Autonomous Driving

**Emeç Erçelik**

Vollständiger Abdruck der von der TUM School of Computation, Information and Technology der Technischen Universität München zur Erlangung eines

**Doktor der Naturwissenschaften (Dr. rer. nat.)**

genehmigten Dissertation.

**Vorsitz:** Prof. Dr. Stephan Günnemann

**Prüfer*innen der Dissertation:**

1. Prof. Dr.-Ing. habil. Alois Christian Knoll
2. Assoc. Prof. Dr. Nazım Kemal Üre

Die Dissertation wurde am 07.06.2022 bei der Technischen Universität München eingereicht und durch die TUM School of Computation, Information and Technology am 04.11.2022 angenommen.

## Acknowledgement

I want to thank my supervisor, Professor Alois Knoll, for giving me the opportunity to work on such an exciting topic. His trust and support of my work have been a great source of motivation.

I would like to thank Alexander Lenz for his support, fruitful discussions on science and life, and being always there when I needed help. Amy Bücherl and Ute Lomp have also been great supporters, especially for the organizational matters at the chair, and I want to thank them.

I want to thank all my colleagues at the Chair of Robotics, Artificial Intelligence, and Real-time Systems. It was always lovely and inspiring to talk to them. During this work, Esra İçer and Burcu Karadeniz have become my great supporters, friends, mentors, and family by all means. Discussions with Okan Köpüklü have always been inspiring, and I am glad to have met him during my PhD. Sina Shafaei and Morteza Hashemi have been there from the first day I started at the chair. They have been great mentors and friends. Working with Etienne Müller, Daniel Auge, Christoph Segler, and Bare Luka Zagar was full of fun and made things more straightforward than they were. I want to express my sincere gratitude to all of them for making my work easier. I also want to thank all of my thesis students, especially Ahmet Burakhan Koyuncu, Maria Dreher, and Mingyu Liu. I enjoyed working with all of them a lot.

I want to thank Johannes Handloser for being a supportive colleague during my industrial collaboration experience. I also want to express my gratitude to Sladjana Martens and Noah Klarmann. Being in a team with them was a great chance.

I want to thank Ekim Yurtsever for his outstanding mentorship and collaboration. My PhD experience started before this work at the Neuroscientific System Theory group of TUM. I want to thank all my colleagues at NST and especially Cristian Axenie, Florian Mirus, Ziyed Tayeb, Prof. Jörg Conradt, Nikolai Waniek, Lukas Everding, and Christoph Richter. In addition, Simmag group at the Istanbul Technical University has been my default supporter for years. I want to thank my lifelong mentor Prof. Neslihan Serap Şengör. Without her help, support, and guidance, I would not be able to reach where I am. I also want to thank Berat Denizdurduran. He has always supported me and became an elder brother in my scientific life. Hasan Özdemirci is another valuable member of Simmag team who has helped me a lot.

My family has been the source of motivation for my entire academic life. I want to thank my mothers, Naciye and Nimet, my fathers, Mustafa and İbrahim, my brother Uğur, and my sisters Eda and Leyla. I want to thank my little daughter, Ece, for being the best gift while reaching my PhD work's end. Finally, I would like to thank my wife, Büşra, for helping me realize my dreams and make good decisions, and sacrificing part of her life for my academic life. This work would not be possible without her.

Garching, June 07, 2022 *Emeç Erçelik*

## Abstract

Perception is a vital task in enabling autonomous driving function, as it is crucial for understanding the context, making safe decisions, and planning. As one of the 3D perception problems, 3D object detection aims to detect objects in the environment with their positions, sizes, and orientations. Recently, state-of-the-art 3D object detection methods mostly rely on either lidar point clouds or a combination of lidar point clouds with other sensory data such as camera images and radar point clouds. Current deep 3D detector networks mainly focus on learning from single time step data ignoring the previously-processed sequential data and preceding estimations. However, preceding time steps are important for extending sensors' point of view. Further, lidar point clouds are too sparse for far-away objects and lidar sensors can see objects from a single perspective leading to an incomplete perception of the shape of objects. Moreover, dynamic objects may create occlusions and the resulting loss of information leads to the missing of previously-detected objects.

The use of sequential data can help enhance object feature quality obtained from single-frame feature extractors at successive time steps. Sequential frames allow for changes in the angle of sight of the ego vehicle as it moves, which provides an opportunity for the extractor to complete missing parts of objects. While sequential data processing has been intensely investigated for 2D video object detection and action recognition problems, its use for 3D object detection remains understudied.

In this work, we propose three approaches to improve 3D object detection results using the sequential data. First, we aggregate scene-level feature maps of multiple frames to obtain a richer representation of the entire scene. This method helps pinpoint contextual changes between consecutive frames and compensates for loss of information. Next, as an alternative to the scene-level approach, we present an object-level method, where temporal object representations are constructed using regional object features from preceding time steps. This method helps remedy bounding box flickers occurring in consecutive frames. Finally, for pre-training 3D object detection networks, we adopt scene flow, an inherently-temporal perception task. Self-supervised scene flow training provides motion-aware backbone initialisation for the 3D detection network using unlabelled data. In this way, the motion pattern-awareness provided by the scene flow training helps the 3D detector to distinguish objects from other objects and the background. Our experiments on different datasets with several 3D detectors indicate that sequential data processing does improve the representation ability of 3D detection networks, which leads to enhanced 3D detection accuracy.

## Zusammenfassung

Die Umfeldwahrnehmung ist ein wichtiger Teil des autonomen Fahrens, da sie entscheidend für das Verstehen des Kontexts, das Treffen sicherer Entscheidungen und die Planung ist.

Als eines der 3D-Wahrnehmungsprobleme zielt die 3D-Objekterkennung darauf ab, Objekte in der Umgebung mit ihren Positionen, Größen und Ausrichtungen zu erkennen. Aktuelle 3D-Objekterkennungsmethoden basieren meist auf Lidar Punktwolken oder einer Kombination von Lidar Punktwolken mit anderen Sensordaten, wie Kamerabildern und Radar-Punktwolken. Heutzutage konzentrieren sich die tiefe 3D-Detektornetzwerke hauptsächlich auf das Lernen von Daten aus einem einzigen Zeitschritt und ignorieren die zuvor verarbeiteten sequentiellen Daten und vorherigen Schätzungen. Die vorherige Zeitschritte sind jedoch wichtig, um den Blickwinkel der Sensoren zu erweitern. Außerdem sind Lidar Punktwolken für weit entfernte Objekte zu spärlich. Lidar Sensoren können Objekte nur aus einer einzigen Perspektive sehen, was zu einer unvollständigen Wahrnehmung der Form von Punktwolken führt. Darüber hinaus können dynamische Objekte Verdeckungen verursachen. Der daraus resultierende Informationsverlust führt dazu, dass zuvor erkannte Objekte in den nachfolgenden Zeitschritten nicht erkannt werden.

Die Verwendung sequenzieller Daten ermöglicht es, die Qualität von Objektmerkmalen zu verbessern, die aus Einzelbild-Merkmalsextraktoren in aufeinanderfolgenden Zeitschritten extrahiert werden. Die Verwendung von sequenziellen Einzelbildern ermöglicht Änderungen des Sichtwinkels des Ego-Fahrzeugs, während es sich bewegt, was dem Extraktor die Möglichkeit gibt, fehlende Teile von Objekten zu ergänzen. Während die sequenzielle Datenverarbeitung für die 2D-Video-Objekterkennung und die Erkennung von Handlungen intensiv untersucht wurde, ist ihre Verwendung für die 3D-Objekterkennung noch wenig erforscht.

In dieser Arbeit werden drei Ansätze aufgeführt, um die 3D-Objekterkennung mit Hilfe von sequenziellen Daten zu verbessern. Um eine umfassendere Darstellung der gesamten Szene zu erhalten, werden zuerst Merkmale auf Szenenebene von mehreren Einzelbildern zusammengefasst. Diese Methode hilft kontextuelle Änderungen zwischen aufeinanderfolgenden Bildern zu erkennen und Informationsverluste zu kompensieren. Als Nächstes stellen wir als Alternative zum Ansatz auf Szenenebene eine Methode auf Objektebene vor, bei der zeitliche Objektdarstellungen unter Verwendung regionaler Objektmerkmale aus vorangegangenen Zeitschritten konstruiert werden. Diese Methode hilft, das Flackern der 3D-Boxen in aufeinanderfolgenden Bildern zu beheben. Für das Vortraining von 3D-Objekterkennungsnetzwerken verwenden wir einschließlich den Szenenfluss, der eine inhärent zeitliche Wahrnehmungsaufgabe repräsentiert. Das selbstüberwachte Training des Szenenflusses bietet eine bewegungsbewusste Initialisierung des 3D-Erkennungsnetzes unter Verwendung unmarkierter Daten. Auf diese Weise hilft das Bewegungsmusterbewusstsein, das durch das Training des Szenenflusses bereitgestellt wird, dem 3D-Detektor, Objekte von anderen Objekten und dem Hintergrund zu unterscheiden. Die Experimente an verschiedenen Datensätzen mit mehreren 3D-Detektoren zeigen, dass die sequenzielle Datenverarbeitung die Darstellungsfähigkeit von 3D-Detektionsnetzwerken verbessert, was zu einer höheren 3D-Detektionsgenauigkeit führt.

# Contents

Chapter 1

# Introduction

Autonomous vehicles rely highly on perception mechanisms to understand the environment and make safe decisions. Accurate perception is especially vital due to the dynamic nature of the traffic. Advancements in computational resources, deep learning-based computer vision methods, and availability of public datasets have accelerated data-driven computer vision research for the perception of objects in the traffic [Zam+21]. Multiple perception tasks, such as object detection, segmentation, depth estimation, and tracking, are considered for self-driving cars [Guo+21]. Further, several sensory systems, including camera, lidar, radar, GPS , and IMU , have been used extensively for achieving good perception results [Cui+21a]. Even though perception models have improved greatly over the years [GLU12a], available models are still far-away from being perfect.

Object detection aims to detect objects in the visible field of sensors with their positions, sizes, and classes [Jan+20]. 2D object detection mainly positions objects on the image plane using single or multi-view images [Ren+15; Lin+17b; He+17; Liu+16; Red+16]. However, 2D object localisation results lack the 3D depth understanding as well as realistic object shapes, both of which are important for planning the future movements of the ego vehicle. Hence, one of the aims of 3D object detection is to accurately position objects in the 3D space. 3D object detection methods use different sensory data, such as monocular or stereo camera images and lidar point clouds. Monocular image-based [Liu+21b; Par+21; Shi+21c; Lu+21b] and stereo image-based [Wan+21h; RVK22; Che+20c] methods have preserved their popularity since cameras are cheaper than lidar sensors, and handling camera images is more straightforward than processing unstructured point cloud data. However, lidar point cloud-based methods [ZT18; Qi+18; Zhe+21c; Den+20; He+20a] provide superior detection accuracy thanks to lidar's accurate depth measurements. Recently, several studies have also investigated the use of pseudo-point clouds for bridging the gap between the two sensor data processing methods [Wan+19; Par+21]. The sparsity of lidar point clouds [Qi+18; Zhe+21c; Den+20; He+20a], requirement of high computation power [Lan+19], and the processing of unordered and unstructured point clouds [ZT18; Qi+17a; Qi+17b] pose substantial challenges for 3D object detection using lidar point clouds, which hinder obtaining perfect detection.

In recent years, lidar-based 3D object detection methods have achieved considerable enhancements in 3D detection performance [GLU12a; Cae+20; Sun+20]. Most state-of-the-art 3D detectors rely on single-frames of lidar data while estimating object bounding boxes [YZK21; Shi+20a; Zhe+21c; PMR20]. Even though results are encouraging, relying on single frames brings several challenges. First, traffic scenarios are dynamic and objects occlude each other. Therefore, some objects in the visible field will be missed by the lidar sensor for a time interval, which can cause mistakes in decisions. Second, quality of a lidar point cloud that is reflected from an object can vary in time due to the sparsity of lidar data. This can result in changing object representations between sequential frames and instigate flickers. Further, the sparsity of lidar point clouds might result in the missing of far-away objects, which are represented mostly with a few lidar points. This is a major reason behind hav-

ing multiple difficulty levels for measuring the performance of these detection methods in public benchmarks [GLU12a; Cae+20; Sun+20]. Combined, these limitations become more significant in dense traffic scenarios due to the limited angle of sight of the ego vehicle.

Using data from multiple consecutive frames has been shown to be effective for some image-based computer vision tasks such as 2D object detection, multi-object tracking, and action recognition. For 2D RGB object detection, object features [Kan+17a; Den+19; SLB19; Wu+19; SN16; LLT17] as well as scene feature maps [LZ18; BTS18; Zhu+18; Nin+17; Kim+21] from consecutive frames are processed temporally to obtain better-quality detection results. Recognising actions depending on single frames remains challenging, since multiple actions can share the same scene. However, the course of action through successive frames can improve the identification of actions. Therefore, many action recognition methods also process RGB video data to make reliable action estimations [CZ17; FPZ16; Wan+16; Xie+18; ZSB18; KWR19; Jia+19; Yan+20a]. 2D multi-object tracking methods mainly aim to match objects in two successive frames [Bew+16; WBP17]. On the other hand, using more than two frames helps capture objects missed in intermediate frames, which leads to better tracking results [Kan+17b; Kra+21; Pan+20].

Video object detection and perception methods accompany multiple inherent issues such as multi-frame feature alignment, motion blur, and occlusion. However, several solutions are proposed to these thanks to well-studied deep learning techniques for image processing [Jia+21b]. Compared to 2D perception tasks, multi-frame 3D object detection remains as an understudied field, not only due to the difficulty in temporally fusing unstructured point cloud features but also due to a lack of data availability. However, the latter point has now been addressed thanks to datasets such as nuScenes [Cae+20] and Waymo [Sun+20], and KITTI tracking [GLU12a] have recently made it possible to study temporal processing of lidar point clouds. Further reasons behind the increasing attention in this field will be detailed later on in Section 3.3. Despite increasing attention in this area, multiple issues remain unaddressed: (i) handling misaligned feature maps from consecutive frames in dynamic traffic scenarios and (ii) temporal processing of successive point clouds. To address these gaps, in this work, we investigate approaches to processing point cloud features from sequential frames for obtaining enhanced 3D object detection quality in autonomous driving cases.

3D object detection is mostly a data-driven problem, where annotated, large datasets are required [Shi+21b; YZK21; Car+20]. However, obtaining annotations for lidar data is challenging and costly [Sun+20; Cae+20; Dai+17]. To address this, unsupervised and self-supervised point cloud processing methods have been investigated to learn point representations without needing expert annotations [Che+21; WLN21; MOH20]. Despite being a self-supervised learning method, the use of the inherent structure of the sequential point clouds and learned motion representations have not yet been considered for 3D object detection. The scene flow problem aims to estimate motion vectors of individual points in a scene using structural differences between two successive point clouds [LQG19; Wan+20f; Bau+21]. Such a network training construct motion-aware representations in the network. Making use of the learned motion-based point cloud representations between consecutive point clouds would help distinguishing objects in the environment, and such pre-training for 3D object detection would provide better 3D bounding box quality. This would also help decrease the need for annotated, large datasets.

## 1.1 Motivation and Research Questions

In this work, we aim to answer three main research questions (RQs) related to the use of sequential point clouds for the benefit of 3D object detection.

In doing so, we first investigate temporal fusion of single-frame 3D feature maps of sequential point clouds. Unstructured point clouds can be transformed into a structured shape using grid-based projection methods, which allows using convolutional neural networks (CNNs) for point cloud processing and outputs 3D feature maps from single frames. However, two consecutive feature maps would have regional shifts causing misalignment. Related to this, we aim to answer the following research question:

**Research Question 1** (Scene-level multi-frame fusion) *How can we process consecutive scene feature maps to improve 3D object detection quality?*

Second, we investigate the fusion of object features from sequential frames to improve 3D object detection. The feature map of an entire scene represents all objects from the scene including the background as well as objects from the regions of interest, which can cause noise during fusion. In a different direction from the RQ1, we shape our second research question as follows:

**Research Question 2** (Object-level multi-frame fusion) *How can we temporally process objects' point cloud features from sequential frames to improve 3D object detection quality?*

Third, we investigate self-supervised scene flow pre-training for improving 3D object detection. Scene flow helps learning motion-based representations from sequential point clouds. The learned motion-related features can be also useful for 3D object detection networks, which would make object representations more easily distinguished from the background. Thus, we form our third research question as follows:

**Research Question 3** (Self-supervised scene flow pre-training) *How can we make use of point cloud representations learned through an inherently-temporal perception task to improve 3D object detection quality?*

## 1.2 Thesis Outline

This thesis consists of eight chapters. This first chapter provided an introduction to the thesis and detailed the main research questions of interest. Chapter 2 introduces the main concepts related to 3D object detection as well as the scene flow tasks. This chapter also introduces sensory data used commonly for autonomous driving perception tasks in addition to the datasets and the metrics that we used in this thesis. It also presents common 3D object detection network structures, point cloud processing methods, and bounding box representations. Next, Chapter 3 presents literature related to 3D object detection and scene flow tasks. This chapter summarises the state-of-the-art single-frame and multi-frame 2D and 3D object detection and tracking approaches, perception methods that make use of sequential data, and point cloud processing methods published at the time of this thesis.

Chapter 4 defines the general problem definition discussed in Chapters 5, 6, and 7. Chapters 5, 6, and 7 provide the main contribution of this thesis related to the research questions and hypotheses detailed in Chapter 4. Briefly, Chapter 5 explains the scene-level multi-frame point cloud approach adopted in this thesis to improve 3D object detection accuracy using feature maps from sequential point clouds. Chapter 6 explains the object-level multi-frame point cloud processing method through which enhanced 3D object detection results are obtained by combining features of the same object from consecutive time steps. Chapter 7 considers the sequential point cloud processing idea from a different perspective to learn inherently-

available point cloud features between two consecutive frames in a self-supervised way. This approach provides better point cloud representations for 3D object detection than only using supervised learning. Finally, Chapter 8 summarises the multi-frame point cloud approaches adopted in this thesis for 3D object detection and discusses thesis findings in context of its limitations areas for future work.

## 1.3 Thesis Contributions

This work contributes to the problem of processing sequential lidar point clouds for enhanced 3D object detection quality by answering research questions given in Section 1.1 through Chapters 5, 6, and 7. Chapter 5 discusses how sequential point cloud feature maps generated by CNNs can be used temporally to obtain 3D detection results. We propose that the receptive fields of convolutional recurrent neural networks can align successive point cloud features while also aggregating them in time. Our experimental results show that using temporal point cloud features improves 3D bounding box estimation accuracy. The temporal fusion of RGB image features further boosts the performance using spatial alignment with point cloud features through 3D anchor boxes.

Even though aggregating scene-level feature maps in time provides performance increases, alignment issues that limit further improvements remain. Therefore, Chapter 6 discusses how temporal point cloud feature fusion can be done at the object level. We hypothesise that object-level features can be accumulated in time and this helps obtaining a richer representation for the object of interest. Our detailed experiments in Chapter 6 indicate that features of the same object from sequential frames can be processed with recurrent layers and the resulting temporal object-level features yield better 3D detection quality. To the best of our knowledge, our work was the first study for object-level temporal fusion of point cloud features for the 3D object detection.

Processing sequential point clouds from previous frames aims to strengthen the features of the current frame, which are then used for 3D perception. Therefore, in Chapter 7 we hypothesise that the motion features obtained from two sequential point clouds can provide a good initial representation for 3D object detection. This relationship between perception and motion helps distinguish objects more clearly from the background and other objects. Our experiments support our idea and show that the scene flow-based pre-training helps obtaining better 3D detection quality.

This work has been partly published in international peer-reviewed journal and conferences.

- Emeç Erçelik, Ekim Yurtsever, Mingyu Liu, Zhijie Yang, Hanzhen Zhang, Maximilian Listl, Pınar Topçam, Yılmaz Kaan Çaylı, Alois Knoll. 3D Object Detection with a Self-supervised Lidar Scene Flow Backbone. European Conference on Computer Vision. Springer, Cham, 2022. [Erç+22]

- Emeç Erçelik, Ekim Yurtsever, Alois Knoll. Temp-Frustum Net: 3D Object Detection with Temporal Fusion. 2021 IEEE Intelligent Vehicles Symposium (IV), 2021 [EYK21b]

- Emeç Erçelik, Ekim Yurtsever, Alois Knoll. 3D Object Detection with Multi-Frame RGB-Lidar Feature Alignment. IEEE Access, 2021, 1-1 [EYK21a]

Papers [EYK21a] and [EYK21b] form the basis of Chapter 6. In these works, we present our object-level temporal feature fusion method for 3D object detection using RNNs. The

quality of object representations in sequential frames varies depending on the visibility of objects to the sensors. We demonstrate that 3D detection accuracy increases by fusing features of the same objects from sequential frames, which compensates for the loss of information due to the limited angle of sight. Chapter 7 is partly published as [Erç+22] in 2022. In [Erç+22], we show that the self-supervised scene flow task can be used to pre-train 3D object detection networks, which helps learning representative motion features. This pre-training based on sequential lidar point clouds improves 3D object detection quality obtained by the supervised training on top of scene flow pre-training.

The following publications are not directly part of this thesis. However, the results have been obtained in the course of this work and these publications investigate similar ideas presented in this thesis.

- Ahmet Burakhan Koyuncu, Emeç Erçelik, Eduard Comulada-Simpson, Joost Venrooij, Mohsen Kaboli, Alois Knoll. A Novel Approach to Neural Network-based Motion Cueing Algorithm for a Driving Simulator. 31st IEEE Intelligent Vehicles Symposium, 2020. [Koy+]
- Maria Dreher, Emeç Erçelik, Timo Bänziger, Alois Knoll. Radar-based 2D Car Detection Using Deep Neural Networks. 23rd Intelligent Transportation Systems Conference (ITSC 2020), 2020. [Dre+20]
- Maximilian Kraus, Seyed Majid Azimi, Emeç Erçelik, Reza Bahmanyar, Peter Reinartz, Alois Knoll. AerialMPTNet: Multi-Pedestrian Tracking in Aerial Imagery Using Temporal and Graphical Features. 25th International Conference on Pattern Recognition (ICPR 2020), 2020. [Kra+21]
- Tobias Weber, Emeç Erçelik, Martin Ebert, Alois Knoll. Recognition & Evaluation of Additional Traffic Signs on the example of '80 km/h when wet'. 2019 IEEE Intelligent Transportation Systems Conference (ITSC), 2019. [Web+19]

International workshops listed below have also impacted this work:

- 1st International Workshop on Data Driven Intelligent Vehicle Applications (DDIVA) 2019. Alois Knoll, Emeç Erçelik, Esra Icer, Burcu Karadeniz, Christoph Segler, Sina Shafaei, and Julian Tatsch. Co-located with IEEE Intelligent Vehicles Symposium (IV), 2019.
- 2nd International Workshop on Data Driven Intelligent Vehicle Applications (DDIVA) 2020. Alois Knoll, Emeç Erçelik, Esra Icer, Neslihan Kose, Burcu Karadeniz, Christoph Segler, Sina Shafaei, and Julian Tatsch. Co-located with IEEE Intelligent Vehicles Symposium (IV), 2020.
- 3rd International Workshop on Data Driven Intelligent Vehicle Applications (DDIVA) 2020. Alois Knoll, Walter Zimmer, Emeç Erçelik, Esra Icer, Neslihan Kose. Co-located with IEEE Intelligent Vehicles Symposium (IV), 2021.

Chapter 2

# Background

Perception is a vital task for autonomous vehicles or assistant systems to understand the environment and make plans taking the scene and intentions of the traffic participants into account. Perception can rely on a single sensor and multiple sensors' data, among which the most common ones are the RGB camera images, lidar point clouds, and radar points [Cae+20]. Perception benchmarks such as KITTI [GLU12a], nuScenes [Cae+20], Waymo [Sun+20], and Argoverse [Cha+19] provide two or more synchronised camera images in addition to the point clouds. The nuScenes also provides radar data along with others. We first introduce the sensory data types considered for autonomous vehicles in Section 2.1. Accomplishing 3D object detection is an essential part of the perception for autonomous vehicles, and the main body of this dissertation investigates the multi-frame approaches for 3D object detection. Therefore, we explain the 3D object detection problem and the fundamental strategies for point cloud processing in Section 2.2. Scene flow problem is also essential to understand the motion characteristics of the environment, which is highly related to multi-frame processing. We give a 3D scene flow background in Section 2.3. Current state-of-the-art solutions for such perception problems rely mostly on data-driven approaches. Therefore, methods and neural network architectures depend highly on dataset characteristics. We introduce the datasets used throughout this thesis and the metrics to evaluate models in Section 2.4.

## 2.1 Sensory Data

Monocular and stereo camera images (Fig. 2.1) have been used for several 2D and 3D computer vision tasks such as segmentation [Gra+21; Huo+21], optical flow[JTS21; Chi+21; APM21], depth estimation [Lie+21; Pin+21; Tan+21; YAL21], and 3D object detection

**Figure 2.1:** Stereo images from the KITTI object detection dataset [GLU12a]. Left and right column images are from left and right cameras, respectively.

[Zho+21a; Luo+21; ZLZ21]. An RGB camera image $I$ can be defined as $I \in \mathbb{R}^{H \times W \times 3}$, where $H$ and $W$ are the height and width, respectively. 3 stands for the number of channels in the given image, each of which represents one of the *R, G, B* channels. The image consists of pixels $I = \{p_i \in \mathbb{R}^3 \mid i = 1, ..., H \times W\}$, where $i$ is the pixel index. Images are dense and structured, which can be processed with CNNs, unlike point clouds.

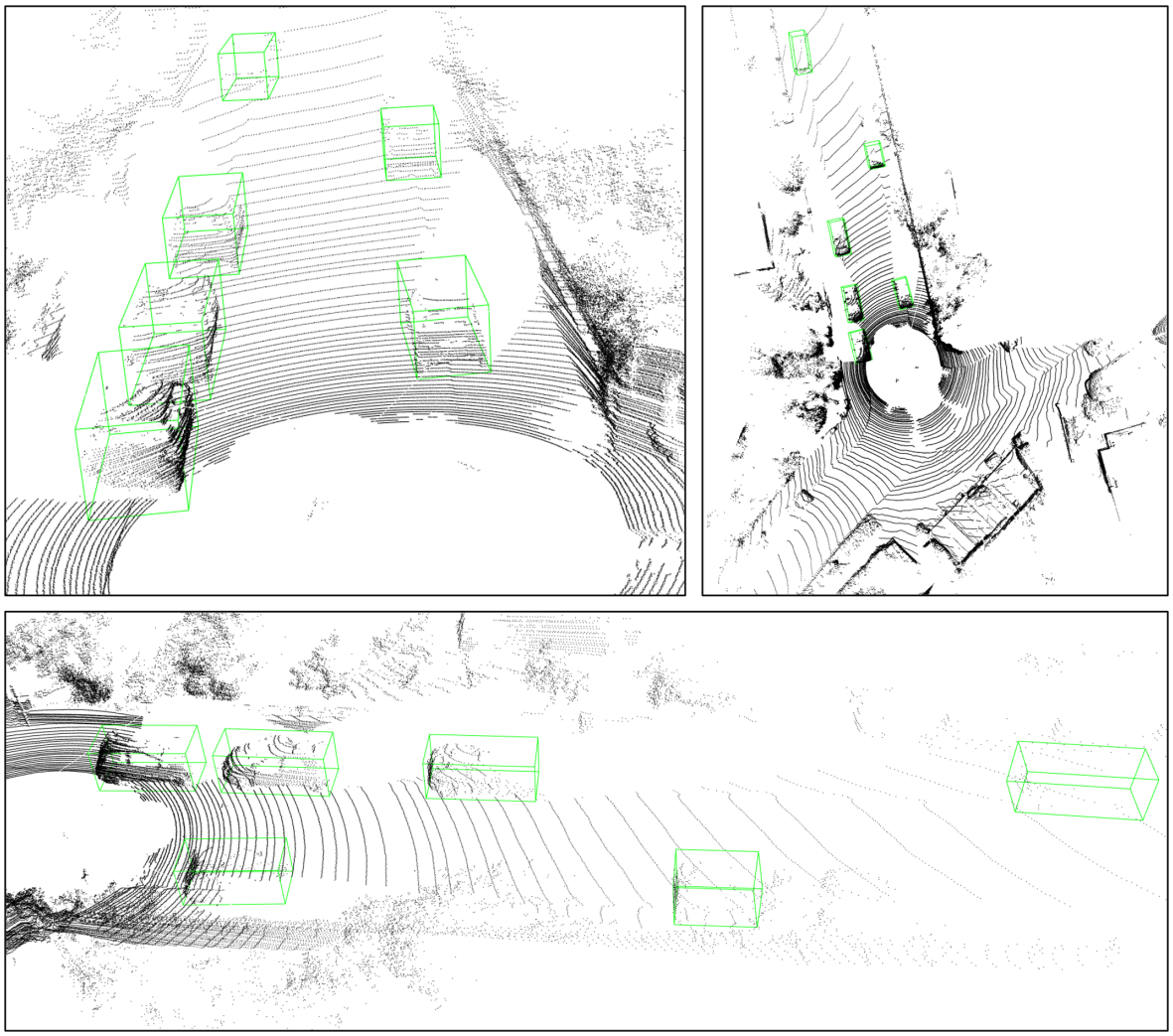


**Figure 2.2:** Lidar point cloud visualisation of the upper row shown in Fig. 2.1 from different views. The green bounding boxes indicate the visible objects in the corresponding RGB image. The lidar sensor can only see at most two sides of objects. Point clouds get sparser with the distance.

Lidar sensors output point clouds in the 3D space, which provide accurate depth information [Wan+19]. Point clouds are scattered around the 3D space sparsely. Objects can only be seen from one point of view, and reflected points originate from one or two sides of objects [YZK21]. Fig. 2.2 and 2.3 show the point cloud visualisations of upper and lower stereo images in Fig. 2.1, respectively. As seen in the figures, objects are not entirely visible to the lidar sensor, and sparsity increases with the increasing distance. A point cloud can be shown as a set of points $P = \{p_i \in \mathbb{R}^4 \mid i = 1, ..., N\}$. $p_i = (x, y, z, r)$ is a single point, and $N$ is the number of points in the point cloud. The $(x, y, z)$ is the 3D coordinates of the point, and $r$ is the reflectance value that depends on the surface properties of the objects [Hua+17]. Since lidar point clouds are unordered and do not have a structure similar to images in a local neighbourhood, it is not straightforward to process with CNNs without pre-processing.

Radar point clouds provide localisation information similar to the lidar point clouds; how-

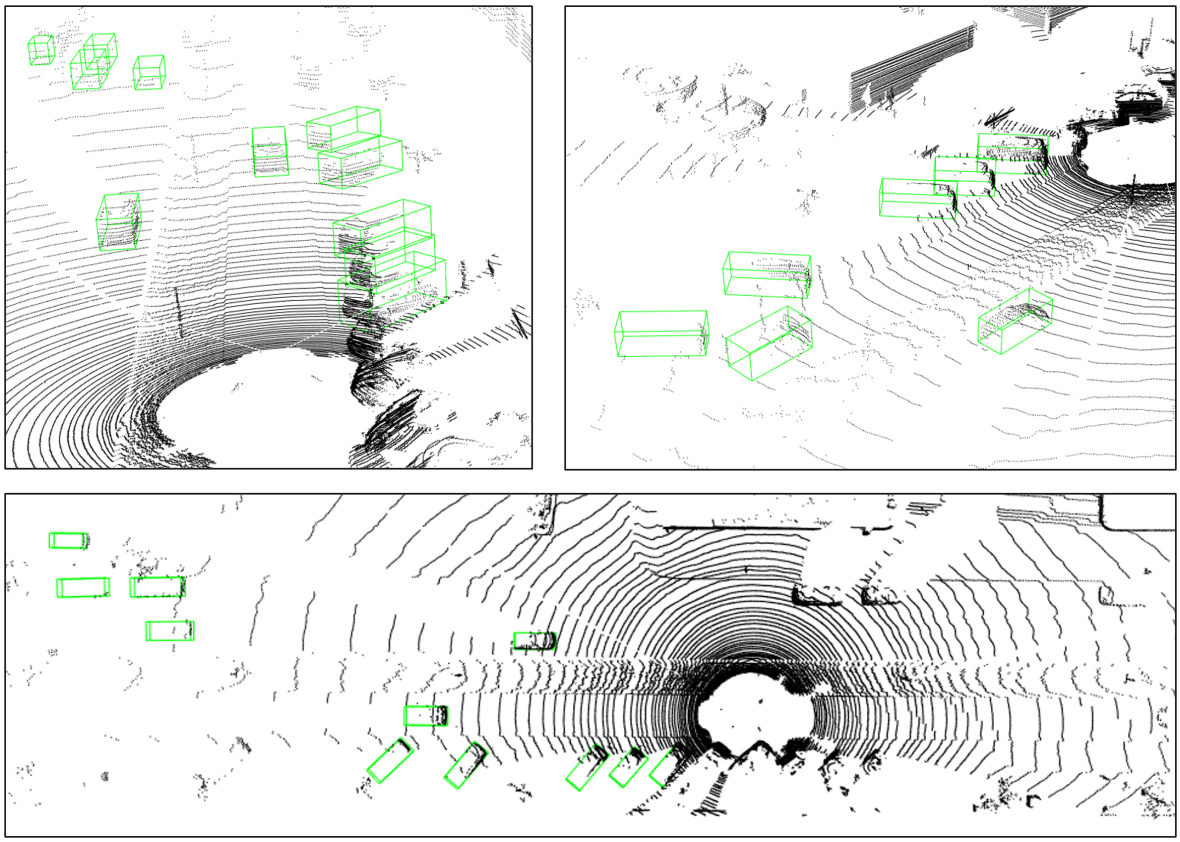


**Figure 2.3:** Lidar point cloud visualisation of the lower row shown in Fig. 2.1 from different views. Green bounding boxes are only drawn for the visible objects in the RGB image.

ever, the elevation information is less accurate due to radar's range in that direction than a lidar sensor [Cae+20]. A radar point cloud can be given as $R = r_i \in \mathbb{R}^5 \mid i = 1, ..., N$, where each point $r_i = (x, y, \sigma, v_x, v_y)$. $(x, y)$ indicates the bird's eye view (BEV) position of the radar point, $\sigma$ is the radar cross section, and $(v_x, v_y)$ is the ego-motion compensated radial velocity in $x$ and $y$ directions. Radar provides accurate positioning and velocity information; however, the sparsity of the data limits its usage for the perception of the whole environment. A comparison between lidar and radar point clouds on the image plane can be seen in Fig. 2.4. The left image shows the lidar points, and the right image shows the radar points, which are highly sparser than lidar points.

## 2.2 3D Object Detection

3D object detection task aims to localise a set of objects $O = o_i | i = 1, ..., M$, where $o \in \mathbb{R}^8$ is an object and $M$ is the number of objects visible in a given single-step data source. An object $o$ consists of a 3D bounding box represented with $(x, y, z, w, h, l, \theta)$ parameters and with an object class. $(x, y, z)$ represents the centre coordinates of the 3D bounding box, $(w, h, l)$ are the width, height, and length of the bounding box, respectively. $\theta$ shows the yaw angle of the object. 3D bounding boxes drawn around car objects with the green lines in a lidar point cloud visualisation can be seen in Fig. 2.3.

**Figure 2.4:** Lidar (left image) and radar (right image) points clouds projected onto the image plane of a scene in the nuScenes dataset [Cae+20]. The point colours indicate the depth values.

### 2.2.1 Network Architectures

Most 3D object detection networks utilise lidar point clouds or fuse them with RGB camera image features. In general, the 3D object detection problem is an extension of 2D object detection, which considers two types of architectures: one-stage [Liu+16; Red+16] and two-stage[He+17; Ren+15]. 3D object detection networks also follow similar architecture designs.

One-stage 3D detection networks mainly consist of a backbone and a detection head [Zhe+21c; Zhe+21b; GDS21; He+20a; SR20; Lan+19; Yan+20b; Wan+20b; Che+20b; Wan+21d; Du+20; Kua+20; Ge+20; NLH21; Cha+21; Liu+20]. The backbone extracts features from the given input (lidar point clouds or RGB images), and the detection head regresses the bounding box parameters by processing the input features. Even though different architectures are considered for the backbones and heads [He+20a; SR20; Lan+19], the backbone is taken as a generic feature extractor and the head consists of task-specific layers. These networks are designed to be computationally efficient compared to two-stage detectors.

Two-stage 3D detectors utilise a proposal generation stage in addition to the backbone and the detection head [Shi+20a; Den+20; Lia+20; PMR20; PMR22; Yoo+20; Yan+19; Hua+20b; Li+20; Zho+20; Shi+20b; Lia+19; Leh+19; Zhu+21; Shi+21b; Wan+21a; LWW21; Qia+21; Yan+21c; Mao+21a]. The proposal generation module takes the features extracted by the backbone and outputs object bounding box candidates. The candidate bounding boxes are rough 3D bounding box estimations with high objectness scores. The proposed 3D bounding boxes are further refined using the features from the candidate boxes' region of interest (RoI). The refinement stage corresponds to the detection head. In general, refinement stages improve the 3D detection quality [SWL19]; however, this mainly adds to the computation time.

In 2D object detection, a common approach is to refine anchor box parameters virtually scattered around the image plane [Ren+15; He+17; Liu+16; Red+16; Lin+17b]. Anchor boxes are densely placed in the image, assuming that the anchors overlap with the ground-truth bounding boxes. Features in the receptive field of an anchor box are used to refine the 2D bounding box parameters instead of regressing the actual value of the bounding box parameters. Therefore, deciding on anchor box shape requires knowledge of the object shapes in the dataset for different classes. The standard approach is utilising different anchor shapes based on average box size in a class and multiple anchors per object class to consistently represent various aspect ratios and the object scales. In a similar approach, [Liu+16; Red+16] utilise reference boxes placed in feature grids, which are class-agnostic but covers different aspect ratios and scales. There are two main drawbacks of anchor or reference boxes

[LD18]. The first is the requirement of many anchor boxes on the image, which increases the processing times and causes an imbalance in positive and negative anchor boxes since most anchors will not overlap with a ground-truth bounding box. The second is tuning many anchor box hyper-parameters, such as the number of anchors, their aspect ratios, and scales. Recently corner- [LD18] and centre-based [Dua+19] approaches have been proposed to remove the need for anchor boxes. CornerNet [LD18] predicts possible object corner positions with a heatmap and a special corner-pooling operation. CenterNet-based methods [ZWK19; Dua+19] predict the centres with heatmaps instead of corners and directly estimate sizes of 2D bounding boxes from the centre position. Therefore, the two approaches do not need tuning anchors or prior knowledge on object bounding boxes.

3D object detection networks follow similar approaches with the 2D counterparts by either using anchors at highly-selective 3D regions [Shi+20a; SR20; Qi+18; Zhe+21c; Den+20; Zhe+21b; GDS21; Yoo+20; He+20a; ZT18] or estimating object centre and sizes from the features in the vicinity of keypoints or estimated centre priors directly [Yan+20b; Che+20b; Wan+21d; YZK21]. The standard approach for the anchor-based methods is to define two perpendicular 3D bounding boxes per class around the yaw angle [SR20; Shi+20a]. In this way, anchors also set a prior knowledge about the object's heading. Another common anchor-based approach is defining one anchor box per class and splitting the yaw angle into several bins [Qi+18].

### 2.2.2 Point Cloud Encoding

Point clouds are unordered sets of points with an irregular format. Due to being unordered, point set features should not depend on the order permutation of the given points [Qi+17a]. Therefore, this data structure requires a particular feature extraction method to provide meaningful features for the complex perception tasks. Early 3D object detection approaches [Che+17; Ku+18] transformed point clouds into grid-based representations in BEV or front-view with handcrafted statistical features placed in grid projections. However, these resulted in sub-optimal detection accuracies due to the loss of information during projection and handcrafted features. Therefore, feature learning from points has become the primary feature extraction method for state-of-the-art 3D object detection networks [Qi+18; Shi+20a; YML18; YZK21]. There have been mainly three frequently-used learning-based feature extraction methods: point-based [Qi+17a], voxel-based [ZT18], and pillar-based[Lan+19].

**Point-based** feature extraction relies on the PointNet architectures [Qi+17a; Qi+17b]. The PointNet extracts a feature vector from a set of points $\mathcal{P} = \{p_1, ..., p_N\}$ such that $p_i \in \mathbb{R}^d$ using equation 2.1. The functions $h$ and $g$ are realised with shared multi-layer perceptrons (MLPs) and the $max$ operation is the max-pooling. The $h$ function outputs a feature vector per point. The max-pooling operation takes the maximum value in each dimension of point feature vectors. Training the MLPs makes learning representations of the given point sets possible. Finally, the $g$ function outputs a feature vector $\mathbf{f} \in \mathbb{R}^m$, which is the representation of the point set $\mathcal{P}$. PointNet encodes the point features in a set without considering their neighbourhood in a metric. However, grouping points in different scales around a centroid can help learn more generalisable local representations. Therefore, PointNet++ learns features of centroids in a given point set within multiple radius neighbourhoods of a centroid. The multi-scale grouping of PointNet++ applies equation 2.1 to several subsets of $\mathcal{P}$ separately. Then, the feature vectors of each subset are concatenated to reach the final representation of the $\mathcal{P}$. PointNet and PointNet++ are for the local feature encoding. *Set Abstraction* (SA) layers are to process point clouds hierarchically[Qi+17b]. SA first samples regional centroids using the farthest point sampling (FPS) method iteratively from a given point cloud $\mathcal{P}$ with $N$ points. FPS ensures that the point clouds have the best coverage with a fixed number of

centroid points ($N'$). Then, centroids are grouped with their $K$ neighbour points. Finally, a PointNet module is applied to $N' \times K$ points that each point is represented with a $d + C$-dimensional vector. $d$ is the point coordinate vector length, and $C$ is the dimension of the feature vector. With the max-pooling operation within each group, PointNet decreases the $N' \times K \times (d+C)$ feature tensor into $N' \times (d+C')$, where $C'$ is the new feature length. The point coordinates $p_i = (x, y, z)$ are shifted to the local region coordinates $p_i' = (x - c_x, y - c_y, z - c_z)$ according to the centroid point ($\hat{p} = (c_x, c_y, c_z)$) of a group. Local coordinates ensure invariance in a point group or a subset.

$$\mathbf{f} = g(\underset{i=1,\ldots,N}{max}(\{h(p_i)\})) \tag{2.1}$$

**Voxel-based** point cloud feature learning splits the 3D space into equally spaced voxels in all three axes [ZT18]. Each voxel contains several points. By learning voxel encodings, the point cloud can be represented structured. Voxel feature encoding (VFE) layer used in VoxelNet [ZT18] processes points similar to the PointNet but limiting the region with voxels instead of a radius neighbourhood. Each point in a voxel is represented with a point vector $p_i = (x, y, z, r, x - c_x, y - c_y, z - c_z)$, where $(x, y, z)$ is the point coordinate, $r$ is the reflectance value returned from the lidar sensor, $(c_x, c_y, c_z)$ is the centroid coordinate of the voxel points. The voxel feature vector $v$ can be calculated following equations 2.2, 2.3, and 2.4 from $N$ points. Points are randomly downsampled if a voxel contains more than $N$ points. $h$ and $g$ functions are realised with fully-connected networks, the $max$ operation is the element-wise max-pooling, $concat$ is the concatenation of vectors. $\mathbf{f}_i$ represents the point-wise feature, and $\mathbf{u}$ represents the global feature of the voxel, which is used to derive the voxel feature $v$. After obtaining features of all the non-empty voxels, 3D convolutions are applied to process the voxel features for the 3D detection of objects. Due to the high percentage of empty voxels, SECOND [YML18] proposes sparse 3D convolutions, which is the common approach for the voxel-based state-of-the-art 3D detectors [Shi+20a].

$$\mathbf{f}_i = h(\mathbf{p}_i) \tag{2.2}$$

$$\mathbf{u} = \underset{i=1,\ldots,N}{max}(\{\mathbf{f}_i\}) \tag{2.3}$$

$$\mathbf{v} = \underset{i=1,\ldots,N}{max}(g(\{concat(\mathbf{u}, \mathbf{f}_i)\})) \tag{2.4}$$

**Pillar-based** feature learning is similar to the voxel-based one, but this encoding scheme improves the processing time while providing similar results [Lan+19]. A pillar is defined as an infinitely-high voxel. Thus, the 3D space is not binned in the elevation axis and requires less hyper-parameter tuning. Pillars are represented on the BEV, and features are extracted per pillar using a PointNet-like pillar encoder. Similar to the VoxelNet [ZT18], the number of points in a pillar is limited with $N$ points to keep a dense tensor shape, and points are randomly sampled if more than $N$ is residing in a pillar. A point input is represented as $p_i = (x, y, z, r, x - c_x, y - c_y, z - c_z, x - c_{px}, y - c_{py})$ similar to the VoxelNet, but adds two additional values. $c_{px}$ and $c_{py}$ represent the centre of a pillar in the BEV along x- and y-axis. Using offsets to $c_{px}$ and $c_{py}$ include information about the points' positions in local pillar coordinate system. Representing points with offsets to the point centroids $(c_x, c_y, c_z)$ embeds the points considering points' distribution in a pillar. As the 9D point vectors are taken as inputs, pillar features are calculated with the equations 2.2, 2.3, and 2.4.

All learning-based point feature encoding methods are widely used in the literature. Some studies also combine two representations to benefit from the advantages of various local context embedding schemes. Frustum PointNet [Qi+18] uses the frustum generation method to obtain object points and then apply cascaded PointNet or PointNet++ architectures to learn

object features. [Yan+20b] applies FPS and SA layers hierarchically to predict object centres. [Yan+19] gets object proposals using PointNet features and then applies voxelisation to extract voxel-based point features from the region. [He+20a] utilises voxel-based encoding for the backbone and point-based encoding for its auxiliary network. PV-RCNN [Shi+20a] fuses voxel features with keypoint features to obtain more discriminative features. [Den+20; Zhe+21c] directly utilise voxel features for the 3D object detection. [YZK21] interchangeably uses pillar-based or voxel-based point feature encoding backbone. [Pie+21] extracts pillar features from a sequence of point clouds to obtain 3D bounding boxes.

### 2.2.3 Geometrical 3D Bounding Box Representation

3D bounding boxes parameters can be defined in different ways [Ku+18]. [Che+17] estimates a bounding box with all the corner positions $b = (c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8)$, where $c$ is the 3D position of each corner of the box. This representation requires 24 parameters in total. The object's orientation is derived from the long side of the 3D box. However, this assumption is not always accurate, especially for pedestrians. The 8-corner parameter definition does not consider the shared coordinates among the corners and redundantly estimates the box's coordinates. [SX16] defines the 3D bounding boxes with $b = (c, w, h, l)$, where $c$ is the centre of the bottom plane and $(w, h, l)$ are the width, height, and length of the 3D box. Since the centre is 3 dimensional, this representation estimates 6 parameters in total. The box orientation is taken as the high-scoring anchor's orientation, which is aligned with the axes of the indoor room. Even though non-square boxes are covered with anchors defined with perpendicular orientations, the axis-aligned orientation assumption is not valid in outdoor cases.

[Ku+18] defines 3D bounding boxes differently from the previously-introduced ones. The bounding boxes are estimated with $b = (c_1, c_2, c_3, c_4, h_1, h_2)$ parameters, where $c_i$ are the bottom plane corners of the bounding box, $h_1$ and $h_2$ are the heights of the bottom and top planes from the ground. In addition, the method estimates the orientation $\theta$ with $(cos(\theta), sin(\theta))$ parameters. However, the $\theta$ value is not directly used. After estimating bottom plane corners, there are four possible directions for the bounding boxes. The closest one among the four possible directions to the $\theta$ is taken as the box orientation. On the other hand, more recently proposed methods [Qi+18; ZT18; Lan+19] follow a different approach and define 3D bounding boxes with $b = (x, y, z, h, w, l, \theta)$, where $(x, y, z)$ are the 3D coordinates of the 3D box centre, $(h, w, l)$ are the height, width, and length of the 3D box, respectively, and $\theta$ is the heading angle of the bounding box. [Qi+18] estimates the $\theta$ by classifying the equally-split bins of $2\pi$ angle and regressing offset values of each bin.

### 2.2.4 3D Bounding Box Annotation Conversion

Similar to 3D bounding box definitions, ground-truth bounding boxes and sensory data are kept in different formats for different datasets as introduced in Section 2.4. Since 3D object detection networks rely on geometric representations for 3D bounding box estimations, designing a generic interface to cover public datasets is essential and challenging. Recent 3D detection toolboxes handle the format-specific data challenges by converting several data formats into a generic one [Con20; Tea20; Zhu+19]. There are also attempts to define the data axes, world coordinates, and ground-truth annotations in a standard format [Han+19]. [Wan+20d] investigates conversion of data and annotation formats for transferring knowledge among datasets for the 3D object detection task. The ISO vehicle coordinate is a frequently-used standard to define the axes in industrial applications [ISO13; Han+19].

However, the bounding box definitions differ from the KITTI dataset's format [GLU12a], and conversion should be applied.

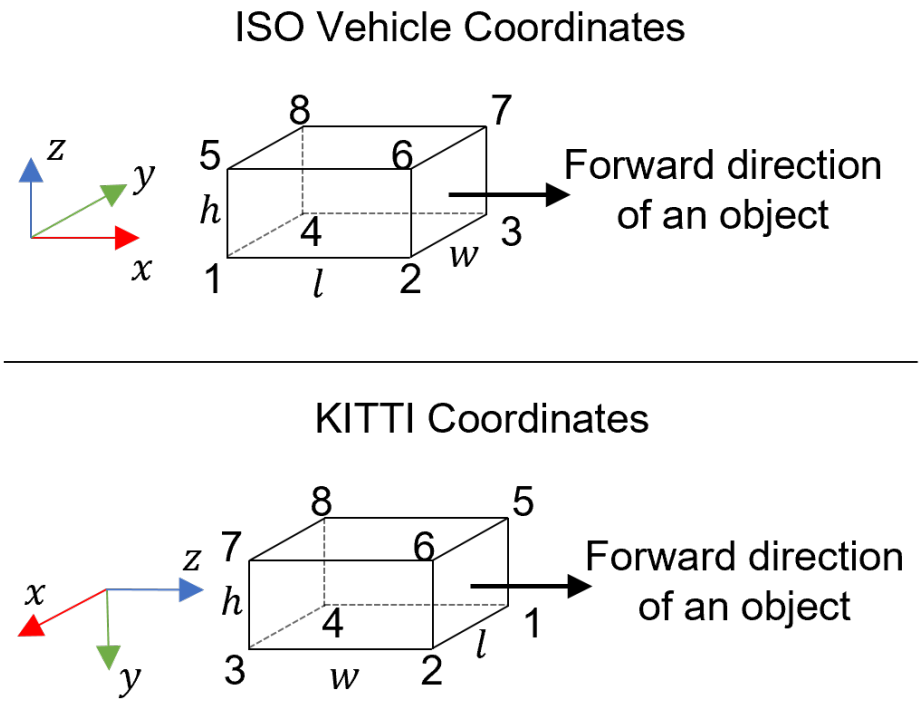



**Figure 2.5:** Comparison of 3D bounding box annotations between the ISO vehicle coordinates and the KITTI coordinates.

For ego vehicle coordinate system defined with the ISO vehicle coordinates [ISO13] in open simulation interface (OSI) [Han+19], the x-axis shows the forward direction of the ego vehicle, the y-axis points left of ego vehicle, and the z-axis points upwards from the ego vehicle as shown in Fig. 2.5. The yaw angle is the rotation angle around the z-axis. Every object provided in labels is annotated in its coordinate system. Using the yaw, pitch, and roll rotation matrices defined with quaternion angles, objects can be rotated to the ego vehicle coordinates. Objects' length ($l$) is the size of the bounding box in its forward direction. Similarly, an object's width ($w$) is its size along the y-axis in the object coordinate, and height ($h$) is its size along the z-axis. The centre of each object is defined as the 3D centre point of the bounding box. For 3D object detection, the pitch and roll angles are ignored, and only the yaw angle is estimated for the orientation. According to the object corner definition in Fig. 2.5, the corners of an object in the object's coordinate system can be given with equation 2.5. The numbers on the bounding box in Fig. 2.5 indicates the column order of $C_{obj}$ matrix in equation 2.5. The object corners can be transformed into ego vehicle coordinates with equation 2.6. The $(x, y, z)$ are the centre coordinates of the 3D bounding box. $R_z$ is the rotation matrix around the z-axis. $C_{ego} \in \mathbb{R}^{3\times8}$ is the corners in the ego vehicle coordinate system. $T_{cam2ego} \in \mathbb{R}^{4\times4}$ is the transformation matrix from camera coordinates to ego vehicle coordinate system. The equation 2.7 transforms the 3D box corners onto the image plane. The $P \in \mathbb{R}^{3\times4}$ matrix is the projection matrix from camera coordinates to the image plane. Similarly, point clouds are given in lidar sensor coordinates and can be transformed into the ego vehicle coordinates with equation 2.8. The $T_{lidar2ego} \in \mathbb{R}^{4\times4}$ is the transformation matrix from lidar coordinate system to the ego vehicle coordinate system. $p_{ego} \in \mathbb{R}^4$ and $p_{lidar} \in \mathbb{R}^3$ are a point in ego vehicle and lidar sensor coordinate systems.

$$C_{obj} = \begin{bmatrix} -l/2 & l/2 & l/2 & -l/2 & -l/2 & l/2 & l/2 & -l/2 \\ -w/2 & -w/2 & w/2 & w/2 & -w/2 & -w/2 & w/2 & w/2 \\ -h/2 & -h/2 & -h/2 & -h/2 & h/2 & h/2 & h/2 & h/2 \end{bmatrix} \tag{2.5}$$

$$C_{ego} = R_z \times C_{obj} + \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{2.6}$$

$$C_{img} = P \times T^{-1}_{cam2ego} \times \begin{bmatrix} C_{ego} \\ 1 \end{bmatrix} \tag{2.7}$$

$$p_{ego} = T_{lidar2ego} \times \begin{bmatrix} p_{lidar} \\ 1 \end{bmatrix} \tag{2.8}$$

Labels are defined in rectified camera coordinates for the KITTI dataset. In rectified camera coordinates, the z-axis points to the forward direction of the ego vehicle, the x-axis points to the right of the ego vehicle, and the y-axis points downwards from the ego vehicle, as shown in Fig. 2.5. Therefore, the yaw angle is defined as the rotation angle around the y-axis. The $h$, $w$, and $l$ values of each object annotation are given along the y-axis, z-axis, and x-axis of the object's coordinate system, respectively. Similarly, the forward direction of an object also points towards its z-axis. The centre of each object annotation $(x, y, z)$ shows the centre of the bottom plane of the 3D bounding box in KITTI. Similar to the OSI object corner coordinates, corners of an object in its coordinate system can be represented with the $C_{obj} \in \mathbb{R}^{3\times8}$ matrix as given in equation 2.9. The column numbers follow the corner numbers given in Fig. 2.5 for KITTI. The $C_{obj}$ can be transformed into the rectified camera coordinates ($C_{rect}$) of the KITTI dataset, in which the annotations are provided, using equation 2.10. $R_y \in \mathbb{R}^{3\times3}$ is the rotation matrix for the yaw angle. The corners in the rectified camera coordinates can be projected onto the image plane using equation 2.11. $P_2 \in \mathbb{R}^{3\times4}$ is the projection matrix from rectified camera coordinates onto the left camera image plane. A point can be transformed into the rectified camera coordinates using equation 2.12. $T_{lidar2cam} \in \mathbb{R}^{4\times4}$ is the transformation matrix from lidar sensor coordinates to the camera coordinates. $R_{rect} \in \mathbb{R}^{3\times4}$ is the transformation matrix to obtain points in the rectified camera coordinates.

$$C_{obj} = \begin{bmatrix} l/2 & l/2 & -l/2 & -l/2 & l/2 & l/2 & -l/2 & -l/2 \\ 0 & 0 & 0 & 0 & -h & -h & -h & -h \\ w/2 & -w/2 & -w/2 & w/2 & w/2 & -w/2 & -w/2 & w/2 \end{bmatrix} \tag{2.9}$$

$$C_{rect} = R_y \times C_{obj} + \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{2.10}$$

$$C_{img} = P_2 \times \begin{bmatrix} C_{rect} \\ 1 \end{bmatrix} \tag{2.11}$$

$$p_{cam} = R_{rect} \times T_{lidar2cam} \times \begin{bmatrix} p_{lidar} \\ 1 \end{bmatrix} \tag{2.12}$$

To align the annotations between OSI and KITTI, coordinate systems of the object annotations are compared. The axes, along which the object sizes and the centres are defined, are different between the considered formats. The heading angle also differs. The lidar point clouds of OSI can be converted into the KITTI format with equation 2.13. Since the KITTI lidar sensor coordinates and the ego vehicle coordinates are defined the same, the OSI lidar points should be transformed into ego vehicle coordinates.

$$p_{rect} = T_{lidar2ego} * \begin{bmatrix} p_{osi} \\ 1 \end{bmatrix} \tag{2.13}$$

The centres in object 3D bounding box annotations defined in OSI format can be converted into KITTI format following equation 2.14. The KITTI centre annotations point to the centre of the bottom plane, whereas it is the bounding box centre for the OSI format. Also, the axis directions are changed. Similarly, there is a difference in object bounding box size definitions. The equations in equation 2.15 can be used. The equation 2.16 shows the conversion of the yaw angles from OSI format to the KITTI format.

$$\begin{aligned} x_{kitti} &= -y_{osi} \\ y_{kitti} &= -z_{osi} + 0.5 * h_{osi} \\ z_{kitti} &= x_{osi} \end{aligned} \tag{2.14}$$

$$\begin{aligned} h_{kitti} &= h_{osi} \\ w_{kitti} &= l_{osi} \\ l_{kitti} &= w_{osi} \end{aligned} \tag{2.15}$$

$$ry_{kitti} = -\theta_{osi} \tag{2.16}$$

The calibration matrices should also be defined in the KITTI format for a complete annotation conversion from OSI to KITTI. $P_i$ matrices for KITTI are the projection matrices from rectified camera coordinates onto the image plane, and $i$ is the camera ID. $P_2$ is for the left RGB camera in the KITTI dataset. However, mostly the projection matrices do not have a correspondence in the OSI format. The equation 2.17 shows the corresponding KITTI $P_2$ projection matrix derived from the OSI calibration matrices. $P_{osi}$ is the projection matrix from camera coordinates to image plane for the OSI format. $T_{cam2ego}$ is the transformation matrix from camera coordinates to the ego vehicle coordinates. The final $4 \times 4$ matrix in equation 2.17 ensures the conversion from OSI to KITTI 3D bounding box corners. Since the rectified camera coordinates are not defined in OSI format, the $R_{0\_rect\_kitti}$ can be defined as an identity matrix to keep up with the KITTI format as given in equation 2.18. Similarly, the transformation matrix $T_{lidar2cam\_kitti}$ from the lidar sensor to the camera is provided in equation 2.19.

$$P_2 = P_{osi} \times T^{-1}_{cam2ego} \times \begin{bmatrix} 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}^T \tag{2.17}$$

$$R_{0\_rect\_kitti} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.18}$$

$$T_{lidar2cam\_kitti} = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \tag{2.19}$$

### 2.2.5 Frustum Generation

The Frustum PointNet network [Qi+18] processes points in the frustum of a 2D bounding box to reduce the search space. Therefore, points in the frustum should be extracted as a pre-processing step. Corners of the 2D bounding boxes are represented with their pixel values on the image plane. The frustum of a 2D bounding box indicates the field of view of the 2D bounding box in the 3D space defined with the camera coordinates. A lidar point can be projected onto the image plane to understand whether the lidar point is in the frustum of a 2D bounding box. The first transformation is from the lidar coordinate system of KITTI into the rectified camera coordinates using equation 2.12. The following operation is projecting points onto the image plane using equation 2.11. Assuming that a 2D bounding box is given with $b = (x_1, y_1, x_2, y_2)$, a point projected onto the image $p_{img} = (x_p, y_p)$ should satisfy the following inequalities to be in the frustum of a 2D bounding box:

$$\begin{aligned} x_p &> x_1, \\ y_p &> y_1, \\ x_p &< x_2, \\ y_p &< y_2 \end{aligned} \tag{2.20}$$

Fig. 2.6 shows an image of a pedestrian cropped into its 2D bounding boxes. The frustum of the 2D bounding box is given with the green lines, which contains the orange points. The black points reside outside of the frustum. This process reduces the number of lidar points used to estimate object 3D bounding box. However, there are still background points in the frustum, which should be eliminated by the 3D object detection network.

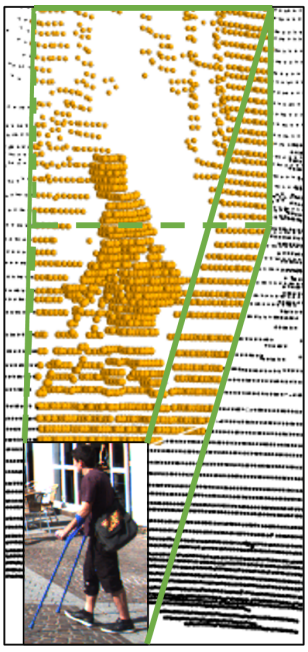


**Figure 2.6:** Extracting frustum point cloud. Green lines represent the frustum of the given 2D image cropped into the 2D bounding box of the pedestrian. The orange points represent the points that reside inside the frustum.

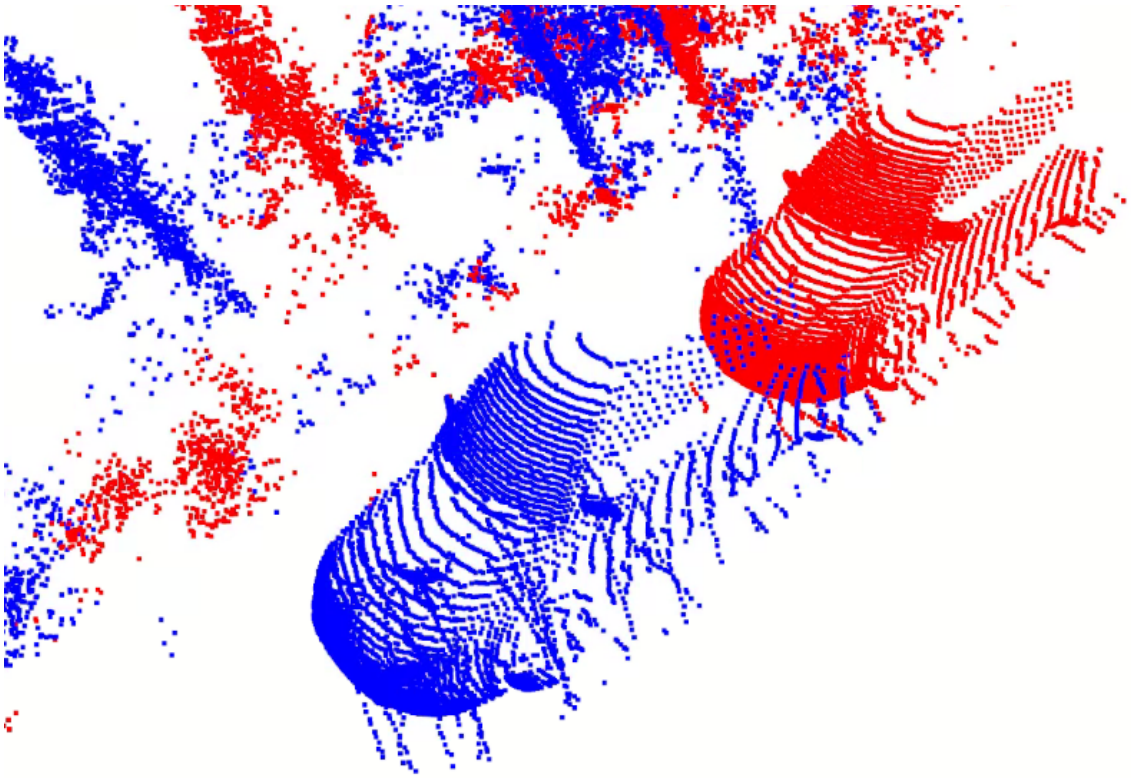


**Figure 2.7:** Movement of a point cloud between two consecutive frames. Red and blue points are from frames $t$ and $t+1$, respectively.

## 2.3 Scene Flow

Scene flow problem is an extended version of optical flow with the third dimension [Ved+05]. The aim is to estimate 3D motion for each element of the given sensor data between two consecutive frames. For lidar point clouds, scene flow is interested in finding motion of each point from frame $t$ to $t+1$, $\mathcal{F}_{t\to t+1}$, in 3D. Thus, the flow vector of a set of point can be shown as $\mathcal{F}_{t\to t+1} = \delta p_i \in \mathbb{R}^3$. In Fig. 2.7, the movement of the points reflected from a car between two frames are shown. Red and blue points are from frames $t$ and $t+1$, respectively. The point motion may originate from the ego vehicle's movement since the lidar points are defined in the ego vehicle's coordinate. In addition, the motion originates from the dynamic objects in the environment. Therefore, as seen in Fig. 2.8, point motion is not uniform in the environment. Some objects are almost static, whereas some others are moving. One of the challenges in scene flow is the limited annotated datasets [MHG15]. This is mainly caused by the difficulty of annotating points one-by-one between two frames. In addition, the points in two successive frames do not correspond to each other in a one-to-one manner. Lidar sensors do not receive reflections from the same points since the ego vehicle's angle of view and objects' position change between two frames. In Fig. 2.9, the propagated points from frame $t$ to $t+1$ with the ground-truth motion vectors are shown with the green colour. The figure shows that the blue and green points do not match one-to-one.

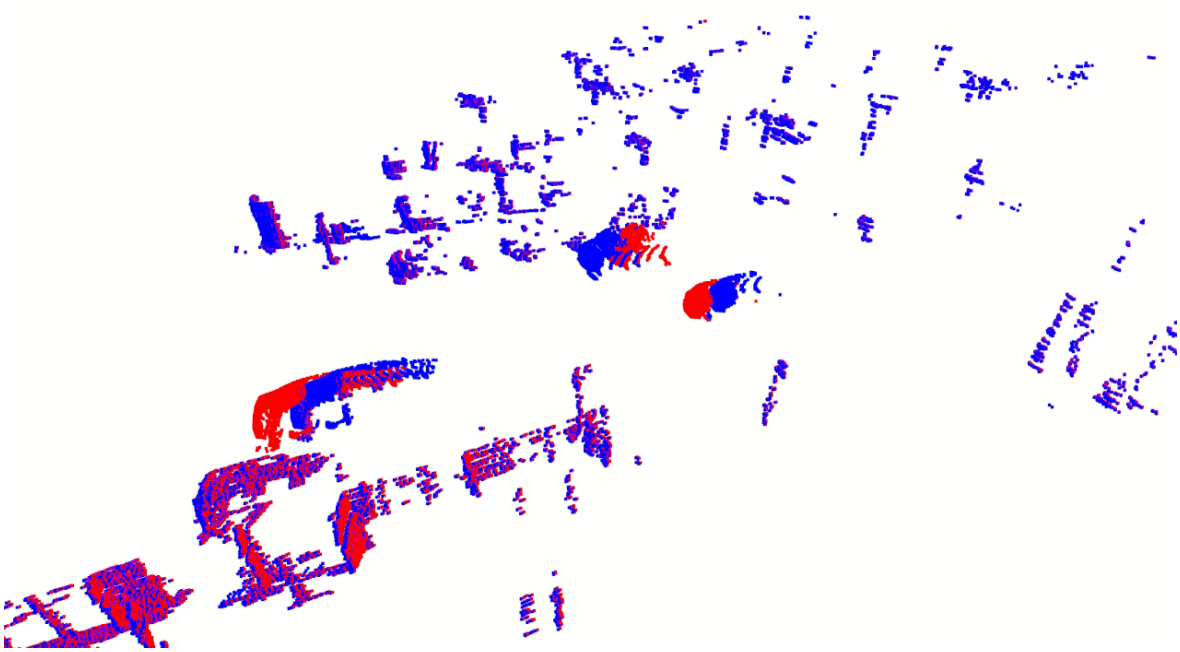


**Figure 2.8:** Scene flow in a dynamic environment. Red points are from the previous, and blue points are from the next frame. Some of the objects have a bigger motion than others, which indicates that the motion is not uniform in the environment.

## 2.4 Datasets

### 2.4.1 KITTI 3D Object Detection Dataset

KITTI 3D Object Detection dataset [GLU12a] has been a commonly-used benchmark for validating 3D detection networks for autonomous driving. The dataset contains stereo RGB images and lidar point clouds as samples from different drives in Karlsruhe, Germany. The frames are not sequential except the RGB images given with the preceding three frames without annotations. Data are collected mainly in good weather and during the day. A *Velodyne HDL-64E* lidar sensor is used to collect point clouds. There are 7481 training and 7518 test frames with their associated data. Only the annotations for the training data are provided. Splitting the training data into the train and validation sets has been a common approach as proposed in [Che+15]. The proposed train-val splits consist of 3712 training and 3769 validation frames.

KITTI benchmark provides annotations for the car, van, truck, pedestrian, person sitting, cyclist, tram, misc, and dontcare classes. Only the car, pedestrian, and cyclist classes are used for evaluations due to the small number of samples from other classes. The dontcare regions are defined only on the 2D images to indicate regions where objects are not annotated to avoid false positives during the training. Object annotations include 2D bounding boxes in pixels, 3D bounding box centre positions $(x, y, z)$, 3D object dimensions $(w, h, l)$, and their orientations. Truncation and occlusion properties of the objects are given, which are used to determine the difficulty levels of the objects.

The average precision (AP) is used for each class and difficulty level for the evaluation. There are three difficulty levels: easy, moderate, and hard, split according to the objects' occlusion, truncation, and distance. The AP values are calculated with an $IoU = 0.7$ for cars and an $IoU = 0.5$ for pedestrians and cyclists. Currently, the official AP values are computed through 40 recall points as suggested in [Sim+19b]. However, when necessary, we also

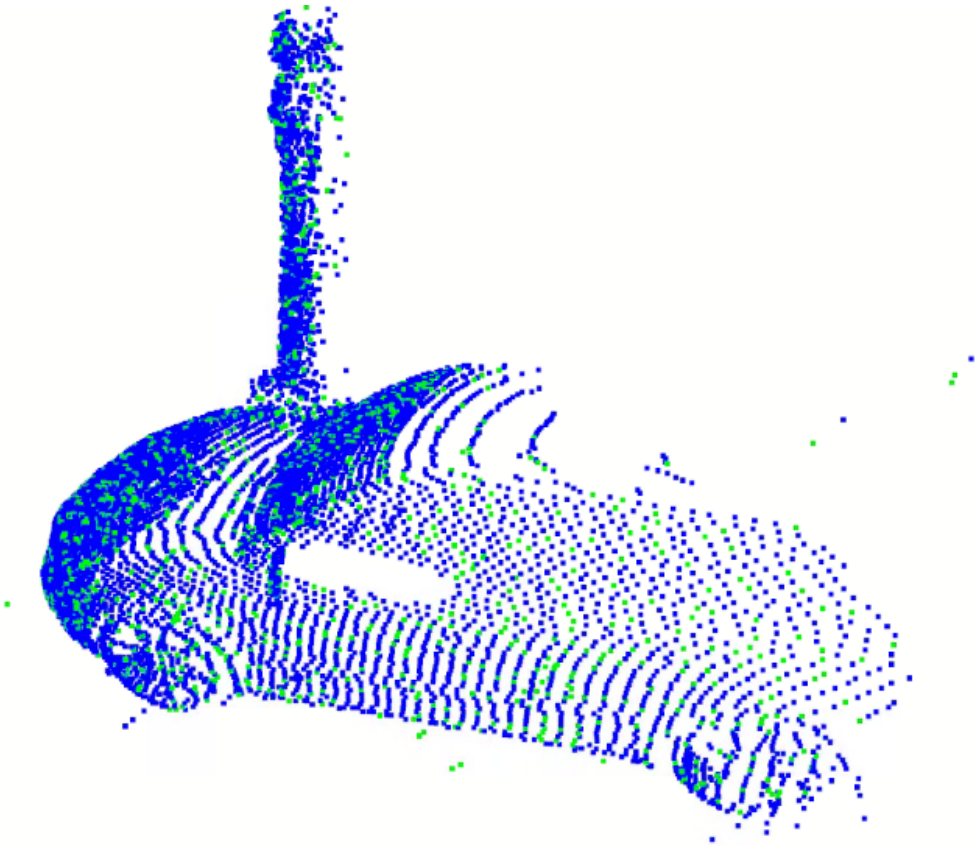


**Figure 2.9:** Green points are propagated from frame $t$ to $t+1$ using ground-truth annotations. Since there is not a one-to-one correspondence between successive frames, green points do not exactly match to the blue points at frame $t+1$.

utilise AP with 11 recall points to compare with previous works, which used 11 recall points to report their results.

### 2.4.2 KITTI Multi-object Tracking Dataset

One of the datasets used in this work is the KITTI Multi-object Tracking Dataset [GLU12a]. The dataset is from the KITTI benchmark suite and provides data from sequential frames, unlike the KITTI 3D object detection dataset. The multi-object tracking dataset contains 21 training and 29 testing drives. The drives are small sections of different journeys in the urban or suburban environment. Among all the data types provided in the dataset, we use the left RGB camera images and lidar point clouds, which are sampled and synchronised at 10 Hz. The lidar point clouds are taken from 64 channel *Velodyne HDL-64E* laser scanner.

The dataset contains the track ID annotations as well as the 2D and 3D bounding box labels of the objects. The annotations are only for the objects visible in the front camera view. We use drives with IDs 11, 15, 16, and 18 in the training set for validation and the rest of the training drives for training since there is no standard train-val split in the community. In total, the training and validation sets contain 6264 and 1239 frames. While defining the train-val split, we tried to keep the object instance ratio in both splits similar. The training split contains 31886 car, 8379 pedestrian, and 1039 cyclist instance annotations and the validation split has 6494 car, 2980 pedestrian, and 809 cyclist instance annotations. Also,

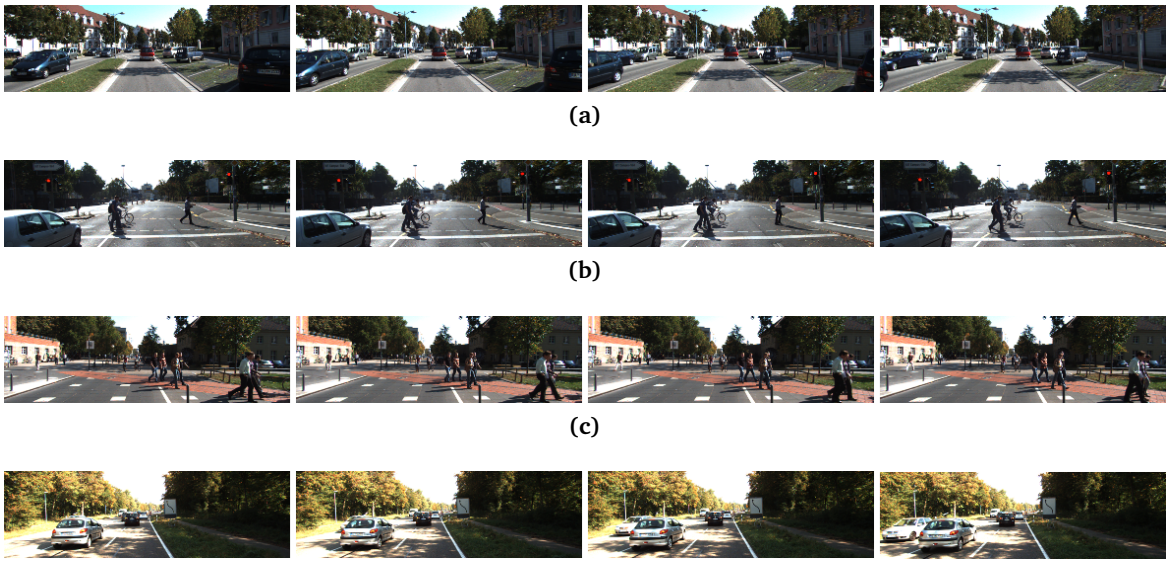


**Figure 2.10:** Successive scenes from our validation split of the KITTI multi-object tracking dataset. (a) Drive 11, (b) Drive 15, (c) Drive 16, (d) Drive 18.

the 92% and 96.3% of objects in the training and validation splits can be followed in more than 10 sequential frames.

We also tried to include different scenarios in the validation to make it challenging enough. Fig. 2.10 shows some successive scenes from the validation drives, which combine urban and suburban drives. The validation data contain objects occluded mainly because of the traffic jam, overtaking and parked cars, pedestrians and cyclists crossing the street.

It is important to note that lidar point clouds are sparse due to the capabilities of outdoor laser scanners. Detecting far-away objects is more challenging since fewer points are reflected from these objects, and they are more prone to occlusions. Table 2.1 summarises the average number of points of the objects for different distance bins. The table shows the average number of points inside objects' ground-truth 3D bounding boxes according to the KITTI difficulty levels for training and validation drives. As expected, objects in the hard category contain fewer points than the others, making them difficult to detect. Also, after 35 meters, the number of reflected points decrease steeply. This relation shows the difficulty of detecting far-away objects. Comparing the training and validation splits, the validation objects are more challenging to detect than training objects, even for the moderate and hard difficulty levels and all distance bins. The dashes in the table indicate that no object instance fits into that category.

### 2.4.3 KITTI Scene Flow Dataset

The KITTI Scene Flow dataset consists of multi-view data and flow ground-truths annotated semi-automatically [MHG15]. The dataset consists of 200 training and 200 test scenes. For the training data, the scene flow, optical flow, and depth disparity ground-truths are available for all pixels. In addition to the smallness of this dataset, the dataset does not contain point clouds and scene flow ground-truths for them. Therefore, the KITTI scene flow data are only appropriate for RGB stereo flow estimation methods. To remedy the lack of point cloud data, [LQG19] generates ground-truth 3D motion labels per point from the raw lidar point clouds of the KITTI scene flow training set. Authors apply ground removal methods on the

**Table 2.1:** Mean number of lidar points in object 3D bounding boxes according to the distance and KITTI difficulty level

| Training split | 0-35 m | 35-50 m | 50 m + |
|---|---|---|---|
| Easy | 375.4 | 78 | - |
| Moderate | 209.1 | 36.8 | 17.6 |
| Hard | 95.6 | 25.3 | 8.3 |
| **Validation split** | **0-35 m** | **35-50 m** | **50 m +** |
| Easy | 304.4 | - | - |
| Moderate | 137.3 | 31.7 | 13.5 |
| Hard | 93.7 | 16.4 | 5.25 |

raw point clouds before annotating since the ground points do not usually contribute to the motion vectors in the scene. Finally, they provide scene flow ground-truth labels per point for 150 training scenes from the KITTI scene flow dataset. Among 150 scenes, 100 frames are used for training, and the remaining 50 frames are used for validation. Since then, the provided point cloud scene flow dataset has been the standard for the point cloud scene flow methods [Wan+20f; MOH20; Lee+20; Bau+19] to fine-tune the deep neural networks on a real-world dataset. An example of the scene flow dataset can be seen in Fig. 2.8. The red points are from the frame $t-1$ and the blue points from the frame $t$. The ground points are removed. Therefore, the ground points growing in a circular shape similar to Fig. 2.3 cannot be seen in Fig. 2.8. The ground points might also decrease the accuracies of some scene flow methods, as stated in [LQG19; Bau+21].

### 2.4.4 nuScenes Dataset

The nuScenes [Cae+20] benchmark provides a large-scale autonomous driving dataset for the 3D object detection task. This dataset consists of 700 training, 150 validation, and 150 testing drives. Each drive consists of 20 second data. Compared to the KITTI dataset, nuScenes consists of more challenging and denser scenes. The data are collected in different lighting and weather conditions. nuScenes contains lidar and radar scans, camera images, and map information. The radar data are synchronised and combined from 5 radar sensors, and there are six camera images from different views per time step. The data are annotated for the samples at 2 Hz. Since lidar scans are sampled at 20 Hz, the annotations cannot be used for every lidar sweep. The common approach is to fuse 10 lidar scans for each keyframe annotation set [Lan+19]. In addition, nuScenes data span 360-degree view, whereas KITTI only provides annotations for the front-view objects. The data are collected from two different cities, another difference from KITTI. The annotations are given for 23 classes, and 10 out of 23 classes are evaluated for the benchmark. The lidar sensor used for the nuScenes is a 32-channel one. For the evaluation, average precision (AP) per class, mean average precision (mAP), and nuScenes detection score (NDS) are used.

### 2.4.5 Metrics

3D AP is the primary metric for evaluating 3D object detection results [GLU12b; Cae+20]. This metric is calculated using the precision/recall curve based on the overlap between pre-

dicted 3D object bounding boxes and ground-truths [Eve+15]. According to [Eve+15], precision and recall definitions are given with equation 2.21. $TP$, $FP$, and $FN$ are true positives, false positives, and false negatives, respectively. The true positive indicates correctly detected objects. The false positive is for wrongly detected objects, and false negative means that the detector missed a ground-truth object. The precision defines how successful the method is with its detections. The recall shows the percentage of the detected objects among all the ground-truth object bounding boxes. The precision/recall curve gives detection precision at every recall point.

$$\begin{aligned} precision &= \frac{TP}{TP+FP} \\ recall &= \frac{TP}{TP+FN} \end{aligned} \tag{2.21}$$

In the KITTI evaluation [GLU12b], the $TP$, $FP$, and $FN$ detections are determined according to the intersection over union (IoU) metric. IoU defines how much two bounding boxes are close to each other. Thus, the intersection of bounding boxes is divided by the union of the two bounding boxes. IoU is calculated with equation 2.22, where $b_i$ is an object bounding box. Intersection and union can define the area or the volume. An object is correctly detected if its IoU with a ground-truth is above a certain threshold. For the KITTI benchmark, the threshold is defined as 0.7, 0.5, and 0.5 for the car, pedestrian, and cyclist classes, respectively. However, for the nuScenes [Cae+20], the matching metric is the centre distances between a detected object and a ground-truth object. The object is correctly detected if the distance is less than 2m on the ground plane.

$$IoU(b_1, b_2) = \frac{b_1 \cap b_2}{b_1 \cup b_2} \tag{2.22}$$

A precision/recall curve is given in Fig. 2.11, which is calculated on the KITTI tracking dataset. The AP is calculated by averaging the precision values at different recall points as given with equation 2.23. In the equation, $R$ is a set of recall points, and $r$ is a recall point, for which the precision value is interpolated. $\rho_{interp}$ defines the interpolation function to obtain precision value at the recall point $r$. The interpolation function is given with $\rho_{interp}(r) = \underset{r':r' \geq r}{max}\, \rho(r')$. The $\rho$ function returns the maximum precision value among all precision values calculated at recall points greater or equal than $r$. $R$ is initially defined for 11 recall points $0, 0.1, ..., 1$ as $R_{11}$ [GLU12b]. Following [Sim+19b], the number of recall points are increased to 40 with $R_{40} = 1/40, 2/40, ..., 1$. Because $R_{11}$ returns a high AP when a single object is detected with a high accuracy at $r = 0$, which means approximately 0.09 AP result with a single detection. This problem is eliminated by removing the $r = 0$ from the calculation. Also, 40 recall points are used instead of 11 to approximate the precision/recall curve better. On the other hand, nuScenes dataset calculates AP with 10 points by removing $r = 0$ from $R_{11}$ set [Cae+20]. mAP for nuScenes is calculated by averaging AP values over the distance thresholds $0.5, 1, 2, 4$ and the set of classes.

$$AP|_R = \frac{1}{|R|} \sum_{r \in R} \rho_{interp}(r) \tag{2.23}$$

In addition to the mAP, the nuScenes dataset defines the nuScenes detection score (NDS), a combination of different metrics given in equation 2.24. This metric takes half of its accuracy from mAP and the other half from the additional true positive (TP) metrics provided in $\mathbb{TP}$ set. mTP is the mean of all five TP metrics defined in [Cae+20].

$$NDS = \frac{1}{10}[5mAP + \sum_{mTP \in \mathbb{TP}} (1 - min(1, mTP))] \tag{2.24}$$

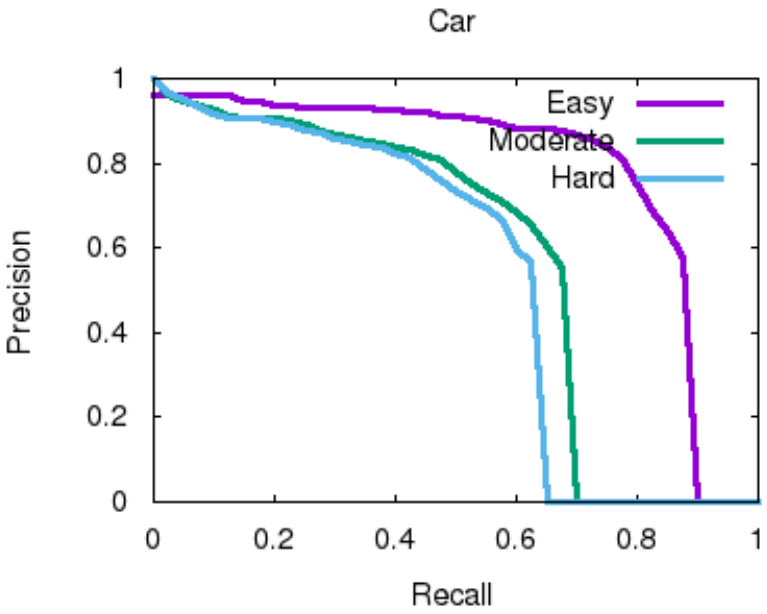



**Figure 2.11:** An example precision recall curve calculated for 41 recall points on the KITTI tracking dataset.

### 2.4.6 Prospective Autonomous Driving Datasets

With increasing interest in realising the autonomous driving function, perception tasks and methods have been studied more frequently. Related to this, several autonomous driving datasets with varying features have been recently released in addition to the datasets previously explained in this section [Cre+22; Sun+20; Cha+19].

A9-Dataset [Cre+22] provides RGB image and lidar point clouds data collected from a set of sensors mounted on a stationary setup on the A9 highway near Munich. The dataset is collected on a three km-long highway section and contains 1098 frames with 14459 annotated 3D objects. There are four subsets in the first dataset release, each of which has different characteristics. The point cloud is collected using a 64-channel lidar sensor. S1 subset contains only camera images from 4 cameras with two types of lenses. Images have a resolution of 1920x1200. S1 subset consists of randomly-sampled frames, whereas S2, S3, and S4 subsets consist of sequences with varying lengths, with a sampling rate of 2.5 Hz. The dataset also considers different weather conditions, which only provides daylight data. Annotated object classes are car, trailer, truck, van, pedestrian, bus, motorcycle, other, and bicycle with a decreasing number of labels, respectively. A9-Dataset provides a different perspective from other datasets since sensors are stationary. Also, the sampling rate of the sequences and varying weather conditions make 3D object detection challenging.

Waymo Open Dataset [Sun+20] is a large-scale dataset with 1000 training and validation scenes and 150 test scenes collected from an ego vehicle. Each scene continues for 20 seconds and consists of lidar point clouds synchronised with camera images. The data are collected with five lidar sensors and five cameras from diverse urban and suburban areas at 10 Hz, which give a dense representation. The dataset contains 2D and 3D bounding box annotations with object track IDs. In total, there are around 12 million lidar and 12 million camera bounding box labels for vehicles, pedestrians, cyclists, and signs. Two difficulty levels are considered for the benchmark metrics.

Argoverse dataset [Cha+19] provides 3D tracking data collected with two lidar sensors and two cameras. The lidar data are sampled at 10 Hz, whereas the camera data are sampled

at 5 Hz. The annotations are provided for 15 classes as 3D bounding boxes and tracking IDs for a total of 11052 object instances. There are 113 drives, each lasting from 15 to 30 seconds. Unlike the previous two datasets, HD maps are also provided, which can further enhance the understanding of the environment.

Chapter 3

# Related Work

## 3.1 Single-frame 3D Object Detection

### 3.1.1 Multi-modal RGB-Lidar Fusion Methods for 3D Object Detection

Multi-modal perception for autonomous driving is essential to avoid any problems due to sensor failures. RGB and lidar fusion has been a common way for multi-modal 3D object detections, generally conducted by feature-level or high-level fusion of RGB image and lidar point cloud information.

MV3D [Che+17] applies feature-level fusion of RGB images and hand-crafted front-view and bird's eye view (BEV) lidar point cloud features. BEV lidar features are used for generating proposals, and the region of interest (RoI) pooling is applied to the fused features for regional features of the 3D proposals. Like MV3D, AVOD [Ku+18] fuses RGB image and lidar BEV features but uses both features for proposal generation and detection. MVXNet [SZT19] projects a point cloud onto the image plane using calibration to obtain image features from a 2D detector corresponding to projected points. The corresponding 2D features of the points are then fused with the voxel features to detect objects. PointPainting [Vor+20] applies a similar method to extend point representations. The technique obtains semantic segmentation scores on the RGB image and adds the segmentation scores to the lidar point information $(x,y,z,r)$ by aligning image pixels and lidar points with projection. Further, a 3D detector is employed to process the extended lidar data. Zhu et al. [Zhu+21] propose a two-stage architecture similar to AVOD. However, the two backbone networks for RGB image feature extraction and lidar point feature extraction are fused in this level in addition to the RoI feature fusion using 2D-3D anchor matching.

Some studies consider the high-level fusion of RGB information with lidar data instead of feature-level fusion. Frustum PointNet [Qi+18] and F-ConvNet [WJ19] extracts frustums of an object of interest using 2D bounding boxes detected on RGB images. In this way, both approaches decrease the search space in the entire point cloud and take points only highly related to an object as inputs. Detection modules further process the frustum points to predict 3D bounding boxes from features of the raw points in the frustum. IPOD [Yan+18] uses semantic segmentation results from RGB images to filter out the background points and generate proposals in the point cloud. Afterwards, a point-based feature extractor generates features from the 3D proposal's points to predict the 3D bounding box.

Our temporal fusion method is based on the Frustum PointNet architecture, which applies high-level RGB and lidar frustum fusion. In this way, we obtain object features using lidar points representing the object of interest given by the 2D bounding box.

### 3.1.2 Lidar-based Methods for 3D Object Detection

Due to the unordered and sparse structure of the point clouds, it is not straightforward to process them directly with structured CNNs. Therefore, there are two main methods to obtain features from the point clouds: voxel-based and point-based.

VoxelNet [ZT18] introduces feature learning from points in a grid cell of 3D space called a voxel. A feature encoding layer learns to extract features from points, and 3D convolutional layers process the obtained 4D feature tensor to predict 3D bounding boxes. SECOND [YML18] improves the speed and accuracy by using sparse convolutions and data augmentation with ground-truth objects added to different frames from the database. Voxel R-CNN [Den+20] also follows the voxelisation strategy but introduces a two-stage detection with region proposals on the bird's eye view (BEV) plane to reach the feature representation capability of point-based approaches. Pointpillars [Lan+19] uses pillars, the infinite-height version of voxels, to increase the detection speed with high accuracy. The network learns pillar features from the points inside a pillar, and the 2D BEV representation of the pillar features are processed by a 2D CNN followed by a single-stage detection head. [Wan+20e] extends Pointpillars by fusing front-view pillar representations with BEV pillar features.

Point-based 3D detectors apply PointNet [Qi+17a] or PointNet++ [Qi+17b] architectures to obtain learned features from a set of points. Unlike [Qi+18], which uses 2D bounding boxes to reduce the sampling space for point clouds, 3Dssd [Yan+20b] applies a Euclidean and a feature distance-based sampling of points to have better point representation. These features are further concatenated and processed with an anchor-free head. Point-GNN [SR20] builds a hierarchical graph instead of a point sampling. The method extracts graph features with multi-layer perceptrons (MLPs) by iterating node and edge features. The detector considers nodes as centres of object proposals and applies a box merging method to avoid redundant bounding boxes of the same object.

STD [Yan+19] combines point-based processing with voxelisation. The proposals are generated using point-based features. The proposal features are converted into a dense and structured representation to decrease the computation load while having high object detection accuracy. SA-SSD [He+20a] mainly works on voxel features. However, the authors introduce an auxiliary network based on points obtained from 3D voxel tensors. The auxiliary network is trained with segmentation and centre estimation losses to learn better representations for 3D voxel features. PV-RCNN [Shi+20a] and its improved versions [BC20; Shi+21b] utilise point and voxel features at the same time for region proposals and corresponding keypoint features. In this way, they combine the representation strengths of both two-stage voxel-based networks and point feature extraction.

We utilise and extend Frustum PointNet [Qi+18] since it provides object-specific features without additional computation compared to the presented point- and voxel-based approaches in this subsection.

## 3.2 Multi-frame 2D Computer Vision Approaches on RGB Data

### 3.2.1 Video Object Detection and Tracking

Even though single-frame 2D object detection methods provide good results [Ren+16; He+17], they can miss objects in specific video frames detected in previous frames. Recent studies also propose multi-frame information fusion for 2D tracking, especially to infer motion from several frames instead of two.

Kang et al. [Kan+17a] generate tubelets from 2D object proposals. The object-level features from tubelets are processed by LSTMs in forward and backward directions to improve the detection performance. Similarly, in [Kan+17b], bounding boxes are propagated to the following frames using the motion guidance generated via tubelets from multiple frames. [Den+19] utilises attention mechanisms to aggregate object features in the current frame with those stored in the memory for better 2D object detection results. [SLB19] proposes a relation block between the target and supporting frame proposal features to improve the detection results in the target frame. [Wu+19] measures the similarities between the proposal features in video frames and concatenates similar proposal features to obtain a richer representation. [SN16] and [LLT17] utilise RNNs to fuse object-level features obtained by a single-frame proposal generation process.

Contrary to the object-level temporal feature aggregation, Liu et al. [LZ18] use convolutional LSTMs to fuse features from the entire scene of successive frames. [BTS18] combines scene-level features in a reference frame with a supporting frame using an attention-like architecture. Similarly, [Zhu+18] utilises flow fields to aggregate scene features from video frames. Ning et al. [Nin+17] combine visual features with bounding box information using LSTMs in video frames to reach precise location and tracking prediction. [Kim+21] also fuses scene-level features with an attention network combined with motion features generated with LSTMs in successive frames. TF-Blender [Cui+21b] temporally aligns the target and neighbouring frames individually and the adjacent frames among themselves as a scene-level method. This way, the feature contribution from previous neighbouring frames to the current frames is determined according to the relation between neighbouring frames themselves. [Zho+21c] tries to match the salient parts of the template with the search image features. By merging the salient regions, the method uses correlations to estimate the object bounding box in the target image.

Kraus et al. [Kra+21] combine context features from two successive frames with graph-level features and motion features from multiple frames to track objects in aerial images. [Pan+20] generates bounding tubes from front-view images and applies 3D convolutions to the tube feature tensor of the object of interest. This helps keep the object inside the tube and track without an additional single-frame 2D detection. [Zen+21b] uses a transformer to keep track of the feature of objects in the scene through multiple frames. The query list is updated recursively to avoid object switches. Similarly, [Chu+21] utilises a transformer architecture on graphs in various frames to track 2D objects. [Yan+21a] also takes advantage of an encoder-decoder transformer for tracking the object given with a template in the upcoming frames. The proposed model directly predicts the bounding box of the template object and updates the object templates dynamically, which removes the need for matching in any kind.

### 3.2.2 Action Recognition

2D action recognition problem requires combining the context from multiple frames since the state of a single frame can lead to different action decisions, as explained in [CZ17] by Carreira and Zisserman. For this reason, they propose utilising 3D convolutions that process the video clip in two branches are concatenated to predict action in the scene, which represent spatial and temporal streams similar to [FPZ16]. Using a similar architecture, Wang et al. [Wan+16] train the network on frame segments of a video clip using a 2D CNN. [Xie+18] and [ZSB18] investigate a combination of 2D and 3D CNNs to decrease the computation cost while keeping the accuracy high. With a similar aim, Kopuklu et al. [KWR19] apply an attention mechanism to combine 2D spatial and 3D temporal features effectively. Jiang et al. [Jia+19] propose combining spatio-temporal context with motion features in a ResNet-like architecture to improve action recognition accuracy using 2D CNNs. The architecture

proposed in [Yan+20a] learns spatial, temporal, and flow features in a pyramid network independent of the type of backbone to extract scene features. [Shi+21a] adds the unsupervised and weakly-supervised losses to train the video action detection network. It tries to learn action foreground regions and complete action scores instead of action class.

### 3.2.3 Segmentation

As the 2D object detection task, segmentation is prone to distortion in the RGB images, such as occlusions. Therefore, utilisation of the motion helps estimate the pixel classes and separate the instances.

Garnot and Landrieu [GL21] combine time-series satellite image features of different scales using a temporal attention module. They argue that using temporal feature maps helps obtain better border estimation between fields. Mao et al. [Mao+21c] utilise a transformer-based architecture to process spatio-temporal video data for better object segmentation in a semi-supervised way. The multi-frame encodings obtained from the transformer-based module guide the object mask estimation in the search image. Ji et al. [Ji+21] get discriminative features from the video images and optical flow outputs using a cross-attention module. The temporal information is brought into the context by the motion estimations. These are further combined with the appearance features through multiple message passing modules. [Lia+21b] utilises masks from the previous frames and the object-specific query from the current frame to generate a future object template for the video object segmentation. The temporal object template helps the object segmentation alignment by providing a coarse mask, refined with the proposal mask generated in the current frame. [Zha+21c] follows an iterative approach to refine object masks through multiple frames using a spatio-temporal aggregation module that finds the local correlations between the object encodings and the frame features.

Wang et al. [Wan+21g] tackle the video instance segmentation, object detection, and tracking problems simultaneously using a spatio-temporal graph architecture. The graph nodes from the reference and target frames are iterated through a message passing network to obtain temporal features. Afterwards, the node features are used to get segmentation and detection results and the edge features to match the objects. [CCS21] proposes a multi-task method for multi-object tracking and segmentation, which uses a long-range cost to make assignments between frames. The cost is based on the IoU, Euclidean distance, and optical flow, which decreases ID switches due to missed objects in a sequence. On the other hand, [Ke+21] utilises features from previous frames with a cross-attention network instead of only relying on the temporal data for assignments. The proposed method enhances the feature quality of the object of interest in the current feature map using the foreground and background feature prototypes from previous frames with its cross-attention module. Yang et al. [Yan+21b] utilise only the motion information to segment objects in complex images, where it is challenging to localise objects of interest. They use the self-supervised consistency between two reconstructed motion maps from the feature embeddings.

### 3.2.4 Other Computer Vision Problems

Temporal context is important for several other problems concerning the 2D video data. Utilising multiple frames helps understand the scene and object relations and their dynamic arrangements. Also, temporal cues have been utilised for ensuring the consistencies between frames while employing unsupervised and self-supervised learning techniques.

[Con+21] proposes a spatio-temporal transformer to find the relation between objects in video frames using a graph structure. The spatial encoder generates the context-refined features within the frame, and the temporal decoder combines the relationship between frames to predict a relationship graph. [Ten+21] also utilises a transformer-like architecture for additional temporal fusion with relation- and object-level features. The video features are obtained with a 3D CNN on top of single-frame detection and relation estimation for target features. For learning the correspondences of patches in video frames, [ZJH21] also proposes a spatio-temporal graph, which is trained with a contrastive loss, to capture neighbour relations and the long-term similarities between frames. Similarly, [ASS21] takes object features from a 3D CNN applied to video frames. For the video understanding problem, the method employs a graph message passing model on the object features, taken as nodes of the graph.

[Fen+21] employs a teacher-student approach with unsupervised learning in a representation learning problem. The method additionally brings in temporal teach networks for additional temporal supervision to the student network, followed by a knowledge transformer that determines the importance of the temporal teacher networks. [Li+21d] applies temporal convolutions for the trajectory forecasting through the graph features from multiple frames. In addition, the proposed method predicts future trajectories region-wise convolutions on the temporal graph features of the observed trajectories. [Wu+21b] applies graph-based multi-view matching for the pose estimation problem in a different approach. Even though the method does not consider the temporal features, the multi-view matching is similar to temporal matching between successive frames.

## 3.3 Multi-frame 3D Object Detection

Multi-frame 3D object detection started to receive attention with the availability of 3D detection datasets that provide sequential lidar and RGB data, such as nuScenes [Cae+20] and Waymo Open [Sun+20]. Even though the KITTI 3D object detection benchmark [GLU12a] includes RGB images from three successive frames, it does not provide lidar sequences. On the other hand, the KITTI multi-object tracking dataset offers sequences for necessary data but has been only utilised for tracking tasks. Therefore, using multi-frame features has been mostly limited to 2D video perception and 3D tracking tasks until recently.

We split the multi-frame 3D object detection methods into four main groups. Even though the groups have obvious overlaps, the main approaches are different. Methods like [YZK21] and [Sun+21] use the multi-frame information implicitly without directly fusing the features. In the second group, methods utilise recurrent layers or 3D convolutions for aggregating features across the whole scene from multiple frames. Methods in the third group align the scene-level features with more specialised mechanisms like attention modules before aggregating. The last group aggregates the object features from multiple frames instead of the scene-level features. The object-level features are associated with attention-like mechanisms or tracking information between frames.

CenterPoint [YZK21] utilises centre map predictions in two successive frames to perform 3D detection and tracking as points. Even though the model does not explicitly aggregate features from multiple time steps, it merges points from the previous frames to the current frame for the velocity estimation. [Sun+21] utilises object occupancy map history to determine the proposals in the upcoming frame. To improve the proposal accuracy, the odometry data are also considered as well as the Kalman filter outputs for the fused tracklets. [MY21] uses odometry-based flow information to fuse feature maps of successive time steps for the 3D object detection. [Pie+21] utilises the point cloud and RGB sequences while estimating object 3D bounding boxes. The point cloud features are marked with a time-stamp, their

positions are compensated with the ego-motion. The resulting point cloud is used to create the PointPillars pseudo-image representation. These RGB features from different network levels are added to the different hierarchical levels of the 3D detection network through dynamic connections determined by an attention-like mechanism. The resulting multi-level and multi-modal features estimate the 3D bounding boxes.

Luo et al. [LYU18] utilise a voxel-based deep neural network for 3D detection, tracking, motion forecasting tasks. They use 3D convolutions for processing features across time on their BEV voxel features. [El +18] and [MZ20] employ directly convolutional LSTMs on BEV features to have spatio-temporal feature aggregation. Similarly, Huang et al. [Hua+20a] propose a custom-designed LSTM to aggregate scene-level features in the previous time steps for improved 3D object detection and apply ego-motion compensation for the alignment between frames. The point features from successive frames require alignment due to the ego vehicle and the traffic participants' movements. The previously mentioned studies implicitly handle this problem with the recurrent architectures or receptive fields of 3D convolutions.

In addition to the RNN- and 3D convolution-based approaches, attention mechanisms have also been considered for temporal scene-level feature aggregation. Kumar and Alstouhi investigate non-local and spatio-temporal context networks for temporal 3D detection [KA20]. [Yin+20] and [Yin+21] combine a spatio-temporal transformer architecture with a convolutional gated recurrent unit (GRU) to propagate the scene-level feature in time. They utilise a spatial graph network and GRU cells to increase the receptive field of pillar features spatially. The pillar node features are further processed with a 2D CNN on BEV, which is the input of their feature aggregation network. Afterwards, a spatial transformer provides features to the convolutional GRU. A temporal transformer further aligns the outputs of the spatial transformer and the convolutional GRUs temporal features. [Yua+21] also aggregates BEV representations of raw point clouds from multiple frames using a temporal-channel transformer architecture to obtain 3D detections. SDP-Net [Zha+20b] is a multi-task (scene flow, tracking, and detection) network that takes sequential BEV maps for multi-frame processing. The multi-frame feature maps are aggregated using the BEV flow and ego-motion estimations. Also, a height correction among sequential scenes is applied using ground plane estimation. The flow is considered only for the voxels containing an object. The object bounding boxes are predicted using the final aggregated BEV feature maps. [Hu+20] calculates a temporal occupancy map using Bayesian filtering using point cloud sweeps. Then, the method fuses the occupancy map with the point pillar features for the final 3D object detection results.

As an object-level multi-frame method, Yang et al. [Yan+21c] propose an attention module to align and aggregate features from multiple frames. First, a single frame 3D detector is used to obtain 3D bounding boxes. Features from those regions are stored in the memory and used later to refine the 3D box in the detector's second stage. The attention module finds the corresponding feature maps in the multiple frames. Then, the features are aggregated with another attention module. Qi et al. [Qi+21] tackle the automatic 3D labelling problem using a sequence of lidar point clouds. The proposed method applies a single-frame 3D detector and a 3D multi-object tracker to obtain 3D bounding boxes and tracking IDs. Then, they refine the 3D boxes by merging the points from static and dynamic objects in successive frames through foreground segmentation, box regression networks, and sequence encoders. Xiong et al. [Xio+21b] extend the CenterPoint [YZK21] backbone with the foreground context modelling and a spatio-temporal graph aggregation module. The foreground context modelling block is used as an attention mechanism to understand the foreground regions. The region proposals from the sequential feature maps are further processed by the spatio-temporal graph aggregation module, which considers adaptive similarity measurements between object nodes in successive frames for the final 3D detection results.

These methods also show the usefulness of the multi-frame approaches to improve the quality of 3D detections. However, methods that utilise scene-level feature aggregation are required to align features from successive frames in dynamic traffic. On the other hand, the object-level features still rely on additional mechanisms for picking correct feature sets from the previous frames. The accuracy of these different modules significantly impacts the final detection quality.

## 3.4 3D Multi-object Tracking

Utilising 3D bounding box and appearance information through successive frames is a common approach for the 3D multi-object tracking problem. Fantrack [Bas+19] learns a similarity map between appearances of detections in two consecutive frames and their bounding box parameters using CNNs. The similarity map is further used to obtain object associations for tracking. AB3DMOT [Wen+20a], on the other hand, takes 3D bounding boxes from a lidar-based detector and propagates object positions with a 3D Kalman filter. Afterwards, the Hungarian method matches the propagated and detected objects reaching better 3D tracking results than Fantrack. This allows the model to be aware of objects that were only detected by one of the modalities. CenterTrack [ZKK20] takes two successive images as inputs with the detections in the past frame to predict detections in the current frame with 2D and 3D centre offsets. Using the temporal features and the predicted offsets, it can do the association and track the objects. [Sha+18b] applies geometry- and appearance-based costs for 3D bounding boxes between the propagated and detected objects in two successive frames. [Hu+19] combines 3D detection and 3D tracking tasks on monocular video frames using LSTMs for tracking. [LYY21] proposes a joint detection and tracking network for tracking the objects. The network jointly creates voxel features from two successive frames. Then, the features are used to estimate the motion between two frames with a centre-based approach used for the object association, removing the need for heuristic approaches. [Sim+19a] utilises RGB features and the semantic segmentation results for the 3D tracking of objects. These features extended using the semantic class information aligned with the ego-motion improve the tracking results efficiently.

In addition to using 2D appearance features for associating objects, temporal 3D appearance and motion features help improve tracking quality. mmMOT [Zha+19] learns to fuse image and lidar appearance features from detections. The fused features and the single-modal features are used to predict the correlation of objects and an adjacency matrix for matching. EagerMOT [KOL21] matches two frames' 2D and 3D boxes with a joint-fusion strategy. Gnn3dmot [Wen+20b] also employs 2D and 3D appearance features in addition to 2D and 3D bounding box features from multiple frames. A graph neural network (GNN) processes the multi-frame and multi-modal features of the objects in two successive frames to obtain affinity matrices. [Zhe+21a] also uses 3D box features to match a given object with the point cloud features in the upcoming frame's search area. The MLVSNet [Wan+21j] tries to match 3D target point cloud features with the voting features in a search area. An attention module generates target-guided features in the search area for multiple vote clusters for this task. The features of the vote clusters are used to predict the final 3D bounding box of the target object in the given search area of the next frame. [SLS20] utilises a 3D detector to jointly estimate 3D bounding box pairs from two successive frames and directly estimates the correspondences without using a matching algorithm.

## 3.5 Multi-frame Point Cloud Approaches

Due to the sparse nature of lidar point clouds, aggregating point cloud series has become a standard approach to 3D vision problems such as segmentation, pose estimation, and registration. TempNet [Zho+21b] proposes temporal backbone feature aggregation for the 3D segmentation problem. The aggregation module finds keypoint neighbours in preceding frames and combines the point features using an attention mechanism. A partial update module decides to update keyframe features according to successive frame consistency. Meteornet [LYB19] groups lidar points from multiple frames according to their neighbourhoods to learn temporal point representations. The temporally grouped point features through Meteornet's early and late fusion versions are further used for classification, semantic segmentation, and scene flow estimation heads.

StickyPillars [Fis+21] uses transformer attention to learn the correlation between point cloud graph nodes for the point cloud registration problem. The self-attention is used for spatial feature learning, and the cross-attention is applied to the keypoints from two frames to match the points. Similarly, PCAM [Cao+21] learns cross-attention matrices between two point clouds, which multiple point-based layers predict. The final attention scores are used to compute the correspondences between the point clouds to be registered. FIRE-Net [Wu+21a] jointly extracts features from target and source point clouds to match points instead of extracting single-frame features and finding correlations. FIRE-Net shares features of two point clouds at local and global levels. A hybrid graph is constructed between two point clouds with k-Nearest Neighbor (kNN) edges for the regional relations. The global relation is learned through the feature correlation matrices. [Den+21] proposes a loss metric match two point clouds, which considers the proximity of points through randomly-selected intersection lines instead of calculating the nearest neighbours of two point clouds. HRegNet [Lu+21a] also utilises hierarchical features with keypoint downsampling for coarse to fine registration between two point clouds.

[Hua+21] trains an online and a target network from two sequential point clouds of an object. The method utilises the consistency loss between two point clouds for learning the point representations. The target network's parameters are updated using the running average of the online network, both of which predicts the point cloud embeddings. SGM-Net [Wei+21] tries to learn rotation-invariant point features in a local neighbourhood using dense connections. Even though the proposed method is tested only with single-frame 3D perception tasks, the dense rotation-invariant representations might be necessary for assigning keypoints from multiple frames. [Wei+21] applies spatio-temporal attention to keypoint feature patches from multiple frames to solve 4D semantic segmentation and 3D action recognition problems. [Zen+21a] aims at improving 3D pose estimation for complex poses by aggregating temporal pose graph nodes using a 1D convolution through the temporal channel with an attention-like feature processing.

## 3.6 3D Attention Modules for Point Cloud Processing

Attention mechanisms play an essential role in finding feature correspondences and relations within a frame and between multiple frames. Especially self-attention networks have attracted attention in the computer vision community [Han+20] after the success of transformer attention modules in natural language processing [Vas+17]. This mechanism has been considered in different fields for different problems such as 2D video classification [Wan+18; Arn+21], image generation [Par+18; Che+20a], and image recognition [Dos+20]. However, attention modules for unstructured point cloud processing have only been recently

started to be investigated.

VoTr [Mao+21b] proposes a 3D detection backbone that consists of submanifold and sparse voxel multi-head attention modules. The submanifold self-attention produces the attended features for the non-empty voxels, whereas the sparse self-attention module extracts features of empty voxels. This helps capture the long-range context features within the point cloud. [Li+21b] creates dense 2D features maps from sparse voxel-based 3D feature maps. The dense features are obtained from 2D sparse feature maps using a deformable convolutional tower and mask-guided attention for a single-stage 3D detector. The mask-guided attention module aims to learn the background and foreground separation and enhances dense features in foreground regions. Wang et al. [Wan+21i] propose a projective transformer to align the lidar features with the RGB camera features for the 3D object detection task. 3DETR [MGJ21] also predicts 3D bounding boxes from downsampled point sets. The network uses a transformer encoder for per-point features and a decoder to estimate box parameters. Pointformer [Pan+21] is also a 3D detection backbone based on local and global transformer attention modules, which work on the features of local points and the keypoints. [BHC21] applies self-attention and deformable attention modules to the region-wise voxel or point set features. The attended features provide a context-aware richer feature set used for obtaining 3D object detections.

Pointr [Yu+21] is based on a transformer attention module for the point cloud completion task. The point-transformer uses graph-based features of the point proxies, which are point embeddings of a local region. Also, the model investigates kNN-based geometry-awareness in the transformer blocks as the inductive bias of the transformers. 3DVG-Transformer [Zha+21e] proposes a transformer-based cross-modal attention module between text description features and point cloud bounding box features to estimate the confidence scores of the objects described in the text. [Ran+21] applies multiple low-level and high-level attention modules to centroids and their neighbour points using point coordinates and features for the point cloud classification and segmentation tasks. [Zha+21d] proposes self-attention layers as point cloud processing backbones for several 3D vision tasks such as object classification, object part segmentation, and semantic scene segmentation. The point transformer layers are applied through all downsampling and upsampling point processing layers with the relevant skip connections to capture hierarchical context features. Similarly, [Hui+21] learns point descriptors globally using attention modules for the keypoints in all downsampling levels within the kNN vicinity for the place recognition problem.

## 3.7 Unsupervised and Self-supervised Point Cloud Processing Methods

Supervised learning is the dominant approach to train deep neural networks to accomplish computer vision tasks. However, annotating available large-scale data is a time-consuming, expensive, and complicated process, which requires expertise [He+20b; Car+20]. Moreover, providing annotations on the 3D point cloud data is more cumbersome than its 2D counterparts [Dai+17]. Therefore, unsupervised or self-supervised approaches have been extensively considered for 2D computer vision tasks [Xio+21a; Mus+21; CCC21; Gan+21; Zha+21a] and 3D point cloud processing. These approaches either learn the hidden structure [Wan+21f] in the data or use the data structure with cycle consistencies [CCC21; MOH20].

[Ngu+21] proposes using a new metric to measure discrepancies between two point clouds. The calculated discrepancy metric is used to train point cloud autoencoders, which are used for several downstream tasks such as point cloud reconstruction, registration, and generation. [Che+21] tries to learn the point cloud representations by removing or disorganising and reconstructing parts of a point cloud. The method first predicts the points belonging

to the disorganised part in the original point cloud and then reconstructs points in the abnormal region. In this way, the trained feature extractor learns how to represent point clouds and is used for a point cloud downstream task such as shape classification and part segmentation. [Jia+21a] uses the semantic segmentation predictions of unlabelled points as pseudo-labels for the contrastive loss calculation. The additional contrastive training helps learn more representative point embeddings for the point cloud segmentation problem. [WLN21] utilises a generative method based on graph point encoders for learning point representations and provides point segmentations.

Pre-training backbones on large annotated datasets has been intensively used for the 2D computer vision deep neural networks [Ren+16; He+17]; however, this approach has not been investigated extensively for 3D point clouds due to the limited annotated datasets. The unsupervised and self-supervised methods have recently solved such problems by training backbones (point cloud feature extractors) for complex 3D vision tasks. [Zha+21b] extracts global features from two augmented versions of the same depth map. Then, the difference between the global representations is used for training the feature extractor. [Rao+21] generates two scenes using the same synthetic object points, whose features are learned with an encoder-decoder architecture. The object features from two synthetic scenes train a 3D detector with a contrastive loss. Similarly, [Lia+21a] uses the transformations of the same real-world scene to learn similar feature embeddings for the same regions following a contrastive objective, which enforces geometry awareness. [Wan+21f] removes a set of points from the original point cloud, which is occluded in the camera view. Then, the method tries to reconstruct the removed points, which helps learn point representations used for point cloud downstream vision tasks. Differently, FAST3D [Fru+21] uses scene flow information to estimate box and object flow. This way, it can propagate the object labels in time and use the propagated bounding boxes as pseudo-labels for further training.

## 3.8 3D Scene Flow Estimation on Point Clouds

Scene flow problem is first proposed as an extension of 2D optical flow considering the motion of points in the third dimension [Ved+05]. Scene flow requires estimating motion vectors of 3D points in the 3D space. The motion vectors from the former to the next frame are obtained mainly by processing two sequential point clouds. However, there are not necessarily one-to-one point correspondences between two point clouds, making annotating the data and applying supervised learning approaches difficult.

Similar to the optical flow problem, monocular and stereo camera data have been used to estimate the 3D pixel motion [HD07; VSR15; HR20]. Also, combining lidar point clouds with RGB camera data has been investigated [Ris+20; Liu+21a]. Currently, most methods consider supervised learning for estimating 3D flow vectors per lidar point [LQG19; Wan+21b; Jun+21; Lee+20; Gu+19; Bau+19; Wan+20f; PBM20; Li+21c; DB20; Wu+20]. Others also employ attention strategies to enhance the regional feature relation and fill in missing correspondences [Wan+21c; Li+21a; Wan+21e]. However, obtaining scene flow annotations are extremely difficult. Therefore, many studies investigate self-supervised learning approaches using the cycle consistencies [MOH20; Zua+20; KER21; LLX21; PHL20; LZL21; Bau+21]. The scene flow problem is closely related to motion-based 3D vision problems such as multi-object tracking and motion segmentation, considered for multi-task training [Bau+21; OR21; Goj+21].

In an early stereo flow approach, Huguet and Devernay [HD07] obtain scene flow on the sequential stereo images. A numerical optimisation method matches the estimated optical flow between the sequential left and right images in the 3D space. [VSR15] obtains motion

fields with the motion of segmented stereo image parts. Further segmentation from coarse to fine provides the 3D motion using the geometrical correspondences. [Sha+18a] also takes the available depth information into account in addition to consecutive RGB images. The method predicts object centres, translation and rotation of the objects, which result in the object segmentations. The segmented objects are tracked through the consecutive images and final scene flow, and segmentation masks are derived using the available information. [Bat+19] fuses the lidar point cloud with the stereo images. Pixels and lidar points are matched, and lidar is used to correct the gap and obtain denser flow maps. RAFT-3D [TD21] extends the RAFT optical flow estimation network [TD20]. RAFT-3D takes sequential RGB-D data and tries to construct the point correspondences between two frames by iteratively estimating the optical flow, rotation and translation of objects.

PWC-Net [HR20] estimates the 3D motion and the depth map jointly using successive monocular images. The method uses a disparity-based self-supervised loss, calculated using the left and right camera images during training time. [HR21] takes advantage of an additional third frame in time on top of the PWC-Net. Additionally, convolutional LSTM layers pass the information among the triplet frames to jointly estimate the depth and scene flow. MonoComb [SSU20] uses the optical flow between two sequential monocular images to find the correspondences between the two depth estimations, which helps estimate the dense scene flow map. [Ris+20] fuses sparse lidar point cloud with the monocular RGB image features in a pyramid structure for the dense scene flow. CamLiFlow [Liu+21a] fuses the lidar point cloud features with the RGB image features. However, the method jointly estimates the optical flow and scene flow from the camera and lidar point cloud branches, whose features are fused through several layers in both directions.

Compared to the previously introduced dense scene flow approaches, the sparse scene flow methods aim to predict the motion of the lidar points. The straightforward training method is to apply supervised learning on the available synthetic [May+16] or real-world datasets such as KITTI Scene FLow [MHG15]. Flownet3d [LQG19] learns flow embeddings from two consecutive point clouds with downsampling and upsampling layers. The method mainly considers the neighbourhood of points in two scenes for the flow embeddings and optimises the network with scene flow ground-truths to predict motion vectors for each point. PillarFlow [Lee+20] outputs the grid flow vectors on BEV using the pillar features from two frames. Hplflownet [Gu+19] generates a lattice structure from two successive point clouds processed by bilateral convolutional layers with down- and up-sampling. In this way, the method can process large point clouds at once. [Bau+19] applies cylindrical projection to the point clouds, which are then processed by the fully convolutional layers in a down- and up-sampling manner. The predicted scene flow is further used for the odometry prediction. FlowNet3D++ [Wan+20f] introduces new point-to-plane and cosine distance losses to the FlowNet3D to take the geometry into account for the scene flow estimation. [Li+21c] tries to ensure motion consistency in the local neighbourhood of points on top of FlowNet3D architecture. FlowMOT [Zha+20a] estimates point motion with a FlowNet3D-like scene flow network and then uses the motion to track objects. FESTA [Wan+21e] proposes spatio-temporal point-based attention modules to overcome the sampling inconsistencies within a set and between two point clouds.

Self-supervised learning methods have also been investigated intensively for the 3D scene flow since obtaining 3D motion vectors per point is not straightforward for large datasets. [MOH20] utilises a self-supervised cycle consistency loss by propagating points forward and backwards to recover the initial point cloud. The position difference between the cycle-propagated point cloud and the original point cloud trains the network to learn the scene flow. [Zua+20] utilises a contrastive approach and the cycle consistency on the sub-sampled point regions in the upcoming frame and the flow-propagated points. FlowStep3D [KER21]

finds the global correlation between the source and target for coarse scene flow results. The rough motion estimation is refined through local refinement units iteratively, propagating flow embeddings using a GRU layer. The method utilises the Chamfer and the regularisation losses for self-supervised learning. [LZL21] makes use of shape reconstruction losses in addition to the Chamfer loss for the self-supervised 3D scene flow learning with 3D graph convolutional neural network and edge convolutions. [PHL20] constructs a graph out of points in two frames. The graph edge weights are optimised using the distances between the closest points of two frames and a smoothness function based on graph Laplacian, ensuring a similar motion in a neighbourhood. On the other hand, SLIM [Bau+21] considers the point motions in the whole scene. After the pillar-based feature encoding, an update block iteratively estimates the point flow, motion segmentation of points, and a transformation matrix between two frames based on odometry.

Considering all the introduced methods, the 3D scene flow is an extremely important multi-frame to capture motion information for the perception of the environment using a small-scale labelled or a large-scale unlabelled dataset.

## 3.9 Discussion

Most 3D object detection methods rely on single-frame data (lidar and RGB), as introduced in Section 3.1. The main reasons for that were the lack of datasets providing time-sequenced data, under-investigated attention-like alignment methods for point clouds, and computation power that can handle multiple frames simultaneously. However, using single frames causes loss of information, as shown with the methods introduced in Section 3.2 for the 2D counterparts. For several 2D perceptions tasks, using consecutive frames improves the quality of the final results. The difference between two successive frames might be high in autonomous driving due to the dynamic traffic participants and moving ego vehicle. Also, the data sampling rate affects the usability of the time-series data. However, the autonomous driving datasets provide the synchronised RGB images and lidar point clouds with *10* or *20 Hz* sampling rates [GLU12a; Cae+20; Sun+20]. On the other hand, 3D point cloud processing is geometrically more complex than 2D data processing, requiring finding correspondences between frames as introduced in Section 3.5.

In Chapter 5, we investigate scene-level multi-frame aggregation for 3D object detection. We directly make use of RGB and lidar point clouds aligned in multiple frames using the receptive fields of convolutional recurrent networks, which has also been used for 2D multi-frame detection [LZ18]. Like the RNN-based 3D multi-frame detection methods [MZ20; El +18], we investigate scene-level feature aggregation in time. Differently, we examine aggregating RGB-lidar feature maps temporally and their contribution to the 3D detection quality. Our method is one of the early approaches in multi-frame 3D object detection and shows that the 3D detection results benefit from the temporal RGB-lidar feature map fusion.

Chapter 6 proposes a multi-frame lidar-RGB fusion for object-level 3D object detection. Our method aggregates object features obtained from consecutive lidar point clouds and RGB images in time similar to the 2D counterparts [LLT17; Wu+19; Kan+17b]. Compared to the scene-level approaches introduced in Section 3.3, our method solves the feature alignment issue at the object level. We investigate the multi-modal approach for object-level features contrary to the concurrent 3D-MAN [Yan+21c] approach, which uses lidar-scans for obtaining object features and aggregating in time. Our multi-modal object-level temporal feature fusion method is the first approach that improves the 3D object detection using multi-frame RGB-lidar object features.

Chapter 7 proposes using the scene flow task for better 3D object detection. Learning

geometric point representations is vital for 3D point cloud-based downstream perception tasks introduced in Section 3.5. Due to the difficulty of obtaining annotations on 3D point clouds, unsupervised and self-supervised learning methods provide a sound basis for the point cloud representation learning, as explained in Section 3.7. Therefore, we employ a self-supervised scene flow method to train a 3D detector's backbone without annotations. Unlike the pre-training techniques [Zha+21b; Rao+21; Lia+21a; Wan+21f], we use sequential data to teach the backbone geometry and motion-related features, which are essential for the perception in a dynamic traffic scene. SLIM [Bau+21] also concurrently investigates using self-supervised scene flow for 3D downstream tasks, but the method does not consider a complex perception task such as 3D object detection. FAST3D [Fru+21] proposes using scene flow for 3D object detection to augment the point clouds from multiple frames instead of point representation learning. Unlike these approaches, we propose a simple pre-training method for the 3D object detection using a self-supervised scene flow task, which can be applied to different 3D detectors, as shown in Chapter 7.

Chapter 4

# Problem Definition

This work aims to utilise data from successive time steps for the 3D detection in the frame of interest. Given a set of data $D^t$ sampled at frame $t$, a 3D detector can be given as $B^t = h(g(D^t))$. Here, $g$ represents the feature extractor and $h$ stands for the 3D detection head, which takes the single-frame features and estimates the 3D bounding box parameters. $B^t$ is a set of 3D bounding boxes given as $B^t = \{b_i^t \in \mathbb{R}^7 | i = 1, ..., n\}$ as explained in Subsection 2.2.3. Our aim is to find a multi-frame feature extractor $f$, where $B^t = h(f(\{D^t, D^{t-1}, D^{t-2}, ...\}))$. $D^t$ can consist of different modalities such as a pair of a point cloud and an image $D^t = (P^t, I^t)$, where $P \in \mathbb{R}^{n \times 4}$ and $I \in \mathbb{R}^{H \times W \times 3}$ respectively. $n$ indicates the number of points sampled in a frame. $H$ and $W$ are the height and width of an RGB image. The similarity between the data points in the set $\{D^t, D^{t-1}, D^{t-2}, ...\}$ will decrease in a larger set since the environment constantly changes in an autonomous driving scenario. Therefore, we need to limit the number of sequential data points in the set with a fixed number of time steps, which can be defined as $S^t = \{D^t, D^{t-1}, ..., D^{t-\tau+1}\}$ for $\tau$ number of time steps. For this goal, we propose three different strategies in this work.

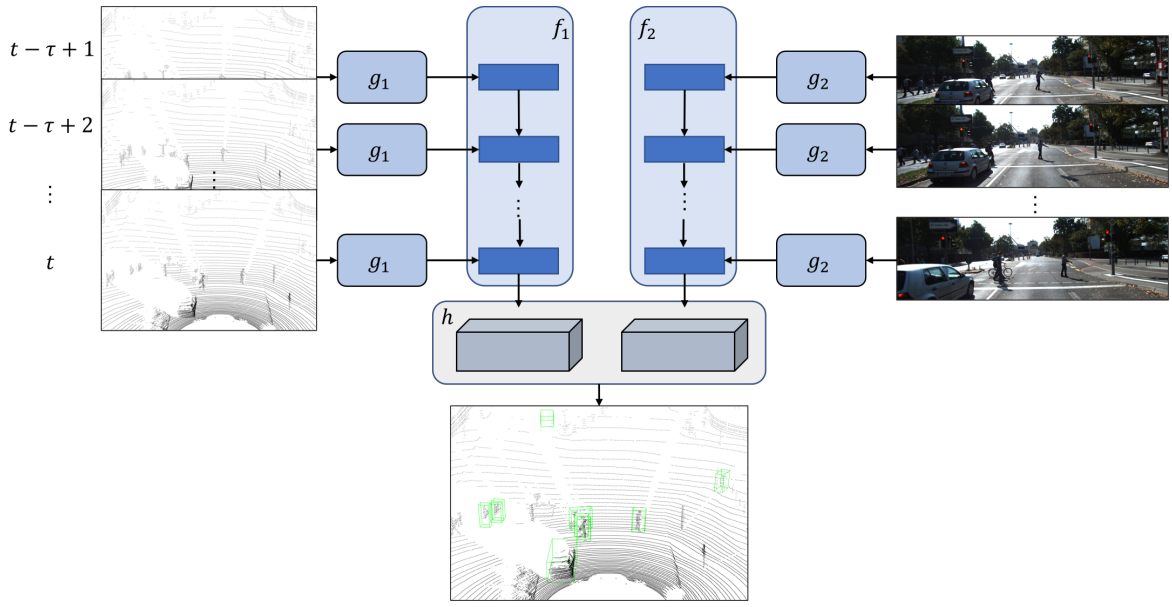



**Figure 4.1:** Multi-frame scene-level aggregation using lidar point cloud and image features. $g$ is a feature extraction network, $f$ is the multi-frame aggregation network, and $h$ is the detection head. $h$ function estimates the objects using temporal feature maps from two modalities.

In Chapter 5, we propose to temporally process sequential lidar point cloud and image feature maps for obtaining 3D detection results on the final frame such that $B^t = h(f(\{g(D^t), g(D^{t-1}), ..., g(D^{t-\tau+1})\}))$, where $f$ is the multi-frame feature aggregation function, $g$ is a single-frame feature extractor, and $h$ is the 3D detection head. The proposed method is illus-

trated in Fig. 4.1. We aim to to generate a richer representation at frame $t$ using temporally-preceding feature maps. Here we utilise the lidar point cloud-RGB image pair $D = (P, I)$ and discuss their temporal contribution together and separately. $g$ generates feature maps for the entire scene, and $f$ accounts for the feature offset between sequential scene-level feature maps due to the motion.

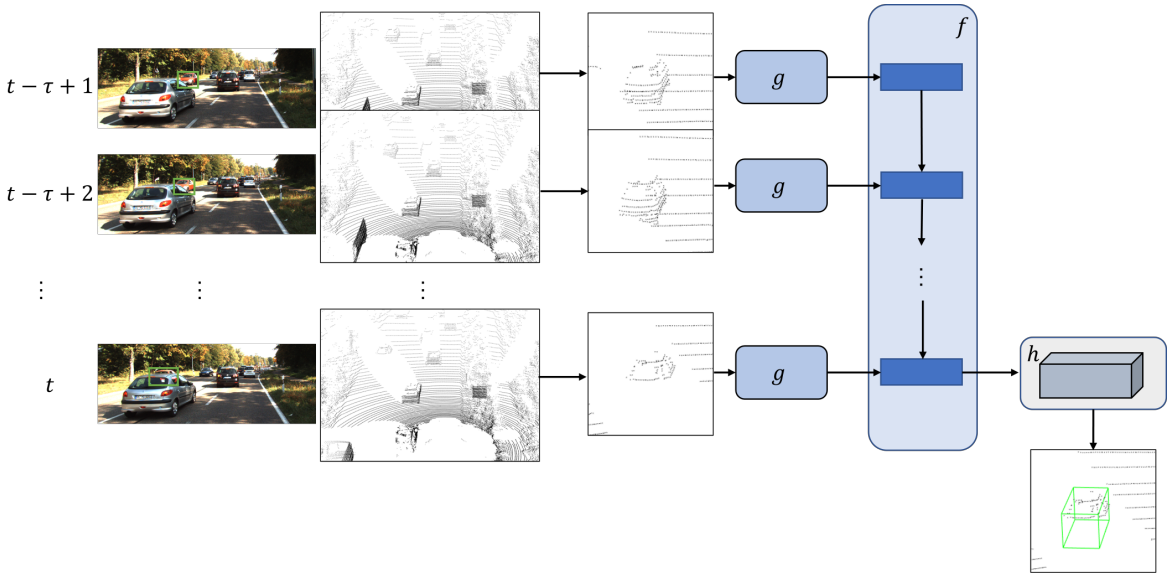



**Figure 4.2:** Multi-frame object-level aggregation using lidar point cloud and 2D RGB detections. $g$ is a regional feature extractor, $f$ is the object-level multi-frame aggregation network, and $h$ is the detection head. $h$ estimates the bounding box parameters using the temporal object feature.

In Chapter 6, we propose a temporal processing method to process sequential object-level features as illustrated in Fig. 4.2. The 3D bounding boxes are obtained with $B^t = h(f(\{g(R_i^t), g(R_i^{t-1}), ..., g(R_i^{t-\tau+1})\}))$ function. $R_i \subset D$ represents data from a smaller region, which is assumed to be from the region of an object. In this setting, $f$ learns the relation between object-level features extracted by the regional feature extractor $g$. To reach this aim, we need to define regional correspondences between successive frames. For the temporal object-level feature fusion, the regional associations are obtained using 2D detections from the RGB images and $D = (P, I)$.

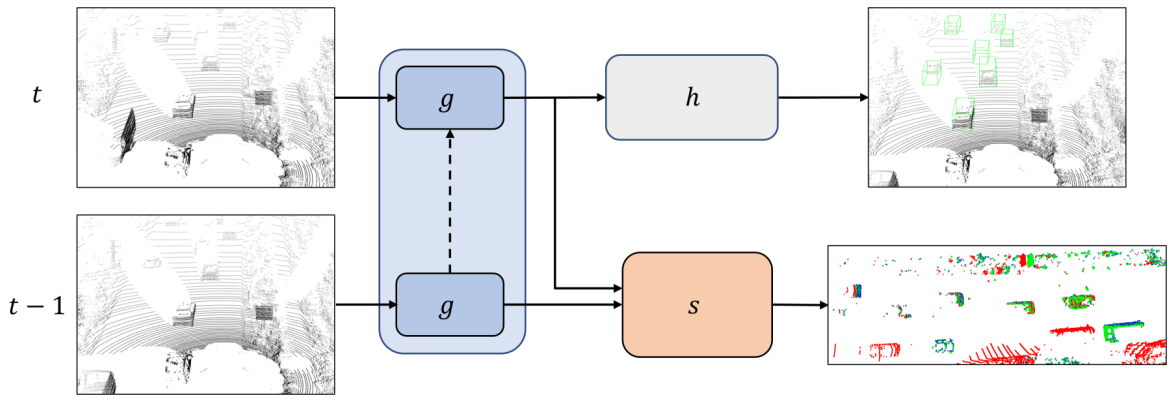



**Figure 4.3:** Scene flow-based self-supervised backbone pre-training. The backbone $g$ is pre-trained through the scene flow head $s$ that process sequential frames. The output of the $s$ network is flow estimations. Colours in the point cloud: red for frame $t-1$, blue for frame $t$, and green for propagated point clouds from $t-1$ to $t$. The pre-trained motion-aware backbone can be used for 3D object detection, which will help distinguish objects based on differences in the motion patterns. $h$ is the 3D detection head that processes motion-aware features from $g$.

Chapter 7 considers the sequential data processing idea from another perspective. Scene flow problem relies inherently on sequential point clouds such that $\mathcal{F}_{t-1\rightarrow t} = s(P^t, P^{t-1})$. $\mathcal{F}_{t-1\rightarrow t} = \{d_i \in \mathbb{R}^3\}$ represents the motion vectors of the points in frame $t-1$ to frame $t$. $s$ is the scene flow estimation network that takes two successive point clouds as input and outputs the flow estimations. In this work, we relate the scene flow problem with the 3D object detection by using learned point-wise motion representations for detecting objects in the 3D space, as summarised in Fig. 4.3. The scene flow definition can be reformulated as $\mathcal{F}_{t-1\rightarrow t} = s(g(P^t), g(P^{t-1}))$, where $g$ is the feature extractor from a given point cloud. The scene flow head, $s$, constructs the relation between two consecutive feature sets and estimates the motion vectors. We propose to use the motion-aware backbone $g$ for the 3D object detector such that $B^t = h(g(D^t))$. Thus, the 3D object detection uses the motion features provided by $g$ and learned through $s$, even providing detections from a single frame. In this way, the 3D detector can recognise the objects of interest from other objects and the background using the difference in their motion patterns embedded in $g$.

Chapter 5

# Multi-frame Scene-level 3D Object Detection

In traffic, an ego vehicle's perception changes at each time step due to its and other traffic participants' movements. As introduced in Section 3.1, most 3D detection methods rely on single frame data collected in a specific time interval. However, considering the sampling rates of sensors given in Section 2.4, two successive frames have ample overlapping space with different fields of view. We propose a method to obtain a richer representation of the current frame using the scene features from the previous time steps in addition to the last sample as summarised in Fig. 5.1.

## 5.1 Methodology

We aim to obtain a spatio-temporal feature map of the scene from sequential lidar point clouds and RGB images. The spatio-temporal feature map is more representative than the spatial feature maps obtained from a single time step since it aggregates information from several perspectives. We use a two-stage 3D object detector [Ku+18] to evaluate the proposed method. A point cloud (PC) and an RGB image sampled at time step $t$ are given as $P^t = \{\mathbf{p}_i | i = 1, ..., k\}$ and $I^t \in \mathbb{R}^{H \times W \times 3}$, respectively. Here $H$ and $W$ are the height and width of the RGB image, and $k$ is the number of points in a frame. A feature extractor, $g$, learns scene feature maps $F^t_{PC} = g_{PC}(P^t)$ and $F^t_{RGB} = g_{RGB}(I^t)$ for the lidar point cloud and the RGB image, respectively. The region proposal network (RPN), $r$, generates 3D bounding box proposals $\hat{B}^t = r(F^t_{PC}, F^t_{RGB})$ from the scene feature maps of two modalities, where $\hat{B}^t = \{\mathbf{b}^t_i | i = 1, ..., l\}$. At this point, our spatio-temporal feature aggregator, $f$, provides spatio-temporal features for two modalities such that $F^{[t,t-\tau+1]}_{PC} = f_{PC}(F^t_{PC}, F^{t-1}_{PC}, ..., F^{t-\tau+1}_{PC})$ and $F^{[t,t-\tau+1]}_{RGB} = f_{RGB}(F^t_{RGB}, F^{t-1}_{RGB}, ..., F^{t-\tau+1}_{RGB})$. The following refinement module (the second stage), $h$, outputs the 3D detections $B^t = h(F^{[t,t-\tau+1]}_{PC}, F^{[t,t-\tau+1]}_{RGB}, \hat{B}^t)$ by using the fused regional spatio-temporal features from point clouds and RGB images obtained from the region of the proposals.

### 5.1.1 Generating Inputs

For the multi-frame 3D object detection task, we extend the AVOD 3D detector [Ku+18], which takes a lidar point cloud ($P^t$) and an RGB image ($I^t$) as inputs. The lidar point cloud is converted to a BEV map instead of directly using the point coordinates as inputs. The BEV map is generated according to the height of points. The $[-40, 40] \times [0, 70]$ BEV region is split into 0.1 m-long square grids. Each grid is split into five equal height intervals between $[0, 2.5]$ meters, and the maximum height of the points in that interval is taken as the feature of that grid cell. In addition to the 5 height channels, a density channel is added, which calculates

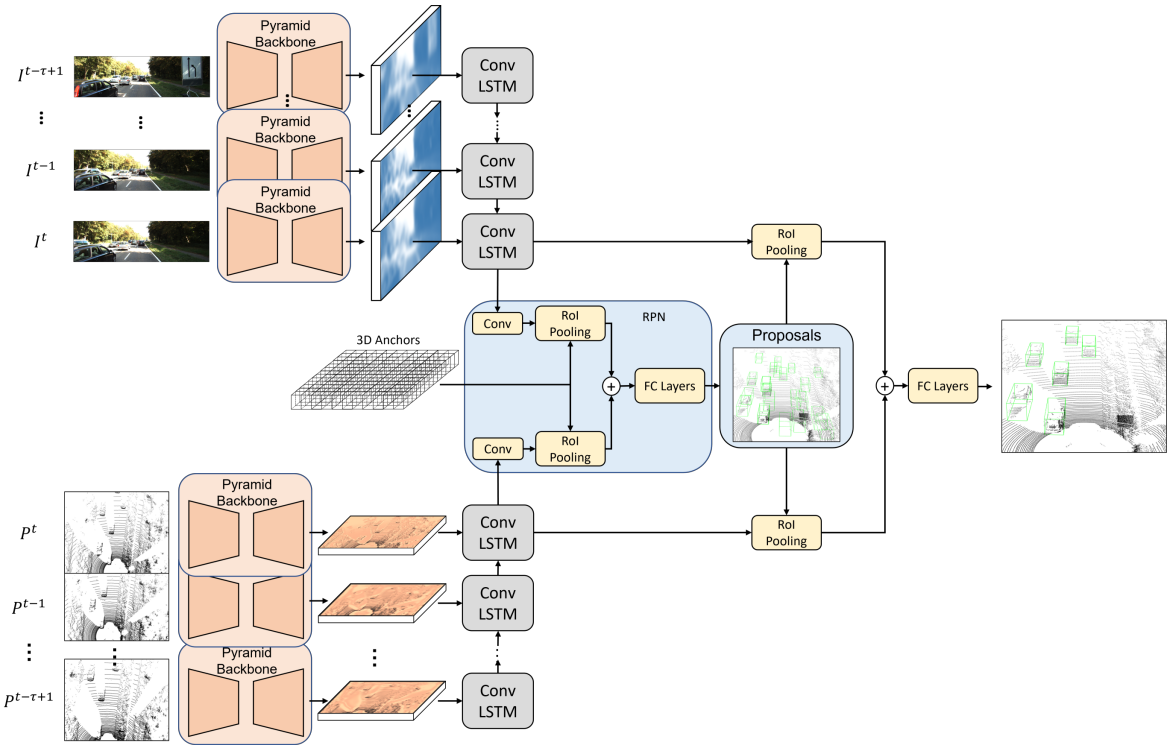



**Figure 5.1:** Our scene-level temporal aggregation method is applied to the AVOD 3D detector [Ku+18]. Instead of relying on single-frame feature maps, we generate temporal feature maps for the 3D object detection. We take sequential RGB images ($\{I^{t-\tau+1}, I^{t-\tau+2}, ..., I^{t}\}$) and BEV maps ($\{P^{t-\tau+1}, P^{t-\tau+2}, ..., P^{t}\}$) as inputs. The pyramid backbone is an encoder-decoder feature extractor. The pyramid backbone outputs feature maps for single-frames, and the Conv LSTM aggregates the single-frame feature maps for two modalities in time. The temporal feature maps are used for the RPN and 3D detection head. RPN extracts features from the non-empty anchor regions with RoI pooling and fuses image and lidar features to estimate object proposals. The 3D detection head similarly extracts regional features from the temporal feature maps using proposal boxes. Afterwards, the regional temporal features from two modalities are fused and fed into FC layers for the final 3D detections.

the point density with $min(1.0, \frac{log(N+1)}{log16})$. $N$ is the number of points in a grid cell. In this way, the point clouds are encoded with heuristic feature embeddings, which have been found as useful features [Ku+18; Che+17]. Our multi-frame 3D detector takes images and lidar point clouds the same way but from multiple sequential frames. This is shown in Fig. 5.1 with the multi-frame inputs to the Pyramid Backbone layers.

### 5.1.2 Pyramid Backbone

The AVOD backbone network ($g$) follows a pyramidal encoder-decoder architecture [Lin+17a] for learning scene features $F_{PC}$ from the BEV map and $F_{RGB}$ from the RGB image. The same backbone architecture processes the image and BEV point cloud inputs, but the backbone layers are not shared between two modalities ($g_{PC}$ and $g_{RGB}$). The encoder is a modified VGG-16 [SZ15] network and does not use the convolutional layers after *conv-4* layers of VGG-16. Therefore, the size of the final feature map is decreased to $\frac{M}{8} \times \frac{N}{8} \times D$ from the original $M \times N \times D$. This causes representing small objects (such as pedestrians) with less than a pixel in the final feature map. Therefore, a decoder with upsampling layers is used following the encoder to have a high-resolution feature map with the same size as the input map, $M \times N \times D'$. Each upsampling operation is followed by concatenation with the corresponding encoder layer of the same size and a $3 \times 3$ convolution layer. We use the same

pyramid backbone as the AVOD 3D detector (Fig. 5.1) to obtain scene-level features from the RGB images and the BEV maps.

### 5.1.3 Region Proposal Network

The region proposal network (RPN) takes the feature maps generated by the backbone for two modalities ($F_{PC}$ and $F_{RGB}$). It estimates the 3D bounding box proposals in the scene using the fused feature maps. We also keep the same RPN architecture used for the AVOD, shown inside the light blue box of Fig. 5.1. The RPN consists of three main parts: 3D anchor generation, feature fusion, and proposal estimation.

3D anchors are generated as proposal candidate boxes above the BEV ground plane. The axis-aligned 3D bounding box representation of [SX16] are scattered on the BEV map with 0.5-meter intervals as explained in Subsection 2.2.3. The 3D bounding box sizes are determined by each class's average 3D ground-truth box sizes in the training set.

$F_{PC}$ and $F_{RGB}$ feature maps are further processed by $1 \times 1$ convolutional layers to reduce the channel dimension to save memory during computation. This is followed by a Region of Interest (RoI) pooling similar to [Ren+16]. The 3D anchors, which contain at least one point, are projected onto the BEV map and RGB image plane. The corresponding feature maps from the region of projected anchors are cropped and bilinearly resized into $3 \times 3$ features. Thus, the regional anchor features from two different modalities are encoded into the same feature dimensions.

The regional features from two modalities are fused with an element-wise mean operation and fed into fully-connected (FC) layers to estimate the objectness scores and refine the anchors by offset estimations. After a non-max suppression (NMS) operation, the remaining candidate boxes are taken as 3D bounding box proposals generated by the first stage of the 3D detector.

### 5.1.4 Temporal Aggregation

The AVOD 3D object detector generates two high-resolution feature maps, $F^t_{PC}$ and $F^t_{RGB}$ for two sensor modalities. These are obtained at each time step, where $t$ denotes the time step the features generated at. The scene-level contextual information embedded in the feature maps changes at every time step due to moving objects in the scene and the moving ego vehicle. Therefore, the successive feature maps can be aggregated through time to obtain richer representations of the environment and benefit from the sensors' varying fields of view.

We aim to obtain a temporal feature map $F^{[t,t-\tau+1]}$, which is a richer representation of the scene than the current feature map. $F^{[t,t-\tau+1]}$ contains the contextual information of the frame $t$ as well as its preceding frames. This helps compensate for the loss of information in the frame $t$ due to occluded and moving-away objects with the information from previous frames. We apply the the spatio-temporal feature aggregation to the sequences of two data modalities separately such that $F^{[t,t-\tau+1]}_{PC} = f_{PC}(F^t_{PC}, F^{t-1}_{PC}, ..., F^{t-\tau+1}_{PC})$ and $F^{[t,t-\tau+1]}_{RGB} = f_{RGB}(F^t_{RGB}, F^{t-1}_{RGB}, ..., F^{t-\tau+1}_{RGB})$. The $f$ function is realised with the convolutional bottleneck LSTM layers (Conv LSTM) introduced in [LZ18]. We show the unrolled Conv LSTM in Fig. 5.1 that takes feature maps from frames $t-\tau+1$ to $t$ and outputs a final feature map $F^{[t,t-\tau+1]}$ to the RoI Pooling layer for the 3D detection head. The Conv LSTM works similar to the original LSTM units [Gre+16], except the gates are realised with convolutional instead of feed-forward layers. In this approach, the receptive fields of the convolutional filters learn how to account for the feature mismatches due to the motion offset.

### 5.1.5 3D Detection Head

The 3D object detection head is the second stage of the AVOD detector, which refines the object proposals using the feature maps from the proposal regions. The temporal feature maps $F_{PC}^{[t,t-\tau+1]}$ and $F_{RGB}^{[t,t-\tau+1]}$ from two modalities are fed into the RoI pooling layers separately to extract proposal features. Like the RoI pooling operation explained in Subsection 5.1.3, the 3D box proposals are projected onto the BEV map and the image plane to have regions of interest. Then, the RoI pooling crops and resizes the features from the projected 3D proposal regions. The extracted spatio-temporal features from two modalities are concatenated and further processed with the FC layers. The process is visualised in Fig. 5.1. The FC layers estimate the 3D bounding box offsets to refine the proposals and predict the bounding box's class scores. The 3D detections are post-processed with an NMS module to remove the redundant detections.

The final 3D bounding boxes are represented with $B^t = b_i = (c_1, c_2, c_3, c_4, h_1, h_2)$ as explained in Subsection 2.2.3. The orientation of the 3D box is encoded with the angle $\theta$, which is estimated with the pair of $(cos(\theta), sin(\theta))$. Since 4 corners encode a 3D bounding box, its heading can be one of the four directions defined by the rectangular shape of the 4 corners. The closest angle among the four directions to the estimated $\theta$ is taken as the orientation of the 3D bounding box.

## 5.2 Implementation Details

This section gives the details of the dataset, the metrics, and the hyperparameters used for the experiments, which are explained in Subsection 5.3.

### 5.2.1 Dataset

Our multi-frame scene-level 3D detection method can be trained with a dataset that consists of sequential point clouds and images. KITTI 3D object detection dataset is not appropriate for our approach since it does not contain video data, as explained in Subsection 2.4.1. On the other hand, KITTI multi-object tracking dataset consists of 21 annotated drives with sequential data. Hence, we use the KITTI multi-object tracking dataset to validate our method as the dataset details given in Subsection 2.4.2. We use the four annotated drives for validation and the rest for training.

### 5.2.2 Metrics

Average precision (AP) is the most commonly-used evaluation metric for the 3D object detection methods introduced in Subsection 2.4.5. We follow the KITTI 3D object detection practices and calculate AP per class and difficulty level through 40 recall points. As usual for the KITTI dataset, two mostly-annotated classes (car and pedestrian) are detected. We calculate the 3D AP and the BEV AP. 3D AP measures the accuracy of the 3D bounding boxes, whereas the BEV AP measures the accuracy of 3D bounding boxes projected onto the BEV plane. Therefore, BEV AP shows how accurately oriented 2D BEV bounding boxes are detected.

### 5.2.3 Loss function

The RPN and the 3D detection head are trained end-to-end with a multi-task loss. RPN estimates the offset between an anchor box and a corresponding ground-truth bounding box. Offsets are given for each bounding box parameter as $(\Delta c_x, \Delta c_y, \Delta c_z, \Delta w, \Delta h, \Delta l)$, which is the difference between the anchor parameters and the ground-truth parameters. Smooth L1 loss ($L_{off-RPN}$) is used for the proposal offset regression. In addition to the proposal box parameters, the RPN gives objectness scores for anchors to identify whether they belong to the background or the foreground. The cross-entropy loss ($L_{obj-RPN}$) is used for the objectness score training.

The 3D detection head outputs refined 3D bounding boxes, class scores, and orientation angles. Different from the RPN, the 10-parameter $(c_1, c_2, c_3, c_4, h_1, h_2)$ bounding box representation is used for the refined boxes. Therefore, the regression targets are defined with the $(\Delta c_{1x}, \Delta c_{1y}, \Delta c_{1z}, \Delta c_{2x}, ..., \Delta h_1, \Delta h_2)$, which are the differences between the proposal coordinates and the ground-truth coordinates. The corners of the proposal box and the ground-truth box are matched with the closest corner while calculating the regression targets. For the regression of the bounding box and heading angle offsets, Smooth L1 losses are used ($L_{box-Det}$,$L_{ang-Det}$). Similar to the RPN, the cross-entropy loss is used for the classification ($L_{cls-Det}$). The total loss can be given with equation 5.1. $\alpha$, $\beta$, and $\gamma$ are the loss weights for the bounding box regression, classification, and the angle regression, respectively.

$$L_{total} = \alpha L_{off-RPN} + \beta L_{obj-RPN} + \alpha L_{box-Det} + \gamma L_{ang-Det} + \beta L_{cls-Det} \tag{5.1}$$

### 5.2.4 Training

During training, the proposal boxes from RPN and the refined boxes from the detection head are associated with the ground-truth bounding boxes to calculate the losses introduced in Subsection 5.2.3. For the RPN, this association is done by projecting the 3D anchors and ground-truths onto the BEV plane and calculating 2D IoU values between them. Anchors that overlap a 3D car bounding box with a 2D IoU above 0.5 are considered objects. Anchors with a 2D IoU below 0.3 are taken as background. For the pedestrian ground-truths, the 0.5 car threshold is decreased to 0.45.

The proposal boxes are assigned to the ground-truth boxes for the 3D detection head training, similar to the RPN training. Association between the ground-truth boxes and the proposal boxes are done only if their 2D BEV IoU is above 0.65 for car class and 0.55 for pedestrian class. For the loss calculation, the weights are chosen as $\alpha = 5$, $\beta = 1$, and $\gamma = 1$.

The redundant proposal boxes and refined bounding boxes are discarded using the NMS. RPN keeps 1024 proposals for all classes with a 2D BEV IoU threshold of 0.8 for the training. During inference, 300 box proposals for the car class and 1024 for the pedestrian class are kept. The refined bounding boxes from the 3D detection head are post-processed by the NMS with a 0.01 threshold to remove the redundant boxes.

The RPN has two FC layer heads for regression and classification heads. Each regression and classification head is a two-layer FC network with $[256, 256]$ units. The 3D detection head has three FC layers with $[2048, 2048, 2048]$ units. The final output regresses the bounding box parameters and orientation and provides the class scores.

The crop and resize operations in the RoI pooling stage resize the cropped features into $3 \times 3$ dimensions for the RPN and $7 \times 7$ dimensions for the 3D detection head. The $1 \times 1$ convolution operation for RPN decreases the number of channels of the feature map to 1, whereas, for the 3D detection head, the feature map number of channels are kept as $D' = 32$.

The network parameters are optimised using the Adam optimiser with a $10^{-4}$ learning rate, which is decayed by 0.8 for every $30k$ iterations. In total, the training takes $120k$ iterations. Tensorflow is used for training and validation. For each step, a mini-batch of 512 RoIs for RPN and 1024 RoIs for the detection head is used to train the network end-to-end. Separate networks are trained for the car and pedestrian classes. The input BEV map is constructed to have $700 \times 800 \times 6$ size.

## 5.3 Experiments

We conduct several experiments to show the usefulness of our scene-level multi-frame approach for the 3D object detection task. In our experiments, we use the KITTI tracking dataset. In our first experiment, we compare our multi-frame AVOD with the baseline AVOD [Ku+18]. Our multi-frame AVOD utilises the temporal feature maps for the RPN and the 3D detection head. We provide our results for the car and pedestrian classes on the KITTI tracking validation set. In the following ablation experiment, we investigate the contribution of temporal feature maps from the image and the point cloud modalities separately. For this experiment, we use the single-frame feature map for one modality and the multi-frame feature map for the other modality. In a second ablation study, we increase the Conv LSTM kernel size to determine the effect of the receptive field size on the results. We compare our method's car detection results with the baseline on the KITTI validation set for the ablation study. Moreover, we add another convolutional layer to the baseline's feature extractor to match the architecture depth of the multi-frame version. Hence, we can argue that the improvement arises from the temporal processing but not increased depth.

## 5.4 Results & Discussion

This section shows our main quantitative and qualitative results and discusses them. Following the direct comparison of our method with the baseline on car and pedestrian classes, we also ablate our approach on the car class for each validation drive to observe the effect of scene-level temporal fusion for two modalities.

### 5.4.1 Quantitative Results

Tables 5.1 and 5.2 show car and pedestrian 3D and BEV detection scores for the AVOD baseline [Ku+18] our multi-frame AVOD. We trained both networks on the KITTI tracking training set and obtained results on the KITTI tracking validation set with $AP_{R_{40}}$ metric. Our multi-frame approach outperforms the baseline with a large margin for all difficulty levels for car and pedestrian classes.

Our multi-frame AVOD outperforms the baseline with a large margin on the 3D detection task on the car class. The most significant improvement is in the easy difficulty level with a 9% increase, but enhancements in the moderate and hard levels are as considerable with 7% and 4.5%, respectively. Compared to the 3D detection task, improvement on BEV detection is limited. We think that the single-frame version can already reach good detection results on the BEV since it is a relatively more straightforward task than the 3D detection, where only a 2D BEV oriented bounding box is estimated instead of the 3D shape. However, our multi-frame approach shows a significant improvement in the hard difficulty level objects, which are represented with a smaller number of points or occluded on the image plane.

**Table 5.1:** 3D and BEV $AP_{R_{40}}$ car detection scores on the KITTI tracking validation set for the AVOD baseline and our multi-frame AVOD.

| Car | 3D AP | | | BEV AP | | |
|---|---|---|---|---|---|---|
| Method | Easy | Mod | Hard | Easy | Mod | Hard |
| AVOD Baseline | 55.71 | 39.02 | 35.69 | 81.05 | 63.03 | 54.39 |
| Multi-frame AVOD | **64.83** | **46.38** | **40.38** | **81.66** | **63.45** | **63.31** |
| Improvement | +9.12 | +7.36 | +4.69 | +0.61 | +0.42 | +8.92 |

For the pedestrian class, the multi-frame approach outperforms the baseline on all difficulty levels and for 3D and BEV detection tasks. Like the car class, the most significant improvement is obtained for the easy difficulty. The performance gain is considerable for all 3D detection levels, even though limited compared to the car class. This is due to the smaller region occupied by the pedestrians on the image plane and BEV feature maps. Furthermore, the BEV AP improvement is minor compared to the 3D AP.

**Table 5.2:** 3D and BEV $AP_{R_{40}}$ pedestrian detection scores on the KITTI tracking validation set for the AVOD baseline and our multi-frame AVOD.

| Pedestrian | 3D AP | | | BEV AP | | |
|---|---|---|---|---|---|---|
| Method | Easy | Mod | Hard | Easy | Mod | Hard |
| AVOD Baseline | 29.27 | 32.39 | 31.27 | 53.38 | 53.69 | 53.65 |
| Multi-frame AVOD | 31.87 | 34.07 | 32.38 | 53.58 | 53.82 | 53.74 |
| Improvement | +2.60 | +1.68 | +1.11 | +0.2 | +0.13 | +0.09 |

### 5.4.2 Qualitative Results

The qualitative results indicate that detection based on temporal feature maps enhances the 3D detection accuracy remarkably compared to the single-frame detection. Qualitative results show evidence for the situations where the multi-frame detection performs better and prevents flickers and missing objects. Fig. 5.2 presents baseline and multi-frame detection of 5 subsequent frames. The red bounding boxes are detections of the networks, and the ground-truths are not shown to see the detected boxes clearly. As shown in the figure, the baseline misses the parked cars that it previously detected. However, the multi-frame network catches the parked cars correctly through the sequence.

Fig. 5.3 shows another scene where the baseline fails to detect a previously-detected car. Read bounding boxes indicate the detections. In this scenario, pedestrians cross the street in front of the ego vehicle, which impedes the far-away car. On the other hand, the multi-frame AVOD successfully detects the same car in consecutive frames. This shows that accumulating feature maps help detect objects under occlusions. Similarly, the baseline misses the far-away car, which is represented with a smaller number of points than other cars, in the middle frame of Fig. 5.4. Contrarily, the multi-frame AVOD detects the car in all frames.

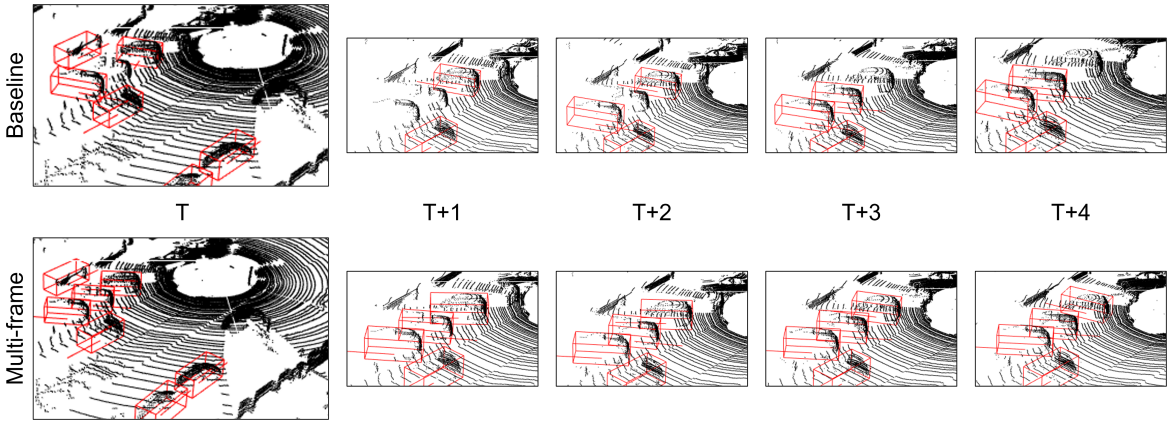



**Figure 5.2:** Qualitative comparison between the baseline and the multi-frame AVOD detectors. Baseline misses all of the previously-detected parked cars at least for one frame. On the other hand, the multi-frame AVOD can consistently detect all the parked cars through the subsequent frames.

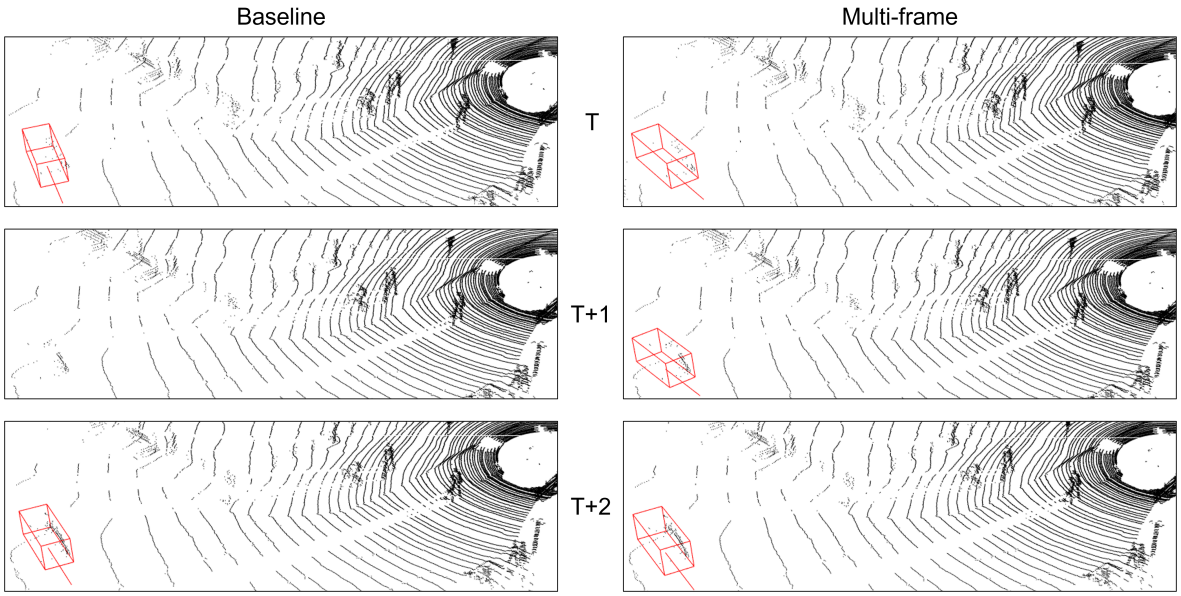



**Figure 5.3:** Qualitative comparison between the baseline and the multi-frame AVOD detectors. Red bounding boxes are the detections. Baseline AVOD misses the previously-detected car at $T+1$, which which the pedestrians occlude. However, multi-frame AVOD consistently detects the same car in consecutive frames.

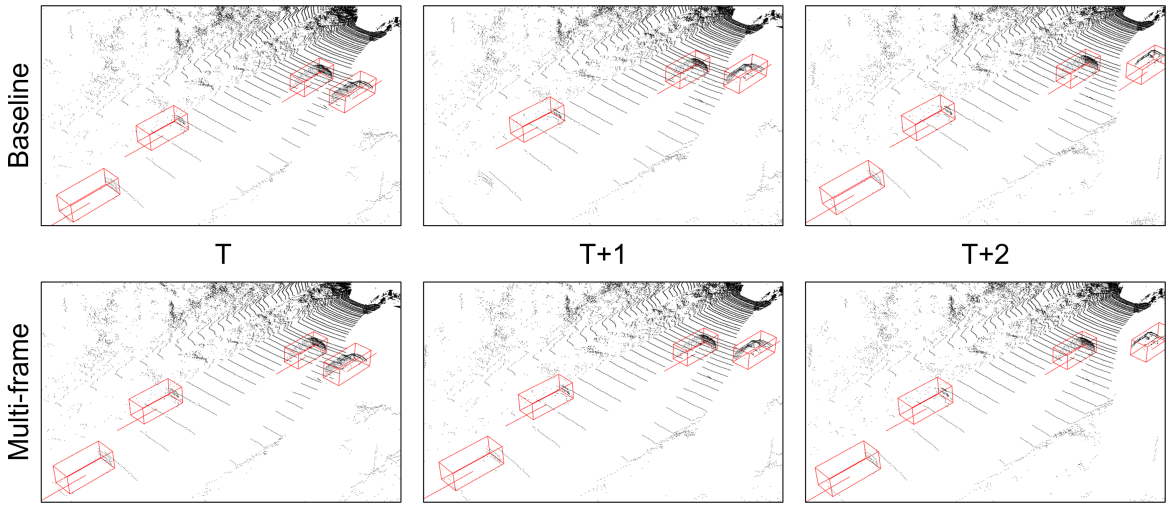



**Figure 5.4:** Qualitative comparison between the baseline and the multi-frame AVOD detectors. Red bounding boxes are the detections. The baseline could not detect the far-away car in the middle frame, but multi-frame AVOD successfully sees it.

### 5.4.3 Ablation Studies

In this section, we compare the contribution of two modalities to the multi-frame detection results by ablating one of the modalities. For this experiment, we use the single-frame feature map in one data modality (image or lidar BEV) while keeping the multi-frame feature map for the other modality. We report our results for the car class on the KITTI validation set. Each model is trained for four times, and the checkpoint best performing on the moderate difficulty level is considered for the comparison. We also provide the average metrics of all training sessions in a separate table. In a second ablation study, we compare the effect of the Conv LSTM receptive field by increasing the convolution kernel size.

Table 5.3 shows the 3D $AP_{R_{40}}$ comparison between the baseline and the ablated versions of our multi-frame method (Ours). We give the AP scores obtained for the baseline. The scores of the multi-frame AVOD versions are presented with delta values on top of baseline scores. Results of the only temporal image or lidar feature maps are given separately. The results show that adding the Conv LSTM only to the image feature map branch improves all difficulty levels' baseline 3D detection scores.

On the other hand, adding the Conv LSTM to only the lidar feature map branch consistently improves the 3D AP scores. The best 3D detection results for the easy and hard difficulty levels obtained with only temporal lidar feature maps maps surpass the complete multi-frame network with a small value. The contribution of the temporal image branch is still apparent, which can be seen from the moderate level for the complete model. The contribution of the temporal image features is 0.65% points. Our ablation study shows that the biggest BEV AP improvement for the hard difficulty level is obtained with the temporal image feature maps. The improvement using the temporal lidar feature map over the baseline is minor. Moreover, the full model consisting of both the temporal image and lidar features perform consistently better than the baseline for all difficulty levels, unlike the ablated multi-frame versions.

In Table 5.4, we investigate how consistent the training of multi-frame AVOD is. For this, we train each multi-frame version and the baseline for four times and provide the average AP scores here. Using single-modality temporal feature maps performs consistently better than the baseline, as indicated with the average scores. The improvement obtained with the temporal lidar feature map is more than with the temporal image feature map for 3D AP. The

**Table 5.3:** 3D $AP_{R_{40}}$ car detection results for ablated multi-frame AVOD on the KITTI tracking validation set.

| | Temporal Feature Map | | 3D AP | | | BEV AP | | |
|---|---|---|---|---|---|---|---|---|
| Method | Image | Lidar | Easy | Mod | Hard | Easy | Mod | Hard |
| Baseline | | | 55.71 | 39.02 | 35.69 | 81.05 | 63.03 | 54.39 |
| Ours | ✓ | | +2.45 | +5.01 | +2.82 | +0.53 | +0.38 | +8.72 |
| Ours | | ✓ | **+9.34** | +6.71 | **+4.96** | +0.6 | +0.27 | +0.02 |
| Ours | ✓ | ✓ | +9.12 | **+7.36** | +4.69 | **+0.61** | **+0.42** | **+8.92** |

BEV AP scores are similar except for the hard difficulty level, which was also the case for the best-performing checkpoints given in Table 5.3. On average, the complete temporal model significantly performs better than the ablated models on all categories.

**Table 5.4:** Average 3D $AP_{R_{40}}$ scores from four times-repeated trainings for car class. The ablated versions of multi-frame AVOD is compared with the baseline AVOD on the KITTI tracking validation set.

| | Temporal Feature Map | | 3D AP | | | BEV AP | | |
|---|---|---|---|---|---|---|---|---|
| Method | Image | Lidar | Easy | Mod | Hard | Easy | Mod | Hard |
| Baseline | | | 55.79 | 38.75 | 35.69 | 79.97 | 62.00 | 54.49 |
| Ours | ✓ | | +1.17 | +5.22 | +2.94 | +1.57 | +1.35 | +5.69 |
| Ours | | ✓ | +7.55 | +6.40 | +4.05 | +1.62 | +1.28 | +2.83 |
| Ours | ✓ | ✓ | **+9.54** | **+6.90** | **+4.06** | **+5.87** | **+2.30** | **+7.86** |

Table 5.5 demonstrates the ablation results of Conv LSTM's convolution kernel sizes. The receptive field of the convolution operation is an important parameter to capture the changing patterns and the motion of objects through successive time steps. However, a bigger kernel size increases the number of parameters, which might cause overfitting to the data. Our results suggest that the bigger kernel size improves our multi-frame AVOD in the 3D AP scores for all difficulty levels. However, the improvement is not as evident for BEV detection.

**Table 5.5:** Comparison of multi-frame AVOD 3D $AP_{R_{40}}$ scores for the car class with different Conv LSTM kernel sizes. Results are obtained on the KITTI tracking validation set.

| | Convolutional Kernel Size | | | 3D AP | | | BEV AP | | |
|---|---|---|---|---|---|---|---|---|---|
| Method | $3 \times 3$ | $5 \times 5$ | $7 \times 7$ | Easy | Mod | Hard | Easy | Mod | Hard |
| Ours | ✓ | | | 64.83 | 46.38 | 40.38 | 81.66 | 63.45 | 63.31 |
| Ours | | ✓ | | +2.32 | +0.43 | +0.67 | -0.12 | -0.17 | -0.24 |
| Ours | | | ✓ | +0.16 | +0.69 | +4.9 | +8.84 | +0.04 | +0.01 |

The multi-frame AVOD's feature extractors have more layers than the baseline due to the convolutional LSTMs. Hence, we investigate the baseline's accuracy with the same depth. We

**Table 5.6:** Ablation by adding extra convolutional layers to the baseline to obtain the same depth with the multi-frame version. 3D $AP_{R_{40}}$ car detection results on KITTI validation set.

| | Additional Conv. Layer | | | 3D AP | | | BEV AP | | |
|---|---|---|---|---|---|---|---|---|---|
| Method | $1 \times 1$ | $3 \times 3$ | Conv. LSTM | Easy | Mod | Hard | Easy | Mod | Hard |
| Baseline | | | | 55.71 | 39.02 | 35.69 | 81.05 | 63.03 | 54.39 |
| Baseline | ✓ | | | +1.55 | +2.24 | +0.57 | +0.47 | +0.22 | +8.61 |
| Baseline | | ✓ | | +5.58 | +1.42 | 2.91 | +0.21 | -0.07 | -0.14 |
| Ours | | | ✓ | +9.12 | +7.36 | +4.69 | +0.61 | +0.42 | +8.92 |

add a $1 \times 1$ or $3 \times 3$ convolutional layer to the baseline's feature extractors. Table 5.6 shows the comparison for our ablation study on the KITTI validation set car class. The convolutional layer added to the baseline improve the 3D detection scores of the baseline. However, the improvement is much inferior to the multi-frame AVOD's increment. This shows that the accuracy boost originates mainly from the multi-frame processing but not from increasing the depth of the neural network architecture.

As a result, accumulating single-modality features in time enhances the baseline 3D detection results. The temporal lidar feature map provides a higher 3D AP score than the temporal image feature map. This is an expected result considering the accurate depth information provided by the lidar sensor. However, the temporal image feature map achieves better BEV scores, which suggest that the hand-crafted lidar BEV inputs are not optimal. Nevertheless, using both temporal feature maps improves the baseline with a large margin. Also, we think that considering larger Conv LSTM kernel sizes help capture the contextual changes from consecutive frames better. However, this might also require a regularisation for training due to the larger number of parameters.

### 5.4.4 Discussion

This chapter proposed a method to aggregate RGB image and lidar point cloud features in time to improve 3D object detection results. The scene-level feature maps obtained from two different modalities are aggregated using Conv LSTMs. The resulting temporal feature maps are used for the RPN and the 3D detection head. Our temporal aggregation method improves the 3D object detection scores significantly compared to the baseline for both car and pedestrian classes.

Comparing the multi-frame AVOD with the baseline AVOD, temporal feature maps improve the 3D AP scores for all difficulty levels. The improvement is limited for the BEV detection except for the hard difficulty level. We think the baseline already performs well on the BEV detection, which is a relatively easier task than the 3D detection. However, the multi-frame temporal feature maps provide better-quality BEV detections for the hard objects with a large margin. Since the loss function optimises only for 3D object detection without favouring a specific difficulty level, the multi-frame AVOD performs slightly better than the easy and moderate BEV detection baseline. However, the multi-frame AVOD is superior to the baseline for cars and pedestrians with a significant improvement.

The ablation studies demonstrate that the temporal lidar point cloud features improve the full multi-frame model. However, temporal image features improve the BEV detection

scores, especially for hard objects. This improvement is consistent with the average detection scores. We think that the structured multi-frame image features become more important for the far-away objects, which are represented only with fewer lidar points. In addition, the multi-frame model consistently performs better than the baseline shown with the average AP scores.

Furthermore, having a larger convolution filter size of the Conv LSTM improves the 3D detection scores while performing slightly worse than the $3 \times 3$ filter size multi-frame version for the BEV AP. A bigger filter size increases the receptive field by introducing more parameters to be learned. Therefore, we think an additional regularisation for multi-frame training with bigger kernel sizes might help improve the BEV AP scores more.

Having hand-crafted lidar point cloud features also limits the performance of the Conv LSTMs to modify lidar feature extraction to have motion awareness. This can be improved by introducing a learnable feature extractor. All in all, our scene-level aggregation with Conv LSTMs provides better-quality 3D bounding boxes shown with the quantitative and qualitative results.

Chapter 6

# Multi-frame Object-level 3D Object Detection

Aggregating information from multiple sequential scenes provides an extended environment awareness to the 3D object detection network. This process is critical when moving objects hinder a part of the view and limit sensors' fields of view. Therefore, we propose using lidar-based object features from multiple scenes for estimating 3D bounding boxes as summarised in Fig. 6.1.

## 6.1 Methodology

Our method aims to use lidar point clouds and RGB images from multiple sequential frames to estimate better 3D bounding boxes on the last frame of the sequence. We utilise a point cloud $P^t = \left\{\mathbf{p}_i | i = 1, ..., k\right\}$ and an RGB image $\mathbf{I}^t \in \mathbb{R}^{H \times W \times 3}$ obtained from the scene $t$. $k$ is the number of lidar points, $\mathbf{p}_i \in \mathbb{R}^3$ is a single lidar point. $H$ and $W$ represent the height and width of the RGB image. We require a set of 2D bounding boxes $O^t = \left\{\mathbf{o}_i^t | i = 1, ..., l\right\}$ and a set of unique object track IDs $U^t = \left\{\mathbf{u}_i^t | i = 1, ..., l\right\}$ predicted on $I^t$ by a 2D detector and tracker. $\mathbf{o}_i \in \mathbb{R}^4$ is a 2D bounding box with $l$ the number of visible objects on $I_t$. The 2D tracker, $h$, takes $O^t$ and $O^{t-1}$ with unique object IDs, $U^{t-1}$, from frame $t-1$ and provides the new set of tracking IDs defined as $U^t = h(U^{t-1}, O^t, O^{t-1})$, where $\mathbf{u}_i \in \mathbb{Z}^+$.

We want to estimate 3D bounding boxes $B^t = \left\{\mathbf{b}_i^t | i = 1, ..., l\right\}$ using our method at frame $t$, which is the last frame of a $\tau$-length RGB-lidar input sequence $S^t = \{(P^t, U^t, O^t), ..., (P^{t-\tau+1}, U^{t-\tau+1}, O^{t-\tau+1})\}$. $\mathbf{b}_i^t \in \mathbb{R}^7$ is a single bounding box defined with its 3D centre $(x, y, z)$, box sizes width, height, and length, and the orientation $\theta$.

### 6.1.1 2D Detection

The Frustum PointNet approach utilises a set of 2D bounding boxes for extracting frustum points. Therefore, we need a 2D detector $f$ to get 2D bounding boxes given as $O^t = f(I^t)$ visible on the image. The 2D bounding boxes are represented with the top left $(x_1, y_1)$ and the bottom right $(x_2, y_2)$ corner locations of a rectangle on the x- and y-axes of the given RGB image. The 2D detector works on single frames and does not consider any temporal relations between sequential frames. As a result of 2D detection, a high recall value is needed since the quality of 3D detection depends highly on detecting objects on the image. The 2D bounding boxes reduce the search space in the entire point cloud. Hence, missing a 2D bounding box means also missing the 3D object.

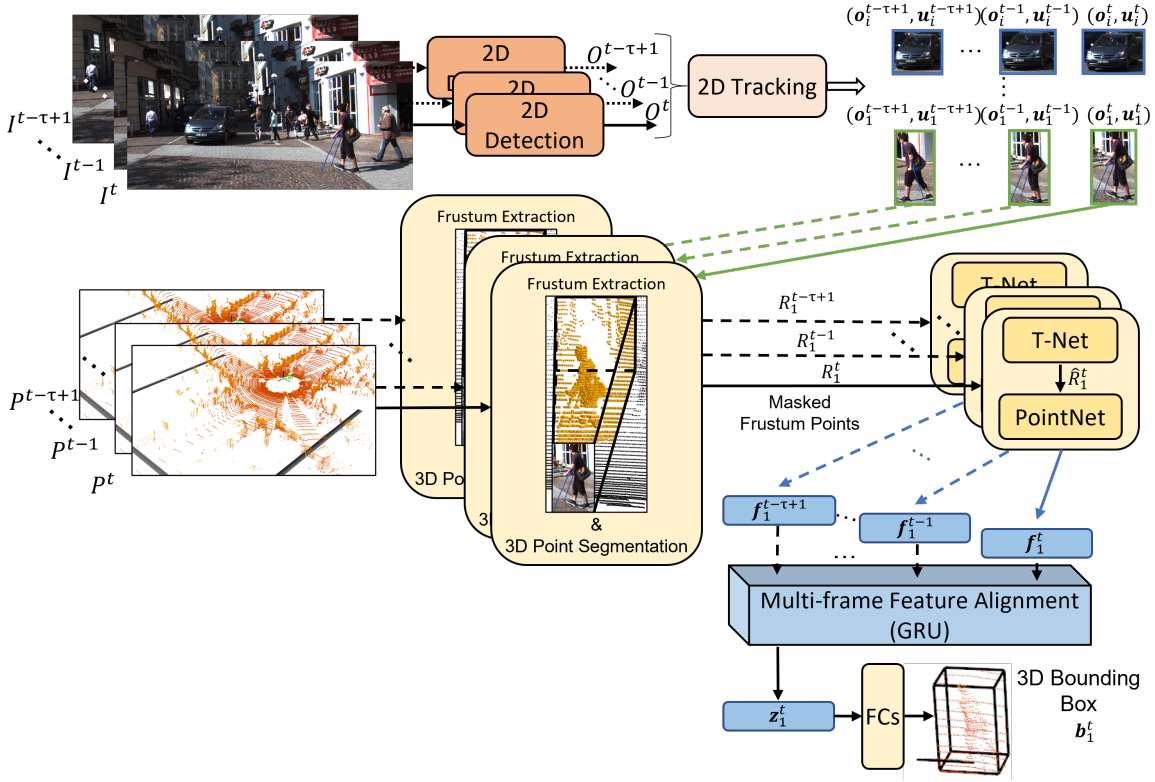



**Figure 6.1: Multi-frame object-level 3D detection:** A 2D detector and a tracker provide 2D bounding boxes $\{O^{t-\tau+1}, ..., O^t\}$ and object track IDs $\{U^{t-\tau+1}, ..., U^t\}$ for the given images $\{I^{t-\tau+1}, ..., I^t\}$ separately. The detected objects, $\{(o_i^{t-\tau+1}, u_i^{t-\tau+1}), ..., (o_i^t, o_i^t)\}$, are further processed by our method based on Frustum PointNet [Qi+18]. We show the detection process only for $i = 1$ to be concise. Lidar points in the frustum of a bounding box are extracted (orange-painted points) from the individual lidar point clouds, $\{P^{t-\tau+1}, ..., P^t\}$, using the 2D bounding boxes. The 3D point segmentation module predicts object point sets, $\{R_1^{t-\tau+1}, ..., R_1^t\}$, on the given frames leaving the background points out. The following T-Net and PointNet take the masked frustum points and outputs object-level features $F_1^{[t-\tau+1,t]} = \{f_1^{t-\tau+1}, ..., f_1^t\}$ of the object $\{o_1^{t-\tau+1}, ..., o_1^t\}$. A GRU-based multi-frame feature alignment module fuses the object feature vectors in $F_1^{[t-\tau+1,t]}$. A set of FC layers estimates the 3D bounding box parameters $b_1^t$ using the multi-frame feature, $z_1^t$, of the object $o_1^t$.

### 6.1.2 2D Tracking

Our method processes the $\tau$-length sequential object-level features to obtain more representative object features and better 3D bounding box estimation. Therefore, we need to match detected objects at frame $t$ to the detected objects in the previous frames. In this way, we can use the same object's features from multiple frames to obtain a better feature vector for that object.

We achieve unique track IDs on the 2D data using a 2D tracker, which is not a part of the Frustum PointNet [Qi+18] architecture. So, a 2D tracker takes 2D bounding boxes in two successive frames, $O^t$ and $O^{t-1}$, with the object track IDs in the previous frame, $U^{t-1}$, and provides the track IDs $U^t$ for the objects at frame $t$. The top row of Fig. 6.1 summarises the 2D detection and tracking process.

### 6.1.3 Single-frame Frustum-level RGB-Lidar Fusion

Frustum PointNet is a single-frame 3D detection network that processes an RGB image-lidar point cloud pair sampled at one time step. The RGB image is fused with the lidar data to scale down the lidar point cloud $P_i$ into a set of frustum points using the object 2D bounding box $o_i$. The frustum point cloud can be defined as $P_i = \{p_i | i = 1, ..., m\}$, where $m$ is the number of points in the new set.

The PointNet [Qi+17a] layers are order-invariant and take a fixed number of inputs. Therefore, $k$ number of points are sampled from $P_i$ before processing. The sampling strategy of the point cloud affects the object features and the 3D detection results. Also, point sampling serves as an augmentation method, when $m > k$. If $m \leq k$, $P_i$ is re-sampled. We sample lidar points randomly following [Qi+18].

### 6.1.4 Background of Frustum PointNet [Qi+18]

The Frustum PointNet architecture consists of two main models. The v1 model is based on the PointNet[Qi+17a], and the v2 model is based on the PointNet++[Qi+17b]. The network aims to estimate amodal 3D bounding boxes from a set of points $P_i$ from the entire point cloud $P^t$. The 3D detector works only on single frames and takes an RGB image and a lidar point cloud from only a time step.

The architecture consists of 3 main parts: frustum generation, point-based segmentation, and amodal 3D box estimation module. The frustum generation part provides a set of object-related point clouds, $P_{O^t} = \{P_1, P_2, ..., P_l\}$, from the frustum of the objects' 2D bounding boxes, $O^t$. A separate 2D detector outputs 2D object bounding boxes. The frustum point set represents the object of interest given with the 2D bounding box. However, the object frustum still contains background points and points of other objects, as seen in Fig. 6.1 with the orange-painted points in the Frustum Extraction box.

In the point-based 3D instance segmentation part, a 3D Point Segmentation module aims to eliminate the irrelevant frustum points. This module predicts the probability of being a part of the object of interest for all frustum points, $P_i^t$. The probabilities are used to mask the frustum points. The remaining points $R_i^t = \{p_i | i = 1, ..., n\}$ are assumed to be obtained from the object of interest that the final amodal box estimation module can estimate correct 3D bounding box parameters. $n$ is the number of remaining and $n \leq k$.

A T-Net and an amodal 3D box estimation PointNet construct the amodal 3D box estimation module. The T-Net predicts a new coordinate system for the $R_i^t$ by estimating the centre

residuals for the actual centre of the object of interest. The amodal 3D box estimation PointNet further processes the translated points $\hat{R}_i^t$ to obtain the object-level feature $\mathbf{f}_i^t$. We call $\mathbf{f}_i^t$ as an object-level feature since the feature vector is obtained from a set of points highly-related to the object of interest, $\mathbf{o}_i^t$. The amodal 3D box estimation PointNet consists of a set of multi-layer perceptrons (MLPs) and a max-pool operation, which downscales features of multiple points into a single feature vector. Two following FC layers are applied further to the $\mathbf{f}_i^t$ for the prediction of 3D bounding box $\mathbf{b}_i^t$.

### 6.1.5 Temporally Fusing Object-level Features for Refining 3D Detection

The 3D object detection modules utilise a subset of point clouds for estimating the parameters. However, the sampled points can change with multiple factors such as the density in the traffic, field of view of the ego vehicle, and the occlusion state of objects. These factors can also decrease the quality of the object feature vector of an object,$\mathbf{f}_i$, which is an embedding of the estimated bounding box parameters.

As a solution to above-mentioned problem, we propose having a function $g$, which maps a set of object-level features $F_i^{[t-\tau+1,t]} = \{f_i^{t-\tau+1}, ..., f_i^t\}$ from the same object to a temporal object feature $\mathbf{z}_i^t$ such that $\mathbf{z}_i^t = g(F_i^{[t-\tau+1,t]})$. $\tau$ represents the sequence length. We realise our multi-frame feature alignment module, $g$, with the gated recurrent units (GRUs) to process the set of features in time, as shown at the bottom of Fig. 6.1. Another FC layer is applied to the output of the GRU cells to construct the temporal object feature vector $\mathbf{z}_i^t$.

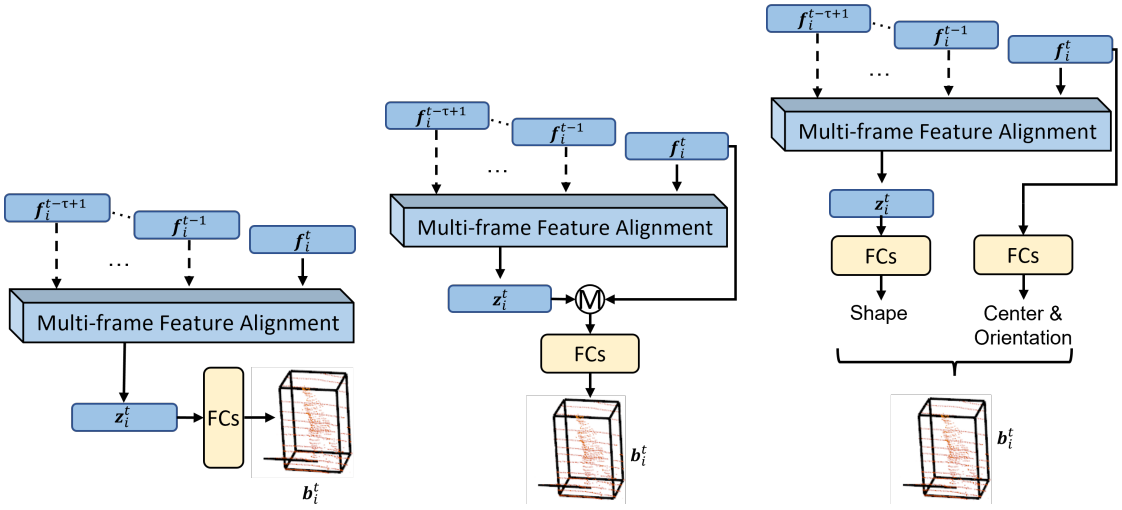
To provide the location awareness, we also utilise the centre residuals estimated by the T-Net as a part of the $\mathbf{f}_i^t$ and investigate its effect on the results. We assume that adding this information to the feature vector provide awareness of the centre shift of the bounding box. For the experiments, we concatenate the centre residuals, $\mathbf{r}_i \in \mathbb{R}^3$, with the $\mathbf{f}_i^t$.

### 6.1.6 Multi-frame Feature Alignment Strategies

We expect the temporal feature vector $\mathbf{z}_i^t$ to be inherently more representative since it is constructed by a set of feature vectors from multiple frames. However, the resulting $z_i^t$ can be misleading if the information from previous frames is not representative enough. Therefore, we consider different strategies to use $\mathbf{z}_i^t$ for the estimation of bounding box parameters.

The straightforward strategy is to use $\mathbf{z}_i^t$ for directly estimating the bounding box parameters by applying the following FC layers, as shown in Fig. 6.2a. This strategy is named one branch (OB) in the figure and the experiments. On the other hand, we can still benefit from the object feature vector $\mathbf{f}_i^t$ since it contains information related to its current state. As a second strategy, we apply a *mean* operation to $\mathbf{z}_i^t$ and $\mathbf{f}_i^t$. The resulting feature vector is processed again with the same FC layers shown in Fig. 6.2b for the bounding box parameter estimation.

Either the movement of the ego vehicle or other traffic participants cause a change of bounding box centres and orientations. At the same time, the shape parameters remain the same between frames for the same object. Therefore, we predict the object shape using the $\mathbf{z}_i^t$ and the rest of the parameters using the $\mathbf{f}_i^t$ as a third alignment strategy. The features are processed by FC layers separately to estimate parameters, as shown in Fig. 6.2c. We concatenate the two outputs for constructing the bounding box parameters. Predicting shape parameters using $\mathbf{z}_i^t$ has an auxiliary effect on training shared layers between $\mathbf{f}_i^t$ and $\mathbf{z}_i^t$. Therefore, using the multi-frame feature also trains for the centre and orientation estimation and for obtaining a more representative object feature vector $\mathbf{f}_i^t$ from the preceding MLPs.




**(a)** One branch (OB) **(b)** Two branch (TB) **(c)** Ours

**Figure 6.2:** Multi-frame feature alignment strategies for the temporal features.

We call this approach Ours in the experiments and results. None of the strategies requires additional trainable parameters, and we use the same FC-layer shape for all of them.

## 6.2 Implementation Details

In this section, we explain the details about our implementation, namely the dataset, the metrics, the loss functions, and our training setup. We also describe our experiments in this section and provide their results in Section 6.4.

### 6.2.1 Dataset

The KITTI 3D object detection dataset has been commonly used to benchmark the 3D object detection methods. However, the dataset does not provide sequential lidar data that we can use for the training and validation of our approach. Therefore, we utilise the KITTI multi-object tracking dataset with the same train-val split as explained in Section 6.4.

### 6.2.2 Metrics

The average precision (AP) metric is commonly accepted for evaluating 3D object detection networks. As suggested for the KITTI 3D object detection dataset evaluation[1], we use the AP calculated with 40 recall points as explained in [Sim+19b]. Similarly, the car IoU threshold is taken as 0.7. Pedestrian and cyclist IoU thresholds are 0.5.

[1] http://www.cvlibs.net/datasets/kitti/eval_object.php

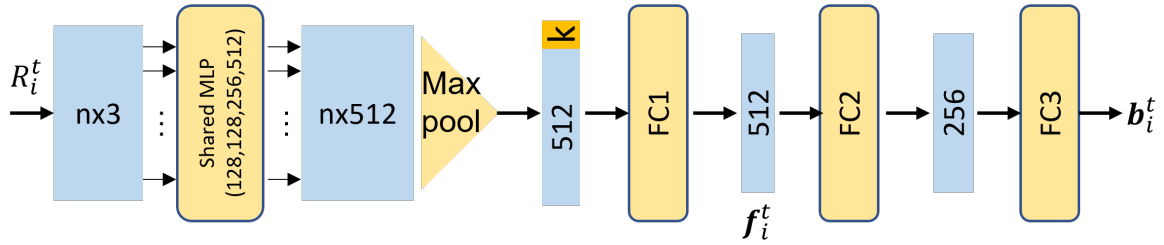



**Figure 6.3:** The architecture of the Amodal 3D Box Estimation PointNet of Frustum PointNet v1 model [Qi+18]

### 6.2.3 Loss function

The Frustum PointNet [Qi+18] architecture optimises the trainable network parameters using a multi-task loss function for the point segmentation and detection tasks. We mainly use the original loss function and, at the same time, apply a cosine distance loss to the sequential object features.

The Frustum PointNet multi-task loss function optimises different parts of the network end-to-end. The final loss function is calculated with the mask, centre, heading class, size class, heading regression, size regression, the T-Net centre regression, and the corner losses as indicated with $L_{mask}$, $L_{centre}$, $L_{h-c}$, $L_{s-c}$, $L_{h-r}$, $L_{s-r}$, $L_{T-c}$, and $L_{corner}$, respectively. Softmax loss is used for the classification, and smooth-L1 loss is used for the regression losses. The cosine distance loss is calculated between the two successive object features $\mathbf{f}_i^t$ and $\mathbf{f}_i^{t-1}$ to force a consistency between the features generated by a different set of points. The loss calculation is given in equation 6.1. $\upsilon$ and $\omega$ are the sequential feature vectors used for the distance calculation, and $l$ is the vector dimension.

$$L_{cos}(\upsilon,\omega) = 1 - \frac{\sum_{k=1}^{l} \upsilon_k \omega_k}{\sqrt{\sum_{k=1}^{l} \upsilon_k^2}\sqrt{\sum_{k=1}^{l} \omega_k^2}} \tag{6.1}$$

The final multi-task loss function is given in equation 6.2. $\alpha$, $\beta$, and $\gamma$ define the weights for the 3D box losses, the corner loss, and the cosine distance loss, respectively.

$$L_{total} = L_{mask} + \alpha(L_{centre} + L_{h-c} + L_{s-c} + L_{h-r} + \quad L_{s-r} + L_{T-c} + \beta L_{corner}) + \gamma L_{cos} \tag{6.2}$$

### 6.2.4 Training

The architecture of the amodal 3D box estimation PointNet is given in Fig. 6.3 as implemented in Frustum PointNet. The shared MLP, with (128,128,256,512) unit layers, generates the point features from the segmented points. These are reduced to the object-level features with a max pool operation. $k$ represents the number of object classes, and the estimated object probabilities are concatenated with the object-level features. The resulting tensor is fed into three FC layers. We take the output of FC1 as the object feature $\mathbf{f}_i^t$. Features of the same object from the previous frames, $\{f_i^{t-\tau+1}, ..., f_i^{t-1}\}$, are fed into the network from memory. In the ablation studies, we also take the outputs of max pool operation and the FC2 as the object features and provide the results. FC2 and FC3 layers in Fig. 6.3 are shown as FCs in Fig. 6.2. The rest of the network is the same as the Frustum PointNet v1 model. The multi-frame feature alignment module (Fig. 6.2) contains a GRU layer with the same number of units as the preceding FC layer, followed by an FC layer. We only use $(x, y, z)$ values of the points, $p_i$, without the reflection value.

During training and validation, 1024 points are randomly sampled from the frustum of a 2D bounding box. We train the model with a batch size of 32 for 100 epochs using the Adam optimiser with a momentum value of 0.9 and a learning rate of 0.001. Since we are interested in using the object-level features from multiple frames, we take 2D ground-truth bounding boxes and track IDs while extracting frustums of objects and matching them in successive frames. We train the network with each parameter set at least five times and take the best-resulting checkpoint for evaluation. The training and validation steps took place in a Docker container, which runs Python 3.6 and Tensorflow 1.15, and on a single Nvidia RTX 2080 GPU with an Intel Xeon E3-1225 v5 3.30GHz CPU.

## 6.3 Experiments

First of all, we compare our method with the Frustum PointNet baseline on KITTI Tracking Dataset for car, pedestrian, and cyclist classes using the 3D AP metric. Additionally, we provide results of several ablation studies as explained below to make sure that the multi-frame data aggregation is beneficial for a better 3D object detection:

1. We utilise perturbed ground-truth 2D bounding boxes with the ground-truth track IDs for the state-of-the-art comparison. As a first ablation, we take the ground-truth 2D bounding boxes and predict the track IDs using SORT [Bew+16] and Deep SORT [WBP17] 2D tracking methods.

2. We conduct another ablation study to determine the sequence length ($\tau$). The $\tau = 1$ means adding another FC layer instead of the GRU to ensure that the improvement does not originate from the additional depth in the network.

3. The sequential object-level features, $F_i^{[t-\tau+1,t]}$, can be fused with different strategies as explained in Subsection 6.1.6. We also provide the results from the proposed strategies.

4. To force the object features, $\mathbf{f}_i^t$, from successive frames to be similar, we apply cosine distance loss and provide results.

5. For our main experiments, we utilise GRU layers for processing sequential inputs. We compare our main results obtained using GRU cells with the LSTM and convolutional layers to process the multi-frame inputs. The LSTM layer has the same number of units as the GRU layer. For the convolutional approach, we utilise two convolutional layers for obtaining $\mathbf{z}_i^t$ from $F_i^{[t-\tau+1,t]}$.

6. The main results are obtained by taking object-level features from the FC1 layer. However, it is possible to take the features after the max pool operation or the FC2 layer. The max pool output gives a 515-dimensional vector for $k = 3$, and the FC2 output is a 256-dimensional vector. We call the max pool version the $global$, and the FC2 version the $fc2$ in the ablation results. We keep the same structure for $global$ and $fc2$ as the $fc1$. Hence, after obtaining $\mathbf{z}_i^t$ from the $global$ features, we still apply the FC1, FC2, and FC3 layers to get bounding box estimation.

7. Our primary approach is based on object-level features. The Segmentation PointNet predicts the points more related to the object of interest. To compare this with the object-level multi-frame feature alignment, we skip the segmentation part and obtain object features directly from the frustums, which we call scene-level in the results.

**Table 6.1:** Pedestrian and Cyclist 3D AP results on the KITTI tracking validation data with IoU=0.5.

| Method | Pedestrian | | | Cyclist | | |
|---|---|---|---|---|---|---|
| | Easy | Moderate | Hard | Easy | Moderate | Hard |
| SECOND [YML18] | 57.1 | 56.3 | 49.4 | 83.1 | 57.2 | 54.6 |
| PointPillars [Lan+19] | 54.8 | 54.3 | 48.2 | **86.7** | 43.9 | 43.2 |
| MVXNet [SZT19] | 38.5 | 40.9 | 35.2 | 68.6 | 32.4 | 32.6 |
| AVOD-FPN [Ku+18] | 32.8 | 33.6 | 32.3 | 21 | 13.1 | 9.1 |
| Frustum PointNet v1 [Qi+18] | 56.8 | 56.5 | 50.4 | 81.3 | 59.4 | 58.8 |
| Ours | **67.5** | **60.4** | **53.6** | 82.5 | 65.3 | 65 |
| Ours (w/ Cent) | 64.5 | 59.4 | 52.7 | 85.1 | **74.5** | **67.3** |

## 6.4 Results & Discussion

This section provides the main quantitative and qualitative results for our multi-frame object-level feature aggregation method and the ablation results introduced in Section 6.3.

### 6.4.1 Quantitative results

Table 6.1 compares our method's 3D detection performance with the baseline and the other 3D detectors in the literature for pedestrian and cyclist classes. *Ours* is our main approach explained in Subsection 6.1.6. The only difference of *Ours (w/ Cent)* is using the centre predictions from T-Net as described in Subsection 6.1.5. As shown in Table 6.1, our method outperforms the baseline in all difficulty levels for the pedestrian and cyclist classes with a large margin. Also, our approach provides better results than the compared state-of-the-art 3D detectors in most of the difficulty levels for both classes. For the cyclist class, adding centre predictions (*Ours (w/ Cent)*) to the object-level features improves the results further on top of our multi-frame feature aggregation method (*Ours*). As stated in Subsection 2.4.2, the cyclist class is the least represented class in the KITTI tracking dataset. We assume that the centre-awareness better represented bounding box locations in the lack of data. In addition, pedestrian and cyclist classes reflect fewer points since their shapes are smaller and more prone to occlusions than the cars. We give the results on car class in Table 6.2. Our method still outperforms the baseline Frustum PointNet for the moderate difficulty level. However, the performance difference is less evident for the car compared to the other classes.

### 6.4.2 Qualitative results

We also check the particular cases that our method outperforms the baseline on the KITTI tracking validation set. Fig. 6.4 shows a scenario that two pedestrians approach each other. The left-most inline drawing shows a larger region, and the right-most four scenes show the zoomed-in region of pedestrians as a sequence. The ground-truths are visualised only in the first inline drawing for clarity. The baseline Frustum PointNet on the first row confuses the two pedestrians and locates one of them wrongly. However, our method can localise the pedestrians correctly and keep the correct bounding boxes through the sequence.

Similar to the pedestrian case, we visualise the comparison for the cyclist class between the baseline and our method in Fig. 6.5. The left-most scene shows a broader range, and

**Table 6.2:** Car class 3D AP results on the KITTI tracking validation data with IoU=0.7.

| Method | Car 3D AP | | |
|---|---|---|---|
| | Easy | Moderate | Hard |
| MVXNet [SZT19] | **88.4** | 59.8 | 57.5 |
| AVOD-FPN [Ku+18] | 59.2 | 43.6 | 38.4 |
| Frustum PointNet v1 [Qi+18] | 83.1 | 65.4 | **65.1** |
| Ours | 83.1 | **72.2** | **65.1** |
| Ours (w/ Cent) | 83.7 | 65.6 | 64.1 |

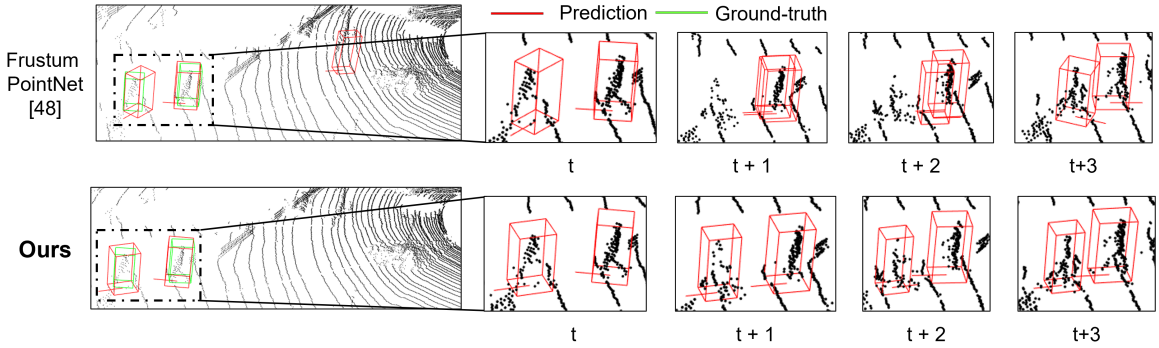



**Figure 6.4:** Comparison of the Frustum PointNet[Qi+18] baseline and our multi-frame approach (*Ours*) qualitatively for the pedestrian class. The red 3D boxes are the predictions, and the green ones are the ground-truth boxes. We show the ground-truth boxes only on the left-most drawing visualising a larger region. The baseline misses the correct localisation of two pedestrians approaching each other in the following frames. However, our method can detect both the pedestrians correctly in a sequence from the beginning.

the right-most four scenes are the sequential zoomed-in lidar visualisations. The red box indicates the predictions, and the green boxes show ground-truth boxes. Baseline localises the object correctly in frame $t$ even though the box orientation is wrong. In the following frames, the baseline misses the cyclist entirely. However, our method can detect and keep the object bounding box correctly from the frame $t$ on.

We also provide our qualitative results for the car class, as seen in Fig. 6.6. The figure shows the detections of the Frustum PointNet baseline and our method on a scene sequence. The broader scene on the left-most hand side visualises detected cars at frame $t$. The ground-truth boxes and detections are drawn with green and red colours, respectively. Both networks can detect the car object at frame $t$. However, the baseline misses the car at $t+1$ and wrongly detects it at $t+3$. On the other hand, the detections provided with *Ours* can detect the car object consistently through the successive frames. We also visualise the ground-truth boxes at the last frame for a better comparison.

All in all, the qualitative results indicate that building multi-frame relations between object features helps more consistent and stable detections for occluded objects or objects that reflect a smaller number of points.

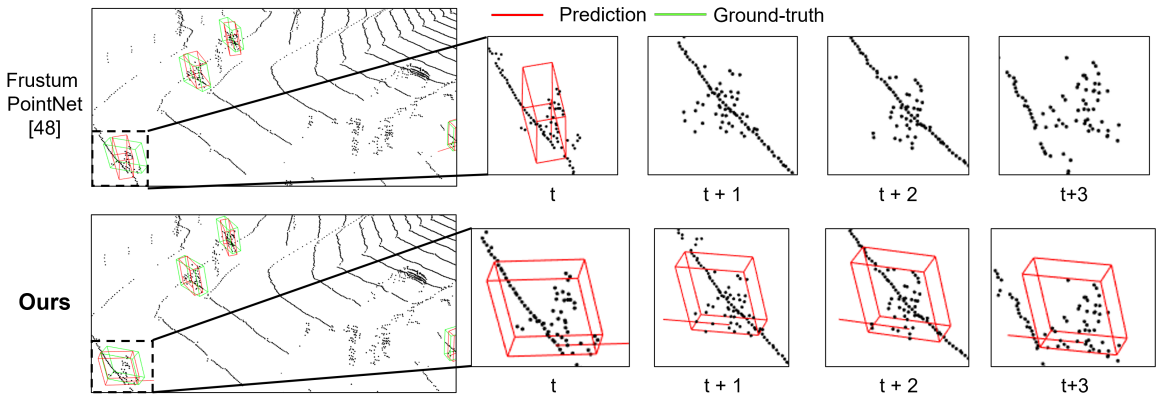



**Figure 6.5:** Comparison for the cyclist class qualitatively between the baseline Frustum PointNet [Qi+18] and *Ours*. The green bounding boxes indicate the ground-truth labels, whereas the red ones are for the detections. The left-most scene is a broader view of the frame $t$. We show a sequence of zoomed-in lidar visualisations of the cyclist with the right-most four visuals. The Frustum PointNet baseline localises the cyclist initially, even though the orientation is wrong. However, it cannot detect the object in the following frames. On the other hand, our method can detect and keep the bounding boxes correctly from the first frame onwards.

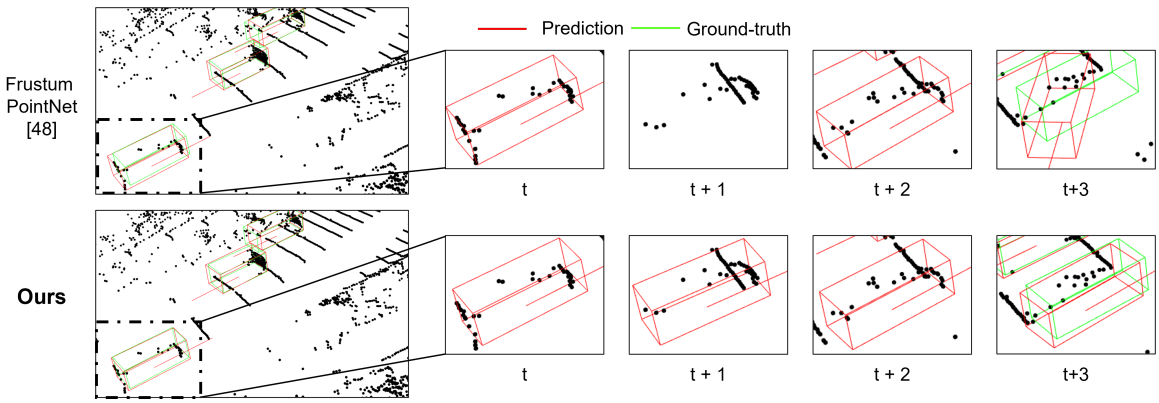



**Figure 6.6:** Qualitative 3D detection results for the car class using the Frustum PointNet baseline and our multi-frame feature alignment method (*Ours*). Green and red colours indicate the ground-truth and detected bounding boxes, respectively. Even though the baseline detects the car at frame $t$, it misses the object at frame $t+1$ and estimates an inconsistent bounding box at frame $t+3$. However, our method can detect the car object consistently through multiple frames. For a better comparison, we also show the ground-truth bounding box at frame $t+3$.

**Table 6.3:** Performance based on 2D tracking on the KITTI tracking validation set. E: Easy, M: Moderate, H: Hard

| Gt 2D Det + | SORT | | | Deep SORT | | | Gt Tracking | | |
|---|---|---|---|---|---|---|---|---|---|
| **Car** | E | M | H | E | M | H | E | M | H |
| Ours | **60.5** | **53.6** | **48.7** | 58.9 | 52.1 | 47.3 | 83.1 | 72.2 | 65.1 |
| Ours (w/ Cent) | 60.2 | 52.9 | 48.2 | 58.1 | 51.1 | 46.7 | 83.7 | 65.6 | 64.1 |
| **Pedestrian** | E | M | H | E | M | H | E | M | H |
| Ours | 27.2 | 26.2 | 22.7 | 28.3 | 26.9 | 23.2 | 67.5 | 60.4 | 53.6 |
| Ours (w/ Cent) | 30.6 | 34.1 | 30.3 | **31.6** | **34.9** | **30.9** | 64.5 | 59.4 | 52.7 |
| **Cyclist** | E | M | H | E | M | H | E | M | H |
| Ours | 42 | 41.1 | 41.7 | 34.6 | 37.5 | 38.4 | 82.5 | 65.3 | 65 |
| Ours (w/ Cent) | **55.4** | **61.5** | **55.5** | 51.4 | 59.2 | 53.2 | 85.1 | 74.5 | 67.3 |

### 6.4.3 Ablation Studies

This section provides the results of the additional experiments explained in Section 6.3 to investigate our method further and show our approach's effectiveness.

#### Performance Based on 2D Tracking

Prediction of the correct track IDs among multiple frames is significant for the success of our method. We use the track IDs for associating the object features that belong to the 2D bounding box. Therefore, we ablate the network performance using the SORT [Bew+16] and the Deep SORT [WBP17] 2D trackers. We only use the predicted tracking IDs for the evaluation of this experiment. We show the results in Table 6.3. Using the track IDs provided by SORT and Deep SORT significantly affects the performance. It decreases the 3D AP values obtained on the validation set compared to the ground-truth track IDs for all difficulty levels and classes. This also suggests that our method would require fine-tuning with the 2D tracking results and an attention mechanism to raise awareness of faulty track IDs.

Compared to the SORT, Deep SORT additionally utilises the 2D image features from the object region. However, we do not fine-tune the Deep SORT backbone on KITTI but use the pre-trained feature extractor on the COCO dataset. SORT performs better for car and cyclist classes; however, Deep SORT outperforms the SORT on the pedestrian class.

#### Sequence Length Ablation

Multi-frame feature fusion can be done on a frame sequence with an arbitrary length. However, the object features become more diverse in the longer sequences. Therefore, we evaluated the effects of the $\tau$ parameter on the detection performance, and the results can be seen in Table 6.4. We obtained the best results for pedestrian and car classes with $\tau = 3$ and the cyclist class with $\tau = 8$ for the medium difficulty level. Even though the best performance for the cyclist is with $\tau = 8$, $\tau = 3$ provides a closer performance.

We also compare our results with $\tau = 1$, which means having an FC layer instead of the GRU units to match the architecture depth. However, the additional layer did not benefit, and the AP scores with $\tau = 1$ are even lower than the baseline. This shows that the improvement originates from the temporal feature processing instead of the additional depth in the network on top of the Frustum PointNet baseline.

**Table 6.4:** Performance with different sequence lengths in 3D AP for three classes on KITTI tracking validation set. E: Easy, M: Moderate, H: Hard

| | | Pedestrian | | | Cyclist | | | Car | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | $\tau$ | E | M | H | E | M | H | E | M | H |
| Frustum PointNet | - | 56.8 | 56.5 | 50.4 | 81.3 | 59.4 | 58.8 | 83.1 | 65.4 | 65.1 |
| Ours | 1 | 52.6 | 47.6 | 46.7 | 80 | 56.6 | 55.5 | 77.1 | 62.1 | 55.5 |
| Ours | 2 | 54.9 | 50.6 | 49.4 | 77.3 | 64.5 | 64 | 83 | 65 | 63.7 |
| Ours | 3 | **67.5** | **60.4** | **53.6** | 82.5 | 65.3 | 65 | 83.1 | **72.2** | 65.1 |
| Ours | 4 | 58.1 | 52.4 | 50.9 | 82.8 | 58.1 | 57.5 | 83.5 | 64.7 | 64 |
| Ours | 5 | 52.5 | 49.8 | 48.6 | 82.8 | 63.8 | 57.8 | **85.5** | 65.2 | 63.6 |
| Ours | 6 | 56.4 | 50.7 | 49.4 | **83.1** | 57.3 | 56.6 | 80.9 | 64.8 | 63.7 |
| Ours | 7 | 53.8 | 48.9 | 42.7 | 78.8 | 62.3 | 56.1 | 82.8 | 66.3 | **65.3** |
| Ours | 8 | 56.9 | 50.7 | 49.2 | 81 | **66.1** | **65.1** | 79.2 | 64.3 | 63.1 |
| Ours | 9 | 51.5 | 47.7 | 41.6 | 78.8 | 52.9 | 47.5 | 80.7 | 64.4 | 63.2 |
| Ours | 10 | 55.2 | 49.4 | 43.6 | 76.4 | 56.6 | 56.2 | 80.7 | 64.4 | 63.2 |
| Ours | 15 | 51.7 | 47.2 | 41.2 | 83 | 56.5 | 55.8 | 82.4 | 65.8 | 64.1 |

#### Ablation of the Feature Alignment Strategies

The temporal object-level features host information from multiple frames. As explained in Subsection 6.1.6, it is possible to estimate the bounding box parameters differently using the temporal object features. In Table 6.5, the results of different alignment strategies are given. We give here the best performing $\tau$ results for all feature alignment methods. The one branch (OB) utilises only the multi-frame feature and does not explicitly get the current feature. We think this is the reason for performing the worst among all strategies. The two branch (TB) strategy merges outputs of the temporal feature and the current feature processing branches with a mean operation. Even though adding the current-frame feature vector improves the results, the detection accuracy is still below the performance of the *Ours* strategy. We use the temporal object features to estimate objects' shape and the current feature vector for estimating the other box parameters for the *Ours* strategy. The shape estimation performs as an auxiliary loss for predicting other box parameters since the preceding layers are shared. As a result, *Ours* performs better than the strategies given in 6.5.

#### Ablation of the Cosine Distance Loss

The cosine distance loss ensures having similar object features from the sampled lidar points in two successive frames, as explained in Subsection 6.2.3. We show the training results with cosine distance loss in Table 6.6 for all different multi-frame alignment strategies and for adding centre residuals to the feature vector. Training with the cosine distance loss improves the results for pedestrian and cyclist classes compared to the results without cosine distance in Table 6.5. However, the 3D detection accuracy for the car class is lower than the training without the cosine distance loss. Using such a similarity metric also helps decrease the gap of *OB* and *TB* versions with the *Ours*. This indicates that the feature extractor can extract similar object features from different sets of points in two successive frames, which reduces the need for the awareness of the features in the current frame.

**Table 6.5: Ablation of the Feature Alignment Strategies:** AP on KITTI validation for three classes

| **Car** | $\tau$ | Easy | Moderate | Hard |
|---|---|---|---|---|
| Ours | 3 | 83.1 | **72.2** | **65.1** |
| Ours (w/ Cent) | 4 | **+0.6** | -6.6 | -1 |
| OB | 3 | -3.2 | -10.5 | -4.9 |
| TB | 4 | -0.9 | -9.7 | -3.4 |
| **Pedestrian** | $\tau$ | Easy | Moderate | Hard |
| Ours | 3 | **67.5** | **60.4** | **53.6** |
| Ours (w/ Cent) | 3 | -3 | -1 | -0.9 |
| OB | 6 | -9.7 | -9.1 | -3.8 |
| TB | 4 | -9.8 | -8.6 | -2.9 |
| **Cyclist** | $\tau$ | Easy | Moderate | Hard |
| Ours | 3 | 84.9 | 65.3 | 65 |
| Ours (w/ Cent) | 4 | **+1.3** | **+9.2** | **+2.3** |
| OB | 4 | -7.4 | -14.6 | -14.2 |
| TB | 4 | -4.4 | -1.6 | -1.9 |

**Table 6.6: Training with the cosine loss:** 3D AP results on the KITTI tracking validation data for three classes.

| | | Cosine Loss | | |
|---|---|---|---|---|
| **Car** | $\tau$ | Easy | Moderate | Hard |
| Ours | 4 | **82.5** | 65.5 | **64.1** |
| Ours (w/ Cent) | 3 | -1.3 | **+0.1** | -0.2 |
| OB | 3 | -3.2 | -3.4 | -3 |
| TB | 3 | -1 | -1.7 | -1.5 |
| **Pedestrian** | $\tau$ | Easy | Moderate | Hard |
| Ours | 4 | 65.7 | **63.3** | **56.4** |
| Ours (w/ Cent) | 4 | -0.1 | -4.2 | -3.8 |
| OB | 6 | +0.3 | -4.2 | -3.8 |
| TB | 6 | **+2.3** | -2.7 | -2.4 |
| **Cyclist** | $\tau$ | Easy | Moderate | Hard |
| Ours | 3 | 85.6 | 65.9 | 65.3 |
| Ours (w/ Cent) | 6 | **+1.9** | **+10.9** | **+4.4** |
| OB | 3 | -6.1 | -1.5 | -1.3 |
| TB | 3 | -3.3 | -0.6 | -0.8 |

**Table 6.7:** Layer type ablation for the temporal object feature fusion. Results are obtained on KITTI tracking validation data using the 3D AP metric.

| Temporal Fusion Type | Ours | | |
|---|---|---|---|
| **Car** | Easy | Mod | Hard |
| GRU | 83.1 | **72.2** | **65.1** |
| GRU (w/ Cent) | 83.7 | 65.6 | 64.1 |
| Conv | 82.5 | 64.6 | 63.3 |
| Conv (w/ Cent) | **84.6** | 65.2 | 63.7 |
| LSTM | 83.5 | 65.7 | 64.6 |
| LSTM (w/ Cent) | 83.1 | 65.7 | 64.4 |
| **Pedestrian** | Easy | Mod | Hard |
| GRU | **67.5** | **60.4** | **53.6** |
| GRU (w/ Cent) | 64.5 | 59.4 | 52.7 |
| Conv | 62.2 | 56.8 | 50.6 |
| Conv (w/ Cent) | 64.8 | 57.4 | 52.1 |
| LSTM | 64.3 | 52.7 | 51.4 |
| LSTM (w/ Cent) | 65.7 | 58.5 | 51.9 |
| **Cyclist** | Easy | Mod | Hard |
| GRU | 84.9 | 65.3 | 65 |
| GRU (w/ Cent) | 86.2 | 74.5 | 67.3 |
| Conv | 84.8 | 58.1 | 57.8 |
| Conv (w/ Cent) | 86.9 | 69 | 68.8 |
| LSTM | **87** | 69 | 68.4 |
| LSTM (w/ Cent) | 85.4 | **76.5** | **69.4** |

#### Layer Type Ablation for Multi-frame Feature Alignment

We mainly use a recurrent layer with GRU cells for our multi-frame object feature alignment method. It is also possible to use LSTM layers, but LSTMs provide similar results with GRU for shorter sequences than natural language processing problems usually require [Chu+14; Yin+17]. Also, training with GRU layers is faster than training with LSTMs. However, we justify our selection in an ablation study by comparing GRU-based, LSTM-based, and convolutional layer-based temporal feature fusion as the results given in Table 6.7. We conduct all the experiments with the *Ours* and *Ours (w/ Cent)* versions. GRU-based temporal feature fusion outperforms the convolution-based for all classes and arrangements in the moderate difficulty level. Overall performance with the convolution-based fusion is worse than the recurrent layer-based ones. Using LSTM layers improves the cyclist class results, but GRU-based temporal fusion provides the best results for the car and pedestrian. The overall performance of the GRU-based temporal feature fusion is better than the proposed LSTM- and convolution-based fusion methods.

#### Feature Depth Ablation

It is possible to take the object features from several depth levels in the network, as explained in Subsection 6.2.4. Table 6.8 gives the results obtained with the $global$, $fc1$, and $fc2$ features, which mean taking the features from the output of the max pool operation, FC1

**Table 6.8:** Feature depth ablation results on KITTI tracking validation data with 3D AP metric for three classes.

| Feature Length | $global$ | | | $fc1$ | | | $fc2$ | | |
|---|---|---|---|---|---|---|---|---|---|
| **Car** | Easy | Mod | Hard | Easy | Mod | Hard | Easy | Mod | Hard |
| Ours | 82.9 | 65.4 | 64.1 | 83.1 | **72.2** | **65.1** | 81.2 | 65.5 | 64.21 |
| Ours (w/ Cent) | 82.4 | 65.2 | 63.6 | **83.7** | 65.6 | 64.1 | **83.7** | 65.7 | 64.2 |
| **Pedestrian** | Easy | Mod | Hard | Easy | Mod | Hard | Easy | Mod | Hard |
| Ours | 68.8 | 62.5 | 55.6 | 67.5 | 60.4 | 53.6 | 57.4 | 51.5 | 50.1 |
| Ours (w/ Cent) | **69.9** | **62.8** | **55.8** | 64.5 | 59.4 | 52.7 | 63.9 | 58.4 | 52.1 |
| **Cyclist** | Easy | Mod | Hard | Easy | Mod | Hard | Easy | Mod | Hard |
| Ours | **89** | 76.6 | 76.4 | 84.9 | 65.3 | 65 | 76.1 | 63 | 57 |
| Ours (w/ Cent) | 88.4 | **77.6** | **77.3** | 86.2 | 74.5 | 67.3 | 86.5 | 75.9 | 68.7 |

layer, and FC2 layer, respectively. The results for the car class are similar for all the layer outputs even though using $fc1$ outperforms the moderate difficulty level with a large margin. $global$ features provide better 3D detection results for pedestrians and cyclists than other features for all the difficulty levels. Pedestrian and cyclist class objects reflect fewer points than cars. Therefore, the fusion of sequential object features for these classes might perform better with lower-level features since $global$ provides a more generic object representation.

#### Ablation of the Feature Extent

We utilise object-level features for generating multi-frame object features. Therefore, the segmented points from a frustum extract object-level features (Fig. 6.3)). The segmented points are assumed to be highly related to the object of interest, as stated in [Qi+18]. To compare our results with the scene-level version of temporal fusion, we skip the segmentation part of the network. This would cause a more general set of features for the object of interest, which might be reasoned from other closer objects and the background. As a result, we obtain the scene-level temporal features from directly the frustum points. We give the results in Table 6.9. The detection accuracy for the scene-level aggregation decreases significantly compared to the object-level version. One reason for this is skipping segmentation causes features from more irrelevant points. This also suggests that the spatial alignment between feature regions is required for the scene-level multi-frame feature aggregation.

### 6.4.4 Discussion

As a result, our multi-frame object feature fusion method provides better 3D detection results on the KITTI dataset. The results are significantly better for the pedestrian and cyclist classes, represented with fewer object instances in the dataset. In addition, pedestrian and cyclist class instances are more prone to occlusion and usually reflect fewer lidar points. Nevertheless, the 2D detection and tracking greatly impact the multi-frame aggregation quality. In those cases, the network should also be fine-tuned for the less accurate data to raise awareness in the multi-frame feature alignment layers. Compared to the baseline, object-level multi-frame feature fusion provides better results with a high margin, mainly to keep

**Table 6.9:** Feature extent ablation on the KITTI tracking validation data with 3D AP metric for three classes.

| | Scene-level | | | Object-level | | |
|---|---|---|---|---|---|---|
| **Car** | Easy | Mod | Hard | Easy | Mod | Hard |
| Ours | 59.7 | 39 | 34.7 | 83.1 | **72.2** | **65.1** |
| Ours (w/ Cent) | 59.4 | 38.1 | 33.8 | **83.7** | 65.6 | 64.1 |
| **Pedestrian** | Easy | Mod | Hard | Easy | Mod | Hard |
| Ours | 34.2 | 31.6 | 27.6 | **67.5** | **60.4** | **53.6** |
| Ours (w/ Cent) | 34.6 | 32.2 | 28.1 | 64.5 | 59.4 | 52.7 |
| **Cyclist** | Easy | Mod | Hard | Easy | Mod | Hard |
| Ours | 39.4 | 46.8 | 46.1 | 84.9 | 65.3 | 65 |
| Ours (w/ Cent) | 40.8 | 48.8 | 43.4 | **86.2** | **74.5** | **67.3** |

the previously detected boxes. Also, using the cosine distance loss shows that it is essential to train the network for extracting similar features for the same object in successive frames.

Chapter 7

# Self-supervised Scene Flow-based 3D Object Detection

3D object detection relies highly on learning good point representations from a set of points in the point cloud using large annotated datasets. However, it is costly to obtain labelled 3D vision datasets, since they require human expertise and time. Recently, there have been self-supervised approaches [Bau+21; MOH20; Tsc+20; PG20] in different domains, which aim to learn representative features using the structure of data without annotations.

Scene flow problem aims at estimating the motion of individual lidar points from the two successive point clouds. Recent methods suggest using self-supervised learning for this task, since it is also difficult to have motion labels for lidar points. This allows the training of the scene flow networks with large datasets.

Our aim is to train the 3D detection backbone on the self-supervised scene flow task to learn representative point features as summarised in Fig. 7.1. Afterwards, we further fine-tune the 3D detection head on the annotated small-scale 3D detection data with the supervised learning. In this way, we expect to obtain a better 3D detection accuracy.

## 7.1 Methodology

We assume that $P^t$ and $P^{t+1}$ are two successive points clouds, where $P^t = \{\mathbf{p}_i | i = 1, ..., k\}$ and $\mathbf{p}_i \in \mathbb{R}^4$. A scene flow function, $s$, estimates the scene flow vectors, $\mathcal{F}_{t \to t+1} = \{d_i\}_M$, for the points in $P^t$, such that $\mathcal{F}_{t \to t+1} = s(P^t, P^{t+1})$. A motion vector is defined as $\mathbf{d}_i^t = \mathbf{p}_i^t - \mathbf{p}_i^{t+1}$ that propagates $\mathbf{p}_i^t$ to the frame t+1. Note that $\mathbf{p}_i^{t+1}$ does not have to exist due to the sparse lidar point clouds. The scene flow function, $s$, utilises a 3D detection backbone, $h$, for extracting point features to estimate motion vectors such that $D = s(h(P^t), h(P^{t+1}))$. We use the cycle consistency loss [MOH20] for training the scene flow function, which also optimises weights of the 3D detection backbone, $h$. In this way, our 3D backbone function, $h$, learns point representations on the large-scale unlabelled data with the self-supervised scene flow task.

The 3D detection network mainly consists of two parts, the backbone, $h$, and the head, $g$. After the self-supervised training, we use the pre-trained backbone, $h$, for the detection such that $B = g(h(P^t))$, where $B = \{\mathbf{b}_i^t | i = 1, ..., l\}$ and $\mathbf{b}_i^t \in \mathbb{R}^7$ is a single bounding box defined with its 3D centre, $(x, y, z)$, box sizes $(w, h, l)$, and the orientation $\theta$. We fine-tune the pre-trained backbone, $h$, and the detection head, $g$, on the small-scale annotated 3D detection data with the supervised learning. Our final aim is to improve 3D detection on quality by initializing the 3D detection network with the point representations learned using the self-supervised cycle consistency loss. The whole pipeline is summarised in Fig. 7.2.

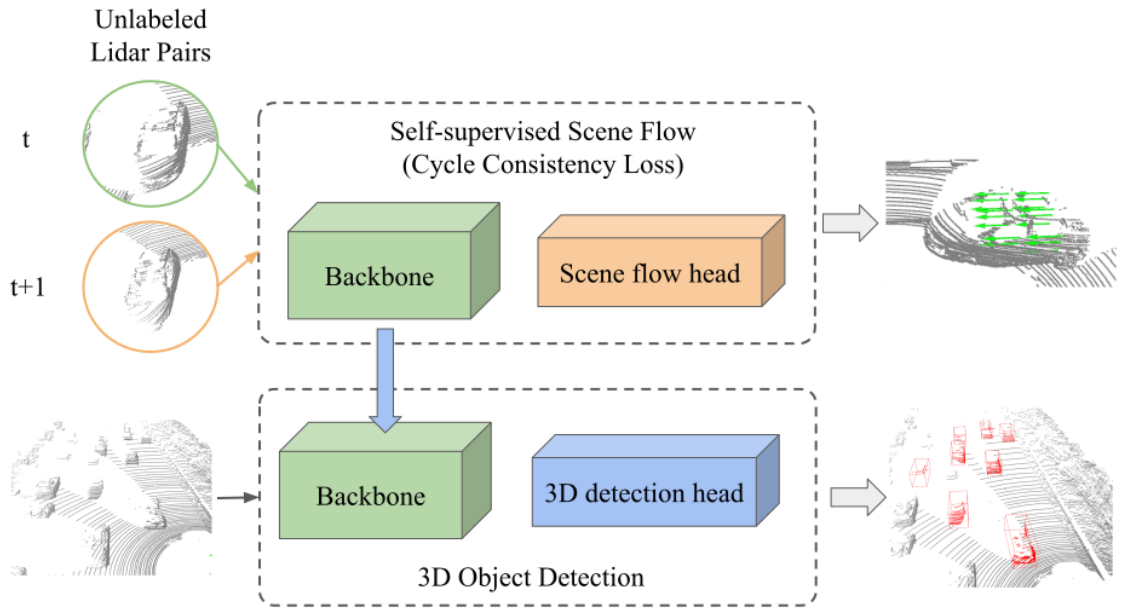



**Figure 7.1:** The annotated 3D vision datasets are difficult to create. Therefore, we propose to train a 3D detection backbone on the self-supervised scene flow task using a large-scale unlabelled lidar point cloud data for learning point representations. We use the cycle consistency [MOH20] between two successive frames. Afterwards, we fine-tune the 3D detection backbone and the head on a smaller-scale labelled 3D detection data, which is expected to improve the detection quality using the learned point representations from the pre-training.

### 7.1.1 Self-supervised Backbone

For the self-supervised 3D detection backbone training, we follow the cycle consistency method proposed in [MOH20] using the FlowNet3D architecture [LQG19]. FlowNet3D uses *setconv* layers as a point feature encoder, $h$, to obtain features of the sampled points from two successive frames. The *setconv* layers are two sets of cascaded three-layer convolutional network. Instead of using the *setconv* layers as a point cloud encoder, we utilise the backbone of a 3D detection network ($h$) for extracting sampled point features and estimating the scene flow, $\mathcal{F}_{t\rightarrow t+1}$. In this way, the self-supervised cycle consistency loss gradients can be backpropagated through the 3D detection backbone, which will learn encoding the point representations. Hence, the 3D detection backbone can provide the point embeddings to the scene flow head as well as the 3D detection head for the scene flow estimations and the 3D detections, respectively.

### 7.1.2 Scene Flow Head

As shown in Fig. 7.2 bottom right, the scene flow head of FlowNet3D consists of several layers to aggregate information from the sampled points of two successive frames. The flow embedding layer combines point features from two frames, which is further processed by a *setconv* layer. A set of following cascaded *set upconv* layers reconstructs the input point shape at frame $t$. A final fully-connected layer estimates the scene flow offsets, $\mathcal{F}_{t\rightarrow t+1}$. The scene flow head is also trained with the self-supervised cycle consistency loss. The *set upconv* layers take skip connections from the *setconv* and the flow embedding layers. We remove the skip connection to the last *set upconv* layer, since we use features from the 3D detection backbone instead of the first *setconv* layer of FlowNet3D.

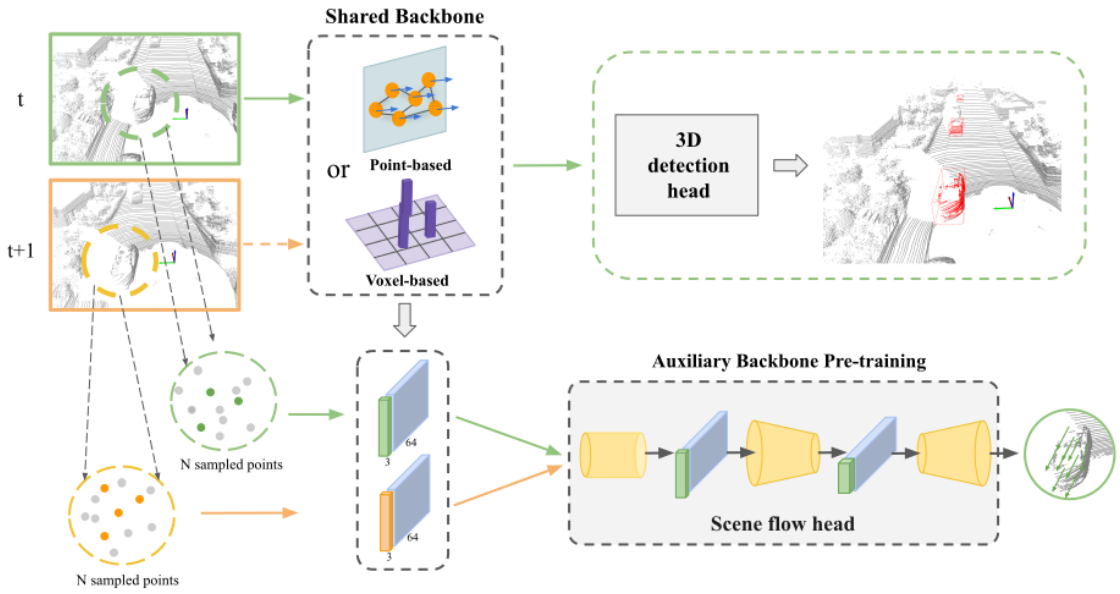



**Figure 7.2:** Our self-supervised scene flow-based pre-training for 3D object detection. The self-supervised scene flow training takes two sequential point clouds and extract point features using the shared 3D detection backbone. This backbone can be arbitrarily one of the point-based or the voxel-based 3D detection backbones. Sampled point sets from two point clouds with the extracted features are fed into the scene flow head based-on FlowNet3D [LQG19] to train the scene flow network with the cycle consistency loss [MOH20]. This is an auxiliary backbone training to help the detection network with learning point representations on a large unlabelled dataset. Afterwards, the 3D detection network (the backbone and the head) is fine-tuned on the labelled smaller-scale dataset using a single-frame approach.

### 7.1.3 Training with Cycle Consistency

The 3D detection backbone, $h$, and the scene flow head, $s$, estimate the scene flow $\mathcal{F}_{t\rightarrow t+1}$. We train both of the network parts using the cycle consistency loss given in [MOH20] in a self-supervised way. First, the forward flow estimation, $\mathcal{F}_{t\rightarrow t+1} = s(h(P^t), h(P^{t+1}))$, propagates points $\mathbf{p}_i^t$ to frame $t+1$ as $\mathbf{p}_i'$. Then, the backwards flow estimation is calculated between the propagated points and the points at frame $t$ such that $\mathcal{F}_{t+1\rightarrow t} = s(h(P^t), h(P^{t+1'}))$. The resulting $\mathbf{p}_i''$ is propagated to the frame $t$ in the backward direction to close the cycle. The inconsistency between the $\mathbf{p}_i$ and the $\mathbf{p}_i''$ at frame $t$ optimises the scene flow network in a self-supervised way.

### 7.1.4 Downstream Task: 3D Object Detection

As shown in Fig. 7.2, the main 3D vision task that we are interested in is 3D object detection. The 3D detection backbone is pre-trained on the scene flow task to learn the point representations. After the pre-training, the 3D backbone provides the point cloud features to the 3D detection head, which estimates the 3D bounding box parameters of objects. The method is independent from the type of 3D detection network, whether it is a point-based or a voxel-based detector. Even though the 3D detection takes a single-frame input, the backbone already learns the consistency between two frames and how to represent point clouds with the geometry-awareness. Therefore, the 3D detection network can be trained on a smaller-scale labelled data with a good backbone initialization to obtain geometry-aware point cloud

features. We show that this assumption holds through extensive experiments given in Section 7.3.

## 7.2 Implementation Details

For the self-supervised scene flow training, point features should be extracted using a point encoder. We choose backbones of 3D detectors as point encoders. The backbone point processing type, point- or voxel-based, is not important as long as the point features can be taken from the backbone. Therefore, we evaluate our method on the Point-GNN[1] [SR20] and PointPillars[2] [Lan+19]. Point-GNN constructs a point-based graph to get features from the lidar point cloud and PointPillars utilises pillar features, which is an infinitely high voxel. We cascade the backbones with the FlowNet3D[LQG19] scene flow head and train the complete architecture with the cycle consistency loss [3][MOH20].

### 7.2.1 Pre-training with Self-supervised Scene Flow

We utilise the FlowNet3D[LQG19] architecture for the scene flow task, which takes two successive point clouds ($t$ and $t+1$) and estimates point cloud motion from frame $t$ to $t+1$. There are four different types of point processing modules in the FlowNet3D, which we also use. The *PointNet Set Abstraction*(*SA*) module aggregates the point features in a defined vicinity to generate keypoint features. *Flow Embedding*(*FE*) module takes keypoint features from two frames and generates flow features for the sampled points from frame $t$. The *PointNet Set Upconv*(*SU*) module reconstructs the initial point shape of the frame $t$. The *upconv* modules take also skip connections from the *SA* modules to have the awareness of multi-scale keypoint features. The PointNet Feature Propagation module is to propagate point features from two levels (the previous layer and the skip connection) without upsampling.

The modified FlowNet3D architecture can be seen in Fig. 7.3. $\mathbf{F}^t \in \mathbb{R}^{n \times m}$ is the feature matrix of points $P^t$ from frame $t$, where $n$ is the number of points and $m$ is the feature length of the points. $F^t$ and $F^{t+1}$ are taken from the backbone. $n = 2048$ for both Point-GNN and PointPillars, whereas $m = 300$ for the Point-GNN and $m = 64$ for the PointPillars. The *SA* modules downsamples the input point clouds to the $n\_points$ keypoints as shown in Fig. 7.3. The keypoint features are generated from at most $n\_sample$ points within its $radius$ neighbourhood through a PointNet with the given MLP layers. The *FE* module generates the flow embedding features for the keypoints from frame $t$ combining keypoint features coming from two successive frames. The flow embeddings for each point from frame $t$ are generated using the frame $t+1$ keypoint features with at most $n\_sample$ keypoints within the $radius$ neighbourhood through the defined MLP layers. The *SU* module upsamples the keypoints to recover the original size. As inputs, it takes the keypoint positions and features from the previous module as well as the skip connections from the *SA* and the *FE* modules. The upsampling is done for the larger keypoint set and the keypoint features are generated from the previous layer keypoint features similarly with at most $n\_smaple$ keypoints within the $radius$ neighbourhood using a PointNet with the defined MLP. It is also possible to have a second MLP, defined with *mlp2*. The feature propagation module combines point and keypoint features through an interpolation operation and an MLP. The two final FC layers estimates the flow vectors from frame $t$ to $t+1$.

[1] https://github.com/WeijingShi/Point-GNN
[2] https://github.com/open-mmlab/mmdetection3d/
[3] https://github.com/HimangiM/Just-Go-with-the-Flow-Self-Supervised-Scene-Flow-Estimation

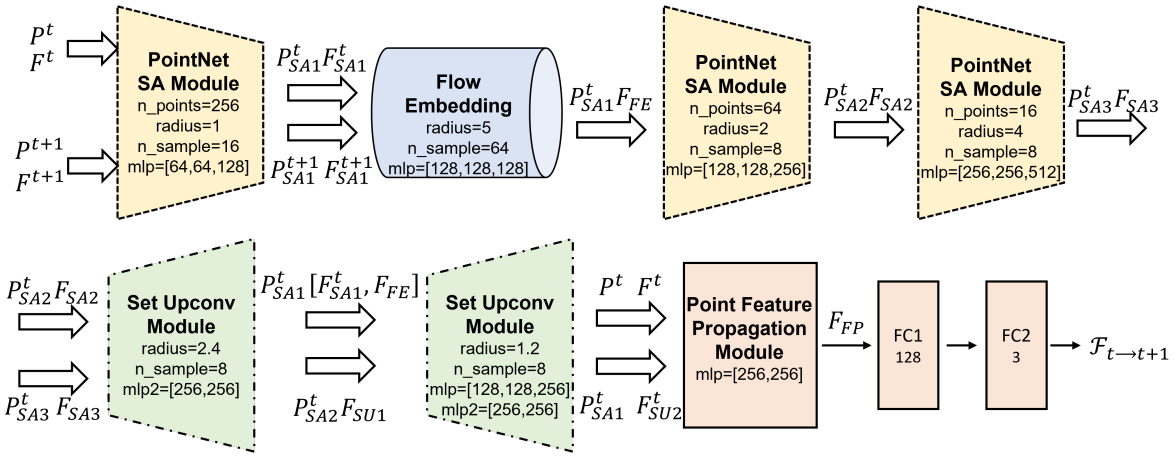



**Figure 7.3:** Architecture of the scene flow head based on FlowNet3D [LQG19].

We use only one *SA* module before the *flow embedding* module, since the features are taken from the 3D detector backbone. Therefore, the final *set upconv* layer and its skip connection are also removed.

**Point cloud backbone:** The point cloud backbone generates the point features either with voxel-based or with point-based point processing methods. Therefore, we utilise two types of point processing methods to show results of our idea. The Point-GNN generates point features from a 3-level GNN. Each level is an independent sub-network and provides keypoint features, which are selected with the Farthest Point Sampling method. The GNN backbone is a point-based processing method, which does not reshape points in a structured representation such as voxelisation. We also apply our method to PointPillars, which has a voxel-based point cloud processing backbone. Similarly, we take the point features from PointPillars' voxel encoder.

When the backbones are used for 3D detection, they provide point features for the entire point cloud as needed for the 3D object detection heads. Therefore, there are no changes in the original 3D detection pipeline. However, we sample $N$ points from each successive frame and take their features from the backbones for the scene flow task. For the Poin-GNN backbone, we use interpolation to generate sampled point features from the keypoint features. The voxel encoder provide voxel features, which we assign to the points of the related voxel.

**Scene flow head:** We consider the whole architecture seen in Fig. 7.3 as the scene flow head since we remove the first SA layer of the original FlowNet3D, which was responsible for generating point features. The head consists of an encoder-decoder architecture, which first generates keypoint features by aggregating point features in different levels and then generates point features using the keypoint features as well as the skip connections. For our method, the scene flow head takes the point features of two successive frames from the backbone and estimates the flow vectors. The final two FC layers are responsible for estimating the 3D flow vectors from the reconstructed point features.

**Training strategy:** The training of the scene flow task and 3D detection are separate processes. The self-supervised scene flow training optimises weights of the backbone and the scene flow head at the same time end-to-end. We use the cycle consistency loss for the optimisation. The initialize the weights of scene flow head with the pre-training on the supervised FlyingThings3D [May+16] dataset. The FlyingThings3D is a simulation dataset and a common approach for pre-training of scene flow networks [LQG19]. We only initialize

our common layers with the FlowNet3D using the pre-trained weights in [MOH20][4] for the numerical stability and having a good starting point for the scene flow training.

We keep the training settings of the 3D detectors for the scene flow training, where possible. The Point-GNN-based scene flow network is optimised using the Stochastic Gradient Descent (SGD) with a learning rate of $6.25 \times 10^{-5}$. The batch size is 4 and $N = 2048$. We continued the training for 80k steps on the enlarged KITTI tracking dataset as explained in Subsection 7.2.3 on a single Nvidia Tesla V100 GPU. For the PointPillars model, we used Adam optimiser with a 0.001 learning rate, 2 batch size, and $N = 2048$. The training took place on two Nvidia RTX 2080 Ti GPUs.

### 7.2.2 3D Detection Fine-tuning

After the self-supervised scene flow training, the 3D detection backbone learns the geometric representations of point clouds. However, the backbone still needs learning point representations also optimised for the 3D detection task. Therefore, we optimise the weights of the 3D detection backbone and the head together end-to-end following the settings of the 3D detectors we use.

**3D detection heads:** We show our results on two 3D detectors, Point-GNN [SR20] and PointPillars[Lan+19], which use point- and voxel-based point encoding methods, respectively. Training with both of the 3D detectors is done in the same way as the regular 3D detection training. Differently, we initialize the backbone weights with the optimised backbone weights from the self-supervised scene flow training. For the Point-GNN we define the layers after the 3-level ($T = 3$ in [SR20]) GNN as the 3D detection head. For the PointPillars, we take the network after the voxel encoders as the 3D detection head.

**Training strategy:** The 3D detection training starts with the backbone weights learned in the self-supervised scene flow task and the 3D head weights initialized randomly. The training of the backbone and the head is done end-to-end using the 3D detection losses explained in Subsection 7.3.3.

We mainly follow an alternating training strategy [Ren+15] between the self-supervised scene flow task and the supervised 3D detection task trainings. The two trianings are repeated alternatingly by initializing the backbone weights, scene flow head weights, and the 3D detection head weights from the last relevant training. The steps of the alternating training are as the following: **(i)** the self-supervised scene flow training to optimise the backbone and the scene flow head weights, **(ii)** the 3D detection training to optimise the backbone and the 3D detection head, where the backbone is initialized with the weights from (i), **(iii)** the self-supervised scene flow training with the backbone weights initialized from the step (ii) and the scene flow head weights initialized from the step **(iii)**, and **(iv)** the 3D detection training, where the backbone weights are initialized from the step (iii) and the 3D detection head weights are initialized from the step (ii). In this strategy, the two tasks are not optimised jointly, but the backbones have the awareness for the local and global point cloud representation on both the scene flow and the 3D detection tasks.

The **Point-GNN** baseline is trained for $700k$ steps with the SGD optimiser and a learning rate of 0.125 on the KITTI 3D detection dataset. For our method, we followed the alternating training strategy and trained the step (ii) for $250k$ steps and the step (iv) for $700k$ steps with the SGD optimiser and a 0.1 learning rate. For all Point-GNN trainings, the batch size is taken as 4 and a Nvidia Tesla V100 GPU is used. We trained the **PointPillars** baseline for 24 epochs with a batch size of 4 on two Nvidia RTX 2080 Ti GPUs using the Adam optimiser with a 0.001 learning rate. For our alternating training, we use the same settings except we decrease the

[4]https://github.com/HimangiM/Just-Go-with-the-Flow-Self-Supervised-Scene-Flow-Estimation

learning rate to 0.0001 for step (iv).

### 7.2.3 Datasets

We train the Point-GNN-based scene flow network on the enlarged KITTI multi-object tracking dataset in a self-supervised way and Point-GNN 3D detection network on the KITTI 3D object detection dataset [GLU12b]. For PointPillars, we use the nuScenes dataset [Cae+20] for both the scene flow and the 3D detection tasks. We don't use any kind of annotations for the self-supervised scene flow trainings. The 3D detection results are provided on the KITTI 3D object detection and nuScenes validation sets.

**KITTI 3D Object Detection:** We use the KITTI 3D object detection dataset for training the Point-GNN 3D detection backbone and head. Since, this dataset does not provide sequential point clouds, it is not suitable for the self-supervised scene flow training. We use the common 3DOP train-val split, which contains 3712 training and 3769 validation frames. In the ablation experiments, we decrease the number of frames in the training set with a percentage to report the performance gap between our method and the baseline. For these ablation studies, we randomly choose the number of frames defined with the specified percentage among the training set and apply the evaluation on the whole validation set. We report our results on this dataset using the 3D AP metric for the easy, moderate, and hard difficulties defined for the KITTI benchmark.

**KITTI Multi-object Tracking:** The KITTI multi-object tracking dataset consists of 21 training and 29 testing drives. Each drive contains sequential data, which we use for the self-supervised scene flow task. We take the drives 11, 15, 16, and 18 for the validation and use the rest of the 17 training drives as well as 29 testing drives for the self-supervised scene flow training since the training does not require any annotations. In total, we use 11902 frames for training, which are sampled at 10 Hz.

**nuScenes:** nuScenes is a large-scale autonomous driving dataset. The nuScenes data are collected in more challenging weather conditions and from denser environments. The whole dataset consists of sequential frames from different sets of drives. Even though the lidar scans are collected at 20 Hz, the annotations are given for the key frames at 2 Hz. We use the nuScenes dataset for training the PointPillars network on the scene flow and 3D detection tasks, combining 10 sequential sweeps as one frame input associated with the key frames. However, we do not use any annotations for the self-supervised scene flow training. 10 classes are annotated in the dataset and mainly the AP per class, mean AP (mAP) among all classes, and the nuScenes detection score (NDS) are used for the evaluation.

### 7.2.4 Loss

This section introduces the loss functions used for training the scene-flow networks as well as the 3D detection networks. Mainly, we use the original loss functions without any changes for the considerede tasks.

**Point-GNN[SR20]:** Point-GNN predicts the object classes as well as the 3D bounding box parameters. The object classes are determined on the estimated probability scores among all classes. Since the Point-GNN detects car objects with a separate network, the class estimation is done among 4 classes. These are the *background*, horizontal and vertical anchor box classes, and a *don't care* class. The 3D bounding boxes are constructed by regressing 7 parameters, which are the object centre $(x, y, z)$, the object size $(w, h, l)$, and the object orientation $\theta$. For the classification, a cross-entropy loss is calculated between the predicted class probabilities and the ground-truth object class. For the regression, the Huber loss (Smooth L1)

is used. In addition, the $L1$ regularisation on the network weights is used to regularise the training. The total training loss is a combination of these three losses as given with equation 7.1, where $\alpha = 0.1$, $\beta = 10$, and $\gamma = 5e-7$ as selected in the original implementation.

$$L_{total} = \alpha L_{cls} + \beta L_{loc} + \gamma L_{reg} \tag{7.1}$$

**PointPillars[Lan+19]:** PointPillars also estimates the object class, 3D bounding box, and the orientation. The classification loss is calculated using the Focal Loss [Lin+17b] with the $\gamma = 2$ and $\alpha = 0.25$. The 7 bounding box parameters are regressed and the regression loss is calculated using the Smooth L1 loss. The orientation of the 3D bounding box is predicted among the discrete direction bins, one horizontal and one vertical. The direction loss is calculated using the cross entropy loss. The total loss is given with equation 7.2, where $a = 1$, $b = 1$, and $c = 0.2$.

$$L_{total} = aL_{cls} + bL_{loc} + cL_{dir} \tag{7.2}$$

**Self-supervised scene flow loss:** We use the cycle consistency loss proposed in [MOH20] for our self-supervised loss function to train our scene flow networks. The cycle consistency loss is a combination of the nearest neighbour loss and the cycle loss. The nearest neighbour loss of a point $\mathbf{p}'_i$ propagated from frame $t$ to $t+1$ is calculated with its nearest point in set $P^{t+1}$. The cycle loss is calculated between the $\mathbf{p}_i$ and $\mathbf{p}''_i$, which is obtained by the forward ($\mathcal{F}_{t\rightarrow t+1}$) and the backward ($\mathcal{F}_{t+1\rightarrow t}$) scene flow function. The final loss is the sum of two losses with weights of 1.

### 7.2.5 Experiments

To show that our method leads to better 3D object detection results, we conduct several experiments:

1. We compare our self-supervised Point-GNN training with the baseline on the KITTI 3D object detection dataset.
2. We compare our self-supervised PointPillars training with the baseline on the nuScenes dataset.
3. As an ablation study, we compare our alternating training self-supervised training strategy with the one step self-supervised training.
4. As a second ablation study, we compare our self-supervised Point-GNN and PointPillars with their baselines when they are trained with a small portion of the annotated data. This shows that our method leads to better results in the lack of labelled data.
5. As a third ablation study, we compare the 3D detection accuracy at an earlier stage of the training and compare the loss values through the training.

## 7.3 Results & Discussion

In this section, we provide the results of our self-supervised pre-training method on two different 3D object detector compared with their baselines as well as the state-of-the-art. In addition, we provide the results of the ablations introduced in Subsection 7.2.5. For all the comparisons with the baselines, we reproduced the baseline detection results. For the comparison with the state-of-the-art, we report the results from the respective papers.

| Car | 3D AP (IoU=0.7) | | | BEV AP (IoU=0.7) | | |
|---|---|---|---|---|---|---|
| Method | Easy | Mod | Hard | Easy | Mod | Hard |
| Point-GNN [SR20] | 90.44 | 82.12 | 77.70 | 93.03 | 89.31 | 86.86 |
| **Self-supervised Point-GNN** | **91.43** | **82.85** | **80.12** | **93.55** | **89.79** | **87.23** |
| Improvement | +0.99 | +0.73 | +2.42 | +0.52 | +0.48 | +0.37 |

**Table 7.1:** Self-supervised Point-GNN comparison with its baseline for car class using the 3D $AP_{R_{40}}$ metric on the KITTI validation set. Our method outperforms the baseline on all.

| | Car AP (IoU=0.7) | | |
|---|---|---|---|
| Method | Easy | Mod | Hard |
| VoxelNet[ZT18] | 81.97 | 65.46 | 62.85 |
| F-PointNet[Qi+18] | 83.76 | 70.92 | 63.65 |
| AVOD[Ku+18] | 84.41 | 74.44 | 68.65 |
| SECOND[YML18] | 87.43 | 76.48 | 69.10 |
| PointPillars[Lan+19] | - | 77.98 | - |
| Point-GNN [SR20] | 87.89 | 78.34 | 77.38 |
| **Self-supervised Point-GNN** | **88.32** | **78.66** | **77.92** |

**Table 7.2:** 3D object detection comparison with the state-of-the-art on the KITTI validation set using 3D $AP_{R_{11}}$ metric.))

### 7.3.1 Main Point-GNN Results

First, we compare our self-supervised Point-GNN with the baseline Point-GNN. We train both detectors for $1400k$ steps as explained in Section 7.2. The results are given in Table 7.1 as obtained on the KITTI 3D object detection validation set. For the self-supervised Point-GNN, we train the scene flow and the 3D detection networks alternatingly. We initialize the 3D detection network's backbone weights with the self-supervised scene flow network's backbone, which are obtained on unlabelled data. Then, we fine-tune the 3D detection network on the annotated dataset. As the results indicate, our self-supervised Point-GNN outperforms the baseline Point-GNN with a large margin in all difficulty levels of the 3D AP and the BEV AP. Our self-supervised pre-training method provides a 2.42% increase in the hard-level 3D AP. This shows that the self-supervised scene flow pre-training helps learning more representative point features especially for difficult objects, which reflect only a small number of points.

In Table 7.2, we compare our methods results with the previous state-of-the-art using the AP metric calculated through 11 recall points. On the car class, we outperform the baseline also in this metric as well as the given state-of-the-art detectors. We calculate the 3D $AP_{R_{11}}$ of the self-supervised Point-GNN as well as the baseline using the same predictions as in Table 7.1. As stated in [Sim+19b], the $AP_{R_{11}}$ can be misleading since a single perfect detection introduces a bias towards higher scores. Therefore, we think that our improvement with our self-supervised method is less obvious in Table 7.2 for the hard difficulty level. Table 7.3 shows the self-supervised Point-GNN 3D AP scores on the KITTI test set. The self-supervised pre-training performs 2% better than the Point-GNN test submission. This also shows that the motion-aware features learned from self-supervised scene flow pre-training helps distinguish difficult objects better.

We visualise our qualitative results in Fig. 7.4. The blue 3D boxes are the predictions of our self-supervised Point-GNN whereas the green ones are obtained by the baseline Point-

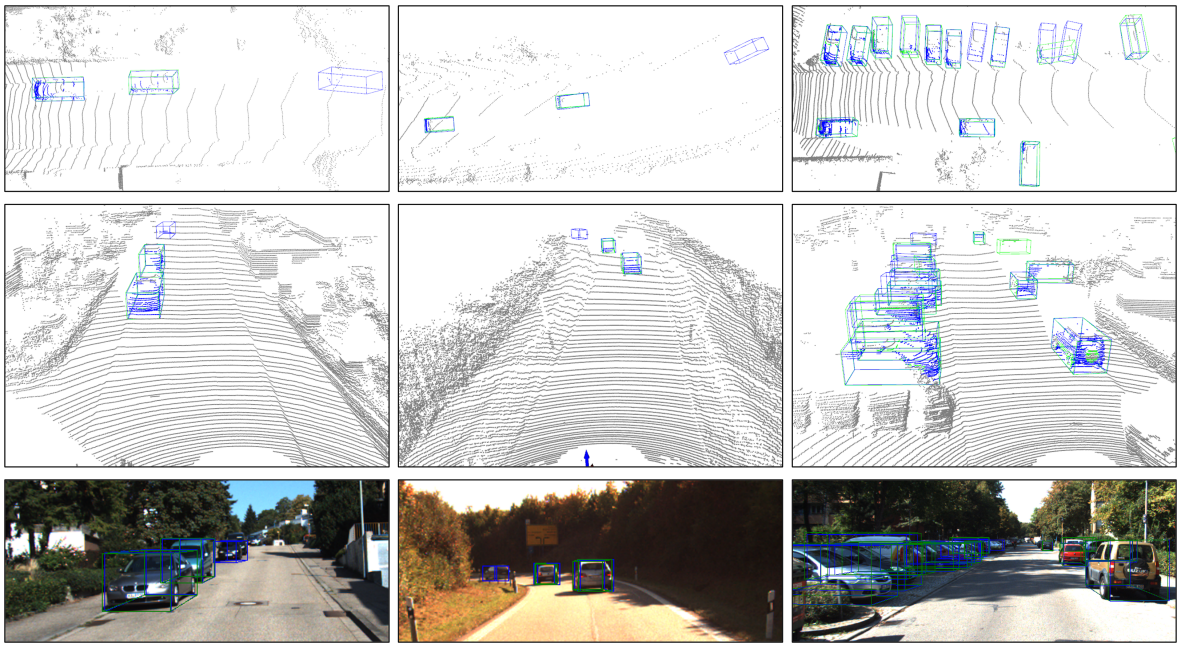


**Figure 7.4:** We compare our self-supervised Point-GNN with the baseline Point-GNN qualitatively on three scenes of KITTI 3D object detection validation set. Baseline and self-supervised Point-GNN predictions are shown with blue and green bounding boxes, respectively. Each column belongs to a different scene.

| Car (IoU=0.7) | 3D AP | | |
|---|---|---|---|
| Method | Easy | Mod | Hard |
| AVOD [Ku+18] | 76.39 | 66.47 | 60.23 |
| F-PointNet [Qi+18] | 82.19 | 69.79 | 60.59 |
| TANet [Liu+20] | 84.39 | 75.94 | 68.82 |
| Associate-3Ddet [Du+20] | 85.99 | 77.40 | 70.53 |
| UBER-ATG-MMF [Lia+19] | 88.40 | 77.43 | 70.22 |
| CenterNet3D [Wan+20a] | 86.20 | 77.90 | 73.03 |
| SECOND [YML18] | 87.44 | 79.46 | 73.97 |
| SERCNN [Zho+20] | 87.74 | 78.96 | 74.30 |
| Point-GNN [SR20] | 88.33 | 79.47 | 72.29 |
| **Self-supervised Point-GNN** | 87.78 | 79.36 | 74.15 |

**Table 7.3:** Self-supervised Point-GNN KITTI Test results: 3D $AP_{R_{40}}$ on Car class.

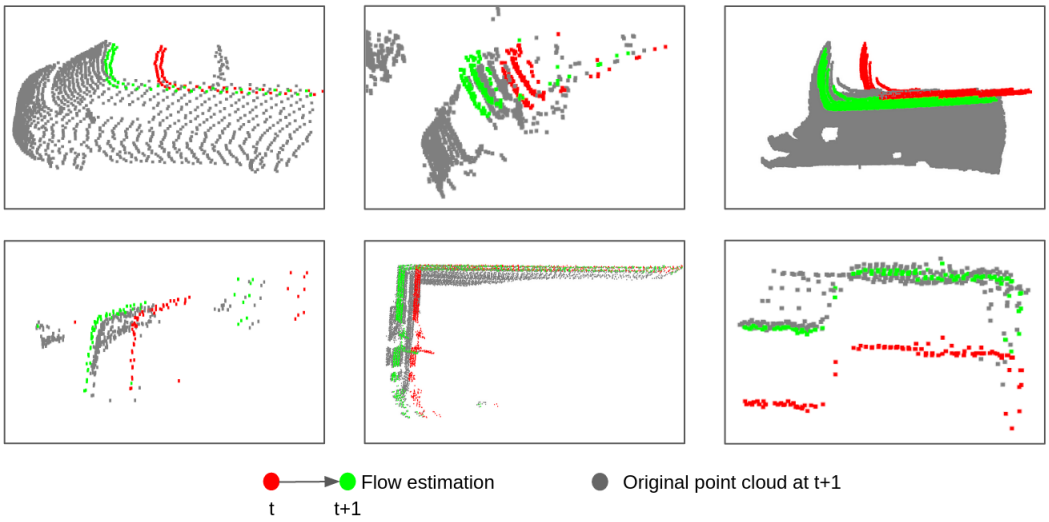



**Figure 7.5:** Sparsely-sampled lidar points (red) at frame $t$ are propagated to the frame $t+1$ (green) using the scene flow estimations of the Point-GNN-based scene flow network. Gray points show the original points at frame $t+1$. The propagated points (green) using flow estimation matches the original points (gray) closely. This indicates that the scene flow network learns useful flow features.

GNN. The first and the second rows show the 3D bounding boxes with the lidar point clouds from the BEV and the front-view, respectively. Each column is from a different scene of the KITTI 3D object detection validation set. We visualise the 3D bounding boxes on the image plane in the last row. In the first and second scenes, our self-supervised Point-GNN can detect the far-away car objects, which are represented with only a small number of lidar points compared to the other objects. On the other hand, the baseline Point-GNN misses them. The third scene is from a denser environment. Our self-supervised Point-GNN can detect the parked cars better compared to the baseline Point-GNN.

We visualize sparse scene flow estimations in Fig. 7.5. The estimations are obtained using Point-GNN backbone and the scene flow head. After the self-supervised scene flow training, we fine-tuned the network on the KITTI scene flow dataset for 100 epochs following [MOH20]. The red and gray points show the original point clouds at frame $t$ and $t+1$, respectively. The scene flow network estimates the flow vectors to propagate frame $t$ points to the frame $t+1$. The sparse propagated points are shown with the green colour. The visualization indicates that our network estimates the scene flow vectors accurately, and covers the object shapes and motion patterns with the propagated sparse points.

### 7.3.2 Main PointPillars Results

We apply the self-supervised pre-training to the voxel encoder of the PointPillars [Lan+19] 3D detector to achieve better 3D detection accuracy. The Table 7.4 compares our self-supervised pre-training method using the PointPillars with the PointPillars baseline as well as the state-of-the-art SECOND [YML18] 3D detector on the nuScenes validation set. We mainly use the nuScenes 3D detection benchmark metrics, which are the mAP, NDS, and AP per class. Our pre-training method improves the PointPillars 3D detection performance in mAP and NDS metrics by 2%. Additionally, our self-supervised PointPillars outperforms the baseline for all classes with considerable margins. Table 7.6 shows the self-supervised PointPillars results obtained on the nuScenes test server compared to the other PointPillars-based submissions in the nuScenes leaderboard. The self-supervised PointPillars also provide better

| Method | mAP | NDS | Car | Ped | Bus | Barrier | T. C. | Truck | Trailer | Moto. |
|---|---|---|---|---|---|---|---|---|---|---|
| SECOND [YML18] | 27.12 | - | 75.53 | 59.86 | 29.04 | 32.21 | 22.49 | 21.88 | 12.96 | 16.89 |
| PointPillars* [Lan+19] | 40.02 | 53.29 | 80.60 | 72.40 | 46.30 | 52.60 | 33.60 | 35.10 | 26.20 | 38.40 |
| **Self-supervised PointPillars** | **42.06** | **55.02** | **81.10** | **74.50** | **49.50** | **54.70** | **34.70** | **38.40** | **29.70** | **38.80** |

**Table 7.4:** 3D object detection comparison on nuScenes validation set. Our self-supervised PointPillars outperforms the baseline on mAP, the nuScenes Detection Score (NDS), and per-class AP metrics. We used mmdetection3d [Con20] checkpoint to obtain the baseline PointPillars scores. T.C. stands for the traffic cone and Moto. is for the motorcycle.

| Car (IoU=0.7) | 3D AP | | | BEV AP | | |
|---|---|---|---|---|---|---|
| Method | Easy | Mod | Hard | Easy | Mod | Hard |
| PointPillars [Lan+19] | 85.41 | 73.98 | 67.76 | 89.93 | 86.57 | 85.20 |
| **Self-supervised PointPillars** | **85.92** | **76.33** | **74.32** | **89.96** | **87.44** | **85.53** |
| Improvement | +0.51 | +2.36 | +6.56 | +0.03 | +0.87 | +0.33 |

**Table 7.5:** Self-supervised PointPillars 3D $AP_{R_{40}}$ results on KITTI validation set for the car class.

3D detection quality than the previously-submitted variants.

In addition to the nuScenes results, we evaluate our self-supervised PointPillars' performance on the KITTI validation set. Table 7.5 compares the baseline PointPillars with our self-supervised version on car class results. Our self-supervised scene flow pre-training helps enhance the 3D detection accuracy significantly. Even though the improvement is minor for the easy 3D AP and BEV AP scores, the self-supervised PointPillars outperforms the baseline with 2% in the moderate and 6% in the hard difficulty level. This improvement is consistent with the Point-GNN results that indicate usefulness of learning the spatio-temporal relation constructed by the self-supervised scene flow pre-training.

For the Point-GNN, we pre-train its whole backbone using the self-supervised scene flow task. However, in PointPillars we train only the voxel encoder, which is a small part of the PointPillars backbone, on the scene flow task. Nevertheless, we obtain a considerable increase in the 3D detection quality on the nuScenes dataset. This suggests that our self-supervised pre-training helps obtaining better point representations also for the 3D object detection. We use the mmedetection3d checkpoint [Con20] for the baseline validation results, which we assume to be optimized for the best detection scores. In addition, we trained our self-supervised PointPillars for the 3D detection with a smaller batch size than the mmdetection3d configurations due to the available GPU RAM constraint. Decreasing the batch size causes sub-optimal results, but our self-supervised PontPillars achieves significantly-improved results.

### 7.3.3 Ablation Studies

In this section, we investigate our self-supervised learning method more in depth through additional ablation experiments. In real-world 3D object detection problems, labelled datasets are usually limited. Therefore, we think that pre-training a part of 3D detector on the unla-

| Method | mAP | NDS | Car | Ped | Bus | Barrier | T. C. | Truck | Trailer | Moto. |
|---|---|---|---|---|---|---|---|---|---|---|
| PointPillars [Lan+19] | 30.50 | 45.30 | 68.40 | 59.70 | 28.20 | 38.90 | 30.80 | 23.00 | 23.40 | 27.40 |
| InfoFocus [Wan+20c] | 39.50 | 39.50 | 77.90 | 63.40 | 44.80 | 47.80 | 46.50 | 31.40 | 37.30 | 29.00 |
| PointPillars+ [Vor+20] | 40.10 | 55.00 | 76.00 | 64.00 | 32.10 | 56.40 | 45.60 | 31.00 | 36.60 | 34.20 |
| **Self-supervised PointPillars** | **43.63** | **56.28** | **81.00** | **73.10** | **37.10** | **58.20** | **47.80** | **36.10** | **41.80** | **35.40** |

**Table 7.6:** Self-supervised PointPillars result on nuScenes test set compared to other PointPillars-based submissions in the nuScenes leaderboard.

| Training Data Size | 1% | | | 5% | | | 20% | | |
|---|---|---|---|---|---|---|---|---|---|
| Car AP (IoU=0.7) | Easy | Mod | Hard | Easy | Mod | Hard | Easy | Mod | Hard |
| Point-GNN | 63.34 | 50.92 | 44.05 | 81.26 | 71.27 | 65.05 | 88.47 | 77.20 | 74.20 |
| **SSL Point-GNN** | **66.47** | **51.42** | **44.63** | **84.04** | **72.69** | **65.93** | **88.65** | **79.52** | **74.87** |
| Improvement | +3.13 | +0.50 | +0.58 | +2.78 | +1.42 | +0.88 | +0.18 | +2.32 | +0.67 |

**Table 7.7:** Self-supervised (SSL) Point-GNN trained with a percentage (1%, 5%, and 20%) of labelled 3D detection data. 3D $AP_{R_{40}}$ results for car class on KITTI validation set.

belled data helps achieving better 3D detection accuracy. We imitate this case using a small part of the annotated data in Subsection 7.3.3. In Subsection 7.3.3, we show that our alternating training strategy indeed provides better results. Our third ablation experiment in Subsection 7.3.3 shows that our pre-training method helps achieving better 3D detection performance already in the early stages of the training. In Subsection 7.3.3, we compare the training and validation loss values through the training of the baseline and our self-supervised Point-GNN.

#### Lack of annotated 3D detection data

We investigate how much improvement our self-supervised method provides in the lack of annotated datasets, while learning point features from the self-supervised scene flow task. For this experiment, we randomly sample 1%, 5%, and 20% of the KITTI 3D object detection training set to train the Point-GNN 3D detector. We train the Point-GNN baseline using the sampled percentage of the training data to obtain the 3D detection accuracy.

For our self-supervised Point-GNN, we train the scene flow network on the unlabelled KITTI tracking dataset with the self-supervised scene flow loss. Initializing Point-GNN's backbone with the weights from the scene flow pre-training, we train the self-supervised Point-GNN using the same sampled 3D detection training data. We do not apply the alternating strategy for these experiments. We do the supervised training of the baseline and the self-supervised Point-GNN for 1718 epochs.

Table 7.7 shows the results of the training on 1%, 5%, and 20% of the annotated data. Our self-supervised method outperforms the baseline in all difficulty levels for all training data samples. This shows that the self-supervised scene flow training provide a good initial representation for the 3D object detection especially when the labelled dataset is small. This is mostly the case in real-world scenarios since the 3D annotation of lidar point clouds is an expensive process. We note that the self-supervised Point-GNN is not trained with the alternating strategy but with the two-step approach.

We also conduct the same ablation study with the PointPillars on the nuScenes dataset. The training split sampling percentages are 2.5%, 5% and 10% for the nuScenes dataset. We train the baseline and self-supervised PointPillars on the small part of the training sets separately. Similar to the Point-GNN ablation experiment, we train the PointPillars voxel encoder on the scene flow task with the self-supervised loss. Before the supervised 3D detection training, we initialize the voxel encoder weights from the self-supervised scene flow training. The baseline's voxel encoder weights are initialized randomly.

Our PointPillars ablation results are given in Table 7.8. In all training experiments, our self-supervised pre-training improves the 3D detection quality for all metrics. This improvement is more obvious for 2.5% training data. Detection scores obtained from the nuScenes validation set are consistent with the previously-introduced results.

| Training Data Size | 2.5% | | | 5% | | | 10% | | |
|---|---|---|---|---|---|---|---|---|---|
| | mAP | Car | Ped | mAP | Car | Ped | mAP | Car | Ped |
| PointPillars | 2.99 | 24.60 | 10.30 | 10.93 | 52.30 | 29.00 | 13.02 | 55.50 | 32.20 |
| **SSL PointPillars** | **4.57** | **30.10** | **10.70** | **11.12** | **53.30** | **29.20** | **13.36** | **56.80** | **33.60** |

**Table 7.8:** Self-supervised (SSL) PointPillars results on the nuScenes validation set. Both the baseline and self-supervised PointPillars are trained only with 2.5%, 5%, and 10% of the nuScenes training split. Our self-supervised pre-training approach improves the results considerably for mAP, car, and pedestrian detection metrics.

| Car | 3D AP (IoU=0.7) | | | BEV AP (IoU=0.7) | | |
|---|---|---|---|---|---|---|
| Method | Easy | Mod | Hard | Easy | Mod | Hard |
| Point-GNN | 90.44 | 82.12 | 77.70 | 93.03 | 89.31 | 86.86 |
| Self-sup. Point-GNN | 91.05 | 82.35 | 77.80 | 93.43 | 89.59 | 87.03 |
| **Self-sup. Point-GNN w/ alternating tr.** | **91.43** | **82.85** | **80.12** | **93.55** | **89.79** | **87.23** |

**Table 7.9:** Ablation for the alternating training strategy between the self-supervised scene flow and the supervised 3D detection trainings. Repeating the scene flow and 3D detection trainings alternatingly improves the 3D detection accuracies on the KITTI 3D detection validation set.

#### Alternating training strategy

Our self-supervised pre-training strategy applies alternating training between the self-supervised scene flow and the supervised 3D detection tasks. Training the scene flow network, initializing weights of the 3D detection backbone from the scene flow network's backbone, and then further fine-tuning the 3D detection network on the labelled 3D detection is a one step of our self-supervised pre-training method. We repeat this step twice while obtaining the alternating training strategy results as explained in Section 7.2.

We summarise our alternating training ablation results in Table 7.9. The one step self-supervised training (Self-sup. Point-GNN in Table 7.9) already outperforms the baseline Point-GNN in all difficulty levels of 3D and BEV detection metrics. However, the alternating training strategy improves further the 3D detection performance and achieves a big improvement in the hard difficulty level. The backbone of the scene flow network is initialized from the 3D detection network's backbone in the alteration training step. Therefore, the self-supervised scene flow network is forced to fine-tune its point features considering the point features learned from the small-scale 3D detection dataset. This step implicitly raises the awareness for the 3D detection-specific point features for the scene flow network. Hence, we think that the improvement is larger in the alteration step than the first self-supervised training step.

#### Shorter supervised training

We also investigate the improvement obtained from the shorter training sessions. In Table 7.10, we summarise the results obtained from a $700k$-step training instead of $1400k$ steps. We report the results from the checkpoint of the smallest validation loss. In the first $700k$ steps, we already observe a large improvement in our self-supervised Point-GNN with alternating training compared to the baseline. We think that this is the effect of a point structure-aware backbone initialization for the 3D object detection task. Additionally, our self-supervised Point-GNN trained for $320k$ steps and it performs already better than the baseline without the alternating training.

| Car | 3D AP (IoU=0.7) | | | BEV AP (IoU=0.7) | | |
|---|---|---|---|---|---|---|
| Method | Easy | Mod | Hard | Easy | Mod | Hard |
| Point-GNN (700$k$) | 88.40 | 79.94 | 77.70 | 93.14 | 89.26 | 86.79 |
| Self-sup. Point-GNN (350$k$) | 89.09 | 80.18 | 77.51 | **93.24** | 89.35 | **86.84** |
| **Self-sup. Point-GNN w/ alternating tr. (**700$k$**)** | **91.23** | **82.44** | **77.87** | 93.22 | **89.38** | 86.77 |

**Table 7.10:** Comparison of self-supervised Point-GNN with the baseline after shorter training on the KITTI validation set. We train the Self-sup. Point-GNN for 320$k$ steps. The baseline and our self-supervised alternating training Point-GNN are trained 700$k$ steps. Both self-supervised versions outperform the baseline.

#### Comparison of the loss values

We also compare the loss values of the baseline Point-GNN training and our self-supervised Point-GNN trainings. Fig. 7.6 shows the training and validation loss values of the baseline Point-GNN with the pre-trained Point-GNNs through the 1400$k$-step training. The training and validation losses are shown with the orange and blue lines, respectively. The x-axis indicates the training steps. Compared to the baseline (the first row), the one-step self-supervised Point-GNN (the second row) has a smoother learning curve. However, the smallest loss of the baseline is 0.053 and the self-supervised Point-GNN is 0.055. On the other hand, the alternated self-supervised Point-GNN achieves 0.043 smallest validation loss. Hence, the improvement in the 3D detection performance over the baseline is more obvious after the alternating training strategy.

### 7.3.4 Discussion

Supervised learning is a well-investigated method for training neural networks for most of the problems as well as 3D object detection. Even though the size of labelled datasets grow, having 3D labels on unstructured lidar point clouds still requires expertise. Even for the experts, the labelling procedure is a time-consuming and expensive process. Therefore, learning from the inherent structure of the data brings might bring further improvement to the 3D detection quality.

In this chapter, we propose a self-supervised method to pre-train 3D detection networks' backbones. Our approach can be applied to all 3D detector architectures that can provide point-wise features. We pre-train the backbone on the self-supervised scene flow task. The scene flow task is relevant to learn point features since the geometrical differences between two adjacent frames are important for both the scene flow and the 3D detection problems. The learned point-wise features at the backbone provides a good initialization for the 3D detector. Therefore, the 3D detector does not have to learn the point features from the beginning only using the labelled data.

Our self-supervised method is simple, but effective. Even though similar methods have been proposed in different fields, the self-supervised learning for such complex vision tasks have been newly considered. We show the effectiveness of our method on two different types of 3D detection networks, Point-GNN and PointPillars, which utilises point- and voxel-level features. We also provide our results on two datasets, KITTI and nuScenes. Even though nuScenes dataset has different characteristics from the KITTI and is more challenging, our self-supervised pre-training method improves the 3D detection quality on both datasets.

Nevertheless, there are still open questions to be considered to increase the benefit from the self-supervised learning. Our scene flow network is restricted to the regional point embeddings to estimate the 3D flow vectors. In addition, our unlabelled number of frames are


**Figure 7.6:** Comparison of loss values between the Point-GNN baseline and our self-supervised Point-GNN. Orange and blue lines show the training and validation loss values. Our self-supervised training achieves a similar validation loss at the end of $1400k$ steps training (Self-sup. Point-GNN) with the baseline. The alternating training strategy, however, achieves a better validation loss than the baseline as well as the one-step self-supervised training.

still limited to about 12$k$ frames. The number of frames can be increased simply collecting new data. However, effects of different types of lidar sensor data on the scene flow and 3D detection quality are still to be investigated and beyond the scope of this work. Additionally, our alternating training strategy improves the results with a large margin. We apply the alternation after the scene flow network and the 3D detection network loss values start to converge. Alternating both trainings more frequently might help learning a joint point representation and hence leads to a larger improvement on the 3D detection quality. In addition, PointPillars voxel encoder shared between the scene flow and the 3D detection networks is small compared to the entire network size. Hence, we think that training a larger portion of the network with the self-supervised scene flow can be more beneficial for the 3D detection task. However, this brings another question: how higher-level regional features can be assigned to the individual points. Since the scene flow cycle consistency loss uses the distance between points from a local neighbourhood, the points will get the same high-level features. One solution might be applying bilinear interpolation, which can be investigated as a future work. All in all, we show that our self-supervised backbone training improves the 3D object detection quality and can be also extended to other vision tasks such as tracking and segmentation.

Chapter 8

# Conclusion and Considerations for Future Work

## 8.1 Discussion of Research Questions

In this work, building on previous knowledge, we considered the main 3D temporal perception problem from three perspectives: (i) aggregating scene-level feature maps from consecutive time steps, (ii) aggregating object-specific feature vectors through sequential frames, and (iii) benefiting from another inherently-sequential perception task for learning better data representations.

In Chapter 5, our goal was to answer the first research question: *How can we process consecutive scene feature maps to improve 3D object detection quality?*.

Feature maps obtained from the entire scene provides regional encodings of the objects at a single time step. Due to moving objects and ego vehicle, there is a disparity between two consecutive frames. This disparity can be beneficial for seeing objects from different perspectives and filling in missing information in the current frame using previous data. In this chapter, we showed that 3D detection based on generated temporal feature map of the entire scene improves 3D object detection results. In our experiments, we used AVOD 3D detector, which takes RGB image and lidar point cloud inputs. Feature maps of two modalities were generated separately. We extended the existing 3D detector with a convolutional LSTM layer, which recurrently processes sequential feature maps and generates a temporal feature map at the end of the sequence. The convolutional LSTM's receptive field can account for the motion offsets between consecutive frames. Further, convolution operation can better capture similar patterns in consecutive frames compared with the FC layers of LSTMs. We added the convolutional LSTMs to the two data branches separately and used the generated temporal feature map for the region proposal network as well as the detection head. Our experiments showed that the multi-frame AVOD outperforms the baseline AVOD with a large margin on all difficulty levels for car and pedestrian classes. Our ablation studies suggested that the biggest temporal contribution originates from the temporal point cloud branch. Compared with the RGB image input, BEV lidar feature map has more accurate depth information, which helps accumulate feature map changes from consecutive frames and learning temporal correspondences more accurately. On the other hand, the BEV map is generated with a heuristic method that takes the height of points in slices as features. This limits feature aggregation through multiple frames. Nevertheless, fusing time-encoded BEV features with image features further enhances the 3D detection accuracy. Using larger Conv LSTM filter size contributes to better 3D detection results. Finally, our multi-frame scene-level 3D object detection method significantly outperformed the baseline method. Our results indicated that processing sequential feature maps from different modalities with the help of convolutional LSTM layers is a beneficial method to improve 3D object detection quality using history data.

In Chapter 6, we aimed to answer our second research question: *How can we tempo-*

*rally process objects' point cloud features from sequential frames to improve 3D object detection quality?*.

Scene-level temporal processing needs to account for the offset between consecutive frames caused by the moving objects. Therefore, the receptive field of the convolutional recurrent layer needs to learn the shifts. On the other hand, objects are represented with features extracted from a certain region where the bounding box parameters are estimated. In each frame of a video, the same object is represented with a different set of features since the data reflected from the object as well as the object state vary. As discussed for the scene-level approach, regional object features that change through multiple time steps can contribute to obtaining better representations.

The object-level feature fusion in time requires matching correct feature sets from successive frames. We proposed the use of RGB image and lidar point cloud data as a solution to this problem and demonstrated that our approach improves the 3D object detection accuracy for car, pedestrian, and cyclist classes. For our multi-frame feature alignment method, we followed Frustum PointNet 3D detector's approach and extend it for multi-frame processing. The Frustum PointNet estimates 2D bounding boxes using single-frame RGB images to reduce the search space in the entire point cloud. Lidar points inside a 2D bounding box frustum are processed with PointNets to separate the foreground (object) from the background. The foreground points are further processed with PointNets to estimate 3D bounding boxes and the PointNet features represent the object's size, position, objectness, and heading.

We used the object-level PointNet features for aggregating object representations in time. In addition to the Frustum PointNet, we estimated object tracking IDs using the RGB video. The unique object tracking IDs were used to match object features from the preceding frames and the current frame. We aligned the features in time using recurrent layers, which output a temporal feature vector of the object. Our assumption was that the temporal feature would be more representative since it is an aggregation across the data history. To validate applicability of temporal object-level features for 3D object detection, we considered three multi-frame feature alignment strategies: (i) we used the temporal object-level feature directly for 3D bounding box estimation, (ii) we followed a residual connection strategy for the temporal and single-frame object-level features, and (iii) we used the temporal object-level feature to estimate shape and the single-frame object-level feature to estimate the rest of the bounding box parameters. We find out that the third strategy provides better results and outperforms the baseline Frustum PointNet for all classes. The shape of the same object remains the same through multiple frames while the rest of the parameters change. Thus, the network can estimate the shape parameters better using a richer aggregated temporal object feature. Additionally, changing temporal features in time forces the network to learn shape-specific feature representations more accurately in a similar way with the dropout. Since the preceding layers of the recurrent layers are shared between the multi-frame and single-frame branches, the multi-frame branch is an auxiliary network to the single-frame parameter estimation head. Hence, the single-frame branch takes time-aware single-frame object features as input. The time-awareness also helps learning better single-frame feature extraction for the layers before the recurrent layers.

The performance increase obtained with our multi-frame method over the baseline was higher for pedestrian and cyclist classes compared with the car class. In the KITTI dataset, the pedestrian and cyclist classes are more underrepresented than the car class. This suggests that applying a multi-frame approach can be more beneficial when annotated data is not available. Since obtaining 3D annotations for the lidar point clouds is a difficult and expensive process, this is an encouraging result for real-world applications, where annotation size is restricted. Finally, we proposed a two-branch training strategy with recurrent layers as a solution and showed that object-level temporal feature aggregation improves the 3D object

detection results over the single-frame approach through extensive experiments.

In Chapter 7, we considered our third and final research question: *How can we make use of point cloud representations learned through an inherently-temporal perception task to improve 3D object detection quality?*

Learning motion representations is important for perception problems in autonomous driving such as for multi-object tracking and optical flow. Motion representations can be also important for 3D object detection and segmentation for distinguishing objects from each other as well as from the background using differences in their motion patterns. Therefore, integrating training for the motion representations into the 3D object detection pipeline can improve the perception accuracy. The scene-level feature aggregation from multiple time steps is expected to learn motion-related features assumed that the motion shifts among the consecutive feature maps perceived by the receptive fields of the convolutional operations. However, this is not explicitly enforced with 3D object detection loss functions. On the other hand, our object-level feature aggregation method considered regional features and motion-related information was handled by the tracking module. Our object-level method only focused on the relation between features from multiple time steps and learning how to construct a richer representation of the object for 3D object detection results. Hence, 3D object detection could benefit from learning motion-related features explicitly. Motion features can be learned with another perception task that explicitly relies on changes on the patterns to reach its goal and the object features become more separable from its environment. Multi-object tracking is a good candidate, however, relying on expert annotations limits its applicability. Scene flow problem, being the 3D version of the well-studied optical flow, arises as another good candidate for this idea since the structural consistency between two consecutive frames makes using self-supervised learning methods possible without the need for large-scale data labels.

We expected self-supervised scene flow to learn initial motion-related point cloud representations for the 3D object detection network. The scene flow network takes two consecutive point clouds and extracts two sets of point cloud features using the 3D detection backbone. With the point features and the scene flow head, the 3D motion vectors of the points can be estimated, which propagate points from the previous frame to the following frame. The self-supervised cycle consistency loss is used to train the 3D detection backbone and the scene flow head inspired from FlowNet3D. A point in the current frame can be propagated to the following frame with a forward flow and can be propagated back with the backward flow. Cycle consistency aims to calculate the difference between the original point and the backward flow. Therefore, the 3D detection backbone is pre-trained with unlabelled dataset to learn motion representations of the given consecutive point clouds. Pre-training of the 3D detection backbone provides a good initialisation for the 3D detection network to learn distinguishable object features, which is trained further with the supervised learning. We followed an interwoven 4-step alternating training scheme for the self-supervised scene flow and 3D object detection. The scene flow network was first trained with self-supervised cycle consistency loss. Initialising the 3D detection backbone from the scene flow training, 3D detection network was trained with the supervised learning. With the initialisation of the backbone and head networks from previous training, the two steps were repeated. Our experiments with Point-GNN and PointPillars 3D detection networks on KITTI and nuScenes datasets showed that the self-supervised learning improves the 3D detection accuracy especially for the objects that are hard to detect. The hard objects are mostly represented with smaller number of points compared to the other categories and they might be far-away or occluded. Therefore, the learned point representations are especially useful for extracting information from the limited input as our experiments indicate. In addition, our self-supervised scene flow

pre-training method considerably improved the baseline in the absence of annotated data for supervised 3D object detection. This result is also consistent with the real-world scenarios, where the annotations are limited. Our alternating training strategy, which has been previously considered for the training of two-stage 2D detectors, further boosts the 3D detection accuracy. We believe that alternating the training of two different networks helps the converging of useful and motion-aware point features in the point cloud backbone.

Initialising backbones of deep neural networks with the weights learned from a large dataset for another task is a common approach in 2D detection. However, this has not been an approach for 3D object detection since a large-scale point cloud dataset similar with the ImageNet [Den+09] is not available. Such an initialisation helps starting with useful data representations and shortens the training time. The training curves of our pre-trained 3D object detection network suggested that the 3D detector can already reach better 3D detection accuracy compared to the baseline in the early steps of the training. This also showed that the self-supervised scene flow pre-training works in a similar direction with the common 2D detection training approach. Finally, our motion-aware pre-training strategy from consecutive point cloud features improved the 3D detection results consistently for voxel-based and point-based 3D detection networks on nuScenes and KITTI datasets. We also showed that the self-supervised scene flow training using consecutive data is a successful method for using sequential data for a 3D object detection perception task.

## 8.2 Limitations

In this work, we proposed two methods to directly use multi-frame inputs for 3D object detection in addition to the motion-aware self-supervised learning method to pre-train a point cloud feature extractor with two consecutive point clouds. Our experimental results demonstrated that the proposed methods outperformed their baselines and improved 3D object detection accuracy. Nonetheless, several limitations must be considered.

- **Heuristic BEV point cloud inputs**
  In Chapter 5, we aggregated sequential feature maps from RGB images and point clouds using Conv LSTMs separately on two branches. Our experimental results showed that using only a Conv LSTM in the point cloud branch provides similar 3D detection performance with the fully-temporal model. Nevertheless, the Conv LSTM can only account for the motion using the hand-crafted lidar point cloud input encodings. We believe that the hand-crafted BEV input generation process, which considers the height of points in horizontal slices as representative features, limits learning how to arrange point cloud features with regards to the feature enhancement and motion capturing. This data encoding method might yield empty fields, which renders Conv LSTMs's learning of motion patterns difficult. Hence, learning-based point cloud features are more suitable for multi-frame feature aggregation with Conv LSTMs.

- **Generalisation over different datasets**
  Available 3D object detection datasets have different characteristics. KITTI data are mainly sampled at 10 Hz with a 64-channel lidar sensor. Annotations are provided for each sampled frame. On the other hand, nuScenes data are sampled at 20 Hz with a 32-channel lidar sensor. Annotations are provided only for every 0.5 seconds. This makes the tuning of hyper-parameters of networks difficult for a generalisable setup. In addition, visible fields defined for benchmarks also differ. The density of point clouds depends on the lidar sensor, which changes the point sampling method parameters as well as the point cloud encoding method. Furthermore, change of the sampling rate of

the data affects the chosen length of the input sequence. A common approach for the nuScenes data is combining unlabelled data sweeps with labelled key frame data, which introduces noisy shapes. This might cause difficulties for the multi-frame approaches in learning the patterns between consecutive frames. Therefore, the models might require a hyper-parameter tuning if applied to another dataset.

- **Sub-optimal matching for object-level feature aggregation**
  Our object-level multi-frame feature alignment method used of 2D bounding boxes and tracking IDs to get features of the same object from multiple frames. Even though our method improved the baseline and the baseline also relied on the 2D bounding boxes, once a 2D object is missed its features will not be aggregated. In addition, the tracking algorithm is still not perfect and can fail in assigning the same tracking ID to the same object. For this case, the network might require an additional fine-tuning with imperfect tracking IDs.

- **Multi-task training of scene flow and 3D detection**
  We trained the scene flow and 3D detection networks, which have a shared backbone, in an alternating way. The 3D object detection training started with motion-aware features, but fine-tuned the network with detection losses. Hence, the scene flow features were overwritten. Even though our alternating strategy improved the results, the optimal training strategy remains an open research question. Multi-task training of both networks is possible, but the unlabelled data for scene flow will not be used for 3D object detection. The bias in the data towards the scene flow task must also be considered.

## 8.3 Considerations for Future Work

Multi-frame processing for 3D detection task has gained attention recently with the increasing availability of labelled datasets. Compared to 2D video object detection, multi-frame 3D object detection remains understudied, mostly since lidar point clouds have not been investigated as extensively as RGB images for temporal processing. In this section, we discuss the potential of sequential data processing for 3D object detection to overcome current limitations in 3D object detection.

Our work showed that multi-frame scene-level feature map aggregation improves 3D detection quality. However, feature learning directly from point cloud data might provide better results than hand-crafted point cloud input embeddings. This would help recurrent layers to learn better representations from point cloud patterns without the information loss brought in by the heuristic feature engineering. The Conv LSTM layer learns temporal relation between consecutive feature maps. However, the receptive field of the Conv LSTM is limited and this leads to the consideration of contextual information instead of the whole object region. Therefore, saliency of the region should also contribute to the multi-frame aggregation, which can be enabled by applying attention mechanisms that takes more informative regions into account.

Aggregating object features from sequential frames requires finding the same object's features in all the frames. Even though using 2D tracking IDs is a successful method, it does not provide the optimum solution. Attention mechanisms have been recently considered for finding correlations between a set of data pieces. Investigating attention architectures for aggregating object-level features remains a promising direction for the future work.

Our self-supervised 3D object detection pre-training method provided encouraging regarding the use of point cloud motion representations. However, our 3D object detector still

relies on single frames. As a future work, extending the 3D detector to take advantage of multiple frames might help increase the benefits gained from learned self-supervised motion representations. Furthermore, GNNs are natural candidates of multi-frame processing by learning correct edge connections between objects of sequential time steps. Such a multi-frame 3D detector would also provide multi-object tracking associations without additional processing since correctly connected graph edges relates the same object through multiple frames.

Multi-frame 3D object detection has not been researched as extensively as single-frame 3D object detection methods, and many research questions regarding sequential data processing remain unaddressed. Multi-frame point cloud processing for better 3D representation learning is a promising research direction for reaching more consistent perception results.

Appendix A

# Supervised Student Theses

The following master's and bachelor's theses as well as guided research works have been advised during this dissertation. Some of the results presented in this dissertation was partly investigated during these works.

1. Guided research: Self-supervised scene flow approaches for 3D lidar point clouds, Zhijie Yang, Nov.2021-Sep 2022
2. Guided research: Analysis of flow-based 3D perception tasks, Hanzhen Zhang, Dec. 2021-Jun. 2022
3. Master's thesis: Spatio-temporal Graph Neural Networks for Object Detection, Maximilian Listl, Nov. 2021-Jun. 2022
4. Master's thesis: Multi-frame self-supervised 3D object detection, Pınar Topçam, Nov. 2021-May 2022
5. Master's thesis: Self-supervised 3D Multi-object Tracking, Burak Tomruk, Nov. 2021-May 2022
6. Bachelor's thesis: Analysis of Self-supervised Scene Flow for Perception, Aysu Konyala, Nov. 2021-Apr. 2022
7. Master's thesis: Spatio-temporal Graph Neural Networks for 3D Object Detection, Mingyu Liu, Oct. 2021-Feb.2022
8. Master's thesis: Attention-based Temporal Frustum Pointnet for 3D Object Detection, Sebastian Grauff, Apr.2021-Jan.2022
9. Master's thesis: Multi-Person and Vehicle Detection and Tracking in Aerial Imagery Sequences using Deep Learning Algorithms, Tsuyoshi Beheim, Apr.2021-Nov.2021
10. Bachelor's thesis: Transfer learning for Temporal 3D Object Detection, Kaan Yılmaz Çaylı, Apr.2021-Aug.2021
11. Bachelor's thesis: Improving orientation estimation for Radar-based object detection, Taner Kurt, Nov.2020-May2021
12. Guided research: Novel methods and challenges of processing time-series data for object detection, Sadık Ekin Özbay, Nov. 2020-Mar.2021
13. Master's thesis: Analysis of 3D Object Association Methods for RGB-D Data, Patrick von Haller, Jun.2020-Feb.2021
14. Master's thesis: 3D object tracking combining LiDAR with temporal image cues, Eda Çiçek, Apr.2020-Dec.2020

15. Bachelor's thesis: Analysis of proposal improvement on Faster RCNN, Salma Mrabet, Jul.2020-Nov2020
16. Master's thesis: 3D object detection using temporal data fusion, Zhiran Yan, Jan.2020-Jul.2020
17. Master's thesis: Efficient object detection and tracking for indoor applications, Robert Walter, Nov.2019-May2020
18. Master's thesis: Multi-Object Tracking in Aerial and Satellite Imagery, Maximilian Kraus, Nov.2019-May2020
19. Master's thesis: Processing video data to improve 2D object detection, Arnd Pettirsch, Oct. 2019-Apr.2020
20. Master's thesis: Deep Learning based People Counting Using 60 GHz FMCW Radar Sensor, Cem Yusuf Aydoğdu, Oct. 2019-Apr.2020
21. Master's thesis: Data fusion from heterogeneous sensor for indoor scenarios, Okan Kamil Şen, Sep.2019-Mar.2020
22. Master's thesis: Radar-based object detection using deep neural networks, Maria Dreher, Jun.2019-Dec.2019
23. Master's thesis: Motion cueing for the control of a car simulator using RNN and RL, Ahmet Burakhan Koyuncu, Apr.2019-Nov.2019
24. Master's thesis: Sensor fusion for object detection on simulator data, Tuba Topaloğlu, Feb.2019-Nov.2019
25. Master's thesis: Sensor fusion through 3D proposals using data generated from simulators, Ruslan Noschajew, Apr.2019-Oct.2019
26. Research internship: Integration of depth information for LSTM-based online video object detection, Mert Keser, Mar.2019-Apr.2019
27. Bachelor's thesis: The recognition and evaluation of traffic signs and its additional information with machine learning in the context of autonomous driving, Tobias Weber, Nov.2018-Apr.2019
28. Guided research: Methods to transfer knowledge from simulation into real-world for object detection problem, Anshul Sharma, Nov.2018-Apr.2019
29. Master's thesis: Evaluation of an object detection model adding recurrency, Daniel Haller, Nov.2018-Apr.2019

# Bibliography


[APM21] Aleotti, F., Poggi, M., and Mattoccia, S. “Learning Optical Flow From Still Images”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2021, pp. 15201–15211.

[Arn+21] Arnab, A., Dehghani, M., Heigold, G., Sun, C., Lučić, M., and Schmid, C. “Vivit: A video vision transformer”. In: *arXiv preprint arXiv:2103.15691* (2021).

[ASS21] Arnab, A., Sun, C., and Schmid, C. “Unified Graph Structured Models for Video Understanding”. In: *arXiv preprint arXiv:2103.15662* (2021).

[Bas+19] Baser, E., Balasubramanian, V., Bhattacharyya, P., and Czarnecki, K. “Fantrack: 3d multi-object tracking with feature association network”. In: *2019 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2019, pp. 1426–1433.

[Bat+19] Battrawy, R., Schuster, R., Wasenmüller, O., Rao, Q., and Stricker, D. “Lidar-flow: Dense scene flow estimation from sparse lidar and stereo images”. In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2019, pp. 7762–7769.

[Bau+19] Baur, S. A., Moosmann, F., Wirges, S., and Rist, C. B. “Real-time 3D LiDAR flow for autonomous vehicles”. In: *2019 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2019, pp. 1288–1295.

[Bau+21] Baur, S. A., Emmerichs, D. J., Moosmann, F., Pinggera, P., Ommer, B., and Geiger, A. “SLIM: Self-Supervised LiDAR Scene Flow and Motion Segmentation”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 13126–13136.

[BTS18] Bertasius, G., Torresani, L., and Shi, J. “Object detection in video with spatiotemporal sampling networks”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 331–346.

[Bew+16] Bewley, A., Ge, Z., Ott, L., Ramos, F., and Upcroft, B. “Simple online and realtime tracking”. In: *2016 IEEE international conference on image processing (ICIP)*. IEEE. 2016, pp. 3464–3468.

[BC20] Bhattacharyya, P. and Czarnecki, K. “Deformable PV-RCNN: Improving 3D object detection with learned deformations”. In: *arXiv preprint arXiv:2008.08766* (2020).

[BHC21] Bhattacharyya, P., Huang, C., and Czarnecki, K. “SA-Det3D: Self-Attention Based Context-Aware 3D Object Detection”. In: *arXiv preprint arXiv:2101.02672* (2021).

[Cae+20] Caesar, H., Bankiti, V., Lang, A. H., Vora, S., Liong, V. E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., and Beijbom, O. “nuscenes: A multimodal dataset for autonomous driving”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 11621–11631.

[Cao+21] Cao, A.-Q., Puy, G., Boulch, A., and Marlet, R. “PCAM: Product of Cross-Attention Matrices for Rigid Registration of Point Clouds”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 13229–13238.

[Car+20] Caron, M., Misra, I., Mairal, J., Goyal, P., Bojanowski, P., and Joulin, A. “Unsupervised learning of visual features by contrasting cluster assignments”. In: *arXiv preprint arXiv:2006.09882* (2020).

[CZ17] Carreira, J. and Zisserman, A. “Quo vadis, action recognition? a new model and the kinetics dataset”. In: *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 6299–6308.

[Cha+21] Chai, Y., Sun, P., Ngiam, J., Wang, W., Caine, B., Vasudevan, V., Zhang, X., and Anguelov, D. “To the Point: Efficient 3D Object Detection in the Range Image With Graph Convolution Kernels”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 16000–16009.

[CCC21] Chang, J.-R., Chen, Y.-S., and Chiu, W.-C. “Learning Facial Representations from the Cycle-consistency of Face”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 9680–9689.

[Cha+19] Chang, M.-F., Lambert, J. W., Sangkloy, P., Singh, J., Bak, S., Hartnett, A., Wang, D., Carr, P., Lucey, S., Ramanan, D., and Hays, J. “Argoverse: 3D Tracking and Forecasting with Rich Maps”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[Che+20a] Chen, M., Radford, A., Child, R., Wu, J., Jun, H., Luan, D., and Sutskever, I. “Generative pretraining from pixels”. In: *International Conference on Machine Learning*. PMLR. 2020, pp. 1691–1703.

[Che+20b] Chen, Q., Sun, L., Wang, Z., Jia, K., and Yuille, A. “Object as hotspots: An anchor-free 3d object detection approach via firing of hotspots”. In: *European Conference on Computer Vision*. Springer. 2020, pp. 68–84.

[Che+15] Chen, X., Kundu, K., Zhu, Y., Berneshawi, A. G., Ma, H., Fidler, S., and Urtasun, R. “3d object proposals for accurate object class detection”. In: *Advances in Neural Information Processing Systems*. Citeseer. 2015, pp. 424–432.

[Che+17] Chen, X., Ma, H., Wan, J., Li, B., and Xia, T. “Multi-view 3d object detection network for autonomous driving”. In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 2017, pp. 1907–1915.

[Che+21] Chen, Y., Liu, J., Ni, B., Wang, H., Yang, J., Liu, N., Li, T., and Tian, Q. “Shape Self-Correction for Unsupervised Point Cloud Understanding”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 8382–8391.

[Che+20c] Chen, Y., Liu, S., Shen, X., and Jia, J. “Dsgn: Deep stereo geometry network for 3d object detection”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 12536–12545.

[Chi+21] Chi, C., Wang, Q., Hao, T., Guo, P., and Yang, X. “Feature-Level Collaboration: Joint Unsupervised Learning of Optical Flow, Stereo Depth and Camera Motion”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2021, pp. 2463–2473.

[CCS21] Choudhuri, A., Chowdhary, G., and Schwing, A. G. “Assignment-Space-based Multi-Object Tracking and Segmentation”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 13598–13607.

[Chu+21] Chu, P., Wang, J., You, Q., Ling, H., and Liu, Z. “TransMOT: Spatial-Temporal Graph Transformer for Multiple Object Tracking”. In: *arXiv preprint arXiv:2104.00194* (2021).

[Chu+14] Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. “Empirical evaluation of gated recurrent neural networks on sequence modeling”. In: *NIPS 2014 Workshop on Deep Learning, December 2014*. 2014.

[Con+21] Cong, Y., Liao, W., Ackermann, H., Rosenhahn, B., and Yang, M. Y. “Spatial-Temporal Transformer for Dynamic Scene Graph Generation”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 16372–16382.

[Con20] Contributors, M. *MMDetection3D: OpenMMLab next-generation platform for general 3D object detection*. https://github.com/open-mmlab/mmdetection3d. 2020.

[Cre+22] Creß, C., Zimmer, W., Leah, S., Fortkord, M., Dai, S., Lakshminarasimhan, V., and Knoll, A. “A9-Dataset: Multi-Sensor Infrastructure-Based Dataset for Mobility Research”. In: 2022. URL: https://a9-dataset.com.

[Cui+21a] Cui, Y., Chen, R., Chu, W., Chen, L., Tian, D., Li, Y., and Cao, D. “Deep learning for image and point cloud fusion in autonomous driving: A review”. In: *IEEE Transactions on Intelligent Transportation Systems* (2021).

[Cui+21b] Cui, Y., Yan, L., Cao, Z., and Liu, D. “TF-Blender: Temporal Feature Blender for Video Object Detection”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 8138–8147.

[Dai+17] Dai, A., Chang, A. X., Savva, M., Halber, M., Funkhouser, T., and Nießner, M. “Scannet: Richly-annotated 3d reconstructions of indoor scenes”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 5828–5839.

[Den+19] Deng, H., Hua, Y., Song, T., Zhang, Z., Xue, Z., Ma, R., Robertson, N., and Guan, H. “Object guided external memory network for video object detection”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 6678–6687.

[Den+09] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. “ImageNet: A Large-Scale Hierarchical Image Database”. In: *CVPR09*. 2009.

[Den+20] Deng, J., Shi, S., Li, P., Zhou, W., Zhang, Y., and Li, H. “Voxel R-CNN: Towards High Performance Voxel-based 3D Object Detection”. In: *arXiv preprint arXiv:2012.15712* (2020).

[Den+21] Deng, Z., Yao, Y., Deng, B., and Zhang, J. “A robust loss for point cloud registration”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 6138–6147.

[Dos+20] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”. In: *International Conference on Learning Representations*. 2020.

[Dre+20] Dreher, M., Erçelik, E., Bänziger, T., and Knol, A. “Radar-based 2D Car Detection Using Deep Neural Networks”. In: *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2020, pp. 1–8.

[Du+20] Du, L., Ye, X., Tan, X., Feng, J., Xu, Z., Ding, E., and Wen, S. “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.

[Dua+19] Duan, K., Bai, S., Xie, L., Qi, H., Huang, Q., and Tian, Q. “Centernet: Keypoint triplets for object detection”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 6569–6578.

[DB20] Duffhauss, F. and Baur, S. A. “PillarFlowNet: A Real-time Deep Multitask Network for LiDAR-based 3D Object Detection and Scene Flow Estimation”. In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2020, pp. 10734–10741.

[El +18] El Sallab, A., Sobh, I., Zidan, M., Zahran, M., and Abdelkarim, S. “Yolo4d: A spatio-temporal approach for real-time multi-object detection and classification from lidar point clouds”. In: (2018).

[EYK21a] Erçelik, E., Yurtsever, E., and Knoll, A. “3D Object Detection With Multi-Frame RGB-Lidar Feature Alignment”. In: *IEEE Access* 9 (2021), pp. 143138–143149.

[EYK21b] Erçelik, E., Yurtsever, E., and Knoll, A. “Temp-Frustum Net: 3D Object Detection with Temporal Fusion”. In: *2021 IEEE Intelligent Vehicles Symposium (IV)*. 2021.

[Erç+22] Erçelik, E., Yurtsever, E., Liu, M., Yang, Z., Zhang, H., Topçam, P., Listl, M., Çaylı, Y. K., and Knoll, A. “3D Object Detection with a Self-supervised Lidar Scene Flow Backbone”. In: *Computer Vision – ECCV 2022*. Ed. by Avidan, S., Brostow, G., Cissé, M., Farinella, G. M., and Hassner, T. Cham: Springer Nature Switzerland, 2022, pp. 247–265. ISBN: 978-3-031-20080-9.

[Eve+15] Everingham, M., Eslami, S. A., Van Gool, L., Williams, C. K., Winn, J., and Zisserman, A. “The pascal visual object classes challenge: A retrospective”. In: *International journal of computer vision* 111.1 (2015), pp. 98–136.

[FPZ16] Feichtenhofer, C., Pinz, A., and Zisserman, A. “Convolutional two-stream network fusion for video action recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 1933–1941.

[Fen+21] Feng, W., Wang, Y., Ma, L., Yuan, Y., and Zhang, C. “Temporal Knowledge Consistency for Unsupervised Visual Representation Learning”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 10170–10180.

[Fis+21] Fischer, K., Simon, M., Olsner, F., Milz, S., Gross, H.-M., and Mader, P. “Stickypillars: Robust and efficient feature matching on point clouds using graph neural networks”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 313–323.

[Fru+21] Fruhwirth-Reisinger, C., Opitz, M., Possegger, H., and Bischof, H. “FAST3D: Flow-Aware Self-Training for 3D Object Detectors”. In: *Proc. British Machine Vision Conference (BMVC)*. 2021.

[Gan+21] Ganeshan, A., Vallet, A., Kudo, Y., Maeda, S.-i., Kerola, T., Ambrus, R., Park, D., and Gaidon, A. “Warp-Refine Propagation: Semi-Supervised Auto-labeling via Cycle-consistency”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 15499–15509.

[GL21] Garnot, V. S. F. and Landrieu, L. “Panoptic Segmentation of Satellite Image Time Series with Convolutional Temporal Attention Networks”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 4872–4881.

[Ge+20] Ge, R., Ding, Z., Hu, Y., Wang, Y., Chen, S., Huang, L., and Li, Y. “Afdet: Anchor free one stage 3d object detection”. In: *arXiv preprint arXiv:2006.12671* (2020).

[GLU12a] Geiger, A., Lenz, P., and Urtasun, R. “Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2012.

[GLU12b] Geiger, A., Lenz, P., and Urtasun, R. “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.

[Goj+21] Gojcic, Z., Litany, O., Wieser, A., Guibas, L. J., and Birdal, T. “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.

[Gra+21] Graber, C., Tsai, G., Firman, M., Brostow, G., and Schwing, A. G. “Panoptic Segmentation Forecasting”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2021, pp. 12517–12526.

[Gre+16] Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., and Schmidhuber, J. “LSTM: A search space odyssey”. In: *IEEE transactions on neural networks and learning systems* 28.10 (2016), pp. 2222–2232.

[Gu+19] Gu, X., Wang, Y., Wu, C., Lee, Y. J., and Wang, P. “Hplflownet: Hierarchical permutohedral lattice flownet for scene flow estimation on large-scale point clouds”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 3254–3263.

[Guo+21] Guo, Z., Huang, Y., Hu, X., Wei, H., and Zhao, B. “A Survey on Deep Learning Based Approaches for Scene Understanding in Autonomous Driving”. In: *Electronics* 10.4 (2021), p. 471.

[GDS21] Gustafsson, F. K., Danelljan, M., and Schon, T. B. “Accurate 3D Object Detection using Energy-Based Models”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 2855–2864.

[Han+20] Han, K., Wang, Y., Chen, H., Chen, X., Guo, J., Liu, Z., Tang, Y., Xiao, A., Xu, C., Xu, Y., et al. “A survey on visual transformer”. In: *arXiv preprint arXiv:2012.12556* (2020).

[Han+19] Hanke, T., Hirsenkorn, N., van-Driesten, C., Garcia-Ramos, P., Schiementz, M., Schneider, S., and Biebl, E. “A generic interface for the environment perception of automated driving functions in virtual scenarios”. In: *Internet: https://www.hot. ei. tum. de/forschung/automotive-veroeffentlichungen* (2019).

[He+20a] He, C., Zeng, H., Huang, J., Hua, X.-S., and Zhang, L. “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.

[He+20b] He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R. “Momentum contrast for unsupervised visual representation learning”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 9729–9738.

[He+17] He, K., Gkioxari, G., Dollár, P., and Girshick, R. “Mask r-cnn”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2961–2969.

[Hu+19] Hu, H.-N., Cai, Q.-Z., Wang, D., Lin, J., Sun, M., Krahenbuhl, P., Darrell, T., and Yu, F. "Joint monocular 3D vehicle detection and tracking". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 5390–5399.

[Hu+20] Hu, P., Ziglar, J., Held, D., and Ramanan, D. "What you see is what you get: Exploiting visibility for 3d object detection". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 11001–11009.

[Hua+17] Huang, P., Cheng, M., Chen, Y., Luo, H., Wang, C., and Li, J. "Traffic sign occlusion detection using mobile laser scanning point clouds". In: *IEEE Transactions on Intelligent Transportation Systems* 18.9 (2017), pp. 2364–2376.

[Hua+20a] Huang, R., Zhang, W., Kundu, A., Pantofaru, C., Ross, D. A., Funkhouser, T., and Fathi, A. "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.

[Hua+21] Huang, S., Xie, Y., Zhu, S.-C., and Zhu, Y. "Spatio-temporal Self-Supervised Representation Learning for 3D Point Clouds". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 6535–6545.

[Hua+20b] Huang, T., Liu, Z., Chen, X., and Bai, X. "Epnet: Enhancing point features with image semantics for 3d object detection". In: *European Conference on Computer Vision*. Springer. 2020, pp. 35–52.

[HD07] Huguet, F. and Devernay, F. "A variational method for scene flow estimation from stereo sequences". In: *2007 IEEE 11th International Conference on Computer Vision*. IEEE. 2007, pp. 1–7.

[Hui+21] Hui, L., Yang, H., Cheng, M., Xie, J., and Yang, J. "Pyramid Point Cloud Transformer for Large-Scale Place Recognition". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 6098–6107.

[Huo+21] Huo, X., Xie, L., He, J., Yang, Z., Zhou, W., Li, H., and Tian, Q. "ATSO: Asynchronous Teacher-Student Optimization for Semi-Supervised Image Segmentation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2021, pp. 1235–1244.

[HR20] Hur, J. and Roth, S. "Self-supervised monocular scene flow estimation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 7396–7405.

[HR21] Hur, J. and Roth, S. "Self-Supervised Multi-Frame Monocular Scene Flow". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 2684–2694.

[ISO13] ISO, D. "8855: 2013-11: Straßenfahrzeuge-Fahrzeugdynamik und Fahrverhalten-Begriffe". In: *Berlin. DOI* 10 (2013), p. 1941260.

[Jan+20] Janai, J., Güney, F., Behl, A., Geiger, A., et al. "Computer vision for autonomous vehicles: Problems, datasets and state of the art". In: *Foundations and Trends® in Computer Graphics and Vision* 12.1–3 (2020), pp. 1–308.

[Ji+21] Ji, G.-P., Fu, K., Wu, Z., Fan, D.-P., Shen, J., and Shao, L. "Full-duplex strategy for video object segmentation". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 4922–4933.

[Jia+19] Jiang, B., Wang, M., Gan, W., Wu, W., and Yan, J. “Stm: Spatiotemporal and motion encoding for action recognition”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 2000–2009.

[Jia+21a] Jiang, L., Shi, S., Tian, Z., Lai, X., Liu, S., Fu, C.-W., and Jia, J. “Guided Point Contrastive Learning for Semi-supervised Point Cloud Semantic Segmentation”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 6423–6432.

[Jia+21b] Jiao, L., Zhang, R., Liu, F., Yang, S., Hou, B., Li, L., and Tang, X. “New Generation Deep Learning for Video Object Detection: A Survey”. In: *IEEE Transactions on Neural Networks and Learning Systems* (2021).

[JTS21] Jiao, Y., Tran, T. D., and Shi, G. “EffiScene: Efficient Per-Pixel Rigidity Inference for Unsupervised Joint Learning of Optical Flow, Depth, Camera Pose and Motion Segmentation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2021, pp. 5538–5547.

[Jun+21] Jund, P., Sweeney, C., Abdo, N., Chen, Z., and Shlens, J. “Scalable Scene Flow from Point Clouds in the Real World”. In: *arXiv preprint arXiv:2103.01306* (2021).

[Kan+17a] Kang, K., Li, H., Xiao, T., Ouyang, W., Yan, J., Liu, X., and Wang, X. “Object detection in videos with tubelet proposal networks”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 727–735.

[Kan+17b] Kang, K., Li, H., Yan, J., Zeng, X., Yang, B., Xiao, T., Zhang, C., Wang, Z., Wang, R., Wang, X., et al. “T-cnn: Tubelets with convolutional neural networks for object detection from videos”. In: *IEEE Transactions on Circuits and Systems for Video Technology* 28.10 (2017), pp. 2896–2907.

[Ke+21] Ke, L., Li, X., Danelljan, M., Tai, Y.-W., Tang, C.-K., and Yu, F. “Prototypical Cross-Attention Networks for Multiple Object Tracking and Segmentation”. In: *arXiv preprint arXiv:2106.11958* (2021).

[KOL21] Kim, A., Ošep, A., and Leal-Taixé, L. “EagerMOT: 3D Multi-Object Tracking via Sensor Fusion”. In: *arXiv preprint arXiv:2104.14682* (2021).

[Kim+21] Kim, J., Koh, J., Lee, B., Yang, S., and Choi, J. W. “Video Object Detection Using Object’s Motion Context and Spatio-Temporal Feature Aggregation”. In: *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE. 2021, pp. 1604–1610.

[KER21] Kittenplon, Y., Eldar, Y. C., and Raviv, D. “FlowStep3D: Model Unrolling for Self-Supervised Scene Flow Estimation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 4114–4123.

[KWR19] Köpüklü, O., Wei, X., and Rigoll, G. “You only watch once: A unified cnn architecture for real-time spatiotemporal action localization”. In: *arXiv preprint arXiv:1911.06644* (2019).

[Koy+] Koyuncu, A. B., Erçelik, E., Comulada-Simpson, E., Venrooij, J., Kaboli, M., and Knoll, A. “A Novel Approach to Neural Network-based Motion Cueing Algorithm for a Driving Simulator”. In: *2020 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, pp. 2118–2125.

[Kra+21] Kraus, M., Azimi, S. M., Ercelik, E., Bahmanyar, R., Reinartz, P., and Knoll, A. “AerialMPTNet: Multi-Pedestrian Tracking in Aerial Imagery Using Temporal and Graphical Features”. In: *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE. 2021, pp. 2454–2461.

[Ku+18] Ku, J., Mozifian, M., Lee, J., Harakeh, A., and Waslander, S. L. “Joint 3d proposal generation and object detection from view aggregation”. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 1–8.

[Kua+20] Kuang, H., Wang, B., An, J., Zhang, M., and Zhang, Z. “Voxel-FPN: Multi-scale voxel feature aggregation for 3D object detection from LIDAR point clouds”. In: *Sensors* 20.3 (2020), p. 704.

[KA20] Kumar, K. C. and Al-Stouhi, S. “Real-time Spatial-temporal Context Approach for 3D Object Detection using LiDAR.” In: *VEHITS*. 2020, pp. 432–439.

[Lan+19] Lang, A. H., Vora, S., Caesar, H., Zhou, L., Yang, J., and Beijbom, O. “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.

[LD18] Law, H. and Deng, J. “Cornernet: Detecting objects as paired keypoints”. In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 734–750.

[Lee+20] Lee, K.-H., Kliemann, M., Gaidon, A., Li, J., Fang, C., Pillai, S., and Burgard, W. “PillarFlow: End-to-end birds-eye-view flow estimation for autonomous driving”. In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2020, pp. 2007–2013.

[Leh+19] Lehner, J., Mitterecker, A., Adler, T., Hofmarcher, M., Nessler, B., and Hochreiter, S. “Patch Refinement–Localized 3D Object Detection”. In: *arXiv preprint arXiv:1910.04093* (2019).

[Li+21a] Li, B., Zheng, C., Giancola, S., and Ghanem, B. “SCTN: Sparse Convolution-Transformer Network for Scene Flow Estimation”. In: *arXiv preprint arXiv:2105.04447* (2021).

[Li+21b] Li, J., Dai, H., Shao, L., and Ding, Y. “Anchor-free 3D Single Stage Detector with Mask-Guided Attention for Point Cloud”. In: *Proceedings of the 29th ACM International Conference on Multimedia*. 2021, pp. 553–562.

[Li+20] Li, J., Luo, S., Zhu, Z., Dai, H., Krylov, A. S., Ding, Y., and Shao, L. “3D IOU-net: IOU guided 3D object detector for point clouds”. In: *arXiv preprint arXiv:2004.04962* (2020).

[Li+21c] Li, R., Lin, G., He, T., Liu, F., and Shen, C. “HCRF-Flow: Scene Flow from Point Clouds with Continuous High-order CRFs and Position-aware Flow Embedding”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 364–373.

[LLX21] Li, R., Lin, G., and Xie, L. “Self-Point-Flow: Self-Supervised Scene Flow Estimation from Point Clouds with Optimal Transport and Random Walk”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 15577–15586.

[Li+21d] Li, S., Zhou, Y., Yi, J., and Gall, J. “Spatial-Temporal Consistency Network for Low-Latency Trajectory Forecasting”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 1940–1949.

[LWW21] Li, Z., Wang, F., and Wang, N. “LiDAR R-CNN: An Efficient and Universal 3D Object Detector”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 7546–7555.

[Lia+21a] Liang, H., Jiang, C., Feng, D., Chen, X., Xu, H., Liang, X., Zhang, W., Li, Z., and Van Gool, L. “Exploring Geometry-Aware Contrast and Clustering Harmonization for Self-Supervised 3D Object Detection”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 3293–3302.

[Lia+19] Liang, M., Yang, B., Chen, Y., Hu, R., and Urtasun, R. “Multi-task multi-sensor fusion for 3d object detection”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 7345–7353.

[Lia+21b] Liang, S., Shen, X., Huang, J., and Hua, X.-S. “Video Object Segmentation With Dynamic Memory Networks and Adaptive Object Alignment”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 8065–8074.

[Lia+20] Liang, Z., Zhang, M., Zhang, Z., Zhao, X., and Pu, S. “Rangercnn: Towards fast and accurate 3d object detection with range image representation”. In: *arXiv preprint arXiv:2009.00206* (2020).

[Lie+21] Lienen, J., Hullermeier, E., Ewerth, R., and Nommensen, N. “Monocular Depth Estimation via Listwise Ranking Using the Plackett-Luce Model”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2021, pp. 14595–14604.

[Lin+17a] Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., and Belongie, S. “Feature pyramid networks for object detection”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 2117–2125.

[Lin+17b] Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P. “Focal loss for dense object detection”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2980–2988.

[Liu+21a] Liu, H., Lu, T., Xu, Y., Liu, J., Li, W., and Chen, L. “CamLiFlow: Bidirectional Camera-LiDAR Fusion for Joint Optical Flow and Scene Flow Estimation”. In: *arXiv preprint arXiv:2111.10502* (2021).

[LZ18] Liu, M. and Zhu, M. “Mobile video object detection with temporally-aware feature maps”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 5686–5695.

[Liu+16] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. “Ssd: Single shot multibox detector”. In: *European conference on computer vision*. Springer. 2016, pp. 21–37.

[LQG19] Liu, X., Qi, C. R., and Guibas, L. J. “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.

[LYB19] Liu, X., Yan, M., and Bohg, J. “Meteornet: Deep learning on dynamic 3d point cloud sequences”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 9246–9255.

[Liu+20] Liu, Z., Zhao, X., Huang, T., Hu, R., Zhou, Y., and Bai, X. “Tanet: Robust 3d object detection from point clouds with triple attention”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 07. 2020, pp. 11677–11684.

[Liu+21b] Liu, Z., Zhou, D., Lu, F., Fang, J., and Zhang, L. “Autoshape: Real-time shape-aware monocular 3d object detection”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 15641–15650.

[Lu+21a] Lu, F., Chen, G., Liu, Y., Zhang, L., Qu, S., Liu, S., and Gu, R. “HRegNet: A Hierarchical Network for Large-scale Outdoor LiDAR Point Cloud Registration”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 16014–16023.

[Lu+21b] Lu, Y., Ma, X., Yang, L., Zhang, T., Liu, Y., Chu, Q., Yan, J., and Ouyang, W. “Geometry Uncertainty Projection Network for Monocular 3D Object Detection”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. 2021, pp. 3111–3121.

[LZL21] Lu, Y., Zhu, Y., and Lu, G. “3D SceneFlowNet: Self-Supervised 3D Scene Flow Estimation Based on Graph CNN”. In: *2021 IEEE International Conference on Image Processing (ICIP)*. IEEE. 2021, pp. 3647–3651.

[LLT17] Lu, Y., Lu, C., and Tang, C.-K. “Online video object detection using association LSTM”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 2344–2352.

[LYY21] Luo, C., Yang, X., and Yuille, A. “Exploring Simple 3D Multi-Object Tracking for Autonomous Driving”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 10488–10497.

[Luo+21] Luo, S., Dai, H., Shao, L., and Ding, Y. “M3DSSD: Monocular 3D Single Stage Object Detector”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2021, pp. 6145–6154.

[LYU18] Luo, W., Yang, B., and Urtasun, R. “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.

[Mao+21a] Mao, J., Niu, M., Bai, H., Liang, X., Xu, H., and Xu, C. “Pyramid r-cnn: Towards better performance and adaptability for 3d object detection”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 2723–2732.

[Mao+21b] Mao, J., Xue, Y., Niu, M., Bai, H., Feng, J., Liang, X., Xu, H., and Xu, C. “Voxel transformer for 3d object detection”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 3164–3173.

[Mao+21c] Mao, Y., Wang, N., Zhou, W., and Li, H. “Joint Inductive and Transductive Learning for Video Object Segmentation”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 9670–9679.

[May+16] Mayer, N., Ilg, E., Hausser, P., Fischer, P., Cremers, D., Dosovitskiy, A., and Brox, T. “A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 4040–4048.

[MZ20] McCrae, S. and Zakhor, A. “3d Object Detection For Autonomous Driving Using Temporal Lidar Data”. In: *2020 IEEE International Conference on Image Processing (ICIP)*. IEEE. 2020, pp. 2661–2665.

[MHG15] Menze, M., Heipke, C., and Geiger, A. “Joint 3D Estimation of Vehicles and Scene Flow”. In: *ISPRS Workshop on Image Sequence Analysis (ISA)*. 2015.

[MGJ21] Misra, I., Girdhar, R., and Joulin, A. “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.

[MOH20] Mittal, H., Okorn, B., and Held, D. “Just go with the flow: Self-supervised scene flow estimation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 11177–11185.

[MY21] Murhij, Y. and Yudin, D. “Real-time 3D Object Detection using Feature Map Flow”. In: *arXiv preprint arXiv:2106.14101* (2021).

[Mus+21] Mustikovela, S. K., De Mello, S., Prakash, A., Iqbal, U., Liu, S., Nguyen-Phuoc, T., Rother, C., and Kautz, J. “Self-Supervised Object Detection via Generative Image Synthesis”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 8609–8618.

[Ngu+21] Nguyen, T., Pham, Q.-H., Le, T., Pham, T., Ho, N., and Hua, B.-S. “Point-set Distances for Learning Representations of 3D Point Clouds”. In: *arXiv preprint arXiv:2102.04014* (2021).

[Nin+17] Ning, G., Zhang, Z., Huang, C., Ren, X., Wang, H., Cai, C., and He, Z. “Spatially supervised recurrent convolutional neural networks for visual object tracking”. In: *2017 IEEE international symposium on circuits and systems (ISCAS)*. IEEE. 2017, pp. 1–4.

[NLH21] Noh, J., Lee, S., and Ham, B. “HVPR: Hybrid Voxel-Point Representation for Single-stage 3D Object Detection”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 14605–14614.

[OR21] Ouyang, B. and Raviv, D. “Occlusion Guided Self-supervised Scene Flow Estimation on 3D Point Clouds”. In: *arXiv preprint arXiv:2104.04724* (2021).

[Pan+21] Pan, X., Xia, Z., Song, S., Li, L. E., and Huang, G. “3d object detection with pointformer”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 7463–7472.

[Pan+20] Pang, B., Li, Y., Zhang, Y., Li, M., and Lu, C. “Tubetk: Adopting tubes to track multi-object in a one-step training model”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 6308–6318.

[PMR20] Pang, S., Morris, D., and Radha, H. “CLOCs: Camera-LiDAR object candidates fusion for 3D object detection”. In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2020, pp. 10386–10393.

[PMR22] Pang, S., Morris, D., and Radha, H. “Fast-CLOCs: Fast Camera-LiDAR Object Candidates Fusion for 3D Object Detection”. In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2022, pp. 187–196.

[Par+21] Park, D., Ambrus, R., Guizilini, V., Li, J., and Gaidon, A. “Is Pseudo-Lidar Needed for Monocular 3D Object Detection?” In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. 2021, pp. 3142–3152.

[Par+18] Parmar, N., Vaswani, A., Uszkoreit, J., Kaiser, L., Shazeer, N., Ku, A., and Tran, D. “Image transformer”. In: *International Conference on Machine Learning*. PMLR. 2018, pp. 4055–4064.

[Pie+21] Piergiovanni, A., Casser, V., Ryoo, M. S., and Angelova, A. “4D-Net for Learned Multi-Modal Alignment”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 15435–15445.

[Pin+21] Pintore, G., Agus, M., Almansa, E., Schneider, J., and Gobbetti, E. “SliceNet: Deep Dense Depth Estimation From a Single Indoor Panorama Using a Slice-Based Representation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2021, pp. 11536–11545.

[PHL20] Pontes, J. K., Hays, J., and Lucey, S. “Scene Flow from Point Clouds with or without Learning”. In: *2020 International Conference on 3D Vision (3DV)*. IEEE. 2020, pp. 261–270.

[PG20] Purushwalkam Shiva Prakash, S. and Gupta, A. “Demystifying Contrastive Self-Supervised Learning: Invariances, Augmentations and Dataset Biases”. In: *Advances in Neural Information Processing Systems* 33 (2020).

[PBM20] Puy, G., Boulch, A., and Marlet, R. “Flot: Scene flow on point clouds guided by optimal transport”. In: *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXVIII 16*. Springer. 2020, pp. 527–544.

[Qi+18] Qi, C. R., Liu, W., Wu, C., Su, H., and Guibas, L. J. “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.

[Qi+17a] Qi, C. R., Su, H., Mo, K., and Guibas, L. J. “Pointnet: Deep learning on point sets for 3d classification and segmentation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 652–660.

[Qi+17b] Qi, C. R., Yi, L., Su, H., and Guibas, L. J. “PointNet++ deep hierarchical feature learning on point sets in a metric space”. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 2017, pp. 5105–5114.

[Qi+21] Qi, C. R., Zhou, Y., Najibi, M., Sun, P., Vo, K., Deng, B., and Anguelov, D. “Offboard 3D Object Detection from Point Cloud Sequences”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 6134–6144.

[Qia+21] Qian, K., Zhu, S., Zhang, X., and Li, L. E. “Robust Multimodal Vehicle Detection in Foggy Weather Using Complementary Lidar and Radar Signals”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 444–453.

[Ran+21] Ran, H., Zhuo, W., Liu, J., and Lu, L. “Learning Inner-Group Relations on Point Clouds”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 15477–15487.

[Rao+21] Rao, Y., Liu, B., Wei, Y., Lu, J., Hsieh, C.-J., and Zhou, J. “RandomRooms: Unsupervised Pre-training from Synthetic Shapes and Randomized Layouts for 3D Object Detection”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 3283–3292.

[Red+16] Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. “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.

[Ren+15] Ren, S., He, K., Girshick, R., and Sun, J. “Faster r-cnn: Towards real-time object detection with region proposal networks”. In: *Advances in neural information processing systems* 28 (2015), pp. 91–99.

[Ren+16] Ren, S., He, K., Girshick, R., and Sun, J. “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.

[Ris+20] Rishav, R., Battrawy, R., Schuster, R., Wasenmüller, O., and Stricker, D. "DeepLiDARFlow: A deep learning architecture for scene flow estimation using monocular camera and sparse LiDAR". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2020, pp. 10460–10467.

[RVK22] Rukhovich, D., Vorontsova, A., and Konushin, A. "Imvoxelnet: Image to voxels projection for monocular and multi-view general-purpose 3d object detection". In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2022, pp. 2397–2406.

[SSU20] Schuster, R., Stricker, D., and Unger, C. "MonoComb: A Sparse-to-Dense Combination Approach for Monocular Scene Flow". In: *Computer Science in Cars Symposium*. 2020, pp. 1–8.

[Sha+18a] Shao, L., Shah, P., Dwaracherla, V., and Bohg, J. "Motion-based object segmentation based on dense rgb-d scene flow". In: *IEEE Robotics and Automation Letters* 3.4 (2018), pp. 3797–3804.

[Sha+18b] Sharma, S., Ansari, J. A., Murthy, J. K., and Krishna, K. M. "Beyond pixels: Leveraging geometry and shape cues for online multi-object tracking". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 3508–3515.

[Shi+21a] Shi, B., Dai, Q., Hoffman, J., Saenko, K., Darrell, T., and Xu, H. "Temporal Action Detection with Multi-level Supervision". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 8022–8032.

[SLS20] Shi, J., Li, P., and Shen, S. "Tracking from Patterns: Learning Corresponding Patterns in Point Clouds for 3D Object Tracking". In: *arXiv preprint arXiv:2010.10051* (2020).

[Shi+20a] Shi, S., Guo, C., Jiang, L., Wang, Z., Shi, J., Wang, X., and Li, H. "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.

[Shi+21b] Shi, S., Jiang, L., Deng, J., Wang, Z., Guo, C., Shi, J., Wang, X., and Li, H. "PV-RCNN++: Point-Voxel Feature Set Abstraction With Local Vector Representation for 3D Object Detection". In: *arXiv preprint arXiv:2102.00463* (2021).

[SWL19] Shi, S., Wang, X., and Li, H. "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.

[Shi+20b] Shi, S., Wang, Z., Shi, J., Wang, X., and Li, H. "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* (2020).

[SR20] Shi, W. and Rajkumar, R. "Point-gnn: Graph neural network for 3d object detection in a point cloud". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 1711–1719.

[Shi+21c] Shi, X., Ye, Q., Chen, X., Chen, C., Chen, Z., and Kim, T.-K. "Geometry-Based Distance Decomposition for Monocular 3D Object Detection". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. 2021, pp. 15172–15181.

[SLB19] Shvets, M., Liu, W., and Berg, A. C. "Leveraging long-range temporal relationships between proposals for video object detection". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 9756–9764.

[Sim+19a] Simon, M., Amende, K., Kraus, A., Honer, J., Samann, T., Kaulbersch, H., Milz, S., and Michael Gross, H. "Complexer-yolo: Real-time 3d object detection and tracking on semantic point clouds". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 2019, pp. 0–0.

[Sim+19b] Simonelli, A., Bulo, S. R., Porzi, L., López-Antequera, M., and Kontschieder, P. "Disentangling monocular 3d object detection". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 1991–1999.

[SZ15] Simonyan, K. and Zisserman, A. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Bengio, Y. and LeCun, Y. 2015. URL: http://arxiv.org/abs/1409.1556.

[SZT19] Sindagi, V. A., Zhou, Y., and Tuzel, O. "Mvx-net: Multimodal voxelnet for 3d object detection". In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 7276–7282.

[SX16] Song, S. and Xiao, J. "Deep sliding shapes for amodal 3d object detection in rgb-d images". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 808–816.

[SN16] Subarna Tripathi Zachary Lipton, S. B. and Nguyen, T. "Context Matters: Refining Object Detection in Video with Recurrent Neural Networks". In: *Proceedings of the British Machine Vision Conference (BMVC)*. Ed. by Richard C. Wilson, E. R. H. and Smith, W. A. P. BMVA Press, York, UK, Sept. 2016, pp. 44.1–44.12. ISBN: 1-901725-59-6. DOI: 10.5244/C.30.44. URL: https://dx.doi.org/10.5244/C.30.44.

[Sun+21] Sun, J., Xie, Y., Zhang, S., Chen, L., Zhang, G., Bao, H., and Zhou, X. "You Don't Only Look Once: Constructing Spatial-Temporal Memory for Integrated 3D Object Detection and Tracking". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 3185–3194.

[Sun+20] Sun, P., Kretzschmar, H., Dotiwalla, X., Chouard, A., Patnaik, V., Tsui, P., Guo, J., Zhou, Y., Chai, Y., Caine, B., 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.

[Tan+21] Tan, S., Wu, Y., Yu, S.-I., and Veeraraghavan, A. "CodedStereo: Learned Phase Masks for Large Depth-of-Field Stereo". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2021, pp. 7170–7179.

[Tea20] Team, O. D. *OpenPCDet: An Open-source Toolbox for 3D Object Detection from Point Clouds*. https://github.com/open-mmlab/OpenPCDet. 2020.

[TD20] Teed, Z. and Deng, J. "Raft: Recurrent all-pairs field transforms for optical flow". In: *European conference on computer vision*. Springer. 2020, pp. 402–419.

[TD21] Teed, Z. and Deng, J. "Raft-3d: Scene flow using rigid-motion embeddings". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 8375–8384.

[Ten+21] Teng, Y., Wang, L., Li, Z., and Wu, G. "Target Adaptive Context Aggregation for Video Scene Graph Generation". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 13688–13697.

[Tsc+20] Tschannen, M., Djolonga, J., Ritter, M., Mahendran, A., Houlsby, N., Gelly, S., and Lucic, M. "Self-supervised learning of video-induced visual invariances". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 13806–13815.

[Vas+17] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. "Attention is all you need". In: *Advances in neural information processing systems*. 2017, pp. 5998–6008.

[Ved+05] Vedula, S., Rander, P., Collins, R., and Kanade, T. "Three-dimensional scene flow". In: *IEEE transactions on pattern analysis and machine intelligence* 27.3 (2005), pp. 475–480.

[VSR15] Vogel, C., Schindler, K., and Roth, S. "3d scene flow estimation with a piecewise rigid scene model". In: *International Journal of Computer Vision* 115.1 (2015), pp. 1–28.

[Vor+20] Vora, S., Lang, A. H., Helou, B., and Beijbom, O. "Pointpainting: Sequential fusion for 3d object detection". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 4604–4612.

[Wan+21a] Wang, C., Ma, C., Zhu, M., and Yang, X. "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.

[Wan+21b] Wang, G., Hu, Y., Wu, X., and Wang, H. "Residual 3D Scene Flow Learning with Context-Aware Feature Extraction". In: *arXiv preprint arXiv:2109.04685* (2021).

[Wan+21c] Wang, G., Wu, X., Liu, Z., and Wang, H. "Hierarchical attention learning of scene flow in 3d point clouds". In: *IEEE Transactions on Image Processing* 30 (2021), pp. 5168–5181.

[Wan+20a] Wang, G., Tian, B., Ai, Y., Xu, T., Chen, L., and Cao, D. "CenterNet3D: An Anchor free Object Detector for Autonomous Driving". In: *arXiv preprint arXiv:2007.07214* (2020).

[Wan+20b] Wang, G., Tian, B., Zhang, Y., Chen, L., Cao, D., and Wu, J. "Multi-View Adaptive Fusion Network for 3D Object Detection". In: *arXiv preprint arXiv:2011.00652* (2020).

[Wan+21d] Wang, G., Wu, J., Tian, B., Teng, S., Chen, L., and Cao, D. "CenterNet3D: An anchor free object detector for point cloud". In: *IEEE Transactions on Intelligent Transportation Systems* (2021).

[Wan+21e] Wang, H., Pang, J., Lodhi, M. A., Tian, Y., and Tian, D. "FESTA: Flow Estimation via Spatial-Temporal Attention for Scene Point Clouds". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 14173–14182.

[Wan+21f] Wang, H., Liu, Q., Yue, X., Lasenby, J., and Kusner, M. J. "Unsupervised point cloud pre-training via occlusion completion". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 9782–9792.

[Wan+20c] Wang, J., Lan, S., Gao, M., and Davis, L. S. "Infofocus: 3d object detection for autonomous driving with dynamic information modeling". In: *European Conference on Computer Vision*. Springer. 2020, pp. 405–420.

[Wan+16] Wang, L., Xiong, Y., Wang, Z., Qiao, Y., Lin, D., Tang, X., and Van Gool, L. "Temporal segment networks: Towards good practices for deep action recognition". In: *European conference on computer vision*. Springer. 2016, pp. 20–36.

[Wan+21g] Wang, T., Xu, N., Chen, K., and Lin, W. “End-to-End Video Instance Segmentation via Spatial-Temporal Graph Neural Networks”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 10797–10806.

[WLN21] Wang, T., Liu, M., and Ng, K. S. “Spatially Invariant Unsupervised 3D Object Segmentation with Graph Neural Networks”. In: *arXiv preprint arXiv:2106.05607* (2021).

[Wan+18] Wang, X., Girshick, R., Gupta, A., and He, K. “Non-local neural networks”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 7794–7803.

[Wan+19] Wang, Y., Chao, W.-L., Garg, D., Hariharan, B., Campbell, M., and Weinberger, K. Q. “Pseudo-lidar from visual depth estimation: Bridging the gap in 3d object detection for autonomous driving”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 8445–8453.

[Wan+20d] Wang, Y., Chen, X., You, Y., Erran, L., Hariharan, B., Campbell, M., Weinberger, K. Q., and Chao, W.-L. “Train in germany, test in the usa: Making 3d object detectors generalize”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 11713–11723.

[Wan+21h] Wang, Y., Yang, B., Hu, R., Liang, M., and Urtasun, R. “PLUME: Efficient 3D Object Detection from Stereo Images”. In: *arXiv preprint arXiv:2101.06594* (2021).

[Wan+20e] Wang, Y., Fathi, A., Kundu, A., Ross, D. A., Pantofaru, C., Funkhouser, T., and Solomon, J. “Pillar-based object detection for autonomous driving”. In: *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXII 16*. Springer. 2020, pp. 18–34.

[Wan+21i] Wang, Z., Zhang, X., Wang, S., Xin, T., Zhang, H., and Lu, J. “Multi-Scale Spatial Transformer Network for LiDAR-Camera 3D Object Detection”. In: *2021 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2021, pp. 1–8.

[WJ19] Wang, Z. and Jia, K. “Frustum ConvNet: Sliding Frustums to Aggregate Local Point-Wise Features for Amodal”. In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2019, pp. 1742–1749.

[Wan+21j] Wang, Z., Xie, Q., Lai, Y.-K., Wu, J., Long, K., and Wang, J. “MLVSNet: Multi-Level Voting Siamese Network for 3D Visual Tracking”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 3101–3110.

[Wan+20f] Wang, Z., Li, S., Howard-Jenkins, H., Prisacariu, V., and Chen, M. “Flownet3d++: Geometric losses for deep scene flow estimation”. In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2020, pp. 91–98.

[Web+19] Weber, T., Ercelik, E., Ebert, M., and Knoll, A. “Recognition & Evaluation of Additional Traffic Signs on the example of’80 km/h when wet’”. In: *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE. 2019, pp. 4134–4139.

[Wei+21] Wei, Y., Liu, H., Xie, T., Ke, Q., and Guo, Y. “Spatial-Temporal Transformer for 3D Point Cloud Sequences”. In: *arXiv preprint arXiv:2110.09783* (2021).

[Wen+20a] Weng, X., Wang, J., Held, D., and Kitani, K. “AB3DMOT: A Baseline for 3D Multi-Object Tracking and New Evaluation Metrics”. In: *arXiv preprint arXiv:2008.08063* (2020).

[Wen+20b] Weng, X., Wang, Y., Man, Y., and Kitani, K. M. “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.

[WBP17] Wojke, N., Bewley, A., and Paulus, D. “Simple online and realtime tracking with a deep association metric”. In: *2017 IEEE international conference on image processing (ICIP)*. IEEE. 2017, pp. 3645–3649.

[Wu+21a] Wu, B., Ma, J., Chen, G., and An, P. “Feature Interactive Representation for Point Cloud Registration”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 5530–5539.

[Wu+19] Wu, H., Chen, Y., Wang, N., and Zhang, Z. “Sequence level semantics aggregation for video object detection”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 9217–9225.

[Wu+21b] Wu, S., Jin, S., Liu, W., Bai, L., Qian, C., Liu, D., and Ouyang, W. “Graph-based 3d multi-person pose estimation using multi-view images”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 11148–11157.

[Wu+20] Wu, W., Wang, Z. Y., Li, Z., Liu, W., and Fuxin, L. “PointPWC-Net: Cost Volume on Point Clouds for (Self-) Supervised Scene Flow Estimation”. In: *European Conference on Computer Vision*. Springer. 2020, pp. 88–107.

[Xie+18] Xie, S., Sun, C., Huang, J., Tu, Z., and Murphy, K. “Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification”. In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 305–321.

[Xio+21a] Xiong, Y., Ren, M., Zeng, W., and Urtasun, R. “Self-Supervised Representation Learning from Flow Equivariance”. In: *arXiv preprint arXiv:2101.06553* (2021).

[Xio+21b] Xiong, Z., Ma, H., Wang, Y., Hu, T., and Liao, Q. “LiDAR-based 3D Video Object Detection with Foreground Context Modeling and Spatiotemporal Graph Reasoning”. In: *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*. IEEE. 2021, pp. 2994–3001.

[Yan+21a] Yan, B., Peng, H., Fu, J., Wang, D., and Lu, H. “Learning spatio-temporal transformer for visual tracking”. In: *arXiv preprint arXiv:2103.17154* (2021).

[YML18] Yan, Y., Mao, Y., and Li, B. “Second: Sparsely embedded convolutional detection”. In: *Sensors* 18.10 (2018), p. 3337.

[Yan+20a] Yang, C., Xu, Y., Shi, J., Dai, B., and Zhou, B. “Temporal pyramid network for action recognition”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 591–600.

[Yan+21b] Yang, C., Lamdouar, H., Lu, E., Zisserman, A., and Xie, W. “Self-supervised Video Object Segmentation by Motion Grouping”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021.

[YAL21] Yang, J., Alvarez, J. M., and Liu, M. “Self-Supervised Learning of Depth Inference for Multi-View Stereo”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2021, pp. 7526–7534.

[Yan+20b] Yang, Z., Sun, Y., Liu, S., and Jia, J. “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.

[Yan+18] Yang, Z., Sun, Y., Liu, S., Shen, X., and Jia, J. "Ipod: Intensive point-based object detector for point cloud". In: *arXiv preprint arXiv:1812.05276* (2018).

[Yan+19] Yang, Z., Sun, Y., Liu, S., Shen, X., and Jia, J. "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.

[Yan+21c] Yang, Z., Zhou, Y., Chen, Z., and Ngiam, J. "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.

[Yin+21] Yin, J., Shen, J., Gao, X., Crandall, D., and Yang, R. "Graph Neural Network and Spatiotemporal Transformer Attention for 3D Video Object Detection from Point Clouds". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021).

[Yin+20] Yin, J., Shen, J., Guan, C., Zhou, D., and Yang, R. "Lidar-based online 3d video object detection with graph-based message passing and spatiotemporal transformer attention". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 11495–11504.

[YZK21] Yin, T., Zhou, X., and Krahenbuhl, P. "Center-based 3d object detection and tracking". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 11784–11793.

[Yin+17] Yin, W., Kann, K., Yu, M., and Schütze, H. "Comparative study of CNN and RNN for natural language processing". In: *arXiv preprint arXiv:1702.01923* (2017).

[Yoo+20] Yoo, J. H., Kim, Y., Kim, J., and Choi, J. W. "3d-cvf: Generating joint camera and lidar features using cross-view spatial feature fusion for 3d object detection". In: *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXVII 16*. Springer. 2020, pp. 720–736.

[Yu+21] Yu, X., Rao, Y., Wang, Z., Liu, Z., Lu, J., and Zhou, J. "Pointr: Diverse point cloud completion with geometry-aware transformers". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 12498–12507.

[Yua+21] Yuan, Z., Song, X., Bai, L., Wang, Z., and Ouyang, W. "Temporal-Channel Transformer for 3D Lidar-Based Video Object Detection for Autonomous Driving". In: *IEEE Transactions on Circuits and Systems for Video Technology* (2021).

[Zam+21] Zamanakos, G., Tsochatzidis, L., Amanatiadis, A., and Pratikakis, I. "A comprehensive survey of LIDAR-based 3D object detection methods with deep learning for autonomous driving". In: *Computers & Graphics* 99 (2021), pp. 153–181.

[Zen+21a] Zeng, A., Sun, X., Yang, L., Zhao, N., Liu, M., and Xu, Q. "Learning skeletal graph neural networks for hard 3d pose estimation". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 11436–11445.

[Zen+21b] Zeng, F., Dong, B., Wang, T., Chen, C., Zhang, X., and Wei, Y. "MOTR: End-to-End Multiple-Object Tracking with TRansformer". In: *arXiv preprint arXiv:2105.03247* (2021).

[Zha+20a] Zhai, G., Kong, X., Cui, J., Liu, Y., and Yang, Z. "FlowMOT: 3D Multi-Object Tracking by Scene Flow Association". In: *arXiv preprint arXiv:2012.07541* (2020).

[Zha+19] Zhang, W., Zhou, H., Sun, S., Wang, Z., Shi, J., and Loy, C. C. "Robust multi-modality multi-object tracking". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 2365–2374.

[Zha+21a] Zhang, Y., Qu, Y., Xie, Y., Li, Z., Zheng, S., and Li, C. “Perturbed Self-Distillation: Weakly Supervised Large-Scale Point Cloud Semantic Segmentation”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 15520–15528.

[Zha+20b] Zhang, Y., Ye, Y., Xiang, Z., and Gu, J. “SDP-Net: Scene Flow Based Real-time Object Detection and Prediction from Sequential 3D Point Clouds”. In: *Proceedings of the Asian Conference on Computer Vision*. 2020.

[ZLZ21] Zhang, Y., Lu, J., and Zhou, J. “Objects Are Different: Flexible Monocular 3D Object Detection”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2021, pp. 3289–3298.

[Zha+21b] Zhang, Z., Girdhar, R., Joulin, A., and Misra, I. “Self-supervised pretraining of 3d features on any point-cloud”. In: *arXiv preprint arXiv:2101.02691* (2021).

[Zha+21c] Zhao, B., Bhat, G., Danelljan, M., Van Gool, L., and Timofte, R. “Generating masks from boxes by mining spatio-temporal consistencies in videos”. In: *arXiv preprint arXiv:2101.02196* (2021).

[Zha+21d] Zhao, H., Jiang, L., Jia, J., Torr, P. H., and Koltun, V. “Point transformer”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 16259–16268.

[Zha+21e] Zhao, L., Cai, D., Sheng, L., and Xu, D. “3DVG-Transformer: Relation modeling for visual grounding on point clouds”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 2928–2937.

[ZJH21] Zhao, Z., Jin, Y., and Heng, P.-A. “Modelling Neighbor Relation in Joint Space-Time Graph for Video Correspondence Learning”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 9960–9969.

[Zhe+21a] Zheng, C., Yan, X., Gao, J., Zhao, W., Zhang, W., Li, Z., and Cui, S. “Box-aware feature enhancement for single object tracking on point clouds”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 13199–13208.

[Zhe+21b] Zheng, W., Tang, W., Chen, S., Jiang, L., and Fu, C.-W. “CIA-SSD: Confident IoU-Aware Single-Stage Object Detector From Point Cloud”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35. 4. 2021, pp. 3555–3562.

[Zhe+21c] Zheng, W., Tang, W., Jiang, L., and Fu, C.-W. “SE-SSD: Self-Ensembling Single-Stage Object Detector From Point Cloud”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 14494–14503.

[Zho+20] Zhou, D., Fang, J., Song, X., Liu, L., Yin, J., Dai, Y., Li, H., and Yang, R. “Joint 3d instance segmentation and object detection for autonomous driving”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 1839–1849.

[ZKK20] Zhou, X., Koltun, V., and Krähenbühl, P. “Tracking objects as points”. In: *European Conference on Computer Vision*. Springer, Cham. 2020, pp. 474–490.

[ZWK19] Zhou, X., Wang, D., and Krähenbühl, P. “Objects as Points”. In: *arXiv preprint arXiv:1904.07850*. 2019.

[ZT18] Zhou, Y. and Tuzel, O. “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.

[Zho+21a] Zhou, Y., He, Y., Zhu, H., Wang, C., Li, H., and Jiang, Q. “Monocular 3D Object Detection: An Extrinsic Parameter Free Approach”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2021, pp. 7556–7566.

[Zho+21b] Zhou, Y., Zhu, H., Li, C., Cui, T., Chang, S., and Guo, M. “TempNet: Online Semantic Segmentation on Large-Scale Point Cloud Series”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 7118–7127.

[Zho+21c] Zhou, Z., Pei, W., Li, X., Wang, H., Zheng, F., and He, Z. “Saliency-Associated Object Tracking”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 9866–9875.

[Zhu+19] Zhu, B., Jiang, Z., Zhou, X., Li, Z., and Yu, G. “Class-balanced Grouping and Sampling for Point Cloud 3D Object Detection”. In: *arXiv preprint arXiv:1908.09492* (2019).

[Zhu+21] Zhu, M., Ma, C., Ji, P., and Yang, X. “Cross-modality 3d object detection”. In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2021, pp. 3772–3781.

[Zhu+18] Zhu, X., Dai, J., Yuan, L., and Wei, Y. “Towards high performance video object detection”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 7210–7218.

[ZSB18] Zolfaghari, M., Singh, K., and Brox, T. “Eco: Efficient convolutional network for online video understanding”. In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 695–712.

[Zua+20] Zuanazzi, V., Vugt, J. van, Booij, O., and Mettes, P. “Adversarial self-supervised scene flow estimation”. In: *2020 International Conference on 3D Vision (3DV)*. IEEE. 2020, pp. 1049–1058.