SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY — INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Multi-Modal 3D Object Detection in Long Range and Low-Resolution Conditions of Sensors

Egemen Kopuz

SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY — INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Multi-Modal 3D Object Detection in Long Range and Low-Resolution Conditions of Sensors

Multimodale 3D-Objekterkennung bei großer Reichweite und geringer Auflösung von Sensoren

Author:	Egemen Kopuz
Supervisor:	Prof. Dr.-Ing. habil. Alois C. Knoll
Advisors:	M.Sc. Walter Zimmer, M.Sc. Xavier Diaz Ortiz
Submission Date:	15.11.2023

I confirm that this master's thesis is my own work and I have documented all sources and material used.

Munich, 15.11.2023 Egemen Kopuz

Acknowledgments

I extend my deepest gratitude to Prof. Knoll for giving me the opportunity to write my thesis under his supervision. I would also like to give my gratitude to my supervisors, Walter Zimmer and Xavier Diaz, for their invaluable guidance, patience, and expertise. Their insightful feedback and unwavering support have been pivotal throughout this journey.

I am equally indebted to my family, whose love and trust have not gone unnoticed. To my parents and my sister whose wisdom and encouragements have been my motivation, and provided me with much-needed relief during the most intense moments. Their unyielding support has been a cornerstone of my success.

Abstract

With the rise of autonomous vehicles and intelligent transportation systems, robust 3D object detection has become essential. These systems often struggle with data sparsity caused by challenges like distant and occluded objects, or low-resolution sensors, which can impair performance. This thesis investigates primarily the impact of temporal information on the prediction accuracy of such objects across two datasets from different domains – specifically, TUMTraf-i [Zim+23b] and OSDaR23 [Tag+23]. We propose the Temporal Fuser (TF) that assimilate prior frames to refine features in bird's eye view level, as well as, the Temporally-Aware Ground Truth Paste (TA-GTP) data augmentation method, which enhances training scenes with moving and rotating virtual objects that are temporally coherent.

These proposed methods have been integrated using our custom-made Temporal Pipeline, built upon BEVFusion [Liu+22], which facilitates swift inference through an Online Caching mechanism for validation and testing phases while ensuring the temporal consistency of all existing augmentation methods for training phase. To ensure our evaluations are contextually relevant to temporal dynamics all the time, we have implemented a novel Temporal Dataset Split Search algorithm. This algorithm finds an optimal division of datasets by considering the custom attributes of objects. It ensures that the divisions are balanced with respect to class variety as well as custom attributes such as distance from the ego position, the number of points within bounding boxes, and levels of occlusion.

Our methods, when combined, yield significant performance improvements across all datasets and various combinations of modalities. We demonstrate their efficacy on objects at different distances, presenting extensive quantitative results, and illustrate how our methods predict distant or occluded objects through detailed visualizations. Furthermore, we have demonstrated our methods' performance in scenarios characterized by a reduced number of point clouds, further simulating sparsity caused by low-resolution sensors. To ascertain the effectiveness of each method individually, we further conducted thorough ablation studies on components of Temporal Fuser (TF), Temporally-Aware Ground Truth Paste (TA-GTP), and Temporal Loading, thereby validating our proposed methods.

Contents

1 Introduction

The evolution of autonomous vehicles and intelligent transportation systems have transformed the landscape of transportation, promising increased safety, efficiency, and convenience. At the heart of this technological developments lies the critical challenge of accurately detecting objects. For this, there has been remarkable progress in recent years, including the utilization of numerous sensors and usage of cutting-edge machine learning techniques. Two fundamental aspects of object detection are 2D and 3D perception. 2D object detection, primarily reliant on cameras, offers cost-effective solutions for detecting objects within the same plane as the sensor but can struggle with depth perception. In contrast, 3D object detection, enabled by technologies like LiDAR and radar, excels at capturing the full spatial extent of objects but tends to be costlier. While cameras are adept at recognizing object attributes, LiDAR and radar sensors are indispensable for precise 3D localization. Most importantly, the integration of these sensors significantly impacts the budget of autonomous systems. Therefore, sensors themselves are specialized to correspond to specific requirements, such as accuracy and range specifications. The balance between these modalities is pivotal, as it directly influences the overall quality of data available for object detection and subsequent decision-making in autonomous vehicles and transportation systems.

One of the main issues of detecting objects is data sparsity, caused by challenging conditions, such as having objects that are far away or occluded by other elements within the environment, or low-resolution sensors. Therefore, selection of sensors becomes the fundamental part of the decision process. As an example, for far away objects, cameras by themselves are usually not enough as their depth estimations become compromised as the range increases, so sensors like LiDARs are generally used for such cases. However, not all the LiDAR devices are suited for it unless they are specifically designed for long-range scanning. Otherwise, their scans of far away objects may be quite sparse to a point that it does not hold any significant information for system to utilize. Therefore, many state-of-the-art approaches utilize multiple sensors like cameras and LiDARs to compensate each other's drawbacks and consolidate their strengths.

In the context of autonomous vehicles, specifically autonomous trains, and intelligent transportation systems, the detection of objects at a distance plays a pivotal role in ensuring safe and efficient operations. Trains often travel at high speeds, and their

immense weight and momentum make sudden stops or evasive maneuvers extremely challenging. Detecting objects at a considerable distance allows the train's control system to anticipate potential obstacles, such as vehicles, pedestrians, or debris on the tracks, and take early preventive actions. This is crucial in preventing accidents and ensuring the well-being of passengers and the surrounding community.

Occlusion is also another an ever-present challenge in real-world transportation environments. Objects, be they other vehicles, infrastructure elements, or pedestrians, are frequently obscured or partially hidden from view by other objects. From a sensor's perspective, the presence of these objects can vary temporally. In other words, an object may be fully or partially visible in one frame but not in the subsequent frames, resulting in either sparse or non-existent cloud points. Therefore, having a means to utilize past temporal information can significantly enhance the detection process. This temporal information proves particularly valuable for mobile vehicles, as their sensor placements are inherently limited, resulting in restricted coverage areas. Conversely, in a static infrastructure environment such as a traffic intersection, additional sensors can be strategically placed in various locations to expand the coverage area. Nevertheless, this expansion comes at the cost of an increased budget, which may not be ideal, especially if the ultimate goal is to implement such a setup in multiple locations across the country.

This thesis and its associated experiments related to the TUMTraf-i Dataset [Zim+23b] were conducted within the scope of the Providentia++ project [Krä+21] led by the Chair of Robotics, Artificial Intelligence, and Real-time Systems at the Technical University of Munich's Department of Informatics. The project is dedicated to researching the flow of information between vehicles and infrastructure along the A9 highway, which extends into urban areas. Its primary goal is to establish a comprehensive digital twin of the existing traffic situation. This innovative approach will serve as a foundation for the development of services that enhance traffic management and improve intelligent transportation systems. In addition, various other experiments, utilizing the OSDaR23 dataset [Tag+23], were carried out in collaboration with SETLabs Research GmbH as part of the consortium project known as safe.trAIn. The safe.trAIn project is an initiative focused on advancing safety and efficiency in driverless rail transport through the development and integration of robust AI technologies. By addressing the unique challenges within this safety-critical environment, this project contributes to the broader objectives of climate protection and sustainable transportation in Europe. The successful implementation of safe.trAIn is expected to pave the way for the widespread adoption of driverless rail vehicles, thereby making significant strides toward a cleaner and more efficient transportation sector.

In conjunction with all the aforementioned projects, the primary objective of this thesis is to demonstrate the effectiveness of implementing temporal-aware data augmentations and fusion networks within a state-of-the-art multi-modal fusion model. This study adopts the work of [Liu+22] as its baseline model and builds upon it.

The key contributions can be summarized as follows:

- **Temporal Fusion Networks:** Convolutional Gated Recurrent Unit Recurrent Networks [Bal+16] and Convolutional Long Short-Term Memory Recurrent Networks [Shi+15] are integrated into the pipeline to effectively fuse bird's eye view features of varying sequential frames.
- **Temporally-Aware Ground Truth Paste Data Augmentation:** A novel and temporally-aware adaptation of the GT-Paste Augmentation method introduced by Yan et al.'s work from [YML18] that pastes virtual objects in the forms of 3D bounding boxes and cloud points from other scenes to the training scenes. Particularly noteworthy in this extended version is the inclusion of randomized, yet temporally consistent, rotation and translation of such virtual objects across sequential frames.
- **Temporal Pipeline:** The training, evaluation and inference pipelines, along with the 2D and 3D data augmentation methods, have been modified to ensure compatibility with sequential data flow, whether it is online or offline. Additionally, Online Caching mechanism is implemented for faster inference pipeline.
- **Temporal Dataset Split Search:** A novel exhaustive search algorithm employing multi-processing techniques to identify temporally optimal splits (consisting training, validation, and test sets) that maintain uniformity in terms of both number of classes and object attributes such as distance from the ego position, point cloud count within bounding boxes, and occlusion levels.

These contributions collectively emphasize the significant impact of leveraging temporal aspects in improving 3D object detection performance, particularly for objects located at a distance or occluded, within the domains of railway train and traffic infrastructure.

2 Background

2.1 Input Data

Data is fundamental to computer vision for creating and representing digital objects or scenes. Popular data types, specifically for the 3D object detection task, include multi-view camera images and LiDAR point clouds. Multi-view images and LiDAR point clouds complement each other, offering diverse perspectives and precise depth information crucial for 3D object detection models.

2.1.1 Multi-View Images

Multi-view imaging involves capturing a scene from multiple angles to obtain a set of images that represent different perspectives. This technique is essential for creating 3D representations of the subjects [HZ01], which is particularly beneficial for tasks such as 3D object detection [PF20] [Hua+22] [RKC18] [Pan+20] [RC20] [Li+22b] [Li+23]. The images are typically synchronized and calibrated to ensure consistency in further processing stages. In this thesis, multi-view images are utilized in 3D object detection models, offering a broader coverage area and better occlusion handling than single-image counterparts [Nin+].

2.1.2 LiDAR Cloud Points

LiDAR (Light Detection and Ranging) devices produce point clouds by measuring the distances to objects with laser pulses, creating high-resolution 3D representations of the environment. These point clouds are known for their high density and accuracy, which makes them invaluable for precision mapping and detailed examination of physical spaces [LI20] [Zam+21]. In this thesis, LiDAR point clouds are utilized in 3D object detection models to provide a nuanced and precise understanding of the environment, offering detailed localization and depth information.

2.2 Bird's Eye View Data Representation

The concept of Bird's Eye View (BEV) data representation is a comprehensive approach to visualizing and interpreting data from an elevated perspective, offering benefits across various fields and applications.

BEV representation has gained popularity for improving 3D detector performance in autonomous driving systems [Zhu+23], as demonstrated by research adopting BEV as a unified representation across single or multiple modalities [Hu+23] [Liu+22] [Lia+22]. In this thesis, we utilized a bird's eye view as a unified representation in order to overcome the view discrepancy between different modalities.

2.3 Deep Learning

Deep Learning (DL) is an advanced computer-based modeling technique within machine learning (ML) and artificial intelligence (AI) fields, known for its proficiency in processing large and unstructured datasets. Tracing its roots to the early neural network models inspired by associationism – a theory of how the brain processes information – deep learning has evolved into sophisticated architectures, such as Convolutional Neural Networks (CNNs) and Transformers, which are at the forefront of current research [WR17] [Alz+21] [MRP21].

Characterized by their layered processing, deep learning models can interpret data at multiple levels of abstraction. Their capacity to learn from specific training data and to streamline analytical model development has led to their widespread adoption in intelligent systems across various fields, including speech recognition, healthcare, natural language processing and autonomous vehicle technology [Shi+23] [JZH21].

2.3.1 3D Object Detection

The detection of 3D objects using deep learning models is a process that involves the identification and localization of objects in 3D space. This is typically achieved through the generation of 3D bounding boxes that encapsulate the location, dimensions and orientation of the objects within a given environment. Deep learning models, trained on large datasets with annotated 3D bounding boxes, learn spatial hierarchies and feature representations essential for generating these boxes and achieving object detection. During the inference phase, these models process input data, which can come from various sources such as LiDAR and multi-view cameras, to predict the 3D bounding boxes for each detected object.

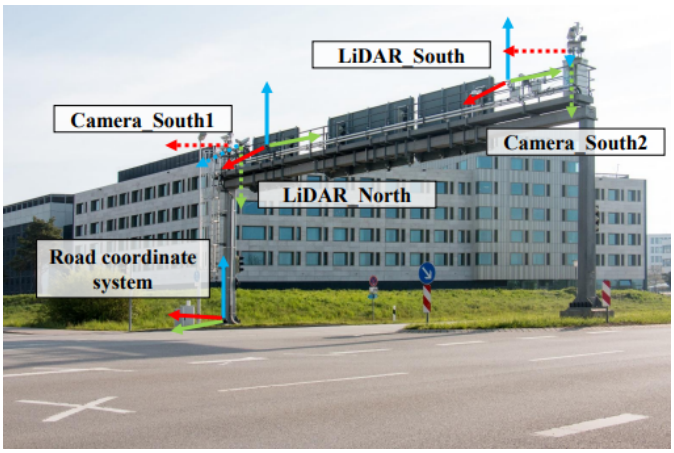


Figure 2.1: Image showing the positions of the sensors used in the TUMTraf Intersection Dataset (TUMTraf-i). Taken from [Zim+23b].

2.4 Datasets

In this thesis, we used two datasets, TUMTraf-i [Zim+23b] and OSDar23 [Tag+23], each from a different domain with unique attributes specific to their use-cases.

2.4.1 TUMTraf Intersection Dataset

The TUMTraf Intersection Dataset (TUMTraf-i), a component of the Providentia++ project [Krä+21], comprises LiDAR point clouds and multi-view camera images from a road intersection near Munich, Germany. The sensors – 2 cameras and 2 LiDAR devices – are mounted adjacently on a gantry 7 meters high to monitor traffic flow at the intersection's center (see Figure 2.3). Sensor specifications are as follows:

- **Camera**: Basler ace acA1920-50gc, 1920×1200, Sony IMX174, glo. shutter, color, GigE with 8 mm lenses.
- **LiDAR**: Ouster OS1-64 (gen. 2), 64 vert. layers, 360° FOV, below horizon configuration, 120 m range, 1.5 - 10 cm accuracy.

The TUMTraf-i dataset consists of 4 separate sequences, from S1 (1^{st} sequence) to S4 (4^{th} sequence), each comprising camera images and corresponding labeled LiDAR recordings. Sequences S1 and S2 each encompass a 30-second duration, depicting scenarios at dusk. Sequence S3 presents a 120-second sequence captured in daylight under sunny conditions. Finally, sequence S4 includes a 30-second recording obtained at night during heavy rainfall.

Figure 2.2: Visualization of 3D box labels and tracks in the TUMTraf Intersection Dataset (TUMTraf-i). The first row shows the labels projected into the two camera images. Below a registered point cloud from two LiDARs contains 3D box labels of the same scene. Taken from [Zim+23b].

A total of 2,400 frames have been captured, each synchronized across all sensors at a rate of 10 Hz using timestamps. The dataset includes 506 unique objects and 57,406 labeled 3D bounding boxes. Each object is categorized into one of the following classes: Car, Truck, Trailer, Van, Motorcycle, Bus, Pedestrian, Bicycle, Emergency Vehicle, and Other. However, we do not use the class "Other" in our studies. Figure 2.2 showcases an example of these objects and their corresponding classes and labeled 3D bounding boxes. In this thesis, we use all the sensors: 2 cameras and 2 LiDARs.

Figure 2.3: Image showing the positions of the sensors used in the Open Sensor Data for Rail 2023 (OSDaR23). Taken from [Tag+23].

2.4.2 OSDaR23 Dataset

Open Sensor Data for Rail 2023 (OSDaR23) is released by DB Netz AG, within the sector initiative Digitale Schiene Deutschland, and the German Center for Rail Transport Research (DZSF) at the Federal Railway Authority (EBA). The sensor setup consists of multiple calibrated and synchronized IR/RGB cameras, LiDAR and a radar sensors front-mounted on a railway vehicle. In this thesis, we use only the output data from high-resolution cameras and LiDAR sensors included in the dataset. Consequently, we utilize 3 high-res cameras with 6 LiDAR (3 long-range, 1 medium-range, and 2 short-range) which are registered together. These sensors have the following specifications:

- **High-resolution Camera**: Teledyne GenieNano 5GigE C4040, 4112x2504.
- **Long-range LiDAR**: Livox Tele-15, 50,000 - 84,000 points per frame.
- **Medium-range LiDAR**: HesaiTech Pandar64, 60,000 - 115,200 points per frame.
- **Short-range LiDAR**: Waymo Honeycomb, 20,000 - 40,000 points per frame.

The OSDaR23 dataset originally comprises 45 separate sequences, which include a total of 1,534 frames and 204,091 annotations. However, the sequences "4_station_pedestrian_bridge_4.4" and "19_vegetation_curve_19.1" were excluded from the study, resulting in a dataset consisting of 1,424 frames. All frames and sensors are synchronized at a rate of 10 Hz using timestamps. The dataset comprises 22 different

classes; however, some lack 3D bounding boxes, featuring only 2D ones. Additionally, certain classes occur either with extremely low frequency or problematic defined 3D bounding boxes. Consequently, we have chosen to utilize only the following classes for our studies: Person, Catenary Pole, Signal Pole, Road Vehicle and Buffer Stop.

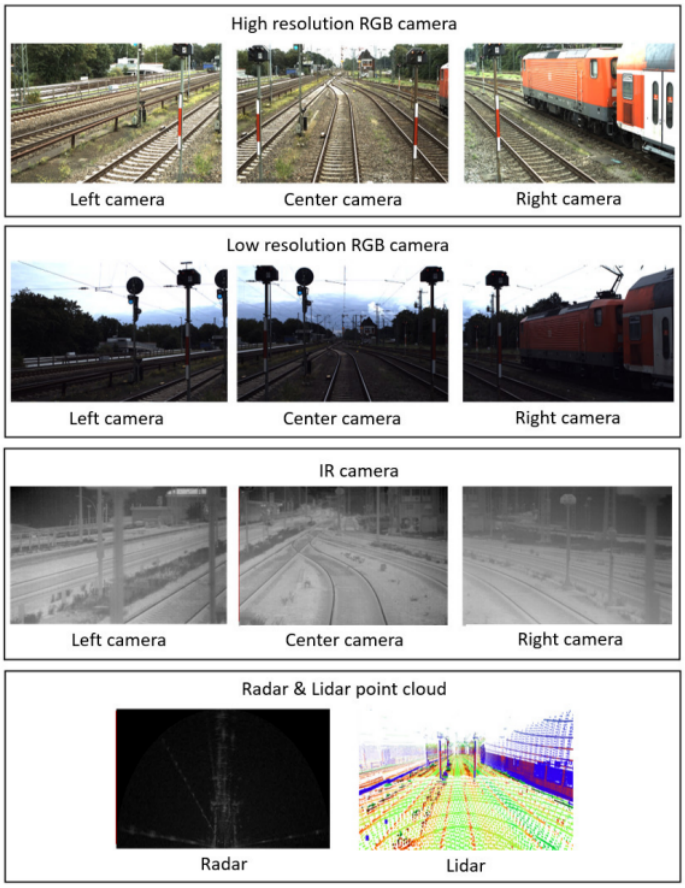


Figure 2.4: Representative samples of high-resolution, low-resolution, and infrared camera images, accompanied by radar and LiDAR data from the Open Sensor Data for Rail 2023 Dataset (OSDaR23) [Tag+23].

3 Related Works

3.1 3D Object Detection

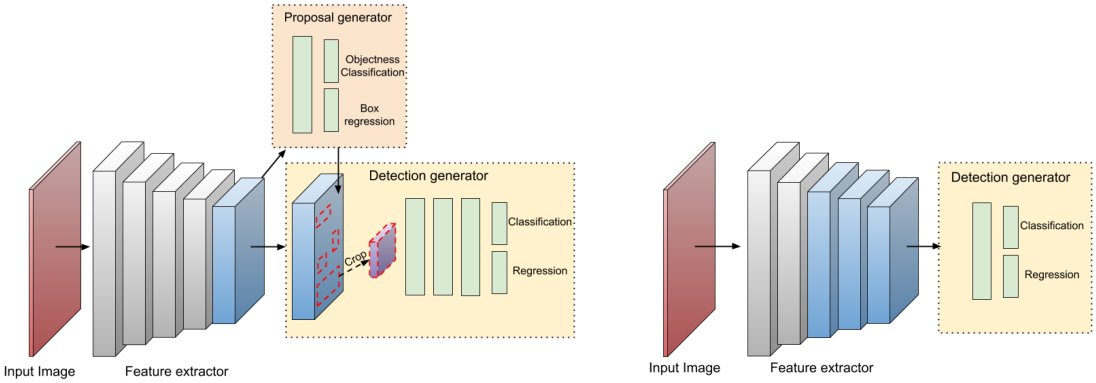
The task of 3D object detection is critical in various fields, including autonomous driving, robotic navigation, and many monitoring applications. Unlike 2D detection, which concerns itself with locating objects in image space, 3D object detection aims to establish the locations, rotations, sizes, and categories of the 3D objects in a scene with respect to the real world, thus demanding a deep understanding of the scene geometry. Models engineered for such detection tasks typically harness data of a multi-modal nature, fusing both images from camera systems and point cloud data garnered from Light Detection and Ranging (LiDAR) sensors. Nonetheless, these multi-modal models predominantly rely on individual sensor-based detection systems, such as those that are exclusively camera-only or LiDAR-only; hence, a preliminary investigation into these modalities is imperative.

3.1.1 Camera-only

Cameras offer a cost-effective solution for 3D object detection systems compared to sensors like LiDAR, hence they are extensively researched and utilized in both academia and industry, employing techniques that range from monocular and stereo to multi-view configurations.

Monocular detection is quite fast but is notably hampered by their inability to directly extract depth information from a single image, leading to sub-optimal predictions. To address this, research on this matter can be categorized by two principal methodologies: single-stage and two-stage detection (See Figure 3.1). Single-stage detection [Lin+18] [BL19] [Bra+20] [Luo+21] performs object classification and bounding-box regression simultaneously without the need for pre-generated region proposals. Conversely, two-stage detection [Ren+16] [Sim+19] [Shi+22] [QWL20] performs these tasks sequentially, initiating with the generation of region proposals followed by classification for each. Although two-stage methods typically achieve higher accuracy, they do so at the expense of reduced inference speed.

Stereo-based detection [LCS19] [Pen+20], on the other hand, is designed to identify 3D objects using a pair of images. These stereo images offer extra geometric con-



(a) Two-stage Faster R-CNN (b) One-stage RetinaNet

Figure 3.1: Comparison between one-stage (single-stage) RetinaNet [Lin+18] and two-stage Faster-RCNN [Ren+16]. Taken from [Car+21].

straints over monocular images, enabling more precise depth estimation. As a result, stereo-based approaches tend to outperform monocular methods in detection accuracy. However, the effectiveness of stereo cameras depends heavily on precise calibration and synchronization, which are often challenging to attain in practical settings.

Recent advancements in multi-view 3D object detection have led to methods [PF20] [Hua+22] [RKC18] [Pan+20] [RC20] [Li+22b] [Li+23] that attempt to utilize a unified representation in bird's-eye view (BEV) space by mapping multi-view images onto it and then applying a BEV-based detector to this integrated feature map for 3D object detection. Specifically, LSS [PF20] which is one of the pioneers, does this by first "lifting" each image individually into a frustum of features for each camera, then "splatting" all frustums into a rasterized bird's eye view space (See Figure 3.2). Nonetheless, the conversion from camera views to BEV can be imprecise without accurate depth information, resulting in imperfect alignment between image pixels and their corresponding BEV space coordinates. The recently introduced EA-LSS model [Hu+23] tries to address this issue by proposing an edge-aware adaptation of LSS, wherein it employs an edge-aware depth fusion and a fine-grained depth module to enhance depth accuracy in regions with rapidly varying depth.

Another important aspect is the choice of a feature extraction backbone model, which is essential for the production of competitive results by the methods mentioned above. The range of available backbone models is extensive, including classic models like ShuffleNet [Zha+17], Xception [Cho17], and DetNet [Li+18] as well as modern ones such as EfficentNet [TL20], and those introduced in various versions of YOLO models [RF18] [WBL22] [TC23a]. Models that incorporate transformers, such as DeiT

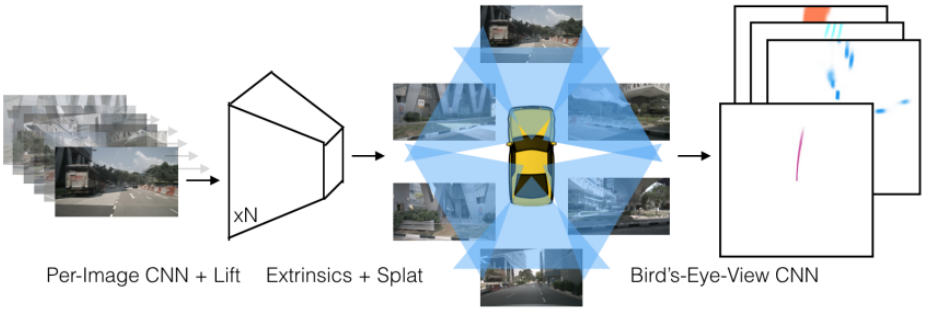


Figure 3.2: The LSS introduced in [PF20] processes multi-view images along with their corresponding extrinsic and intrinsic parameters to produce a frustum-shaped point cloud for each image. These point clouds, embedded with computed depth values, are subsequently transformed into bird's-eye view (BEV) space and then processed for a task via CNN. Taken from [PF20].

[Tou+21] and Swin-T [Liu+21], are also notable. For example, Swin-T employs a multi-stage hierarchical architecture that computes attention within local windows, divided into sub-patches. This design allows for the capture of interactions between different window locations by gradually shifting the partitioning across network levels, thus covering overlapping regions and maintaining efficient, linear computational complexity by localizing self-attention computations [Kha+22]. However, transformers generally require a significant amount of data for training and are known for their high computational complexity [Lin+22].

3.1.2 LiDAR-only

LiDAR sensors have become the cornerstone of high-precision 3D object detection. LiDAR technology offers the ability to capture fine-grained 3D information about the surrounding environment by sending out laser beams and measuring the time it takes for them to return after reflecting off objects. This "time-of-flight" data translates into accurate distance measurements, producing a rich point cloud that precisely represents the scene geometry.

Like camera-based models, LiDAR-based models are primarily classified into single-stage and two-stage types. Single-stage models, such as VoxelNet [ZT17] and PointPillars [Lan+19], process point clouds and flatten them into Bird's Eye View (BEV) space for detection. VoxelNet is an innovative approach that employs 3D sparse voxels and introduces a Voxel Feature Encoding layer to extract features from points within

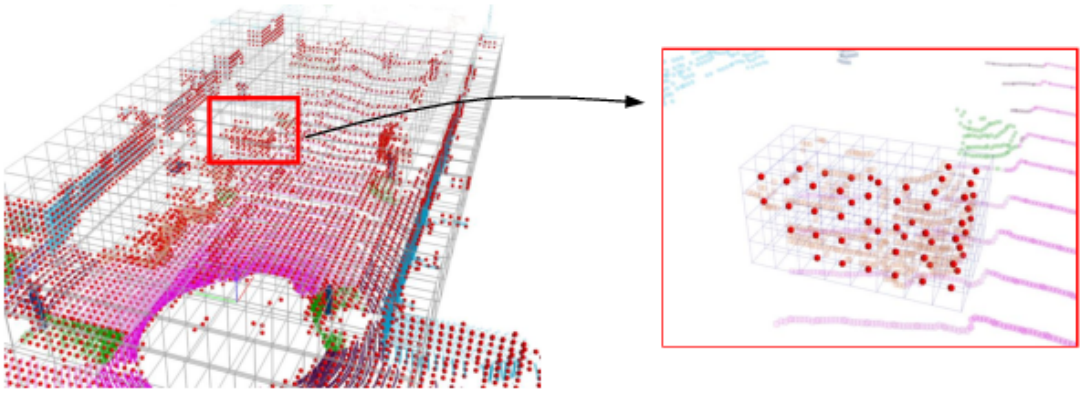


Figure 3.3: An illustration of how voxelization works on cloud points [Xu+21].

voxels for comprehensive volume representation. Conversely, PointPillars employs voxelization using vertically unbounded shape called a pillar, enabling faster encoding. For two-stage models, PointRCNN [SWL19], Fast Point R-CNN [Che+19], PV-RCNN [Shi+21] and LiDAR R-CNN [LWW21] incorporate a RCNN network into one-stage detectors to enhance performance with refinement modules. Beyond classic voxel [ZT17] [Sun+22] [SWL19] [YML18] [Den+21] and pillar [ZT17] [Zhu+19] [Wan+20] representations, there are also others such as cylindrical [Che+20] and spherical [Lai+23] representations, offering distinct solutions to specific problems.

As transformers advance and become suitable for 3D object detection with point clouds, works such as FlatFormer [Liu+23] and DSVT [Wan+23] have become alternatives claiming superior accuracy-latency trade-offs over popular sparse convolutional backbones.

Despite their precision, LiDAR sensors present challenges, the most significant being their cost, which can be prohibitively expensive, and their sparse nature, especially at longer ranges or in environments with fewer surfaces to reflect off. Moreover, processing the high-dimensional point cloud data requires substantial computational resources, often leading to longer processing times; Consequently, the current trend of more sophisticated methods is streamlining model selection, enhancing their suitability for real-time applications.

3.1.3 Multi-modal

The integration of camera and LiDAR data, referred to as multi-modal, seeks to leverage the complementary strengths of both sensor modalities to improve the robustness and accuracy of 3D object detection. While cameras provide high-resolution texture information and color, which is beneficial for object classification, LiDARs offer precise and reliable depth information, crucial for object localization and dimension. Multi-modal 3D detection models have different fusion strategies, categorized as early, late, and deep fusion.

Early-fusion

Early fusion, also termed sensor-level or data-level fusion, combines camera and LiDAR data at the start of the processing pipeline. This method merges raw data prior to feature extraction to create an unified representation, blending detailed visuals from the camera with precise LiDAR depth measurements. Integrating these data sources early aims to maximize the raw potential of each modality, which could enhance performance in object recognition and scene understanding. However, early fusion demands precise alignment of data streams otherwise cross-correlations between data items cannot be exploited, resulting in poor performance. Typically, these methods are executed sequentially: first, 2D detection or segmentation networks extract features from images; next, this information enhances the point cloud data; and finally, the enriched point cloud is input into a LiDAR-based 3D object detector. Consequently, the sequential nature of this process limits the flexibility in processing speed, which crucial for multi-modal models.

There are two types of early fusion: the first leverages knowledge from images to narrow down candidate regions in a 3D point cloud, and the second augments the input point cloud with image features. For the former, Frustum PointNet [Qi+18] initiates this approach, and then Frustum ConvNet [WJ19] makes improvements by applying convolutinal networks on divided grid cells from frustums. Regarding the latter, PointPainting [Vor+20] pioneers this technique by projecting LiDAR point clouds onto the output of an image-only semantic segmentation network and appending the corresponding class scores to each point. However, PointPainting's efficacy is contingent upon the quality and format of the semantic segmentation output, which may significantly increase the risk of false detections. PointPainting++ [Gao+23] mitigates these issues by implementing a weighted classification loss that prioritizes anchors with uncertain semantic information and by integrating a dual-attention module to enhance the refinement of the voxelized point cloud. Additionally, MVP [YZK21b] represents another technique, utilizing semantic information from a camera-

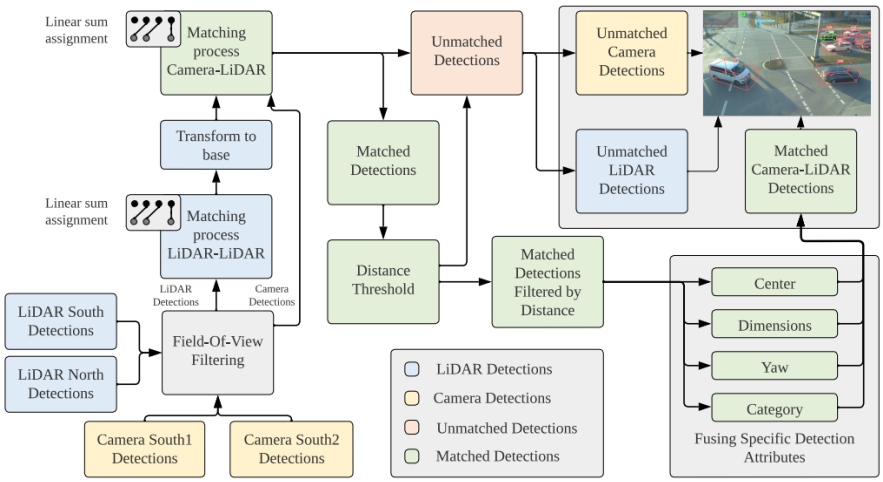


Figure 3.4: Late-fusion Multi-modal 3D object detection pipeline [Zim+23a].

only segmentation network to generate dense 3D point clouds through interpolation.

Late-fusion

In contrast to early fusion, late fusion takes place at a later stage in the processing pipeline. Here, each sensor modality operates independently to produce its own set of bounding boxes, which are then combined at a decision level to produce a final output. This method capitalizes on the strengths of each sensor's processing techniques separately before making a unified inference, allowing for greater flexibility and the possibility of redundancy, which can be beneficial for system reliability. Late fusion is typically less demanding computationally compared to early fusion and can be easier to implement. However, it may not exploit the full potential of the combined data to the same extent as early or deep fusion approaches.

CLOCS [PMR20] exemplifies late fusion, working on pre-NMS (Non Maximum Suppression) output from both 2D and 3D detectors and harnessing their geometric and semantic consistencies to enhance the accuracy of the final 3D and 2D detection results. Fast-CLOCs [PMR22] enhances performance through a lightweight 3D detector-cued 2D image detector (3D-Q-2D) that extracts visual features to significantly improve 3D detection, and by sharing 3D detection candidates as proposals with the 3D-Q-2D, it substantially reduces network complexity. In relation to our thesis, InfraDet3D [Zim+23a], evaluated on one of our datasets TUMTraf-i [Zim+23b], proposes a pipeline that fuses data from two LiDARs through early fusion. Furthermore, detections from monocular cameras are integrated at a later stage to enhance robustness and to improve

the detection of small objects (see Figure 3.4).

Deep-fusion

Deep fusion, often referred to as feature-level or intermediate-level fusion, is a sophisticated approach that merges the strengths of both early and late fusion techniques. It is realized by integrating camera and LiDAR data at different levels within a deep learning architecture. This method allows the network to automatically learn the most effective way to combine features from each sensor modality at various stages of abstraction. The fusion can occur at multiple points within the neural network, leveraging the benefits of joint representations while still allowing individual sensor characteristics to contribute to the final decision. Deep fusion is particularly powerful for complex tasks like 3D object detection, where the combination of hierarchical features from both modalities can lead to a more robust and accurate system. However, it requires significant data and computational resources, along with more complex network architectures, to achieve its full potential.

Numerous efforts have been made in the research community to optimize the deep-fusion process, specifically, on the image and LiDAR features within backbone networks. LiDAR-to-camera transformations establish point-to-pixel correspondences which are then used to fuse features from LiDAR and image backbones using various operators. For example, works of [Lia+20a] and [Lia+20b] utilize continuous convolutions to fuse image and LiDAR feature maps at various resolution levels, while DeepFusion [Li+22a] and CAT-DET [ZCH22] employ an attention mechanism to dynamically capture the correlations between modalities. On the other hand, works such as PointFusion [XAJ18], EPNet [Hua+20b] and MSF-MC [Wan+21] incorporate image-to-LiDAR transformations for point-based detection backbones. This whole fusion can also be done at the output feature maps of backbone networks [Liu+22] [Lia+22]. BEVFusion [Liu+22] is particularly notable, as it addresses and alleviates key efficiency bottlenecks in view transformation through optimized BEV pooling, which includes caching/pre-computation and interval reduction techniques to decrease latency. In parallel, with the same name, BEVFusion [Lia+22] introduces an Adaptive Module (ADP) to enhance up-sampled features through adaptive average pooling and a 1x1 convolution prior to concatenation. Contrary to employing pooling or 3D convolutions for z-dimension compression, this method utilizes a Spatial to Channel (S2C) operation that reshapes a 4D tensor into 3D, thereby reducing computational expense.

There are also transformer-based models, such as TransFusion [Bai+22] and CMT [Yan+23]. The TransFusion model generates queries from the high-response regions of LiDAR features, after which the object queries separately interact with point cloud and image features. However, in CMT, object queries directly engage with multi-modal

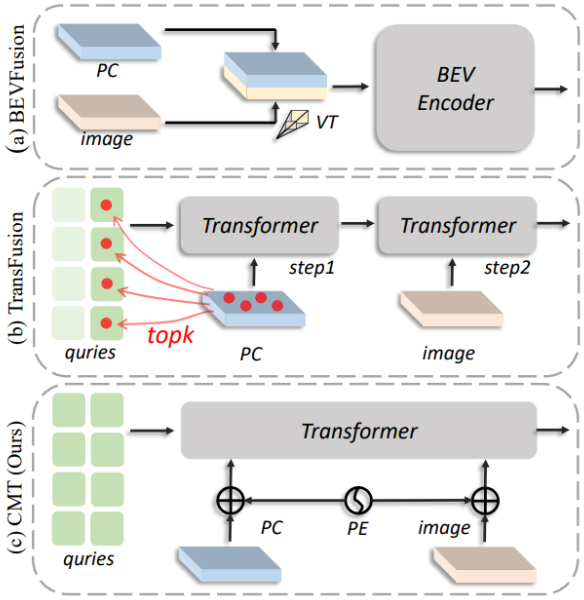


Figure 3.5: Comparison between deep-fusion models BEVFusion [Liu+22], TransFusion [Bai+22], and CMT [Yan+23]. Taken from [Yan+23].

features simultaneously via Position encoding (PE) for alignment (see Figure 3.5).

3.2 Temporal 3D Object Detection

In the previous sections, we explored 3D object detection and many popular methods based on different modalities. As an extension that can be added to these popular methods, the integration of temporal information emerges as a promising research direction for advancement. This is predicated on the hypothesis that leveraging sequential data significantly enhances the accuracy and robustness of detection systems at the expense of processing speed.

Temporal aggregation, a critical feature of this progression, involves the synthesis of data across consecutive frames to construct a richer, more informative representation of the environment. Figure 3.6 illustrates several approaches adopted for temporal aggregation, each with its unique strengths.

Attention mechanisms [Yan+21] [Cai+23], for instance, selectively focus on different parts of the data sequence, thereby enhancing relevant features while suppressing the irrelevant ones, and thus, allocating computational resources efficiently to areas of interest over time.

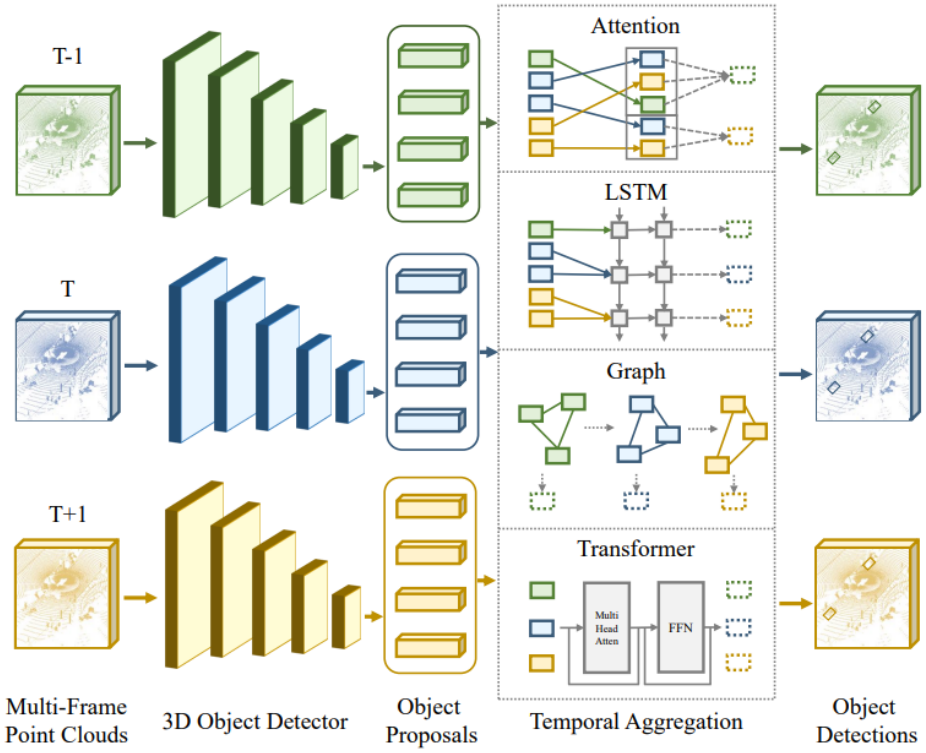


Figure 3.6: An illustration of various temporal aggregation types for multi-frame LiDAR sequences. Taken from [Mao+23].

Transformers, a revolutionary architecture in the field of deep learning, have also been adapted for temporal data processing [Yua+20]. Transformers can weigh the influence of different time steps in the sequence, enabling the model to capture long-range dependencies and temporal dynamics with remarkable effectiveness. However, their computational bulkiness should be considered since sequential data are processed collectively, which may limit feasibility in real-time scenarios.

Long Short-Term Memory (LSTM) [Hua+20a] and Gated Recurrent Unit (GRU) [Yin+20] networks are classes of recurrent neural networks designed to remember information for long periods. In the context of 3D object detection, such networks can track the changes in object position and orientation over time, within either proposals or feature maps. Unlike Transformers, LSTMs and GRUs enable sequential processing, thus offering advantages in real-time applications in terms of speed.

Lastly, research on Graph Neural Networks (GNN) [Zha+20] has extended to include temporal 3D object detection. GNNs can effectively model the evolving relationships

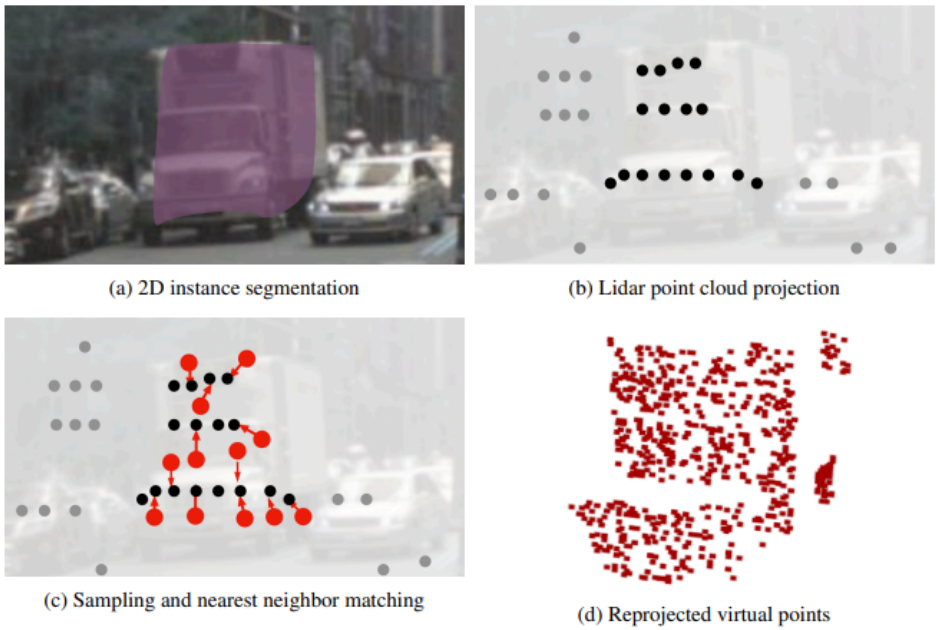


Figure 3.7: Overview of Multi-modal Virtual Point (MVP) generation framework. Taken from [YZK21b].

between objects within a scene, considering both the individual properties of objects and the dynamic topological structure of the scene.

3.3 Data Augmentation

Data augmentation plays a crucial role in enhancing the performance of 3D object detection models. In the field of machine learning, particularly within computer vision, having a vast and varied dataset is essential for training robust models. However, acquiring such diverse datasets, especially for 3D objects, can be challenging. Data augmentation steps in as a practical solution, artificially expanding the dataset by generating new samples from existing data through a variety of transformations so that models can have better generalization.

There are various data augmentation techniques, one of which is the MVP [YZK21b], which uses a series of 2D detections to generate dense 3D virtual points via interpolation, thereby enriching sparse 3D point clouds. This enables the preliminary training of MVP to produce an augmented dataset, which can subsequently be used to train other models. Other examples that follow the same philosophy are PointPainting [Vor+20]

and PointPainting++ [Gao+23], which augment 3D point clouds by integrating semantic information from image segmentation models. Additionally, data-driven point cloud up-sampling techniques, such as those proposed in [Yu+18], [Yif+19], and [Li+19], learn to perform up-sampling through convolutional or generative networks. However, these techniques are unsuitable for real-time applications due to their extensive computational demands.

Beyond basic point cloud augmentation methods like rotation, translation, scaling, or shuffling, the method introduced in SECOND [YML18] as Ground-Truth Paste diversifies training set by sampling ground truth objects along with their point clouds from a database and then pasting them into the frame scene. This method successfully increases the number of objects in each scene, thus simulating objects across a wide array of situations. Furthermore, advancements introduced by LiDAR-Aug [Fan+21] and Real-Aug [Zha+23a] facilitate further extensions of this method by creating more realistic scenarios via learnable techniques.

4 Methodology

4.1 Temporal Dataset Split Search

To train and evaluate models that require sequential frame inputs in their pipelines, it is essential to partition the dataset in a specific manner, ensuring the presence of sufficient continuous sequences in the training, validation, and test sets. Doing this manually is quite a challenge in our case due to the imbalanced class distributions [JK19] in both datasets, TUMTraf-i [Zim+23b] and OsDAR23 [Tag+23]. Besides, balancing class numbers across sets is not enough; we must also consider 3D object attributes, such as their distances from the ego position and whether they are occluded or not. Without the consideration of these attributes, the risk of inadequate generalization arises. For example, if one set predominantly contains objects near the ego position while others feature distant objects, training generalization may be compromised.

Consequently, we introduce a novel exhaustive search algorithm that takes advantage of multiprocessing techniques to find optimal partitions that maintain uniformity in terms of aforementioned object attributes across the partitioned sets.

4.1.1 Definitions

To understand the fundamentals of the algorithm, we need to define some terms:

- List of classes in a dataset:

$$C = \{c_1, c_2, \ldots, c_n\} \tag{4.1}$$

- For every class $c_n \in C$, there are associated weights:

$$w_n = \{w_{c_n}, w_{d_n}, w_{l_n}, w_{o_n}\} \tag{4.2}$$

where: w_{c_n} = the weight for class' occurrences
w_{d_n} = the weight for class' categorized distance
w_{l_n} = the weight for class' categorized count of cloud points
w_{o_n} = the weight for class' categorized occlusion level

- Categorized distances from the ego position, denoted by D:

$$D = \{d_1, d_2, \ldots, d_n\} \tag{4.3}$$

where n is the number of distance categories specific to dataset. For OsDAR23 dataset, values would be

$$D = \{< 50, 50 - 99, 100 - 149, 150 - 200, > 200\}$$

whereas for TUMTraf-i dataset, the values would be

$$D = \{< 40, 40 - 50, > 50\}$$

- Categorized count of cloud points inside 3D bounding boxes denoted by L:

$$L = \{l_1, l_2, \ldots, l_n\} \tag{4.4}$$

where n is the number of categories of point cloud counts inside 3D bounding boxes. Values of L differs depending on the dataset. For OsDAR23 dataset, values are

$$L = \{< 199, 200 - 499, 500 - 999, 1000 - 1999, 2000 - 2999, > 3000\}$$

whereas for TUMTraf-i dataset, the values are

$$L = \{< 19, 20 - 49, > 50\}$$

- Categorized occlusion levels are denoted by O:

$$O = \{o_1, o_2, \ldots, o_n\} \tag{4.5}$$

where n is the number of occlusion categories specific to dataset. Values of O differs depending on the dataset. For OsDAR23 dataset, values are

$$O = \{0 - 25\%, 25 - 50\%, 50 - 75\%, 75 - 100\%, 100\%\}$$

whereas for TUMTraf-i dataset, the values are

$$O = \{\text{not\_occluded}, \text{partially\_occluded}, \text{mostly\_occluded}, \text{unknown}\}$$

- A 3D object b is characterized by the following attributes:

$$b = (c_b, d_b, l_b, o_b) \tag{4.6}$$

where: c_b = the class and $c_b \in C$
d_b = the categorized distance and $d_b \in D$
l_b = the categorized count of cloud points and $l_b \in L$
o_b = the categorized occlusion level and $o_b \in O$

4.1.2 Execution Parameters

In order for our search algorithm to be able to create a search space, the following parameters are manually selected before the execution:

- N_f - the minimum number of frames to be included in pseudo-sequences. For example, if $N_f = 20$, the search algorithm will create pseudo-sequences consisting of 20 sequential frames. However, if the original sequence has less frames then N_f, then the whole original sequence will be taken into account as a pseudo-sequence.
- N_p - the number of permutations of pseudo-sequences to be searched. For example, if $N_p = 100,000$, the search algorithm will consider 100,000 different permutations of pseudo-sequences and calculate a cost for each of them.
- N_w - the number of processes to be used for multi-processing. Each process handles at least $\left\lfloor \frac{N_p}{N_w} \right\rfloor$ permutation. The last process will handle any remained permutation.
- R_t - list of target ratio values for sets as can be described as follows:

$$R_t \in \left\{ (r_1, \ldots, r_n) \mid r \in [0,1], \sum_{i=1}^{n} r_i = 1,\ 0 \leq r_i \leq 1\ \forall i \right\} \tag{4.7}$$

  For example, if $R_t = \{0.8, 0.1, 0.1\}$, the search algorithm will try to create a training set that contains 80% of the data, a validation set that contains 10% of the data, and a test set that contains 10% of the data.

- W - the list of class weights to be used in cost operations. With individual weights of classes defined in equation (4.2), W can be described as follows:

$$W = \{w_1, w_2, \ldots, w_{|C|}\} \tag{4.8}$$

  where $|C|$ is the number of classes in the dataset. These weights are important as they allow putting more emphasis on specific classes and attributes.

- T_c - a Boolean value ensuring all sets contain identical classes as follows:

$$T_c \iff \forall i, j \in S \quad (c \in C_i \iff c \in C_j, \forall c \in C) \tag{4.9}$$

  where S = {train, val, test}; C_i and C_j are the sets of classes present in i-th and j-th sets respectively.

- T_d - the categorized distance threshold value that takes values in the interval [0,1]. This threshold helps eliminate sequence permutations where the training set has a wider range of distance categories than the validation or test sets. The constraints for this filter are:

$$|D_{train}| \cdot T_d \leq |D_{val}|, \quad |D_{train}| \cdot T_d \leq |D_{test}| \tag{4.10}$$

where: $|D_{train}|$ = the number of distance categories in the training set
$|D_{val}|$ = the number of distance categories in the validation set
$|D_{test}|$ = the number of distance categories in the test set

For instance, a T_d value of 1 requires all sets to contain an equal number of distance categories. A value less than 1 allows the validation or test set to have fewer categories. Adjusting this threshold is crucial, especially when N_f has a high value, making it challenging to satisfy the constraints.

- T_l - the categorized count of clouds points threshold value that takes values in the interval [0,1]. This threshold helps eliminate sequence permutations where the training set has a wider range of cloud points categories than the validation or test sets. The constraints for this filter are:

$$|L_{train}| \cdot T_l \leq |L_{val}|, \quad |L_{train}| \cdot T_l \leq |L_{test}| \tag{4.11}$$

where: $|L_{train}|$ = the number of cloud point categories in the training set
$|L_{val}|$ = the number of cloud point categories in the validation set
$|L_{test}|$ = the number of cloud point categories in the test set

For instance, a T_l value of 1 requires all sets to contain an equal number of cloud points categories. A value less than 1 allows the validation or test set to have fewer categories.

- T_o - the categorized occlusion level threshold value that takes values in the interval [0,1]. This threshold helps eliminate sequence permutations where the training set has a wider range of occlusion level categories than the validation or test sets. The constraints for this filter are:

$$|O_{train}| \cdot T_o \leq |O_{val}|, \quad |O_{train}| \cdot T_o \leq |O_{test}| \tag{4.12}$$

where: $|O_{train}|$ = the number of occlusion categories in the training set
$|O_{val}|$ = the number of occlusion categories in the validation set
$|O_{test}|$ = the number of occlusion categories in the test set

For instance, a T_o value of 1 requires all sets to contain an equal number of occlusion level categories. A value less than 1 allows the validation or test set to have fewer categories.

4.1.3 Algorithm Overview

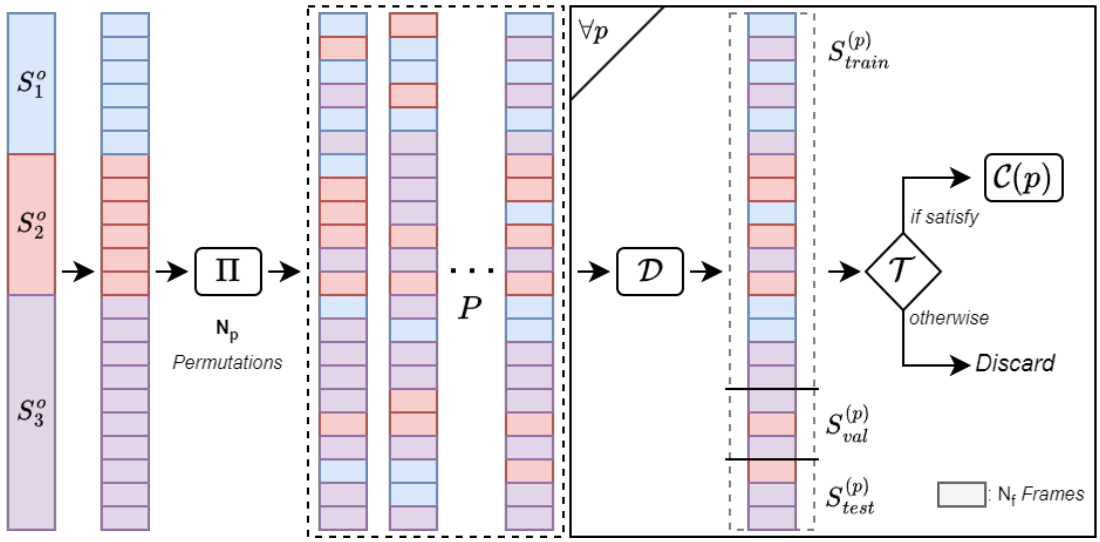


Figure 4.1: Overview of our temporal dataset split search algorithm. A list of original sequences (denoted as S^o) is compiled and segmented into pseudo-sequences of N_f frames each. N_p permutations are created by rearranging these pseudo-sequences. Using operation $\mathcal{D}$, as described in 4.13, each permutation is split into sets: $S_{train}^{(p)}$, $S_{val}^{(p)}$, and $S_{test}^{(p)}$. The algorithm checks if these sets together satisfy the constraints using $\mathcal{T}$ (4.14). When constraints are met, operation $\mathcal{C}$ (4.19) determines the cost, guiding the algorithm to choose the optimal split with the minimum cost.

The search algorithm consists of three main parts: creating permutations of pseudo-sequences, discarding those that do not satisfy the constraints, and calculating costs for the remaining permutations.

Creating Permutations

Each dataset comprises a predetermined list of actual frame sequences. The algorithm aggregates all such frames in the dataset into a list and divides them into pseudo-

sequences with a minimum length of N_f. A pseudo-sequence can be considered a small continuous part of an actual sequence. These divisions are guided by the following constraints:

- Frames within the same pseudo-sequence must originate from the same sequence.
- The number of frames in a pseudo-sequence must be at least N_f.

Consequently, some pseudo-sequences might contain more frames than N_f, especially if forming another complete pseudo-sequence of minimum length N_f is not feasible with the remaining frames. For these pseudo-sequences, the algorithm generates N_p permutations, collectively denoted by P. If we assume that for a specific permutation p, it represents a set of pseudo-sequences of size N_f, then for a given R_t defined in 4.7, the division operation $\mathcal{D}$ for each permutation can be defined as follows:

$$\mathcal{D}(p, R_t) = \left\{ S_{train}^{(p)}, S_{val}^{(p)}, S_{test}^{(p)} \right\} \quad \forall p \in P \tag{4.13}$$

Where, for each permutation $p \in P$, $S_{train}^{(p)}$ represents the set of pseudo-sequences allocated to the training set, $S_{val}^{(p)}$ to the validation set, and $S_{test}^{(p)}$ to the test set.

Discarding Permutations

After the creation of permutations, the operation denoted by $\mathcal{T}$, is applied to each of them to discard those that do not satisfy the constraints. The constraints are defined by the following parameters: T_c (4.9), T_d (4.10), T_l (4.11), and T_o (4.12). The operation $\mathcal{T}$ can be further defined using the constraints:

$$\mathcal{T}(S_{train}^{(p)}, S_{val}^{(p)}, S_{test}^{(p)}) = \begin{cases} \text{keep} & \text{if } p \text{ satisfies } T_c, T_d, T_l, \text{ and } T_o \\ \text{discard} & \text{otherwise} \end{cases} \tag{4.14}$$

Calculating Objective Function

Once permutations have been filtered based on the aforementioned constraints, the next step is to rank the viable permutations based on their potential effectiveness. This is done by calculating a cost for each permutation. The objective is to minimize this cost, ensuring that the datasets derived from the optimal permutation provide a balanced representation in terms of classes, distances, cloud point counts, and occlusion levels.

The cost function $\mathcal{C}$ for a given permutation p aims to measure the differences in attribute distributions between the training set and validation/test sets. This cost is essentially a weighted sum of the differences in distribution of the attributes: class, distance, cloud points, and occlusion. The breakdown of cost function is as follows:

1. For class distribution difference of the n^{th} class:

$$\Delta C^{(p)}_{train,set,n} = |C^{(p)}_{train,n} - C^{(p)}_{set,n}| \tag{4.15}$$

where $set \in \{val, test\}$ and $C^{(p)}_{set,n}$ represents the total count of n^{th} class in the specified *set* for a given permutation p.

2. For distance distribution difference of the n^{th} class:

$$\Delta D^{(p)}_{train,set,n} = |D^{(p)}_{train,n} - D^{(p)}_{set,n}| \tag{4.16}$$

where $set \in \{val, test\}$ and $D^{(p)}_{set,n}$ represents the total count of categorized distances of the n^{th} class in the specified *set* for a given permutation p.

3. For cloud point count distribution difference of the n^{th} class:

$$\Delta L^{(p)}_{train,set,n} = |L^{(p)}_{train,n} - L^{(p)}_{set,n}| \tag{4.17}$$

where $set \in \{val, test\}$ and $L^{(p)}_{set,n}$ represents the total count of categorized cloud points of the n^{th} class in the specified *set* for a given permutation p.

4. For occlusion distribution difference of the n^{th} class:

$$\Delta O^{(p)}_{train,set,n} = |O^{(p)}_{train,n} - O^{(p)}_{set,n}| \tag{4.18}$$

where $set \in \{val, test\}$ and $O^{(p)}_{set,n}$ represents the total count of categorized occlusion levels of the n^{th} class in the specified *set* for a given permutation p.

5. Summing up the distribution differences using weights from W defined in 4.8:

$$\begin{aligned} \mathcal{C}(p) = \sum_{n=1}^{|C|} & w_{c_n} \left(\Delta C^{(p)}_{train,val,n} + \Delta C^{(p)}_{train,test,n} \right) \\ & + w_{d_n} \left(\Delta D^{(p)}_{train,val,n} + \Delta D^{(p)}_{train,test,n} \right) \\ & + w_{p_n} \left(\Delta L^{(p)}_{train,val,n} + \Delta L^{(p)}_{train,test,n} \right) \\ & + w_{o_n} \left(\Delta O^{(p)}_{train,val,n} + \Delta O^{(p)}_{train,test,n} \right) \end{aligned} \tag{4.19}$$

6. The permutation with the least cost value is then picked as follows:

$$p^* = \text{argmin}_p \mathcal{C}(p) \tag{4.20}$$

where p^* is the optimal permutation that minimizes the cost function $\mathcal{C}(p)$.

Lastly, the optimal partitioned sets $\left\{ S^{(p^*)}_{train}, S^{(p^*)}_{val}, S^{(p^*)}_{test} \right\}$ of p^* are selected and then saved for further uses.

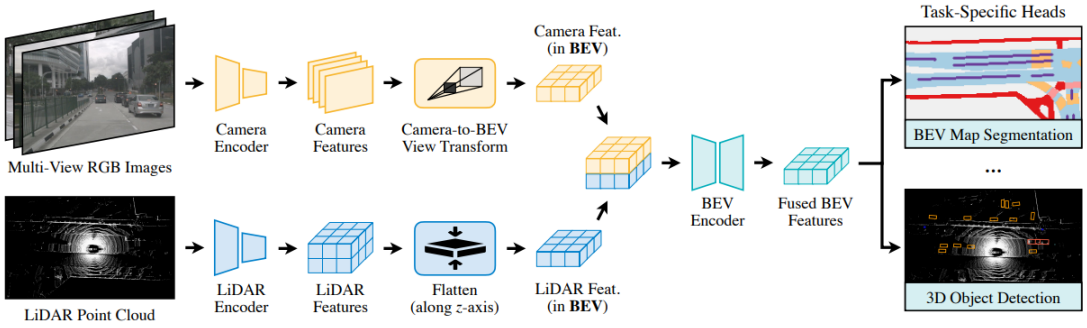


Figure 4.2: BEVFusion, our **baseline** model, extracts features from multi-modal inputs and converts them into a shared bird's-eye view (BEV) space efficiently using view transformations. It fuses the unified BEV features with a fully-convolutional BEV encoder and supports different tasks with task-specific heads. Taken from [Liu+22].

4.2 Temporal Pipeline

4.2.1 General Structure

The methods introduced in this thesis are built upon the works of *BEVFusion* by [Liu+22], which utilizes *MMDetection3D* [Con20] as its framework and incorporates various custom functionalities specifically designed for non-temporal implementations. Consequently, an essential step in this thesis involved converting the entire code base to accommodate our temporal methods. We designate the default BEVFusion model as our baseline and evaluate our methods on the temporal version of the model accordingly. In our implementation, we replaced the BEV Encoder with the Feature Fuser, a simple tensor operation that concatenates feature maps from multiple backbones.

4.2.2 Data Sampling

In object detection domain, the inherent long-tail distribution of classes presents significant challenges. Predominant "head" classes with plentiful data representation can overshadow the less frequent "tail" classes, skewing the model's learning focus [Zha+23b]. For instance, a disproportionate representation of one class can hinder the model's capacity to adequately learn features of underrepresented classes [Zhu+19].

Therefore, we adapt the sampling strategy proposed by [Zhu+19], namely, Class-balanced Grouping and Sampling, to ensure that the models are trained on a balanced dataset with uniform class representation. In regards to temporal compatibility, several

constraints are added to sampling algorithm to ensure that no frame without any prior frames is ever sampled, specifically, for the training pipeline.

4.2.3 Data Loading

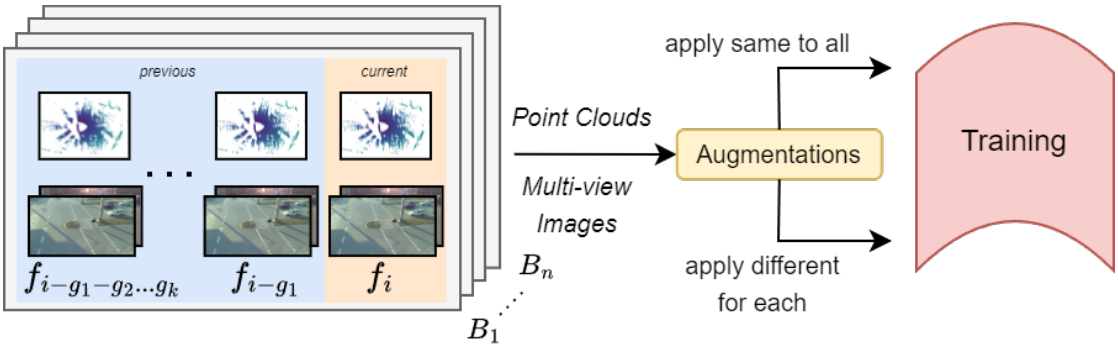


Figure 4.3: Visualization of temporal data loading for training pipeline. Sequential frames, which may contain gaps in their indices, are loaded as a sequential sequence with a specified length denoted by Q. The sum of individual g values must not exceed G. Subsequently, data augmentation methods are applied to these sequences, with the manner of application contingent on the specific configuration of the method, determining whether the effects are applied uniformly across all sequential frames or not.

We introduce a configurable temporal data loading method, allowing models with diverse requirements to customize the queue length and incorporate the possibility of gaps between sequential frames. Let the set F represent a sequence of frames to be loaded. The frame at the current index is denoted by f_i. The set F is constrained by two parameters:

- Queue Length (Q): This parameter specifies that the size of set F must be Q, that is, the number of frames in F should be equal to Q. Not only does this serve as an operational measure, but it also functions as a hyper-parameter that can be fine-tuned to optimize the model's performance. An extended queue length affords the model a broader temporal context, which is beneficial in recognizing objects that are obscured over longer duration. However, this increase in length has a corresponding effect on the model's memory usage and inference time.

  In situations where there are insufficient prior frames to meet the size Q, the system will repeatedly load the same frame. This approach ensures that the

queue consistently maintains its maximum capacity.

- Queue Gap (G): This parameter accounts for the potential presence of gaps between sequential frames in F. While gaps can exist, their cumulative size must not surpass G throughout the sequence. For any two consecutive frames f_{i-k} and f_i in F, where $1 \leq k < Q$, the size of the gap is k. The summation of all such gaps in the sequence can be denoted as:

$$\sum_{j=1}^{Q-1} j \cdot g_j \leq G \tag{4.21}$$

  Here, g_j represents the number of gaps of size j in the sequence. This constraint guarantees temporal coherence, ensuring that abrupt transitions or omitted frames don't compromise the analysis. Moreover, it provides an additional mechanism for training generalization.

4.2.4 Augmentation Methods

Data augmentation is an indispensable technique in deep learning, used to artificially enhance the variability of training datasets, thus tackling the problem of over-fitting while training. In the context of temporal data sequences, the importance of consistent application of augmentation methods across all frames becomes necessity.

Consistency Across Frames

Temporal sequences contain inherent chronological information that establishes relationships between frames. Inconsistent augmentations can disrupt these temporal relationships, leading to misleading or conflicting cues for models. Consider a sequence where one frame has undergone a rotation while the subsequent frame has not; the inconsistency introduced can confuse the model in recognizing the progression of an object's movement or position. Therefore, applying augmentation methods uniformly across frames ensures that the inherent temporal coherence remains undisturbed, leading to improved model robustness and performance.

Image Augmentations

The data augmentation methods detailed below, specifically designed for image inputs, have been modified to ensure consistent augmentation across temporal frames:

- **Image Augmentation in 3D:** As opposed to conventional 2D image augmentations, this method manipulates the image as if it is in 3D space. Cropping as well

Figure 4.4: An output sample image from *Image Augmentation in 3D* method.

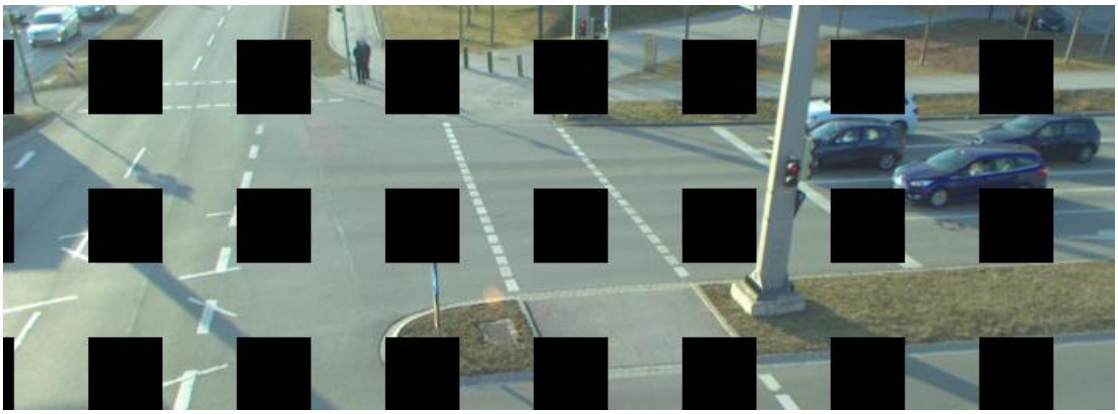


Figure 4.5: An output sample image from *Image Grid Mask* method.

as transformations like flipping, rotations and translations are applied considering the full 3D context by updating the corresponding calibration matrices. A visual example can be seen in 4.4.

- **Image Grid Mask:** This method applies a regular grid of masks to the image, blocking certain sections and forcing the model to learn from partial data. When applied consistently across a sequence, it simulates scenarios where certain sections of data are consistently obscured or missing, such as due to occlusions. A visual example can be seen in Figure 4.5.

Cloud Point Augmentations

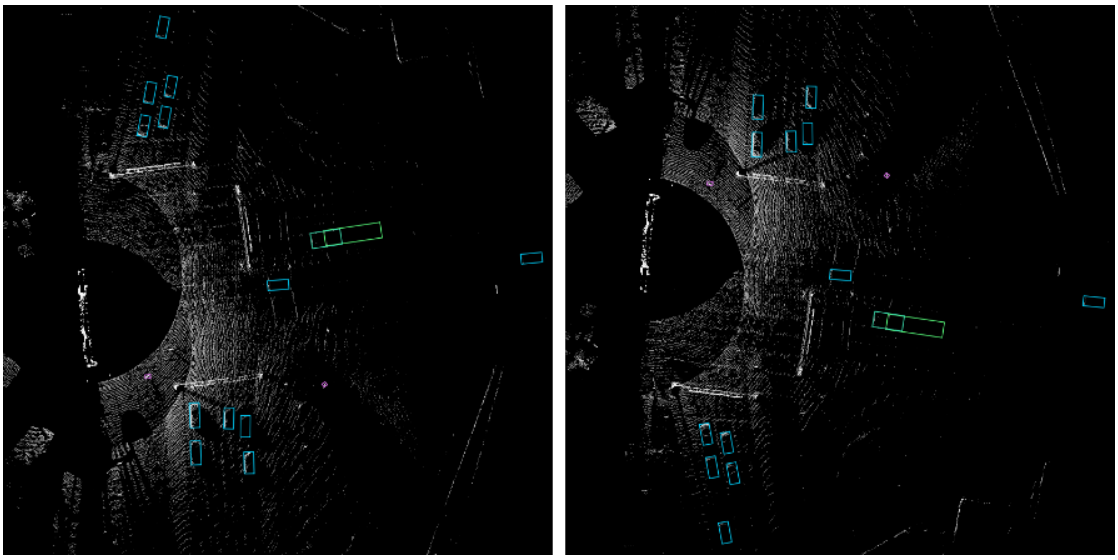


Figure 4.6: A comparison of bird's eye view images of cloud points for *Random Flip 3D* method. Left image represent the original cloud points whereas the right image represents the horizontally flipped version.

Like augmentation methods for images, the following cloud-point-based augmentation methods are also modified to accommodate temporal consistency:

- **Global Transformation (Rotation, Scale and Translation):** This method applies a global rotation, scale, and translation to the entire point cloud together with ground truth bounding boxes. An example can be seen in Figure 4.5.
- **Random Flip 3D:** This technique involves mirroring the point clouds when viewed from a bird's-eye perspective, thereby introducing variability in the object's viewpoints. If consistently applied across consecutive frames, the mirrored perspectives retain their temporal coherence. Nevertheless, in our experimentation, we restricted ourselves to horizontal flips. This restriction arises due to the adjustment in our pre-defined point cloud ranges, which makes vertical flips yield inconsistent augmentations. A visual example can be seen in Figure 4.7.
- **Points Shuffle:** This method randomizes the order of points in the cloud. While this doesn't change the cloud's inherent structure, it provides the model with a varied input structure, promoting robustness against different point orders.

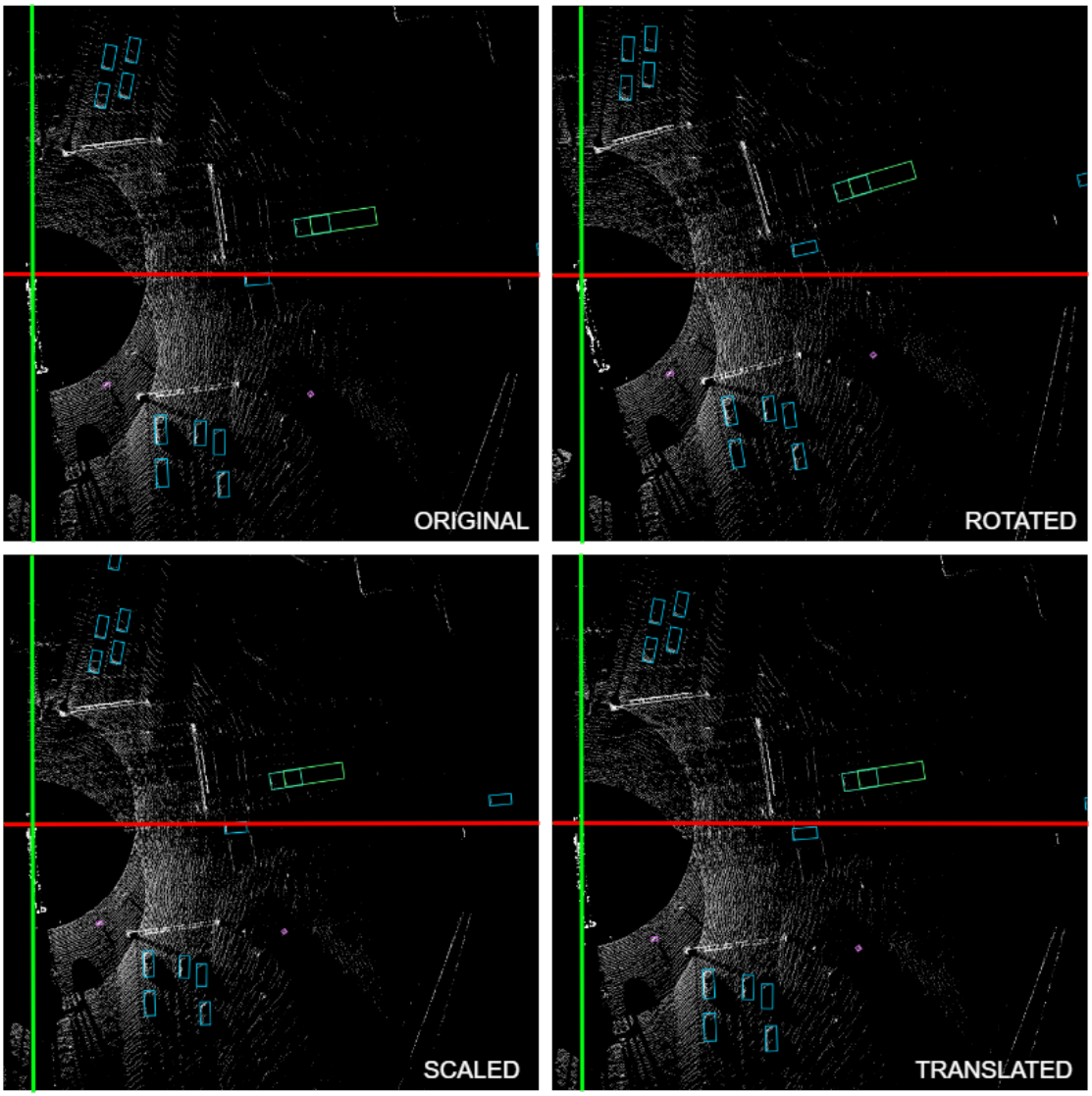


Figure 4.7: A comparison of bird's-eye view images of point clouds using the *Global Transformation (Rotation, Scale and Translation)* method. The upper-left image presents the original point cloud; the upper-right image presents its rotated version; the lower-left image presents the scaled version; and the lower-right image presents the translated version. Coordinate lines are colored red for the X-axis and green for the Y-axis to differentiate effects.

- **Ground Truth Paste:** This method, adapted from [YML18], initializes a database that encompasses labels for all ground truths along with their corresponding point cloud data, sourced from the 3D bounding boxes of these truths within the training dataset. During the training process, specific ground truths are randomly selected from this database and combined into the contemporary training point cloud via concatenation. This method successfully increases the number of objects in each scene, thus simulating objects across a wide array of situations. To guarantee the realism of the results, a post-sampling collision check is performed. Following this check, any objects found in conflict are promptly discarded. An example showcase can be seen in Figure 4.9. Regarding the temporal dimension, we have introduced a variant, namely, "Temporally-Aware Ground Truth Paste Data Augmentation." A more comprehensive discussion on this is available in Section 4.3.

4.2.5 Online Caching

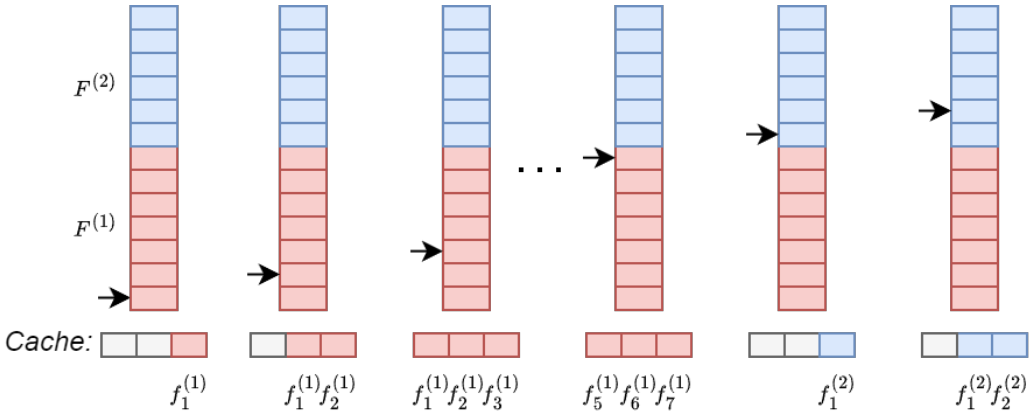


Figure 4.8: Visualization of iterative online caching of frames' feature tensors in validation or test pipeline. Black arrows represent the current iteration step while pointing to corresponding frame and its feature tensor. Each block represents a feature tensor of a given frame denoted by $f_i^{(k)} \in F^{(k)}$ where each F represents the set of frames in a temporal sequence.

In this section we introduce an online caching mechanism for the validation and testing phases of a temporal deep learning model. The rationale behind this mechanism lies in its potential to mitigate the need for recurrent data loading and feature computation, which is both time-consuming and computationally intensive. To facilitate this, a cache with a capacity of Q (see 4.2.3) was established.

The operation of this cache follows a systematic protocol: during each evaluation iteration, the cache accumulates data frames' feature tensors, growing incrementally by each single frame. As soon as the iteration arrives at the conclusion of a sequence, the cache undergoes a reset. Following this, the cache then commences its accumulation of feature tensors from the succeeding sequence. This operational cycle persists until the entirety of the validation or test dataset has been processed. You can see the general work-flow illustrated in Figure 4.8.

There are tangible advantages to this methodology. It offers a marked reduction in the computational overhead. Given that there's no requisite to perpetually load sequential frames afresh and compute their feature tensors, the model can instead utilize the prior feature tensors already present in the cache. This translates to computational savings, and thus faster inference, as demonstrated in Table 5.27.

4.3 Temporally-Aware Ground Truth Paste Data Augmentation

In the previous section, specifically in 4.2.4, we referenced the "Ground Truth Paste" data augmentation method, adapted from [YML18], as an effective technique to augment the number of ground truths within each scene. However, this method has certain limitations. The augmentation procedure is entirely stochastic, neglecting the temporal dimension. Such oversight can generate unrealistic scenarios. To rectify this limitation, we introduce a modified version, namely, Temporally-Aware Ground Truth Paste Data Augmentation, *TA-GTP* for short. This modified method integrates temporal awareness into the augmentation, ensuring that the ground truths selected are temporally consistent, maintaining iterative translation and rotation transformations across sequences. The methodology is detailed as follows:

1. Construct a run-time database encompassing labels for all ground truths and their corresponding point cloud data. This data is derived from the 3D bounding boxes of the ground truths within the training dataset and is automatically generated prior to the training phase.

2. Initiate a sampling algorithm where each class within the training dataset receives several pre-defined configuration parameters as outlined below:

   - *Maximum Sample Count*: Denoted by $A_N^{(c)}$, it is the maximum number of objects that can be present together with sampled ones for a given class $c \in C$. For example, for the class "car," if the $A_N^{(car)}$ is 7 and there are already 5 cars in the frame, then only 2 cars can be sampled and added from the database.
   - *Rotation Values*: They can be defined as:

     $$A_R^{(c)} = [\mu^{(c)}, \sigma^{(c)}] \tag{4.22}$$

     where: $A_R^{(c)}$ = the acceptable rotation parameters for objects of class c
     $\mu^{(c)}$ = the mean rotation value (radian) for class c
     $\sigma^{(c)}$ = the standard deviation of rotation value (radian) for class c

     For any given temporal sequence s, the sampled rotation value $r_s^{(c)}$ for class c can now be expressed using a Gaussian distribution:

     $$r_s^{(c)} \sim \mathcal{N}(\mu^{(c)}, \sigma^{(c)^2}) \tag{4.23}$$

     Where $\mathcal{N}(\mu, \sigma^2)$ denotes a Gaussian random sampling with mean μ and variance σ^2. $r_s^{(c)}$ specifies the change in orientation of the object on z-axis, basically changing its yaw.

- *Translation Values*: They can be defined as:

$$A_T^{(c)} = [t_l^{(c)}, t_u^{(c)}] \tag{4.24}$$

where: $A_T^{(c)}$ = the range of translation parameters for objects of class c
$t_l^{(c)}$ = the minimal translation value (meters) for class c
$t_u^{(c)}$ = the maximal translation value (meters) for class c

For any given temporal sequence s the sampled translation value $t_s^{(c)}$ for class c can be expressed as:

$$t_s^{(c)} \sim U(t_l^{(c)}, t_u^{(c)}) \tag{4.25}$$

Where $U(a, b)$ denotes a uniform random sampling between values a and b. $t_s^{(c)}$ specifies the displacement of the object in meters along the x-axis (front of the bounding box where yaw is $\pi/2$).

3. For every temporal sequence s used as input, a random quantity of objects B is extracted from the database for its initial frame f_1, adhering to the constraints outlined earlier, especially the *maximum sample count*. Formally, the set of extracted objects for sequence s can be symbolized as B_s.

   Within a temporal sequence s, which may consist of multiple frames $f_1, f_2, ...f_n$, each frame f_i will undergo a uniform movement of the sampled objects. This results from copying and pasting objects from the initial frame to the subsequent frames, followed by transformations using the sampled rotation and translation values. This ensures consistency across all frames to maintain temporal coherence.

   The transformation for an object o transitioning from frame f_i to frame f_{i+1} can be defined as:

$$T_{f_i \to f_{i+1}}(o, r_s^{(c)}, t_s^{(c)}) \to o'_{f_{i+1}} \tag{4.26}$$

   where: $T_{f_i \to f_{i+1}}$ = the transformation from frame f_i to frame f_{i+1}
   $o'_{f_{i+1}}$ = the position of object o in frame f_{i+1} after transformation

   This frame-to-frame transformation guarantees that during each transition within sequence s, objects exhibit fluid and consistent movement, further fortifying the temporal coherence in the augmented dataset. An exemplification of this iterative process can be observed in Figure 4.10.

4. The frame-to-frame transformation, as described in the previous step, is reiterated for all temporal sequences within the training steps.

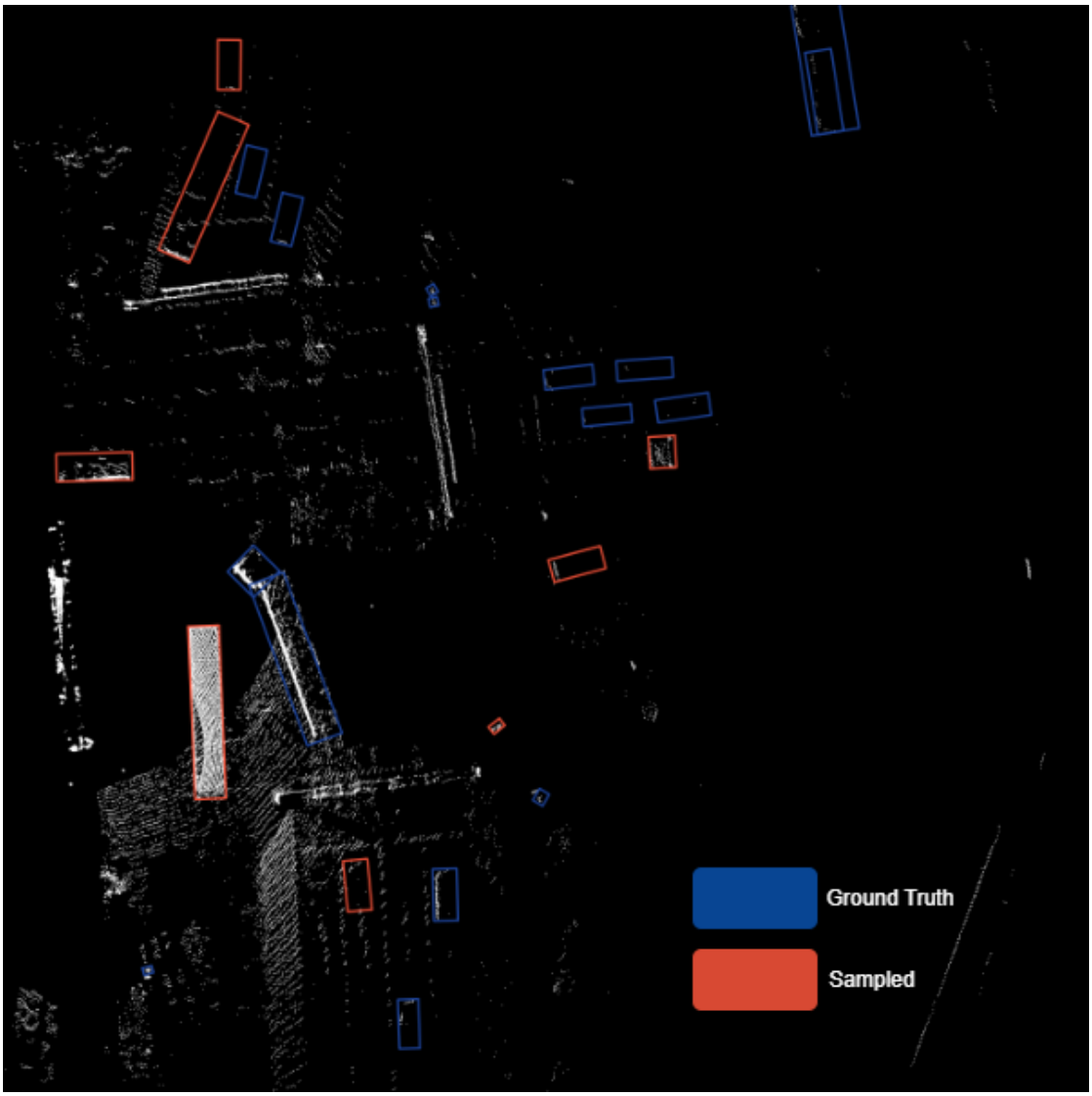


Figure 4.9: A comparison of ground truth objects and objects randomly sampled via the **Temporally-Aware Ground Truth Paste** (TA-GTP) augmentation method, as depicted in bird's-eye view image derived from point clouds. Objects delineated by blue bounding boxes indicate the ground truth, while those with red boxes indicate the sampled objects. Sampled objects are fully identical to their original ground truth attributes such as position, rotation, dimensions and class.

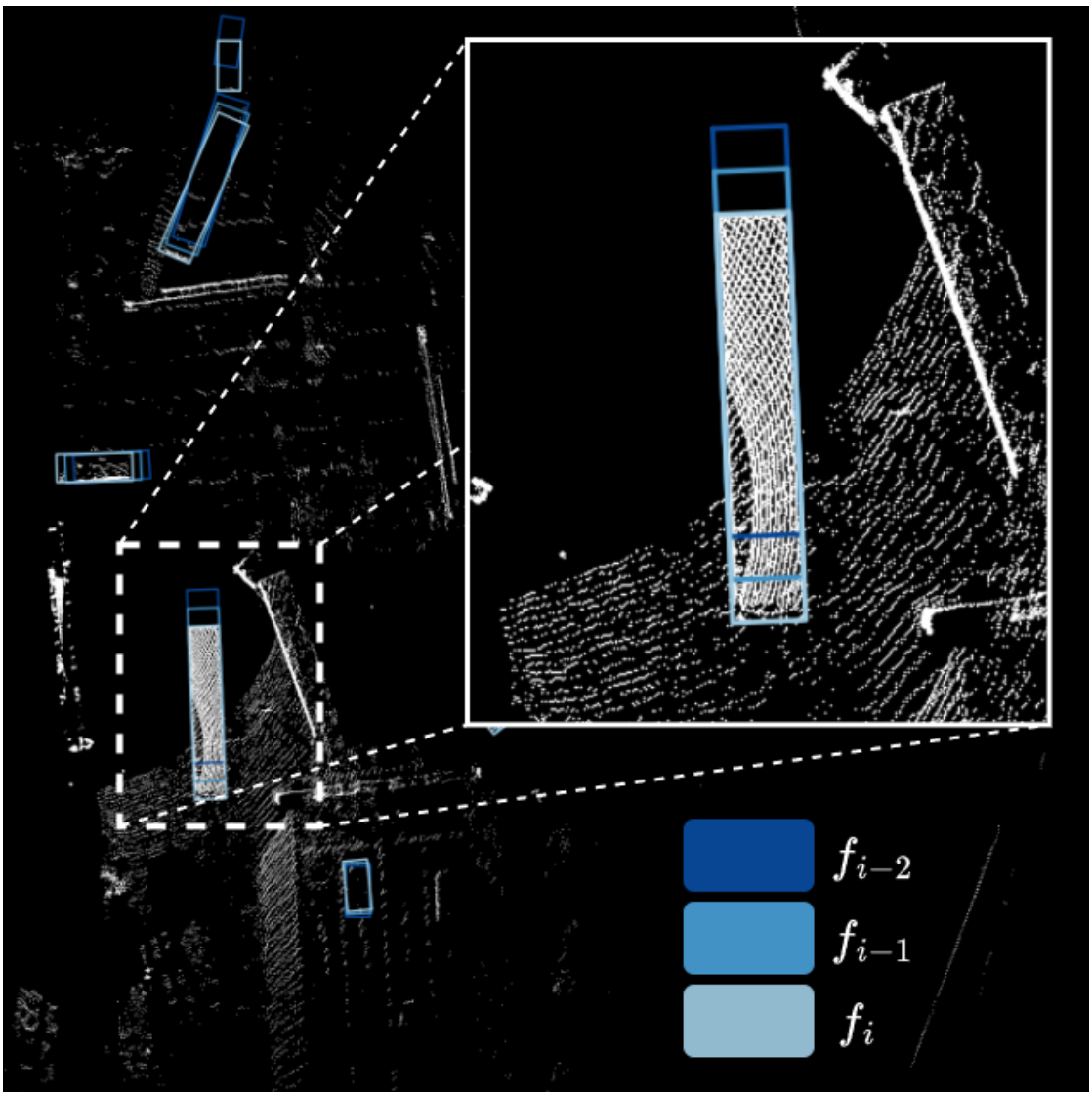


Figure 4.10: A visualization of objects sampled via the **Temporally-Aware Ground Truth Paste** (TA-GTP) augmentation technique is presented, showcasing their historical positions and orientations in bird's-eye view image derived from point clouds. Varied shades of blue represent distinct temporal frames, with darker shades signifying earlier frames. The object inside the focus panel is classified as "Bus" in TUMTraf-i [Zim+23b].

4.4 Temporal Fusion Networks

Convolutional LSTM (Long Short-Term Memory) and Convolutional GRU (Gated Recurrent Unit) networks, which are the extensions of their fully-connected counterparts, represent recurrent architectures tailored for spatial-temporal data. They have convolutional structures in both the input-to-state and state-to-state transitions so that they can encode spatial information as well [Shi+15]. In our case, we integrate these networks to our temporal pipeline since the spatio-temporal behaviour is quite beneficial for fusing temporal features in bird's eye view representations (see Figure 4.11). We shall refer to these networks as the *Temporal Fuser* networks, abbreviated as *TF*.

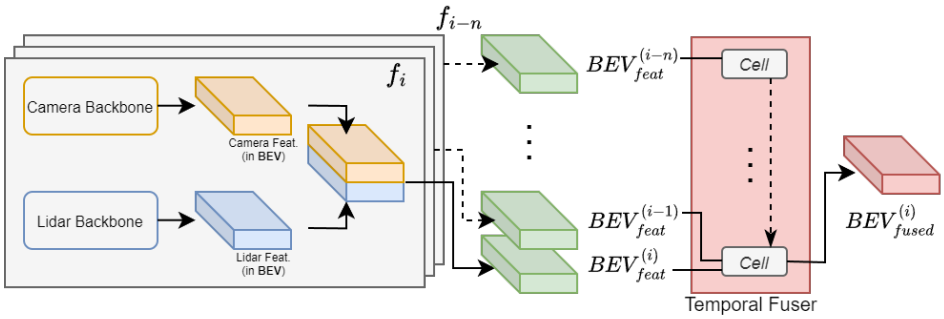


Figure 4.11: An overview of temporal fusion in the detection pipeline. Both the camera and LiDAR backbones produce features in a bird's eye view from their respective inputs. These features are concatenated and represented as BEV_{feat} in the figure. For each frame, denoted by f, in the temporal sequence, features are generated. Beginning with the initial concatenated features $BEV_{feat}^{(i-n)}$ and progressing to the final one $BEV_{feat}^{(i)}$, consecutive pairs of these features are fed into the Temporal Fuser for fusion across the temporal dimension.

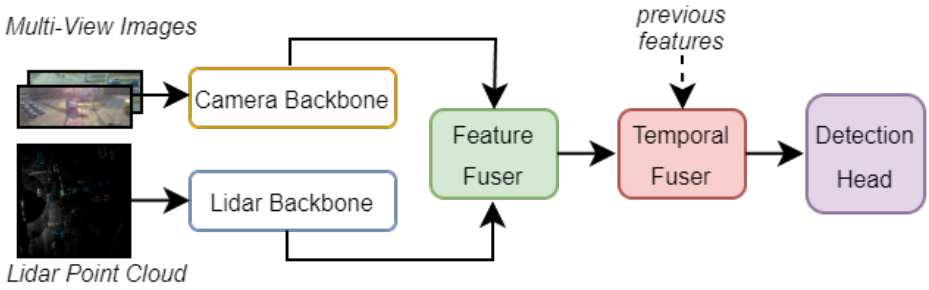


Figure 4.12: Flowchart of integration of Temporal Fuser (TF) on BEVFusion model. Feature Fuser represent an operation that concatenates features of both backbones. Such concatenated features are then given to Temporal Fuser together with previous frames' concatenated features. As a result, temporally fused features are generated and fed into the Detection Head.

4.4.1 Convolutional LSTM

In numerous studies, Long Short-Term Memory (LSTM), a specific Recurrent Neural Network (RNN) architecture, has demonstrated its proficiency in handling long-range dependencies in sequence modeling tasks [HS97] [Gra14].

LSTM's unique feature is its memory cell (or cell state), denoted as C_t, which serves as a repository for state information. This memory cell's interaction, storage, and erasure are managed by three primary gates: input, forget, and output. When a new input is presented, it is integrated into the cell based on the activation of the input gate. Simultaneously, if the forget gate, f_t, is activated, the previous cell state, C_{t-1}, can be discarded. The decision to convey the current cell state, C_t, to the final hidden state, H_t, is governed by the output gate, o_t. Convolutional LSTM introduces convolutional layers to this architecture and can be expressed with the following equations:

$$i_t = \sigma(W_{xi} * X_t + W_{hi} * H_{t-1} + W_{ci} \circ C_{t-1} + b_i) \tag{4.27}$$
$$f_t = \sigma(W_{xf} * X_t + W_{hf} * H_{t-1} + W_{cf} \circ C_{t-1} + b_f) \tag{4.28}$$
$$C_t = f_t \circ C_{t-1} + i_t \circ \tanh(W_{xc} * X_t + W_{hc} * H_{t-1} + b_c) \tag{4.29}$$
$$o_t = \sigma(W_{xo} * X_t + W_{ho} * H_{t-1} + W_{co} \circ C_t + b_o) \tag{4.30}$$
$$H_t = o_t \circ \tanh(C_t) \tag{4.31}$$

where: X_t = the input at timestamp t
i_t = the input gate at timestamp t
f_t = the forget gate at timestamp t
C_t = the cell output at timestamp t
o_t = the output gate at timestamp t
H_t = the hidden state at timestamp t
W = the corresponding weight matrices

4.4.2 Convolutional GRU

The Gated Recurrent Unit (GRU), another type of Recurrent Neural Network (RNN) architecture, has been successfully employed in a variety of sequence modeling tasks due to its ability to retain information over extended sequences while being computationally less demanding compared to LSTM [Chu+14] [FZL16] [Tod+17].

Distinct from the LSTM, GRU simplifies the gating mechanism by utilizing two gates: reset and update gates. These gates determine the extent to which past information is forgotten and how the new input is incorporated into the hidden state. Specifically, the reset gate, r_t, decides how to combine the new input with the previous hidden state, while the update gate, z_t, determines how much of the previous hidden state is retained. The Convolutional GRU adapts the GRU architecture by integrating convolutional layers, which can be described with the following equations:

$$z_t = \sigma(W_{xz} * X_t + W_{hz} * H_{t-1} + b_z) \tag{4.32}$$
$$r_t = \sigma(W_{xr} * X_t + W_{hr} * H_{t-1} + b_r) \tag{4.33}$$
$$\hat{H}_t = \tanh(W_{xh} * X_t + r_t \circ (W_{hh} * H_{t-1}) + b_h) \tag{4.34}$$
$$H_t = (1 - z_t) \circ H_{t-1} + z_t \circ \hat{H}_t \tag{4.35}$$

where: X_t = the input at timestamp t
z_t = the update gate at timestamp t
r_t = the reset gate at timestamp t
$\hat{H}_t$ = the candidate hidden state at timestamp t
H_t = the hidden state at timestamp t
W = the corresponding weight matrices

This architecture allows Convolutional GRU to efficiently capture spatial-temporal dynamics in data while reducing the number of trainable parameters by around 33% when compared with Convolutional LSTM.

5 Evaluation

5.1 Experiment Setup

5.1.1 Configurations

In order to evaluate and compare the performance of the proposed methods with baselines, experiments were mostly standardized under the following configurations:

General Configuration

- **Image Size**: For both TUMTraf-i [Zim+23b] and OSDaR23 [Tag+23], the original images are down-sampled into 256x704 pixels originating from the center.
- **Voxel and Grid Sizes**:
  - For TUMTraf-i dataset, voxel dimensions are

    $$V_x = 0.1, \quad V_y = 0.1, \quad V_z = 0.2$$

    with a consistent grid size of 1600 for both training and testing. Point cloud ranges are

    $$R_x = [-20, 140], \quad R_y = [-80.80], \quad R_z = [-10, 0]$$

    The reason why we shifted the range along the X axis is due to lack of coverage, specifically for the cameras, behind the gantry. Also, we wanted to test whether the models can detect objects are 100 meters away.
  - For OSDaR23 dataset, the voxel dimensions are

    $$V_x = 0.16, \quad V_y = 0.16, \quad V_z = 0.4$$

    with a consistent grid size of 1600 for both training and testing. Point cloud ranges are

    $$R_x = [-6, 250], \quad R_y = [-128.128], \quad R_z = [-3, 13].$$

- **Camera-based Augmentations:** Generally applied augmentations are Image Augmentation in 3D, Image Grid Mask and Image Normalization.
- **LiDAR-based Augmentations:** Generally applied augmentations are Global Transformation (Rotation, Scale and Translation), Random Flip 3D (only horizontally) and Points Shuffle.
- **Hardware**: For the TUMTraf-i dataset, the models are trained on dual *RTX 3090 GPUs*. In contrast, models for the OSDaR23 dataset are trained on a single *RTX 4090 GPU*.

Camera-only Configuration

- **Backbone**: The Swin-T [Liu+21] is adopted as the primary backbone for camera-based models. The FPN [Lin+17] is used to fuse multi-scale camera features, producing a feature map at 1/8 the size of the input.
- **Transformation**: Camera-to-Bird's Eye View (BEV) transformation is achieved using the LSS method [PF20].
- **Detection Head**: The CenterPoint's detection head [YZK21a] is used.
- **Optimizer**: The AdamW optimizer [LH19] is with a learning rate of 6.0e-5 and a weight decay factor of 0.01. Cyclic policy [Smi17] is utilized as a learning rate scheduler.
- **Training Procedure:** These trainings are mostly configured to last 20 epochs.

LiDAR-only Configuration

- **Backbone**: The VoxelNet [YML18] is employed as the backbone for LiDAR-based models.
- **Detection Head**: The TransFusion's detection head [Bai+22] is used.
- **Optimizer**: The AdamW optimizer [LH19] is utilized with a learning rate of 6.0e-5 and a weight decay factor of 0.01. Cyclic policy [Smi17] is utilized as a learning rate scheduler.
- **Training Procedure:** These trainings are mostly configured to last 20 epochs.

Multi-modal (Camera + LiDAR) Configuration

- **Backbones**: Both the Swin-T (from the camera configuration) and VoxelNet (from the LiDAR configuration) are integrated.
- **Detection Head**: The TransFusion's detection head [Bai+22].
- **Optimizer**: The AdamW optimizer [LH19] is utilized with a learning rate of 6.0e-5 and a weight decay factor of 0.01. Cosine annealing [LH17] is utilized as a learning rate scheduler.
- **Training Procedure:** These trainings are mostly configured to last 4 epochs.

Temporal Configuration

- **Queue Length and Queue Gap**: As defined in 4.2.3, the most of the temporal experiments with some exceptions in ablation studies are configured with queue length of 3 and queue gap of 1.
- **Temporal Fuser**: Temporal experiments use Convolutional LSTM (defined in 4.4.1) as a temporal fuser network, with some exceptions of Convolutional GRU (defined in 4.4.2) in ablation studies. These networks are set to use 1 layer only.
- **Optimizer**: The AdamW optimizer [LH19] is utilized with a learning rate of 6.0e-5 and a weight decay factor of 0.01. Cosine annealing [LH17] is utilized as a learning rate scheduler.
- **Training Procedure:** We utilize transfer-learning manually by loading checkpoints from already trained non-temporal camera-only or LiDAR-only models, and then freeze their backbone weights prior to training. These trainings are mostly configured to last 4 epochs.

By employing the aforementioned configurations for the experiments, we aimed to ensure consistency and rigor in the evaluation process, thereby providing a comprehensive understanding of the system's performance across different configurations and datasets.

5.1.2 Temporal Dataset Splits

As discussed in 4.1, our dataset splits had to be carefully generated in order to be compatible with temporal aspects of our methods and techniques. Therefore, we utilized our *Temporal Dataset Split Search* algorithm to find the optimal temporal splits for both of our datasets, namely, TUMTraf-i [Zim+23b] and OSDaR23 [Tag+23].

TUMTraf-I Dataset

The TUMTraf-I dataset contains four sequences with a combined total of 2,400 LiDAR frames. These frames are divided into pseudo-sequences of 20 frames each, denoted as N_f. From these pseudo-sequences, 100,000 permutations, denoted as N_p, were generated through shuffling. We allocated 80% of the frames to the training set and 10% each to the validation and test sets. Thus, the algorithm partitioned each permutation of pseudo-sequences according to these ratios. Upon analysis, we excluded the "Other" class from the search because of its minimal presence in the dataset, affecting temporal sequence partitioning.

For threshold parameters, we set T_c to 1, T_d to 0.95, and T_l to 0.95 to ensure strict category equivalence. However, we chose T_o to be 0.75 because objects in the "mostly occluded" category were scarcely found and concentrated in specific sequences, preventing differentiation in most permutations. As for the weights (as defined in 4.2), we selected the following:

$$
\begin{aligned}
w_{Car} &= \{w_c = 50, w_d = 20, w_l = 10, w_o = 5\} \\
w_{Truck} &= \{w_c = 120, w_d = 48, w_l = 24, w_o = 12\} \\
w_{Trailer} &= \{w_c = 120, w_d = 48, w_l = 24, w_o = 12\} \\
w_{Bus} &= \{w_c = 50, w_d = 20, w_l = 10, w_o = 5\} \\
w_{Van} &= \{w_c = 50, w_d = 20, w_l = 10, w_o = 5\} \\
w_{Bicycle} &= \{w_c = 150, w_d = 60, w_l = 30, w_o = 15\} \\
w_{Motorcycle} &= \{w_c = 150, w_d = 60, w_l = 30, w_o = 15\} \\
w_{Pedestrian} &= \{w_c = 200, w_d = 80, w_l = 40, w_o = 20\} \\
w_{Emergency} &= \{w_c = 50, w_d = 20, w_l = 10, w_o = 5\}
\end{aligned}
\tag{5.1}
$$

We prioritized the weights for vulnerable traffic elements like bicycles, motorcycles, and pedestrians by increasing their overall weight values. The greatest emphasis was placed on the weights corresponding to the number of classes (W_c) and distance categories (W_d).

The algorithm used 6 processes (denoted as N_w) to search through 100,000 permutations, identifying 19 valid ones. The optimal permutation with the lowest cost was selected. Correspondingly, 1,920 frames (96 pseudo-sequences) were allocated for training, 240 frames (12 pseudo-sequences) for validation, and 240 frames (12 pseudo-sequences) for testing. The distributions of the classes across the allocated sets can be observed in Table 5.1.

class	train	val	test
Car	16807 (61.2%)	1991 (55.9%)	2055 (57.0%)
Truck	2002 (7.3%)	253 (7.1%)	306 (8.5%)
Trailer	2163 (7.9%)	285 (8.0%)	339 (9.4%)
Bus	670 (2.4%)	59 (1.7%)	92 (2.6%)
Van	2840 (10.3%)	478 (13.4%)	406 (11.3%)
Bicycle	460 (1.7%)	103 (2.9%)	100 (2.8%)
Motorcycle	526 (1.9%)	130 (3.6%)	78 (2.2%)
Pedestrian	2013 (7.3%)	265 (7.4%)	229 (6.4%)
Total	27481 (79.3%)	3564 (10.3%)	3605 (10.4%)

Table 5.1: Comparison of the number of class objects across the sets of TUMTraf-i. The given percentages for each column correspond to their respective column, except for the last row where the percentages correspond to its row.

The following tables include the statistics about the number of objects with their corresponding attributes for each split.

split set	easy	moderate	hard
train	12516 (41.4%)	10174 (33.7%)	7512 (24.9%)
val	1601 (39.0%)	1343 (32.7%)	1166 (28.4%)
test	1674 (42.5%)	1257 (31.9%)	1010 (25.6%)

Table 5.2: Comparison of the number of objects and their corresponding embedded difficulty levels across the divided sets of TUMTraf-i. The given percentages correspond to their respective rows.

split set	$d < 40m$	$40m \leq d < 50m$	$50m \leq d$
train	12516 (46.3%)	10174 (37.6%)	4333 (16.0%)
val	1601 (45.4%)	1343 (38.0%)	586 (16.6%)
test	1674 (47.5%)	1257 (35.6%)	596 (16.9%)

Table 5.3: Comparison of the number of objects and their corresponding categorized distances across the divided sets of TUMTraf-i. The provided percentages correspond to their respective rows.

split set	$l \leq 20$	$20 < l \leq 50$	$50 < l$
train	14103 (53.0%)	6023 (22.6%)	6495 (24.4%)
val	1854 (54.5%)	858 (25.2%)	690 (20.3%)
test	1560 (46.8%)	926 (27.8%)	844 (25.3%)

Table 5.4: Comparison of the number of objects and their corresponding categorized number of points inside their bounding boxes across the divided sets of TUMTraf-i. The provided percentages correspond to their respective rows.

split set	not occluded	partially occluded	mostly occluded	unknown
train	10730 (71.9%)	2858 (19.1%)	211 (1.4%)	1127 (7.6%)
val	1677 (67.1%)	408 (16.3%)	48 (1.9%)	366 (14.6%)
test	1358 (62.1%)	443 (20.2%)	19 (0.9%)	368 (16.8%)

Table 5.5: Comparison of the number of objects and their corresponding categorized occlusion levels across the divided sets of TUMTraf-i. The provided percentages correspond to their respective rows.

OSDaR23 Dataset

The OSDaR23 dataset contains 45 sequences with a combined total of 1,424 frames. These frames are divided into pseudo-sequences of 25 frames each, denoted as N_f, with any residual frames being integrated into other or forming their own pseudo sequences with smaller sizes. From these pseudo-sequences, 100,000 permutations, denoted as N_p, were generated through shuffling. We allocated 80% of the frames to the training set and 20% to the validation set. Thus, the algorithm partitioned each permutation of pseudo-sequences according to these ratios. Upon analysis, we excluded several classes from the search – namely "signal", "train", "animal", "switch", "bicycle", "crowd,"

"wagons," and "signal bridge" – due to either extremely low frequency or technical issues present in the version of the dataset with which we engaged such as wrongly defined 3D bounding boxes.

For threshold parameters, we set T_c to 1, T_d to 0.9, T_l to 0.9 and T_o to 0.8. As for the weights (as defined in 4.2), we selected the following:

$$\begin{aligned} w_{Person} &= \{w_c = 500, w_d = 30, w_l = 10, w_o = 50\} \\ w_{Signal} &= \{w_c = 250, w_d = 15, w_l = 5, w_o = 25\} \\ w_{CatenaryPole} &= \{w_c = 250, w_d = 15, w_l = 5, w_o = 25\} \\ w_{SignalPole} &= \{w_c = 250, w_d = 15, w_l = 5, w_o = 25\} \\ w_{RoadVehicle} &= \{w_c = 250, w_d = 15, w_l = 5, w_o = 25\} \\ w_{BufferStop} &= \{w_c = 250, w_d = 15, w_l = 5, w_o = 25\} \end{aligned} \tag{5.2}$$

We prioritized the weights for pedestrians by increasing their overall weight values since they are the most crucial elements for detection in this domain. The greatest emphasis was placed on the weight corresponding to the number of classes (W_c).

The algorithm used 6 processes (denoted as N_w) to search through 100,000 permutations, identifying 144 valid ones. The optimal permutation with the lowest cost was selected. Correspondingly, 1,143 frames were allocated for training and 281 frames for validation. The distributions of the classes across the allocated sets can be observed in Table 5.6.

class	train	val
Person	9568 (66.9%)	2839 (66.5%)
Catenary Pole	2381 (16.6%)	728 (17.0%)
Signal Pole	969 (6.7%)	256 (6.0%)
Road Vehicle	823 (5.7%)	243 (5.6%)
Buffer Stop	545 (3.8%)	200 (4.6%)
Total	14286 (77.1%)	4266 (22.9%)

Table 5.6: Comparison of the number of class objects across the divided sets of OsDAR23. The given percentages for each column correspond to their respective column, except for the last row where the percentages correspond to its row.

The following tables include the statistics about the number of objects with their corresponding attributes for each split. It should be noted that the "buffer stop" class of objects is excluded from these tables due to a problem in logging.

split set	$d < 50m$	$50m \leq d < 100m$	$100m \leq d < 150m$	$150m \leq d < 200m$	$200m \leq d$
train	9712 (59.1%)	3324 (20.2%)	1343 (8.2%)	956 (5.8%)	1086 (6.6%)
val	2711 (56.3%)	946 (19.6%)	527 (10.9%)	300 (6.2%)	334 (6.9%)

Table 5.7: Comparison of the number of objects and their corresponding categorized distances across the divided sets of OSDaR23. The provided percentages correspond to their respective rows.

split set	$l \leq 199$	$200 \leq l \leq 499$	$500 \leq l \leq 999$	$1000 \leq l \leq 1999$	$2000 \leq l \leq 2999$	$3000 \leq l$
train	8802 (53.6%)	4053 (24.7%)	2646 (16.1%)	677 (4.1%)	157 (1%)	86 (0.5%)
val	2617 (54.3%)	1202 (24.9%)	797 (16.5%)	160 (3.3%)	21 (0.4%)	21 (0.4%)

Table 5.8: Comparison of the number of objects and their corresponding categorized number of points inside their bounding boxes across the divided sets of OSDaR23. The provided percentages correspond to their respective rows.

split set	0-25 %	25-50 %	50-75 %	75-99 %	100 %
train	11195 (68.2%)	1492 (9.1%)	1790 (10.9%)	1802 (11%)	142 (0.9%)
val	3233 (67.1%)	520 (10.8%)	534 (11.1%)	527 (10.9%)	4 (0.1%)

Table 5.9: Comparison of the number of objects and their corresponding categorized occlusion levels across the divided sets of OSDaR23. The provided percentages correspond to their respective rows.

5.1.3 Evaluation Metrics

We define the metrics for the detection task in the TUMTraf-i and OSDaR23 datasets as follows:

- **2D mean Average Precision (2D mAP)**: A variation of the Average Precision metric that considers the 2D center distance on the ground plane for matching predictions to ground truth objects, instead of using intersection over union (IoU). Matches are made with the nearest center-distance within specific thresholds. The AP is calculated by integrating the precision-recall curve for values greater than 0.1, then averaged over thresholds of 0.5, 1, 2, 4 meters and across classes. This metric is our main evaluation criterion as our models have encoder and decoder networks that work with features in bird's eye view representation which is 2D.
- **2D Intersection-over-Union (2D IoU)**: Measures the overlap between the predicted and ground truth bounding boxes, divided by their union area. It is a standard metric for comparing ground truth with predicted bounding box accuracy.
- **3D Intersection-over-Union (3D IoU)**: Extends the 2D IoU principle to three dimensions, considering the entire volume of the predicted and ground truth bounding boxes.
- **NuScenes Detection Score (NDS)**: Quantifies object detection effectiveness by averaging scores that are the product of an object's confidence score and its IoU with the true bounding box. Scores are adjusted according to the object's sensor range and dimensions, providing an evaluation that highlights real-world variables like proximity, size and orientation.
- **Video RAM (VRAM) Usage**: Quantifies the model's graphics memory consumption in megabytes (MB) during inference. This metric is crucial for evaluating the model's memory efficiency and its impact on the graphics processing unit's (GPU) resource management.
- **Frame-per-Second (FPS)**: Serves as a measure of the model's runtime inference speed, indicating the number of frames processed per second. This metric provides insight into the model's efficiency and suitability for real-time applications.

5.2 Hyper-parameter Tuning

Hyper-parameter tuning is a critical step in the development of machine learning models, particularly for models that utilize complex methods like the Temporally-Aware Ground Paste Augmentation (TA-GTP). The task of optimizing hyper-parameters goes beyond mere adjustment; it involves a strategic exploration of the hyper-parameter space, where the goal is to find a combination that can yield the highest 2D mean-average precision (2D mAP) score while avoiding over-fitting. By leveraging advanced methods like the Tree-structured Parzen Estimator (TPE) [Ber+11] [BYC13] [Yos+20] within the Optuna software framework [Aki+19], we not only systematically approach this optimization problem but also gain insights into the model's behavior under various configurations, which is invaluable for understanding the underlying mechanics of TA-GTP. Hence, the process of hyper-parameter tuning is not simply a step in model development but a cornerstone that supports the construction of robust, effective, and reliable machine learning systems. With that in mind, we applied the hyper-parameter search algorithm for two separate tasks for each dataset:

1. Finding the optimal values for **Maximum Sample Count** (as defined in 4.3) for each class in the dataset. This parameter signifies the upper boundary of objects that can be together with sampled objects for each class within the frame scene. It's important to note that this portion lacked any temporal implementation, and as such, training was carried out on the LiDAR-only baseline model paired with the non-temporal version of TA-GTP. The other training details are described in LiDAR-only configuration, 5.1.1.

2. Finding the optimal values for both **Rotation** (as defined in 4.22) and **Translation** (as defined in 4.24) parameters for each class in the dataset. These parameters decide the manner in which the sampled objects traverse the temporal sequences. For this search, we utilized the most optimal model from the prior search as a starting point. We then introduced a temporal fuser network, specifically the Convolutional LSTM (as described in Section 4.4.1), and followed the temporal configuration described in Section 5.1.1.

TUMTraf-i dataset

In the initial task, we conducted 20 trials, each following the configuration described in Section 5.1.1, and assessed the mean Average Precision (mAP) metric on the validation set. Within these assessments, the mAP values ranged from a minimum of **77.80%** to a maximum of **81.13%**. We determined the optimal Maximum Sample Count values for each class in the TUMTraf-i dataset. These values, denoted as A_N^{class}, are presented in the Table 5.10.

Class	A_N^{class} Value	Importance
Car	14	15%
Trailer	2	4%
Truck	3	6%
Van	4	21%
Pedestrian	0	12%
Bus	1	1%
Motorcycle	0	18%
Bicycle	1	18%
Emergency Vehicle	1	6%

Table 5.10: Optimal Maximum Sample Count values (as defined in 4.3) are presented alongside their respective importance percentages for each class in the TUMTraf-i dataset. These importance values, derived from the hyper-parameter search algorithm, signify the impact of the corresponding parameter on improving performance.

Based on the optimal Maximum Sample Count values from Table 5.10, distinct sampling choices are made for different classes. For instance, "Pedestrian" and "Motorcycle" classes are not sampled, and their importance percentages suggest it's crucial to avoid sampling them. On the other hand, the "Bus" class has a low sample count while having lowest importance value. This might suggest that larger objects like buses are often discarded during collision checks because of their size, thus making them less significant for sampling. The "Trailer" class shows a similar pattern. From this hyper-parameter optimization, we adopted these specific Maximum Sample Count values for all TUMTraf-i experiments that utilizes TA-GTP method.

In the second task, we conducted 25 trials, each following the configuration described in Section 5.1.1, and assessed the mean Average Precision (mAP) metric on the validation set. Within these assessments, the mAP ranged from a minimum of **81.76%** to a maximum of **82.90%**. We determined the optimal **Rotation** and **Translation** values for each class in the TUMTraf-i dataset. These values, represented as $\mu^{(class)}$, $\sigma^{(class)}$, $t_l^{(class)}$, and $t_u^{(class)}$ respectively, are provided in the table 5.11.

Class	$\mu^{(class)}$	$\sigma^{(class)}$	$t_l^{(class)}$	$t_u^{(class)}$
Car	0	0.0851	0	0.2114
Trailer	0	0.1919	0	2.061
Truck	0	0.1229	0	0.1922
Van	0	0.1880	0	0.1244
Pedestrian	-	-	-	-
Bus	0	0.1995	0	1.1107
Motorcycle	-	-	-	-
Bicycle	0	0.2163	0	0.5918
Emergency Vehicle	0	0.1328	0	0.6620

Table 5.11: Optimal **Rotation** and **Translation** values determined for each class in the TUMTraf-i dataset. The values are characterized by $\mu^{(class)}$ (rotation mean), $\sigma^{(class)}$ (rotation standard deviation), $t_l^{(class)}$ (lower translation threshold), and $t_u^{(class)}$ (upper translation threshold). The Pedestrian and Motorcycle classes have missing values due to their exclusion from sampling, as determined by prior hyper-parameter analysis shown in Table 5.10.

Based on the optimal Rotation and Translation values from Table 5.11, certain classes exhibit limited translation freedom compared to others. For example, the "Car" class can only move a maximum of 0.2114 meters from frame to frame. Given that the "Car" class predominates the dataset and often remains stationary awaiting green lights, the algorithm infers limited movement for this class. In contrast, the "Trailer" and "Bus" classes, with their larger and longer dimensions, exhibit higher translation values. This suggests that the algorithm likely shifts them to enhance the effectiveness of data augmentation, possibly because their size and dimensions may limit their sampling rate, leading to being discarded due to the collision check mechanism. Regarding the rotation values, we anticipated minimal fluctuations because objects typically do not undergo sharp rotations between sequential frames; thus, they appear marginally small and consistent with one another.

OSDaR23 dataset

As for initial task designated for the OSDar23 dataset, the general search space was smaller due to the reduced number of selected classes. We conducted 15 trials, each following the configuration described in Section 5.1.1, and assessed the mean Average Precision (mAP) metric on the validation set. Within these assessments, the mAP values ranged from a minimum of **75.36%** to a maximum of **79.73%**. We determined the optimal **Maximum Sample Count** values for each class in the OSDaR23 dataset. These values, denoted as A_N^{class}, are presented in Table 5.12.

Class	A_N^{class} Value	Importance
Person	10	64%
Catenary Pole	6	18%
Signal Pole	2	8%
Road Vehicle	3	9%
Buffer Stop	0	1%

Table 5.12: Optimal Maximum Sample Count values (as defined in 4.3) are presented alongside their respective importance percentages for each class in the OSDaR23 dataset. These importance values, derived from the search algorithm, signify the impact of the corresponding parameter on improving performance.

Based on the optimal Maximum Sample Count values from Table 5.12, the algorithm determines the "Person" class should be sampled preferentially, followed by "Catenary Pole" objects. The imperative of accurately detecting individuals within the railway domain is necessary for ensuring safety, and the algorithm's prioritization of the "Person" class for sampling aligns well with this safety-critical objective. Given the scarcity of "Buffer Stop" objects in the dataset, constituting only 3.8%, the algorithm recommends against sampling them, as they are likely to be over-represented in concentrated areas. In contrast, the algorithm recommends sampling "Signal Pole" and "Road Vehicle" classes, which make up 6.7% and 5.7% of the dataset, respectively, even though their representation is comparable to that of "Buffer Stop" objects. From this hyper-parameter optimization, we adopted theses specific Maximum Sample Count values for all OSDaR23 experiments that utilizes TA-GTP method.

In the second task, we conducted 20 trials, each following the configuration described in Section 5.1.1, and assessed the mean Average Precision (mAP) metric on the validation set. Within these assessments, the mAP ranged from a minimum of **80.80%** to a maximum of **82.13%**. We determined the optimal **Rotation** and **Translation** values for each class in the OSDaR23 dataset. These values, represented as $\mu^{(class)}$, $\sigma^{(class)}$, $t_l^{(class)}$, and $t_u^{(class)}$ respectively, are provided in the table below:

Class	$\mu^{(class)}$	$\sigma^{(class)}$	$t_l^{(class)}$	$t_u^{(class)}$
Person	0	0.1672	0	2.3655
Catenary Pole	0	0.0649	0	0.8370
Signal Pole	0	0.0671	0	3.3953
Road Vehicle	0	0.0709	0	0.8333
Buffer Stop	-	-	-	-

Table 5.13: Optimal **Rotation** and **Translation** values determined for each class in the OSDaR23 dataset. The values are characterized by $\mu^{(class)}$ (rotation mean), $\sigma^{(class)}$ (rotation standard deviation), $t_l^{(class)}$ (lower translation threshold), and $t_u^{(class)}$ (upper translation threshold). The Buffer Stop class has missing values due to its exclusion from sampling, as determined by prior hyperparameter analysis shown in Table 5.12.

Based on the optimal Rotation and Translation values from Table 5.13, the algorithm determined that the "Catenary Pole" and "Signal Pole" classes necessitate minimal rotation for their sampled objects. This implies that their predominantly cylindrical pole-like 3D structures remain largely unchanged by rotation, at least for the major part of their shape. As for the translation values, notable disparities can be seen amongst the classes. A subset of sequences in OSDaR23 dataset have slight ego positional shifts as a consequence of the train's modulation in velocity, either through deceleration or acceleration. This could indicate that the elevated translation values in specific classes are attributable to these variations.

5.3 Quantitative Studies

In this section, we did an extensive quantitative analysis on model performances including the baseline model, described in Section 4.2.1, as well as our own temporal models for both datasets: TUMTraf-i [Zim+23b] and OSDaR23 [Tag+23]. This analysis presents an general overview of the results, detailing both class-specific outcomes and findings across various distance categories.

5.3.1 General Results

TUMTraf-i

	modality	2D mAP ↑	2D IoU ↑	3D IoU ↑	NDS ↑
baseline	C	54.42	87.00	66.88	51.48
w/ TF	C	**55.26**	**87.09**	**68.22**	**51.88**
baseline	L	81.84	85.86	71.72	66.75
w/ TF	L	85.73	85.97	71.60	68.87
w/ TF + TA-GTP	L	**86.91**	**86.19**	**71.90**	**69.59**
baseline	C+L	85.14	86.14	71.40	68.57
w/ TF	C+L	**86.73**	**86.17**	**71.79**	**69.48**

Table 5.14: Quantitative evaluation of different model combinations across multiple modalities using the TUMTraf-i test split. The "Baseline" denotes the model proposed by [Liu+22], which has been adapted for the TUMTraf-i configuration. Here, "TF" represents Temporal Fuser, highlighting the temporal component, and "TA-GTP" refers to Temporally-Aware Ground Truth Paste Augmentation.

The overview of experiments on the TUMTraf-i dataset are presented in Table 5.14, demonstrating that both the Temporal Fuser (TF) and Temporally-Aware Ground Truth Paste (TA-GTP) methods significantly enhance performance across all modalities. In camera-only and multi-modal scenarios, TF exhibits performance gains in all metrics, while LiDAR-only stands out as the optimal modality, achieving the highest metrics scores, especially at 2D mAP – our primary evaluation criterion – reaching a high score of 86.91. On top of that, applying the TA-GTP method to LiDAR-only models demonstrates that they can perform comparably or even surpass multi-modal versions in terms of performance.

All of our models employ a bird's-eye view representation (as described in Section 2.2), which serves as the output from the encoder networks and the input for the

decoder networks; hence, performance gains in metrics designed for 3D space are less effective than their 2D equivalents. For example, 3D IoU does not exhibit marked enhancements in LiDAR-based models, whereas 2D IoU clearly shows improvements in 2D bounding box delineation within the bird's eye view space, especially in the LiDAR-only models. It should be noted that 2D and 3D Intersection over Union metrics are calculated exclusively for matched bounding boxes. As a result, camera-only models exhibit higher 2D IoU values than other models due to their narrower detection margins. However, camera-only models produce lower values for some metrics when compared to those that incorporate LiDAR. This is attributed to the dual-camera setup in the TUMTraf-i dataset, which has a limited field of view (as detailed in Section 2.4). This restricted view is expected to lead to a number of undetected objects within its range, resulting in a marked decrease in performance.

OSDaR23

	modality	2D mAP ↑	2D IoU ↑	3D IoU ↑	NDS ↑
baseline	C	14.00	80.55	**30.12**	34.45
w/ TF	C	**18.83**	**85.15**	29.99	**38.20**
baseline	L	77.56	74.18	53.80	72.39
w/ TF	L	81.79	77.20	**59.67**	73.75
w/ TF + TA-GTP	L	**82.12**	**77.53**	59.47	**75.57**
baseline	C+L	79.54	77.13	60.81	73.19
w/ TF	C+L	**81.77**	**79.37**	**61.14**	**74.26**

Table 5.15: Quantitative evaluation of different model combinations across multiple modalities using the OSDaR23 validation split. The "Baseline" denotes the model proposed by [Liu+22], which has been adapted for the OSDaR23 configuration. Here, "TF" represents Temporal Fuser, highlighting the temporal component, and "TA-GTP" refers to Temporally-Aware Ground Truth Paste Augmentation.

In Table 5.15, we again observe that both the Temporal Fuser (TF) and Temporally-Aware Ground Truth Paste (TA-GTP) methods significantly enhance performance across all modalities. LiDAR-only and multi-modal models can be considered competitive when the Temporal Fuser (TF) is added, with scores of 81.79% and 81.77% in 2D mAP, respectively. However, the LiDAR-only model receives a significant performance boost with the application of the Temporally-Aware Ground Truth Paste (TA-GTP) method, achieving a score of 82.12%. On the other hand, the camera-only model suffers

immensely with the OSDaR23 dataset, as the dataset consists 43.7% of objects being at distance greater than 50 meters, resulting in severe lack of depth information.

5.3.2 Class-wise Results

TUMTraf-i

For our experiments on the TUMTraf-i dataset, we only utilized nine out of ten classes, detailed in Section 2.4, for both training and evaluation purposes. The 2D mean Average Precision (2D mAP) values for each class are presented in Table 5.16.

		2D mAP ↑								
	modality	Car	Trailer	Truck	Van	Pedestrian	Bus	Motorcycle	Bicycle	Emergency
baseline	C	**72.58**	**34.42**	52.41	**63.29**	**58.99**	**50.45**	94.47	50.10	13.04
w/ TF	C	72.49	34.14	**53.02**	60.82	56.88	50.07	**97.17**	**52.29**	**20.48**
baseline	L	89.59	81.02	83.33	83.43	83.46	95.40	86.50	67.94	65.85
w/ TF	L	90.15	**84.13**	**86.06**	81.56	88.03	**97.43**	**91.00**	81.62	71.54
w/ TF + TA-GTP	L	**90.59**	82.56	85.53	**83.19**	**90.78**	95.68	90.39	**83.76**	**79.67**
baseline	C+L	89.85	82.86	**85.47**	81.53	86.92	95.67	89.16	77.25	77.60
w/ TF	C+L	**89.93**	**84.32**	85.23	**83.00**	**90.48**	**96.89**	**89.85**	**81.19**	**79.70**

Table 5.16: Quantitative evaluation of different model combinations on labelled classes across multiple modalities using the TUMTraf-i test split. The "Baseline" denotes the model proposed by [Liu+22], which has been adapted for the TUMTraf-i configuration. Label "Emergency" is short for "Emergency Vehicle".

With Temporal Fuser (TF), camera-only models see a notable improvement in the "Motorcycle" and "Emergency" classes. This suggests that temporal information might be providing richer feature representations for these categories, despite a slight decline in performance for other classes like "Car" and "Van". On the other hand, modalities that include LiDAR, TF contributes to significant improvements in mAP scores across nearly all classes.

The addition of TA-GTP (only applied to the LiDAR-only modality) further enhances the 2D mAP scores for most classes. Notably, "emergency vehicle" class sees a significant increase, suggesting that incorporating TA-GTP is particularly beneficial for infrequent objects.

Overall, the results in Table 5.16 suggest that the inclusion of Temporal Fuser (TF) and Temporally-Aware Ground Truth Paste (TA-GTP) data augmentation significantly enhances object detection performance across different modalities, with LiDAR and combined modalities showing the most promise. Each class responds differently to the model enhancements, which may inform future research on modality-specific optimization in which drawbacks and strengths of sensors can be further observed.

The LiDAR-only and fusion modalities, combined with temporal aspect, appear to be a particularly effective strategies for improving detection in a diverse set of classes, showcasing the potential approaches in complex environments such as traffic scenarios. However, considering that fusion modality introduces additional computational overhead due to its camera structure, LiDAR-only modality is the clear winner here in terms of effectiveness in both performance and cost-effectiveness.

OSDaR23

		2D mAP ↑				
	modality	person	catenary pole	signal pole	road vehicle	buffer stop
baseline	C	28.76	0.01	4.66	20.06	16.53
w/ TF	C	**32.29**	**0.29**	**8.30**	**27.83**	**25.43**
baseline	L	79.99	90.33	75.63	59.57	82.26
w/ TF	L	85.56	**90.99**	**81.32**	65.85	85.20
w/ TF + TA-GTP	L	**86.94**	90.72	80.10	**67.39**	**85.46**
baseline	C+L	86.79	88.85	**73.36**	64.87	**83.83**
w/ TF	C+L	**87.25**	**91.57**	69.98	**66.40**	83.46

Table 5.17: Quantitative evaluation of different model combinations on labelled classes across multiple modalities using the OSDaR23 validation split. The "Baseline" denotes the model proposed by [Liu+22], which has been adapted for the OSDaR23 configuration.

The addition of Temporal Fuser (TF) to the baseline model consistently improves the 2D mean Average Precision (mAP) across all categories (person, catenary pole, signal pole, road vehicle, and buffer stop) and modalities. This indicates that TF significantly contributes to the model's detection capabilities. For the LiDAR modality, the introduction of Temporally-Aware Ground Truth Paste (TA-GTP) alongside TF further enhances the performance, achieving the highest 2D mAP for the person category and showing marginal improvements in road vehicle and buffer stop detection.

Overall, the OSDaR23 dataset results corroborate the findings from the TUMTraf-i dataset in that TF and TA-GTP can provide significant performance enhancements. In particular, the greatest benefits seem to be for dynamic objects such as "person" and "road vehicle" compared to other objects that are static in the world.

5.3.3 Object Distance Category Results

The primary objective of this thesis is to improve the detection of distant objects. Accordingly, this section consolidates experimental results from both datasets to assess the effectiveness of our methods across various distance categories.

TUMTraf-i

	number of ground truth objects									
Distance	Total	Car	Trailer	Truck	Van	Pedestrian	Bus	Motorcycle	Bicycle	Emergency
< 30	446	194	98	67	50	20	16	0	1	0
$30 - 50$	2403	1455	141	208	207	160	66	77	89	0
$50 - 75$	818	435	100	31	159	29	34	0	10	20
> 75	121	35	0	0	80	0	6	0	0	0

Table 5.18: Distribution of Ground Truth Objects by Distance Categories on TUMTraf-I test split.

The TUMTraf-i dataset, consisting nine classes, have varying class distributions across different distance categories, as its test split is detailed in Table 5.18. Cars are the most common object in all distance categories, with a particularly high presence in the "30 - 50" meter range. Some object categories such as "Motorcycle", "Bicycle", and "Emergency" are notably rare or even absent in certain distance ranges. There is a significant increase in the total number of objects detected in the "30 - 50" meter range whereas objects at distances greater than 75 meters are significantly less represented, which indicates the potential boundaries of sensors or fewer occurrences of objects at that range. Pedestrians are more frequently identified in the "30 - 50" meter range but have a much lower presence beyond that. This could reflect the challenges in detecting smaller objects at greater distances. With these in mind, it is likely that every distance category has unique characteristics from which metric values can reflect.

Table 5.19 presents a comprehensive evaluation of various model combinations when applied to the TUMTraf-i's test split, with a focus on aforementioned object distance categories and their performance across multiple modalities.

As the distance to objects increases, there is a corresponding decline in detection performance, a trend commonly attributed to the reduced size of objects and diminished feature information, such as point clouds, at extended ranges. Our temporal methods, namely the Temporal Fuser (TF) and the Temporally-Aware Ground Truth Paste (TA-GTP) augmentation, address this issue by incorporating temporal context, thereby improving performance across different modalities. Notably, in the "50-75" meter

distance category, LiDAR-only models exhibit an increase of 5 to 6 points in 2D mean Average Precision (2D mAP), a significant improvement given the presence of 818 detectable objects within this range.

Models incorporating temporal information through Temporal Fuser, typically exhibit enhanced performance across most metrics when compared to their baseline counterparts. In the distance category "<30" meter, values for the baseline of the camera modality (C) declines harshly upon integrating the Temporal Fuser. This decline may suggest that the camera baseline model predominantly targets objects proximal to the sensors, resulting in significantly compromised detection of distant objects, hence, over-fitting. In this case, however, TF helps generalization over various distances, as it shows performance increases, especially in "30-50" meter distance category where most of the objects lie in. Conversely, LiDAR-only and multi-modal models consistently exhibit performance improvements in all distance categories with the adoption of Temporal Fuser.

TA-GTP, applicable solely to LiDAR-based methods, yields optimal outcomes across all distance categories except beyond 75 meters. Within this range, standalone Temporal Fuser integration reduces 2D mAP performance by 3.42; however, the incorporation of TA-GTP narrows this deficit to a mere 0.7, rendering it competitive with the baseline.

Overall, the TUMTraf-i dataset reveals the intricacies of object detection at various distances, showcasing a diverse range of class distributions and highlighting the challenges in identifying objects at greater ranges. The analysis indicates a sharp decline in detection capabilities at distances beyond 75 meters, suggesting the limits of sensor capabilities or the natural scarcity of objects. The integration of temporal methods like Temporal Fuser (TF) and Temporally-Aware Ground Truth Paste (TA-GTP) has proven to be significantly effective, especially for LiDAR-only and multi-modal models, enhancing the detection performance and mitigating issues related to distance-based detection. While TF shows the potential to improve generalization over various distances, particularly evident in the "30-50" meter range, TA-GTP specifically enhances LiDAR-based detection, maintaining strong performance up to 75 meters. The adoption of temporal context in object detection models is a promising approach to improve accuracy and reliability across different sensor modalities and object distances within infrastructure environments like TUMTraf-i dataset.

Distance		modality	2D mAP ↑	2D IoU ↑	3D IoU ↑	NDS ↑
< 30	baseline	C	**24.30**	86.56	**45.20**	**30.67**
	w/ TF	C	12.58	**86.61**	41.60	23.32
	baseline	L	60.07	72.75	**74.00**	51.65
	w/ TF	L	62.69	73.10	73.71	53.28
	w/ TF + TA-GTP	L	**63.86**	**74.43**	73.82	**54.06**
	baseline	C+L	**62.49**	73.52	**73.82**	**53.19**
	w/ TF	C+L	61.98	**73.61**	73.01	52.93
30 - 50	baseline	C	71.43	85.99	67.49	61.25
	w/ TF	C	**73.74**	**86.49**	**69.25**	**62.58**
	baseline	L	87.23	86.41	74.09	70.28
	w/ TF	L	91.15	86.74	74.42	72.39
	w/ TF + TA-GTP	L	**91.72**	**86.70**	**74.66**	**72.74**
	baseline	C+L	89.26	**86.93**	73.99	71.39
	w/ TF	C+L	**91.49**	86.65	**74.29**	**72.65**
50 - 75	baseline	C	**31.20**	**88.90**	**56.12**	**36.56**
	w/ TF	C	26.75	88.84	56.00	33.83
	baseline	L	74.19	86.84	**68.70**	63.24
	w/ TF	L	79.66	86.84	66.82	66.49
	w/ TF + TA-GTP	L	**80.39**	**86.85**	67.80	**66.60**
	baseline	C+L	80.43	86.96	**68.18**	67.10
	w/ TF	C+L	**81.37**	**86.97**	67.77	**67.44**
> 75	baseline	C	23.73	87.65	46.37	30.42
	w/ TF	C	**23.96**	**88.55**	**49.70**	**30.91**
	baseline	L	**59.43**	**89.93**	**63.63**	**53.85**
	w/ TF	L	56.01	88.47	57.65	50.94
	w/ TF + TA-GTP	L	58.70	88.99	59.98	51.92
	baseline	C+L	52.95	89.56	60.31	49.13
	w/ TF	C+L	**53.41**	**89.91**	**61.12**	**49.51**

Table 5.19: Quantitative evaluation of different model combinations on object distance categories across multiple modalities using the TUMTraf-i test split. The "Baseline" denotes the model proposed by [Liu+22], which has been adapted for the TUMTraf-i configuration. Here, "TF" represents Temporal Fuser, highlighting the temporal component, and "TA-GTP" refers to Temporally-Aware Ground Truth Paste Augmentation.

OSDaR23

	number of ground truth objects					
Distance	Total	Person	Catenary Pole	Signal Pole	Road Vehicle	Buffer Stop
< 50	2537	2302	100	70	65	0
$50 - 100$	750	382	179	58	73	58
$100 - 150$	413	70	221	70	0	52
$150 - 200$	266	60	166	20	20	0
> 200	300	25	62	38	85	90

Table 5.20: Distribution of Ground Truth Objects by Distance Categories on OSDaR23 validation split.

In Table 5.20, which details the distribution of ground truth objects by distance categories in OSDaR23 validation split, we observe that the majority of the objects (2537) are situated within a 50-meter range. Within this proximity, 'Person' objects constitute the most significant proportion, with 2302 individual objects, underscoring the importance of accurately detecting pedestrians is crucial for overall performance. As the distance increases, the frequency of 'Person' detections decreases, with the highest distance range (>200 meters) showcasing a notable diversity in the object types, including a rise in the number of 'Buffer Stop' objects to 90 instances, which is higher than in any other category. This spread of objects suggests varying detection challenges at different distances. The OSDaR23 dataset, to-be-utilized in train-based applications, necessitates the detection of distant objects to enable timely preventive actions, given the significant challenge of abrupt stopping due to the considerable weight and momentum of trains.

In Table 5.21, we see the same pattern we saw in the TUMTraf-i case. The performance generally diminishes with increasing distance. The models with Temporal Fuser (TF) consistently outperform the baseline across almost all metrics and distance categories, emphasizing the effectiveness of incorporating temporal information into object detection models. In particular, the combination of Temporal Fuser and Temporally-Aware Ground Truth Paste Augmentation (TA-GTP) demonstrates superior results in the LiDAR-only modality (L), especially in the "50-100" meter and ">200" meter categories, reflected in almost all the metrics. Overall, the multi-modal (C+L) model shows the best performance when it comes to the detection far away objects. These enhancements suggest that temporal context and advanced data augmentation in which sampled objects traverse with temporal context in mind can significantly bolster model robustness, particularly in challenging scenarios involving more distant objects.

Distance		modality	2D mAP ↑	2D IoU ↑	3D IoU ↑	NDS ↑
< 50	baseline	C	20.20	55.62	43.11	29.68
	w/ TF	C	**24.03**	**56.80**	**46.59**	**35.78**
	baseline	L	73.91	62.57	44.68	69.68
	w/ TF	L	**88.29**	68.93	49.93	73.47
	w/ TF + TA-GTP	L	88.08	**70.76**	**50.85**	**76.42**
	baseline	C+L	81.37	68.43	51.56	71.37
	w/ TF	C+L	**86.68**	**73.14**	**54.20**	**75.18**
50 - 100	baseline	C	38.85	84.39	38.73	**47.99**
	w/ TF	C	**39.34**	**89.17**	**41.71**	47.58
	baseline	L	85.05	81.63	60.42	74.27
	w/ TF	L	86.54	82.54	65.15	**75.09**
	w/ TF + TA-GTP	L	**87.05**	**83.13**	**67.13**	74.96
	baseline	C+L	86.25	82.66	66.95	74.22
	w/ TF	C+L	**87.05**	**86.22**	**68.48**	**74.70**
100 - 150	baseline	C	0.48	60.54	32.99	**22.35**
	w/ TF	C	**0.53**	**61.08**	**34.44**	21.27
	baseline	L	73.87	59.33	46.11	71.07
	w/ TF	L	74.90	61.81	**51.68**	66.80
	w/ TF + TA-GTP	L	**75.88**	**62.88**	51.56	**71.46**
	baseline	C+L	**74.02**	61.52	**51.41**	67.71
	w/ TF	C+L	72.63	**62.72**	51.33	**70.20**
150 - 200	baseline	C	0.00	0.00	0.00	0.00
	w/ TF	C	**0.05**	**22.03**	**78.08**	**7.60**
	baseline	L	49.75	41.03	52.70	**50.66**
	w/ TF	L	51.04	42.29	51.00	45.90
	w/ TF + TA-GTP	L	**52.08**	**43.28**	**55.50**	50.21
	baseline	C+L	50.56	**42.32**	**55.47**	43.99
	w/ TF	C+L	**54.65**	41.25	52.24	**50.88**
> 200	baseline	C	0.00	0.00	0.00	0.00
	w/ TF	C	**4.86**	**76.96**	**13.78**	**29.10**
	baseline	L	78.40	80.76	59.58	73.62
	w/ TF	L	**79.73**	85.63	**64.30**	**74.56**
	w/ TF + TA-GTP	L	78.96	**86.02**	63.07	74.29
	baseline	C+L	76.32	**85.53**	62.75	72.58
	w/ TF	C+L	**80.68**	85.36	**63.26**	**74.67**

Table 5.21: Quantitative evaluation of different model combinations on object distance categories across multiple modalities using the OSDaR23 validation split.

5.4 Ablation Studies

In this section, we show the results of our several ablation studies to see the impact of our methods on performance using TUMTraf-i dataset.

5.4.1 Temporally-Aware Ground Truth Paste Augmentation

The one of the main contributions of our thesis is the Temporally-Aware Ground Truth Paste (TA-GTP) augmentation method, which is highly configurable. Therefore, we conducted an ablation study to assess the impact of incremental additions of these components on our models' performance.

Consistent	Translation	Rotation	2D mAP ↑	2D IoU ↑	3D IoU ↑	NDS ↑
-	-	-	86.15	86.02	71.21	69.02
✓	-	-	86.56	86.04	71.06	69.32
✓	✓	-	86.85	86.08	71.58	69.51
✓	✓	✓	**86.91**	**86.19**	**71.90**	**69.59**

Table 5.22: The results of an ablation study examining the impact of various components of the Temporally-Aware Ground Truth Paste (TA-GTP) method on the performance of the LiDAR-only model using TUMTraf-i test split. The study incrementally introduces consistency enforcement, translation and rotation techniques.

The base temporal model, with none of the TA-GTP components applied, establishes the benchmark with a 2D mean Average Precision (mAP) of 86.15%, a 2D Intersection over Union (IoU) of 86.02%, a 3D IoU of 71.21%, and a NDS of 69.02%. The feature component "Consistent" ensures that once objects are sampled in the initial frame, they are retained across subsequent frames. Thus, temporal frames do not individually resample, but instead, utilize the initially sampled objects. When this feature is introduced, there is a slight increase in 2D mAP to 86.56% and NDS to 69.32%, albeit with a marginal decrease in 3D IoU, suggesting that while enforcing consistency can slightly enhance overall detection precision, it may not significantly affect spatial overlap in three dimensions.

Subsequent integration of translation component into the augmentation method sees further improvements in both 2D mAP (86.85%) and NDS (69.51%), alongside a more pronounced increase in 3D IoU (71.58%), indicating the importance of applying translation transformation for sampled objects. Rotation component appears to be a critical factor, as evidenced by the best recorded in all the metrics with 2D mAP of 86.91%, 2D IoU of 81.19%, 3D IoU of 71.90% and NDS of 69.59%. These enhancements

suggest that applying rotation transformation to sampled objects significantly aids in the model's ability to generalize over various object orientations and positions.

This ablation study reveals that each component of the TA-GTP method contributes positively to the model's performance. Table 5.22 effectively highlights how the incremental inclusion of these components results in gradual improvements in key performance indicators, thereby confirming the efficacy of the TA-GTP method in enhancing LiDAR-only object detection task.

5.4.2 Temporal Loading

Queue Length (Q)	Queue Gap (G)	2D mAP ↑	2D IoU ↑	3D IoU ↑	NDS ↑
2	1	86.46	85.80	71.45	69.30
3	1	86.91	**86.19**	**71.90**	69.59
4	1	**87.36**	86.10	71.18	**69.92**
3	2	86.70	85.99	71.28	69.54
3	1	86.91	**86.19**	**71.90**	69.59
3	0	**87.30**	86.06	71.47	**69.86**

Table 5.23: Comparative analysis of model performance metrics with varying queue lengths (Q) and queue gaps (G) (both defined in 4.2.3) on TUMTraf-i test split.

From the Table 5.23, it is evident that as the queue length (Q) increases from 2 to 4 with a constant queue gap (G) of 1, there is a notable improvement in the 2D mAP, peaking at 87.36% for a queue length of 4. This suggests that a longer queue length may contribute to a more accurate object detection, possibly by providing the model with more temporal context. However, the 2D IoU and 3D reach their highest at a queue length of 3, indicating an optimal balance at this queue length for the intersection-over-union accuracy between the predicted and actual object boundaries.

The NDS, which is a comprehensive metric incorporating several aspects of detection performance, shows improvement as the queue length increases, with the highest score observed at a queue length of 4. This suggests that overall detection performance, taking into account factors such as proximity, size and orientation benefits from a longer temporal queue.

When the queue gap (G) is varied with a constant queue length of 3, the performance does not exhibit a consistent trend. The highest 2D mAP and NDS are observed at the extremes of the gap range (G=0 and G=1), with the scores being 87.30% and 69.86% at G=0, respectively. This implies that utilizing queue gap as a generalization technique

enhances overall detection performance; however, it does not yield a corresponding increase in the accuracy of the predicted bounding boxes, which exhibit random deterioration due to this generalization strategy.

Overall, the table indicates a complex relationship between queue length, queue gap, and model performance. The optimal queue length and gap seem to be dependent on the specific performance metric under consideration. The queue length of 3 with a gap of 1 is a configuration that consistently results in high performance across most metrics, suggesting it as a potential sweet spot for balanced model performance in both 2D and 3D contexts. The nuances observed in this ablation study are critical for possible fine-tuning the temporal loading aspect of the model to achieve optimal performance on the TUMTraf-i.

5.4.3 Cloud Point Sparsity

In this ablation study, we manipulated the point cloud density by reducing the number of points from the original TUMTraf-i data. Specifically, we considered three different densities: the original density (same), a density reduced to half (halved), and a density reduced to a quarter (quartered). This reduction was achieved by systematically sub-sampling the original point cloud data, ensuring that the remaining points were evenly distributed across the volume to maintain a representative subset of the original cloud. As a result, the average count of point cloud data within the TUMTraf-i test split decreased from 60,996 to 30,255 in the reduced by half version, and further diminished to 14,642 in the quarter-size version.

As expected, the performance metrics indicate a trend where a decrease in point cloud density leads to a loss in performance across all evaluated metrics (see Table 5.24). For instance, when the density is halved, there is a noticeable drop in 2D mAP from 85.76% (same density) to 77.45% (halved), while the 3D IoU decreases marginally from 71.68% to 71.04%. This trend is even more pronounced when the point cloud density is quartered, with 2D mAP plummeting to 51.62% and 3D IoU to 62.62%. The differing levels of decrease between 2D mAP and 3D IoU suggest that as the point cloud becomes sparser, it's more difficult for the models to accurately figure out the height of the objects they're mapping compared to how well they can measure the width and depth.

Despite the significant reduction in point cloud density, incorporating temporal version of Temporally-Aware Ground Truth Paste (TA-GTP) augmentation consistently contributes to performance improvements in the models. This phenomenon is evident in the comparison of metrics at each density level. For instance, even when the density is halved, the inclusion of temporal information results in an increase in all the metrics, and a similar pattern with higher gains holds for the quartered density. These

Cloud Point Density	TA-GTP	2D mAP ↑	2D IoU ↑	3D IoU ↑	NDS ↑
same	non-temporal	85.76	85.98	71.68	68.88
same	temporal	**86.91**	**86.19**	**71.90**	**69.59**
halved	non-temporal	77.45	86.32	71.04	64.12
halved	temporal	**77.98**	**86.36**	**71.05**	**64.35**
quartered	non-temporal	51.62	82.47	62.62	48.58
quartered	temporal	**52.85**	**82.61**	**63.93**	**49.38**

Table 5.24: Performance metrics for varying point cloud densities with and without the temporal version of TA-GTP using the TUMTraf-i dataset split. The non-temporal TA-GTP algorithm samples objects independently at each frame, while the temporal version maintains object consistency across sequential frames by initially sampling at the first frame and subsequently adjusting the object trajectories through the sequence using calculated rotation and translation values.

gains, although modest in the context of reduced point cloud density, underscore the importance of temporal dynamics in the models' ability to interpret and analyze the scene. Temporal cues may provide critical information that aids in the continuity and coherence of object detection and tracking, compensating to some extent for the loss of spatial details due to sparser point clouds.

5.4.4 Temporal Fuser Networks

Temporal Fuser	modality	2D mAP ↑	2D IoU ↑	3D IoU ↑	NDS ↑
ConvLSTM	L	**86.91**	**86.19**	**71.90**	**69.59**
ConvGRU	L	86.73	86.16	71.77	69.56
ConvLSTM	C+L	**86.73**	86.17	71.79	**69.48**
ConvGRU	C+L	86.27	**86.34**	**72.55**	69.26

Table 5.25: Performance comparison of Temporal Fuser Networks using Convolutional LSTM and Convolutional GRU models across LiDAR-only and multi-modal modalities using TUMTraf-i test split. Corresponding models are with Temporally-Aware Ground Truth Paste (TA-GTP) augmentation method.

In Table 5.25, we can observe that Convolutional LSTM model outperforms the Convolutional GRU in the LiDAR-only modality, which indicates a stronger performance in terms of both accuracy and consistency of object detection in 2D and 3D spaces. However, when combining Camera and LiDAR modalities (C+L), the results are more nuanced. The Convolutional LSTM still shows superiority in our main metric, 2D mAP, suggesting better overall performance in a multi-modal scenario. Nevertheless, the Convolutional GRU model exhibits a marginal improvement in 2D IoU and a notable increase in 3D IoU over Convolutional LSTM in the combined modality. It should be noted that the Convolutional GRU possesses approximately 33% fewer trainable weights than the Convolutional LSTM, which accounts for the expected decline in performance. This decrement is offset by an increase in inference speed, further examined in Section 5.6.

5.5 Qualitative Studies

5.5.1 Features in Bird's Eye View

In our models, encoders and associated components – such as Camera-to-Bird's Eye View (BEV) transformations in camera-based models – yield feature tensors in BEV output. These tensors exhibit modality-specific variations stemming from the differing fields of view inherent to each sensor. This phenomenon is illustrated in both Figure 5.2 and 5.2, which present a comparison of feature tensors derived from LiDAR-only and camera-only models while demonstrating the corresponding limitations in fields of view.

5.5.2 Predictions

Figure 5.3 presents a visual representation of the model's predictions using the OSDaR23 validation split. This figure is significant as it highlights the model's ability to detect objects at varying distances. For instance, our model successfully detected all the labelled objects within the 200 to 250-meter range, demonstrating its potential for long-range detection. As for TUMTraf-i, Figure 5.4 demonstrates the capability of our model on detecting objects under challenging conditions, such as a dark-colored car at an approximate distance of 75 meters with sparse LiDAR point clouds.

Figure 5.5 further demonstrates the model's capabilities in visual comparison to a baseline using the TUMTraf-i test split. The comparison is structured in a two-column format: the left column displays baseline predictions, while the right column shows those from the TF + TA-GTP model. The top row offers a color-coded classification of predictions. The bottom row highlights occlusion issues, with ground truth labels in dark blue and predictions in red. Notably, our model predicts an additional bicycle, marked in purple, behind the bus. This bicycle, partially occluded and detected by only one of the two LiDAR sensors, illustrates our model's ability to handle partial occlusions and data sparsity.

Figure 5.6 continues the qualitative analysis, highlighting the model's proficiency in detecting occluded objects. It features two examples: a partially occluded bicycle and one at a marginal distance, both marked in purple. Despite the challenges posed by sparse LiDAR points on such targets, the TF + TA-GTP model's accurate predictions underscore its robustness in the infrastructural context of the TUMTraf-i dataset.

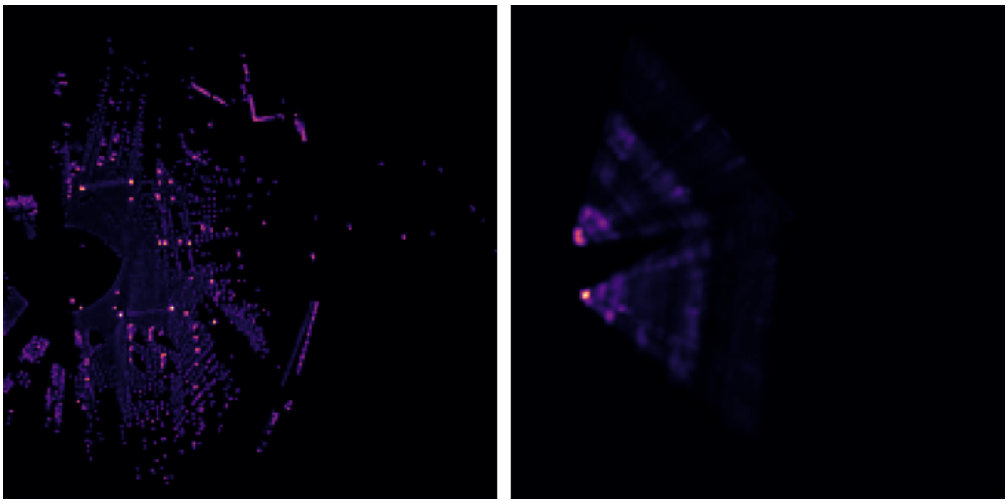


Figure 5.1: Feature visualization in Bird's-Eye View of a sample from TUMTraf-i test split. The left image showcases the feature output from a LiDAR-only temporal model, while the right image showcases the feature output of the same sample from a Camera-only temporal model. These visualizations were generated through normalization of tensor values followed by their summation.

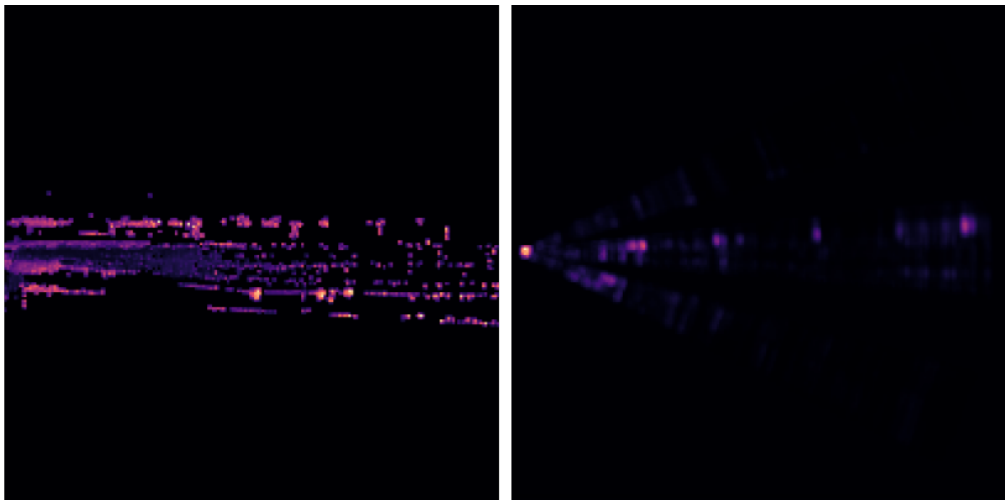


Figure 5.2: Feature visualization in Bird's-Eye View of a sample from OSDaR23 validation split. The left image showcases the feature output from a LiDAR-only temporal model, while the right image showcases the feature output of the same sample from a Camera-only temporal model. These visualizations were generated through normalization of tensor values followed by their summation.

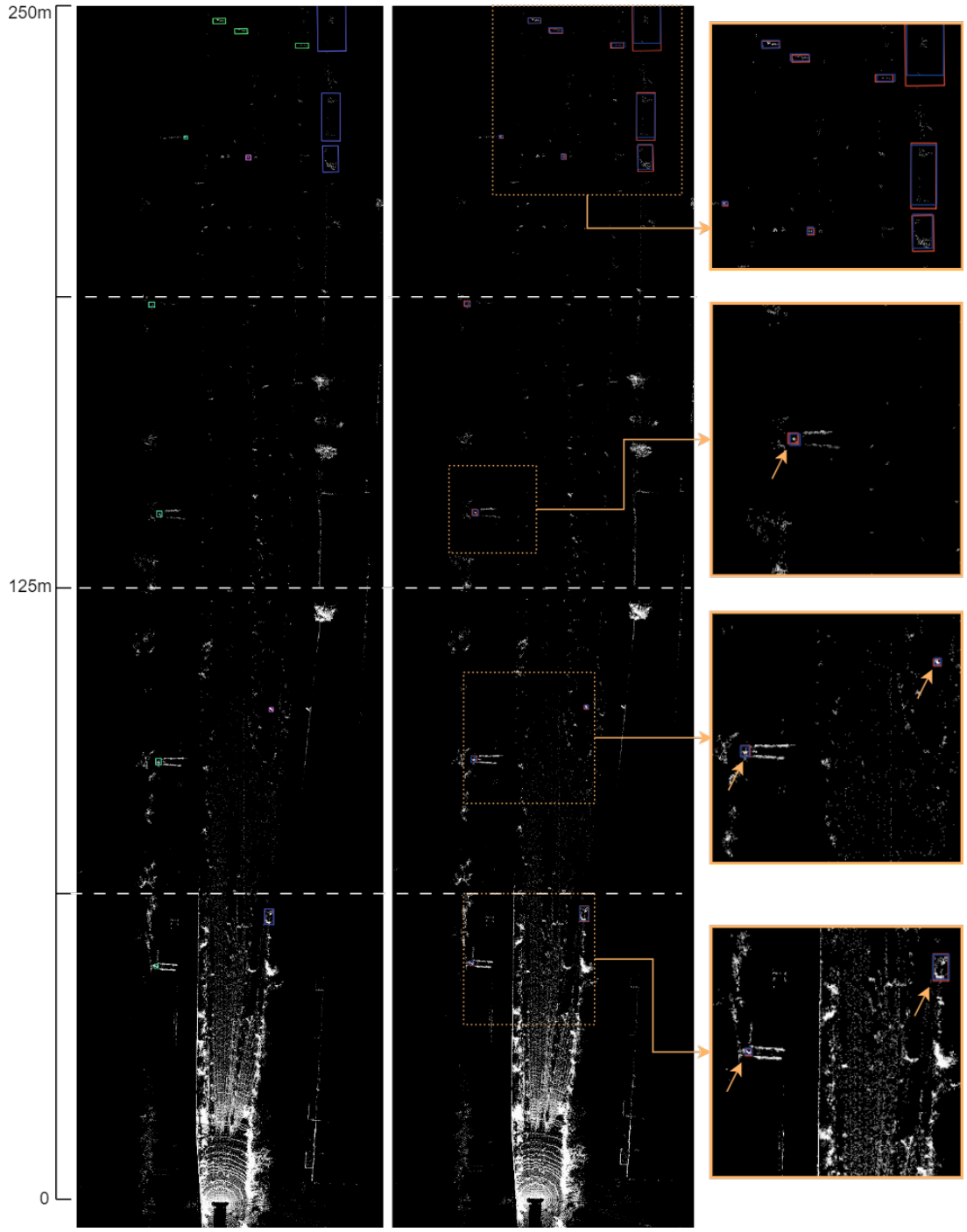


Figure 5.3: Predictions from our best LiDAR-only model (TF + TA-GTP) trained on OSDaR23 dataset. The left image shows predictions color-coded by class whereas the middle image shows ground truth labels in dark blue and predictions in red.

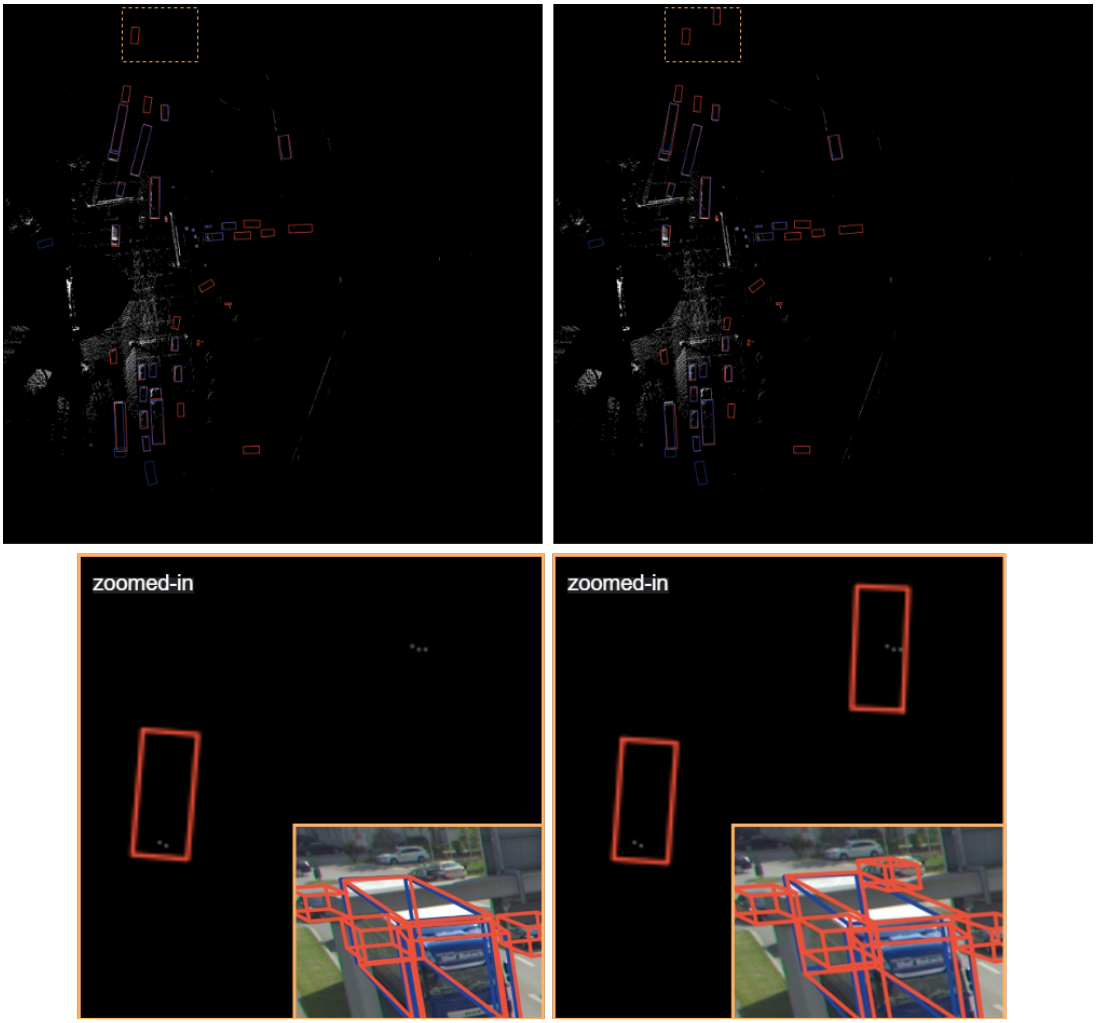


Figure 5.4: Comparison of predictions between the baseline and our best model (TF + TA-GTP) using LiDAR-only modality trained on TUMTraf-i dataset. The left column displays the predictions from the baseline whereas the right column displays the predictions from the best model. The upper images display annotated objects with ground truth labels in dark blue and predictions in red. The lower images present the camera's perspective alongside a magnified bird's eye view (BEV) of the areas outlined by orange dashed boxes in the upper images. Notably, the magnified images together with camera images illustrate that our model manages to predict a dark-colored car despite the sparse LiDAR point clouds at that range.

Figure 5.5: Comparison of predictions between the baseline and our best model (TF + TA-GTP) using LiDAR-only modality trained on TUMTraf-i dataset. The left column displays the predictions from the baseline whereas the right column displays the predictions from the best model. The top row displays images with predictions color-coded by class. The bottom row shows images annotated with ground truth labels in dark blue and predictions in red. The small image in the lower right corner is from the future frame. In these images, we can observe that our best model predicts a partially occluded bicycle (as only one of the LiDAR sensors produces sparse points on it) that is not labeled, highlighting the model's improved detection of objects with occlusion. All the images are cropped versions of the originals.

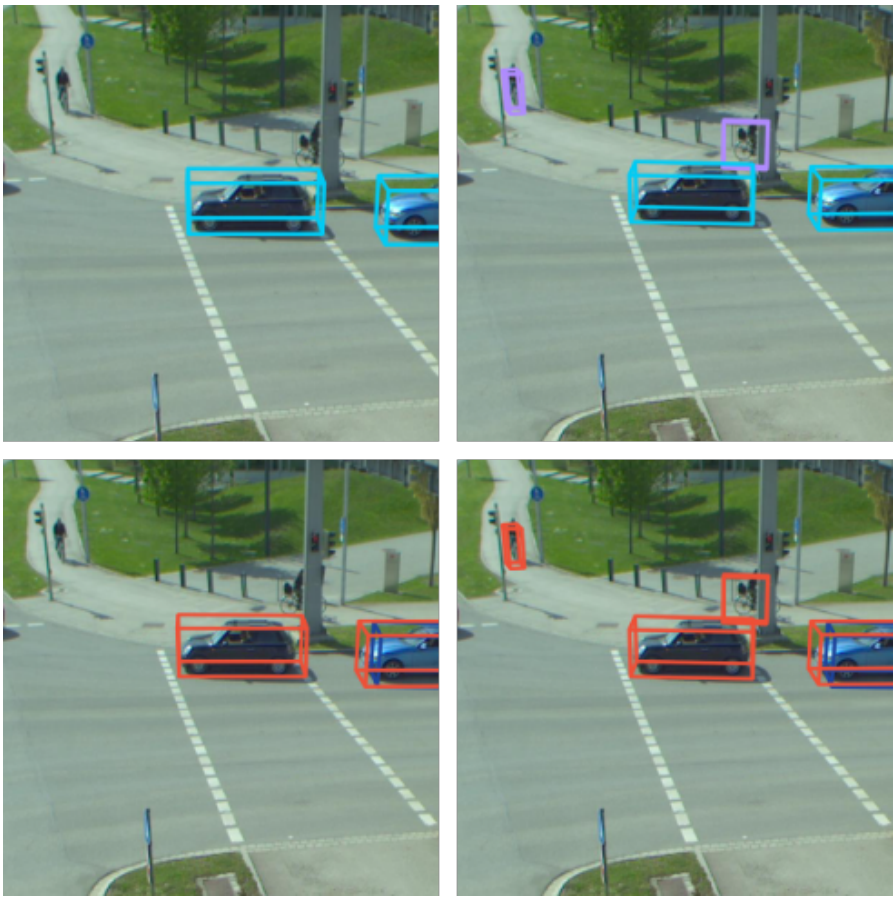


Figure 5.6: Comparison of predictions between our baseline and best model (TF + TA-GTP) using LiDAR-only modality trained on TUMTraf-i dataset. The left column displays the predictions from the baseline whereas the right column displays the predictions from the best model. The top row displays images with predictions color-coded by class. The bottom row shows images annotated with ground truth labels in dark blue and predictions in red. In these images, it is evident that our best model accurately predicts both a partially occluded bicycle and a bicycle at a marginal distance, despite the sparse LiDAR points typically associated with such objects. These images underscore the model's ability to generalize within the infrastructure domain like TUMTraf-i dataset. All the images are cropped versions of the originals.

5.6 Run-time Measurements

In the field of autonomous systems and computer vision, the efficiency and performance of prediction models are pivotal metrics that govern their practical deployment. The run-time measurements presented in Table 5.26 provides critical insight into the comparative performance and efficiency of various baseline and proposed models across different datasets and modalities.

Dataset	Model	Modality	Temporal Fuser	VRAM ↓	FPS_{3090} ↑	FPS_{4090} ↑	2D mAP ↑
TUMTraf-i	baseline	C	-	3371	**13.69**	**35.25**	54.42
	(ours)	C	Conv. LSTM	3371	12.87	32.49	**55.26**
	baseline	L	-	**3313**	**21.44**	**42.51**	81.84
	(ours)	L	Conv. LSTM	3613	12.68	24.62	**86.91**
	(ours)	L	Conv. GRU	3455	13.60	26.21	86.73
	baseline	C+L	-	**3907**	**11.92**	**26.42**	85.14
	(ours)	C+L	Conv. LSTM	3929	8.47	17.53	**86.73**
	(ours)	C+L	Conv. GRU	3915	9.05	18.50	86.27
OSDaR23	baseline	C	-	**3667**	**13.10**	**28.35**	14.00
	(ours)	C	Conv. LSTM	3769	12.34	26.13	**18.83**
	baseline	L	-	**3273**	**10.68**	**19.53**	77.56
	(ours)	L	Conv. LSTM	3575	8.02	14.40	**82.12**
	baseline	C+L	-	**3751**	**7.41**	**16.53**	79.54
	(ours)	C+L	Conv. LSTM	3867	6.15	12.60	**81.77**

Table 5.26: Comparative run-time performance and efficiency metrics of baseline and proposed models across different datasets and modalities. This table showcases the Video RAM (VRAM) usage, frames per second (FPS) performance on NVIDIA GeForce RTX 3090 and 4090 graphics cards, and the 2D mean Average Precision (2D mAP) scores for each model configuration. Models are evaluated using Camera-only (C), LiDAR-only (L), and combined (C+L) modalities, with and without temporal pipeline indicated by a Temporal Fuser type. All of these models utilized Online Caching during the inference.

One of the important aspect reflected in Table 5.26 is the Video RAM (VRAM) usage, a vital metric for assessing the memory efficiency of the models. Lower VRAM consumption is indicative of a model's ability to operate within the constraints of limited-memory systems, which is particularly relevant for embedded systems or applications with simultaneous workloads. The data shows that our models maintain comparable VRAM usage to the baseline models, indicating effective memory management without compromising on model complexity or functionality.

Modality	Online Caching	$FPS_{3090} \uparrow$
C	-	7.0
C	✓	**12.87**
L	-	7.6
L	✓	**13.6**
C+L	-	4.4
C+L	✓	**9.05**

Table 5.27: Comparative performance metrics highlighting the impact of Online Caching (described in Section 4.2.5) on frames-per-second (FPS) on an NVIDIA RTX 3090 GPU across different modalities using TUMTraf-i test split.

Another important metric presented is the frames-per-second (FPS) performance, which is especially important for time-sensitive applications. Higher FPS means that a model is able to process and analyze data faster, a crucial factor for tasks such as real-time video analysis or autonomous vehicle control, where delayed processing can lead to outdated or irrelevant outputs. The table compares the performance of each model on two high-end NVIDIA graphics cards, the GeForce RTX 3090 and 4090. Notably, the introduction of Temporal Fuser resulted in a decline in FPS. Within these networks, the Convolutional GRU marginally outperforms the Convolutional LSTM in FPS, albeit at the cost of a slight decrease in 2D mAP, a performance metric. Multi-modal models, integrating camera and LiDAR inputs, suffer the most as they go below 10 fps, specifically on Nvidia GeForce RTX 3090. The benchmark for acceptable FPS is established at 10 or higher, which aligns with the synchronization rate of 10 Hz for all datasets utilized in this research, namely TUMTraf-i and OSDaR23. It should be noted that the average of number of 3D points in point cloud scans in the TUMTraf-i test split is 60,996, while in the OSDaR23 validation split, it is 203,807. Consequently, LiDAR-based models exhibit differences in frames per second (FPS) from one dataset to another due to the varying loading and processing times associated with point clouds.

In Table 5.27, Online caching (described in Section 4.2.5) demonstrates its usefullness by significantly improving FPS across all modalities: standalone C and L modalities see increases from 7.0 to 12.87 and from 7.6 to 13.6 FPS respectively, while the combined C+L modality's FPS improves from 4.4 to 9.05.

Overall, the data indicates that our proposed models are highly competitive when Online Caching is utilized. Several configurations demonstrate a balanced VRAM usage and FPS performances, underscoring the advancements in creating models that are both efficient and mindful of resource constraints.

6 Future Work

This thesis has presented methodologies that significantly enhance object detection performance on TUMTraf-i [Zim+23b] and OSDaR23 [Tag+23] datasets, particularly noting improvements in the detection of distant objects and those with occlusions. Our experiments have validated the effectiveness of our approaches in multiple modalities; however, the dynamic nature of object detection and machine learning at large presents numerous avenues for future work to build upon the foundations laid by this research.

Our Temporal Fuser (TF) networks are crucial for integrating temporal information and enhancing detection robustness, making them strong candidates for future upgrades. As the field of deep learning progresses rapidly, increasingly sophisticated and powerful networks are emerging. Subsequent research should prioritize the replacement of the current Temporal Fuser networks with these advanced architectures. The adoption of cutting-edge networks could lead to significant enhancements in performance metrics, owing to their superior feature extraction abilities and a more nuanced understanding of temporal dynamics. An improvement could involve incorporating a deformable attention mechanism, as detailed in the work of [Zhu+21] and then further improving such mechanism for both spatial and temporal aspects. Nonetheless, such alternatives might impose more harsher requirements, potentially impacting memory consumption or inference speed, depending on the application. Therefore, it may be advisable to replace our current backbones – VoxelNet [ZT17] for LiDAR and Swin Transformer [Liu+21] for the camera – with faster and more suitable networks like the PointPillars network [Lan+19] for the LiDAR backbone or variants of the YOLO network [TC23b]. In addition, our Camera-to-BEV transformation model LSS [PF20] can be replaced with EA-LSS [Hu+23] for more accurate transformations.

The development of our Temporally-Aware Ground Truth Paste (TA-GTP) method represents a leap forward in augmenting the training data for generalization purposes in object detection. Future iterations of TA-GTP can benefit from transitioning the random techniques to a learnable paradigm, like in the work of [Zha+23a]. By adopting a machine learning or deep learning approaches to determine the placement of sampled objects, the model can learn optimal augmentation strategies that go beyond fixed positioning. Introducing learnable parameters to dictate translation and rotation values for sampled objects can also enable the method to move and rotate objects through temporal sequences in a more realistic manner.

We down-sample our multi-view images to 256x704 pixels in our models, which is considerably small; therefore, studies can be done using different resolution of images in order to see the impact in performance.

The current implementation of the Temporal Fuser lacks an ego-motion calibration step to align bird's eye view features across sequential sequences. TUMTraf-i dataset does not account for ego motion since the sensors are stationed on a stationary gantry. On the other hand, OSDaR23 sensor setup, mounted on a train, does involve motion. Nonetheless, in the sequences, this motion was minimal and did not deteriorate performance. However, for deployment in production environments, incorporating an ego-motion calibration step may be crucial to ensure feature alignment in the presence of considerable displacement between sequential frames.

OSDaR23 dataset has several other sensors such as infrared cameras and radar. These sensors can be integrated into multi-modal model for further potential improvements. We utilized all the LiDARs in the dataset: three long-range, one medium-range, and two short-range. An ablation study can be conducted using different combinations of LiDAR sensors to assess the impact of varying specifications on scan' range. We excluded some of the classes in OSDaR23 datasets due to technical problems related to bounding boxes; therefore, once these problems are fixed, models can be further experimented again with more classes such as animals and trains.

Finally, models can be converted to the ONNX format and then optimized with TensorRT to enhance inference speeds, a critical step for deployment in production. Furthermore, TUMTraf-i models can be deployed to run on s110 gantry bridge [Krä+21] whereas OSDaR23 models can be deployed on the trains to see the real-time capabilities in full action as well as adverse scenarios such as different weather conditions.

7 Conclusion

In the thesis, we address several prevalent challenges in the domain of **3D object detection**, specifically the issues associated with the detection of **distant objects** and those subject to **occlusion**. To mitigate these issues, we introduce a suite of methods and techniques, each tailored to integrate temporal information and required functionalities into distinct components of our pipeline. Through this integration process, we achieve enhanced performance across both our single and multi-modal models applied to our datasets – specifically, **TUMTraf-i** [Zim+23b] and **OSDaR23** [Tag+23]. Demonstrating comparable performance gains across these datasets is crucial, as they differ significantly in terms of sensor configuration and the target domain they represent.

To properly evaluate our datasets, we introduced a novel algorithm, **Temporal Dataset Split Search**, to identify optimal training, validation, and test sets that are balanced in class number and custom attributes, including distance from the ego position, point cloud count within bounding boxes, and occlusion levels. These sets comprise numerous sequential frames, ensuring a sufficient basis for coherent temporal training and evaluation using our **Temporal Pipeline**.

Our **Temporal Fuser** (TF) method has demonstrated improvements in model performances by fusing features from preceding frames with those of the current frame. We have chosen **Convolutional LSTM** and **Convolutional GRU** for this purpose. These networks effectively integrate temporal information from past sequences, thereby enabling models to refine their predictions with increased accuracy. For the TUMTraf-i dataset, increases in 2D mAP for camera-only, LiDAR-only, and multi-modal models were observed as follows: 0.84 (from 54.42 to 55.26), 3.89 (from 81.84 to 85.73), and 1.59 (from 85.14 to 86.73), respectively, when compared to their baselines. In the case of the OSDaR23 dataset, the improvements were more pronounced, with increases of 4.83 (from 14.00 to 18.83), 4.56 (from 77.56 to 82.12), and 2.23 (from 79.54 to 81.77) for the same respective models. In addition to the Temporal Fuser, we introduced the **Temporally-Aware Ground Truth Paste** (TA-GTP), a data augmentation method that enhances training diversity by sampling and simulating the rotation and motion of virtual objects, thereby improving the model's generalization capability for LiDAR-only models. Combining both methods yielded the highest performance, achieving 2D mAP scores of 86.91 on TUMTraf-i and 82.12 on OSDaR23.

Upon examining the performance of our best models with respect to object distance, we observe the anticipated trend of diminishing performance metrics with increasing distance. Nonetheless, models utilizing our methods demonstrate significantly improved performance, even at extreme distances – for instance, for objects located 200 meters or more away in the OSDaR23 dataset. As for the scenarios of significant occlusion where object prediction becomes challenging, our methods provided an advantage to the models over their baselines, as evidenced by our qualitative studies.

However, all of these improvements come at the cost of inference speed. To address this issue at a reasonable level, our **Online Caching** mechanism integrated into our **Temporal Pipeline** enables the features of sequential frames to be cached during validation or testing, allowing future inference steps to utilize these caches without the necessity of re-computation. Consequently, when tested on an Nvidia RTX 4090, the models achieve run-time inference speeds varying between 17.53 and 32.49 FPS (with multi-modal being the slowest and camera-only being the fastest), whereas our multi-modal models perform at less than 10 FPS on an Nvidia RTX 3090.

In summary, this thesis has successfully tackled some of the most pressing challenges in 3D object detection through the development and integration of temporal methods. The significant performance enhancements in detecting distant and occluded objects are testament to the efficacy of our Temporal Fuser and Temporally-Aware Ground Truth Paste methods. Our innovative Temporal Dataset Split Search algorithm further underscores our commitment to methodological rigor by ensuring balanced and coherent dataset utilization. Despite the trade-off with inference speed, our Online Caching strategy has proven effective in optimizing run-times, rendering our approach as a compelling advancement in production. As we conclude, the contributions of this thesis also open avenues for future research to build upon the robust foundation laid herein. This thesis underscores the potential for continued innovation in multi-modal sensor fusion and temporal analysis.

List of Figures

List of Tables

Bibliography

[Aki+19] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama. *Optuna: A Next-generation Hyperparameter Optimization Framework*. 2019. arXiv: 1907.10902 [cs.LG].

[Alz+21] L. Alzubaidi, J. Zhang, A. J. Humaidi, A. Al-Dujaili, Y. Duan, O. Al-Shamma, J. Santamaría, M. A. Fadhel, M. Al-Amidie, and L. Farhan. "Review of deep learning: concepts, CNN architectures, challenges, applications, future directions." In: *Journal of Big Data* 8.1 (Mar. 2021). DOI: 10.1186/s40537-021-00444-8.

[Bai+22] X. Bai, Z. Hu, X. Zhu, Q. Huang, Y. Chen, H. Fu, and C.-L. Tai. *TransFusion: Robust LiDAR-Camera Fusion for 3D Object Detection with Transformers*. 2022. arXiv: 2203.11496 [cs.CV].

[Bal+16] N. Ballas, L. Yao, C. Pal, and A. Courville. *Delving Deeper into Convolutional Networks for Learning Video Representations*. 2016. arXiv: 1511.06432 [cs.CV].

[Ber+11] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl. "Algorithms for Hyper-Parameter Optimization." In: *Advances in Neural Information Processing Systems*. Ed. by J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Weinberger. Vol. 24. Curran Associates, Inc., 2011.

[BL19] G. Brazil and X. Liu. *M3D-RPN: Monocular 3D Region Proposal Network for Object Detection*. 2019. arXiv: 1907.06038 [cs.CV].

[Bra+20] G. Brazil, G. Pons-Moll, X. Liu, and B. Schiele. *Kinematic 3D Object Detection in Monocular Video*. 2020. arXiv: 2007.09548 [cs.CV].

[BYC13] J. Bergstra, D. Yamins, and D. Cox. "Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures." In: *Proceedings of the 30th International Conference on Machine Learning*. Ed. by S. Dasgupta and D. McAllester. Vol. 28. Proceedings of Machine Learning Research 1. Atlanta, Georgia, USA: PMLR, 17–19 Jun 2013, pp. 115–123.

[Cai+23] H. Cai, Z. Zhang, Z. Zhou, Z. Li, W. Ding, and J. Zhao. *BEVFusion4D: Learning LiDAR-Camera Fusion Under Bird's-Eye-View via Cross-Modality Guidance and Temporal Aggregation*. 2023. arXiv: 2303.17099 [cs.CV].

[Car+21] M. Carranza-García, J. Torres-Mateo, P. Lara-Benítez, and J. García-Gutiérrez. "On the Performance of One-Stage and Two-Stage Object Detectors in Autonomous Vehicles Using Camera Data." In: *Remote Sensing* 13.1 (2021). ISSN: 2072-4292. DOI: 10.3390/rs13010089.

[Che+19] Y. Chen, S. Liu, X. Shen, and J. Jia. *Fast Point R-CNN*. 2019. arXiv: 1908.02990 [cs.CV].

[Che+20] Q. Chen, L. Sun, E. Cheung, and A. L. Yuille. "Every View Counts: Cross-View Consistency in 3D Object Detection with Hybrid-Cylindrical-Spherical Voxelization." In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., 2020, pp. 21224–21235.

[Cho17] F. Chollet. *Xception: Deep Learning with Depthwise Separable Convolutions*. 2017. arXiv: 1610.02357 [cs.CV].

[Chu+14] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. *Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling*. 2014. arXiv: 1412.3555 [cs.NE].

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

[Den+21] J. Deng, S. Shi, P. Li, W. Zhou, Y. Zhang, and H. Li. *Voxel R-CNN: Towards High Performance Voxel-based 3D Object Detection*. 2021. arXiv: 2012.15712 [cs.CV].

[Fan+21] J. Fang, X. Zuo, D. Zhou, S. Jin, S. Wang, and L. Zhang. "LiDAR-Aug: A General Rendering-based Augmentation Framework for 3D Object Detection." In: *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 4708–4718. DOI: 10.1109/CVPR46437.2021.00468.

[FZL16] R. Fu, Z. Zhang, and L. Li. "Using LSTM and GRU neural network methods for traffic flow prediction." In: *2016 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC)*. 2016, pp. 324–328. DOI: 10.1109/YAC.2016.7804912.

[Gao+23] Z. Gao, Q. Wang, Z. Pan, Z. Zhai, and H. Long. "PointPainting: 3D Object Detection Aided by Semantic Image Information." In: *Sensors* 23.5 (2023). ISSN: 1424-8220. DOI: 10.3390/s23052868.

[Gra14] A. Graves. *Generating Sequences With Recurrent Neural Networks*. 2014. arXiv: 1308.0850 [cs.NE].

[HS97] S. Hochreiter and J. Schmidhuber. "Long Short-Term Memory." In: *Neural Comput.* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735.

[Hu+23] H. Hu, F. Wang, J. Su, Y. Wang, L. Hu, W. Fang, J. Xu, and Z. Zhang. *EA-LSS: Edge-aware Lift-splat-shot Framework for 3D BEV Object Detection*. 2023. arXiv: 2303.17895 [cs.CV].

[Hua+20a] R. Huang, W. Zhang, A. Kundu, C. Pantofaru, D. A. Ross, T. Funkhouser, and A. Fathi. *An LSTM Approach to Temporal 3D Object Detection in LiDAR Point Clouds*. 2020. arXiv: 2007.12392 [cs.CV].

[Hua+20b] T. Huang, Z. Liu, X. Chen, and X. Bai. *EPNet: Enhancing Point Features with Image Semantics for 3D Object Detection*. 2020. arXiv: 2007.08856 [cs.CV].

[Hua+22] J. Huang, G. Huang, Z. Zhu, Y. Ye, and D. Du. *BEVDet: High-performance Multi-camera 3D Object Detection in Bird-Eye-View*. 2022. arXiv: 2112.11790 [cs.CV].

[HZ01] R. Hartley and A. Zisserman. "Multiple View Geometry in Computer Vision." In: 2001.

[JK19] J. Johnson and T. Khoshgoftaar. "Survey on deep learning with class imbalance." In: *Journal of Big Data* 6 (Mar. 2019), p. 27. DOI: 10.1186/s40537-019-0192-5.

[JZH21] C. Janiesch, P. Zschech, and K. Heinrich. "Machine learning and deep learning." In: *Electronic Markets* 31.3 (Apr. 2021), pp. 685–695. DOI: 10.1007/s12525-021-00475-2.

[Kha+22] S. Khan, M. Naseer, M. Hayat, S. W. Zamir, F. S. Khan, and M. Shah. "Transformers in Vision: A Survey." In: *ACM Computing Surveys* 54.10s (Jan. 2022), pp. 1–41. DOI: 10.1145/3505244.

[Krä+21] A. Krämmer, C. Schöller, D. Gulati, V. Lakshminarasimhan, F. Kurz, D. Rosenbaum, C. Lenz, and A. Knoll. *Providentia – A Large-Scale Sensor System for the Assistance of Autonomous Vehicles and Its Evaluation*. 2021. arXiv: 1906.06789 [cs.RO].

[Lai+23] X. Lai, Y. Chen, F. Lu, J. Liu, and J. Jia. *Spherical Transformer for LiDAR-based 3D Recognition*. 2023. arXiv: 2303.12766 [cs.CV].

[Lan+19] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom. *PointPillars: Fast Encoders for Object Detection from Point Clouds*. 2019. arXiv: 1812.05784 [cs.LG].

[LCS19] P. Li, X. Chen, and S. Shen. *Stereo R-CNN based 3D Object Detection for Autonomous Driving*. 2019. arXiv: 1902.09738 [cs.CV].

[LH17] I. Loshchilov and F. Hutter. *SGDR: Stochastic Gradient Descent with Warm Restarts*. 2017. arXiv: 1608.03983 [cs.LG].

[LH19] I. Loshchilov and F. Hutter. *Decoupled Weight Decay Regularization*. 2019. arXiv: 1711.05101 [cs.LG].

[Li+18] Z. Li, C. Peng, G. Yu, X. Zhang, Y. Deng, and J. Sun. *DetNet: A Backbone network for Object Detection*. 2018. arXiv: 1804.06215 [cs.CV].

[Li+19] R. Li, X. Li, C.-W. Fu, D. Cohen-Or, and P.-A. Heng. *PU-GAN: a Point Cloud Upsampling Adversarial Network*. 2019. arXiv: 1907.10844 [cs.CV].

[Li+22a] Y. Li, A. W. Yu, T. Meng, B. Caine, J. Ngiam, D. Peng, J. Shen, B. Wu, Y. Lu, D. Zhou, Q. V. Le, A. Yuille, and M. Tan. *DeepFusion: Lidar-Camera Deep Fusion for Multi-Modal 3D Object Detection*. 2022. arXiv: 2203.08195 [cs.CV].

[Li+22b] Z. Li, W. Wang, H. Li, E. Xie, C. Sima, T. Lu, Q. Yu, and J. Dai. *BEVFormer: Learning Bird's-Eye-View Representation from Multi-Camera Images via Spatiotemporal Transformers*. 2022. arXiv: 2203.17270 [cs.CV].

[Li+23] Z. Li, C. Zhang, W.-C. Ma, Y. Zhou, L. Huang, H. Wang, S. Lim, and H. Zhao. *VoxelFormer: Bird's-Eye-View Feature Generation based on Dual-view Attention for Multi-view 3D Object Detection*. 2023. arXiv: 2304.01054 [cs.CV].

[LI20] Y. Li and J. Ibanez-Guzman. "Lidar for Autonomous Driving: The Principles, Challenges, and Trends for Automotive Lidar and Perception Systems." In: *IEEE Signal Processing Magazine* 37.4 (2020), pp. 50–61. DOI: 10.1109/MSP.2020.2973615.

[Lia+20a] M. Liang, B. Yang, Y. Chen, R. Hu, and R. Urtasun. *Multi-Task Multi-Sensor Fusion for 3D Object Detection*. 2020. arXiv: 2012.12397 [cs.CV].

[Lia+20b] M. Liang, B. Yang, S. Wang, and R. Urtasun. *Deep Continuous Fusion for Multi-Sensor 3D Object Detection*. 2020. arXiv: 2012.10992 [cs.CV].

[Lia+22] T. Liang, H. Xie, K. Yu, Z. Xia, Z. Lin, Y. Wang, T. Tang, B. Wang, and Z. Tang. *BEVFusion: A Simple and Robust LiDAR-Camera Fusion Framework*. 2022. arXiv: 2205.13790 [cs.CV].

[Lin+17] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. *Feature Pyramid Networks for Object Detection*. 2017. arXiv: 1612.03144 [cs.CV].

[Lin+18] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. *Focal Loss for Dense Object Detection*. 2018. arXiv: 1708.02002 [cs.CV].

[Lin+22] T. Lin, Y. Wang, X. Liu, and X. Qiu. "A survey of transformers." In: *AI Open* 3 (2022), pp. 111–132. ISSN: 2666-6510. DOI: https://doi.org/10.1016/j.aiopen.2022.10.001.

[Liu+21] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo. *Swin Transformer: Hierarchical Vision Transformer using Shifted Windows*. 2021. arXiv: 2103.14030 [cs.CV].

[Liu+22] Z. Liu, H. Tang, A. Amini, X. Yang, H. Mao, D. Rus, and S. Han. *BEVFusion: Multi-Task Multi-Sensor Fusion with Unified Bird's-Eye View Representation*. 2022. arXiv: 2205.13542 [cs.CV].

[Liu+23] Z. Liu, X. Yang, H. Tang, S. Yang, and S. Han. *FlatFormer: Flattened Window Attention for Efficient Point Cloud Transformer*. 2023. arXiv: 2301.08739 [cs.CV].

[Luo+21] S. Luo, H. Dai, L. Shao, and Y. Ding. *M3DSSD: Monocular 3D Single Stage Object Detector*. 2021. arXiv: 2103.13164 [cs.CV].

[LWW21] Z. Li, F. Wang, and N. Wang. *LiDAR R-CNN: An Efficient and Universal 3D Object Detector*. 2021. arXiv: 2103.15297 [cs.CV].

[Mao+23] J. Mao, S. Shi, X. Wang, and H. Li. *3D Object Detection for Autonomous Driving: A Comprehensive Survey*. 2023. arXiv: 2206.09474 [cs.CV].

[MRP21] R. K. Mishra, G. Y. S. Reddy, and H. Pathak. "The Understanding of Deep Learning: A Comprehensive Review." In: *Mathematical Problems in Engineering* 2021 (Apr. 2021). Ed. by A. Ahmadian, pp. 1–15. DOI: 10.1155/2021/5548884.

[Nin+] X. Ning, J. Zhou, J. Cheng, J. Wu, C. Wang, and L. Gu. "Guest Editorial: Multi-view representation learning for computer vision." In: *IET Computer Vision* n/a.n/a (). DOI: https://doi.org/10.1049/cvi2.12176. eprint: https://ietresearch.onlinelibrary.wiley.com/doi/pdf/10.1049/cvi2.12176.

[Pan+20] B. Pan, J. Sun, H. Y. T. Leung, A. Andonian, and B. Zhou. "Cross-View Semantic Segmentation for Sensing Surroundings." In: *IEEE Robotics and Automation Letters* 5.3 (July 2020), pp. 4867–4873. DOI: 10.1109/lra.2020.3004325.

[Pen+20] W. Peng, H. Pan, H. Liu, and Y. Sun. "IDA-3D: Instance-Depth-Aware 3D Object Detection From Stereo Vision for Autonomous Driving." In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 13012–13021. DOI: 10.1109/CVPR42600.2020.01303.

[PF20] J. Philion and S. Fidler. *Lift, Splat, Shoot: Encoding Images From Arbitrary Camera Rigs by Implicitly Unprojecting to 3D*. 2020. arXiv: 2008.05711 [cs.CV].

[PMR20] S. Pang, D. Morris, and H. Radha. *CLOCs: Camera-LiDAR Object Candidates Fusion for 3D Object Detection*. 2020. arXiv: 2009.00784 [cs.CV].

[PMR22] S. Pang, D. Morris, and H. Radha. "Fast-CLOCs: Fast Camera-LiDAR Object Candidates Fusion for 3D Object Detection." In: *2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. 2022, pp. 3747–3756. DOI: 10.1109/WACV51458.2022.00380.

[Qi+18] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas. *Frustum PointNets for 3D Object Detection from RGB-D Data*. 2018. arXiv: 1711.08488 [cs.CV].

[QWL20] Z. Qin, J. Wang, and Y. Lu. *MonoGRNet: A Geometric Reasoning Network for Monocular 3D Object Localization*. 2020. arXiv: 1811.10247 [cs.CV].

[RC20] T. Roddick and R. Cipolla. *Predicting Semantic Map Representations from Images using Pyramid Occupancy Networks*. 2020. arXiv: 2003.13402 [cs.CV].

[Ren+16] S. Ren, K. He, R. Girshick, and J. Sun. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. 2016. arXiv: 1506.01497 [cs.CV].

[RF18] J. Redmon and A. Farhadi. *YOLOv3: An Incremental Improvement*. 2018. arXiv: 1804.02767 [cs.CV].

[RKC18] T. Roddick, A. Kendall, and R. Cipolla. *Orthographic Feature Transform for Monocular 3D Object Detection*. 2018. arXiv: 1811.08188 [cs.CV].

[Shi+15] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-k. Wong, and W.-c. Woo. *Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting*. 2015. arXiv: 1506.04214 [cs.CV].

[Shi+21] S. Shi, C. Guo, L. Jiang, Z. Wang, J. Shi, X. Wang, and H. Li. *PV-RCNN: Point-Voxel Feature Set Abstraction for 3D Object Detection*. 2021. arXiv: 1912.13192 [cs.CV].

[Shi+22] X. Shi, Q. Ye, X. Chen, C. Chen, Z. Chen, and T.-K. Kim. *Geometry-based Distance Decomposition for Monocular 3D Object Detection*. 2022. arXiv: 2104.03775 [cs.CV].

[Shi+23] F. M. Shiri, T. Perumal, N. Mustapha, and R. Mohamed. *A Comprehensive Overview and Comparative Analysis on Deep Learning Models: CNN, RNN, LSTM, GRU*. 2023. arXiv: 2305.17473 [cs.LG].

[Sim+19] A. Simonelli, S. R. R. Bulò, L. Porzi, M. López-Antequera, and P. Kontschieder. *Disentangling Monocular 3D Object Detection*. 2019. arXiv: 1905.12365 [cs.CV].

[Smi17] L. N. Smith. *Cyclical Learning Rates for Training Neural Networks*. 2017. arXiv: 1506.01186 [cs.CV].

[Sun+22] P. Sun, M. Tan, W. Wang, C. Liu, F. Xia, Z. Leng, and D. Anguelov. *SWFormer: Sparse Window Transformer for 3D Object Detection in Point Clouds*. 2022. arXiv: 2210.07372 [cs.CV].

[SWL19] S. Shi, X. Wang, and H. Li. *PointRCNN: 3D Object Proposal Generation and Detection from Point Cloud*. 2019. arXiv: 1812.04244 [cs.CV].

[Tag+23] R. Tagiew, M. Köppel, K. Schwalbe, P. Denzler, P. Neumaier, T. Klockau, M. Boekhoff, P. Klasek, and R. Tilly. *OSDaR23: Open Sensor Data for Rail 2023*. 2023. arXiv: 2305.03001 [cs.CV].

[TC23a] J. Terven and D. Cordova-Esparza. *A Comprehensive Review of YOLO: From YOLOv1 and Beyond*. 2023. arXiv: 2304.00501 [cs.CV].

[TC23b] J. Terven and D. Cordova-Esparza. *A Comprehensive Review of YOLO: From YOLOv1 and Beyond*. 2023. arXiv: 2304.00501 [cs.CV].

[TL20] M. Tan and Q. V. Le. *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks*. 2020. arXiv: 1905.11946 [cs.LG].

[Tod+17] G. Toderici, D. Vincent, N. Johnston, S. J. Hwang, D. Minnen, J. Shor, and M. Covell. *Full Resolution Image Compression with Recurrent Neural Networks*. 2017. arXiv: 1608.05148 [cs.CV].

[Tou+21] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou. *Training data-efficient image transformers distillation through attention*. 2021. arXiv: 2012.12877 [cs.CV].

[Vor+20] S. Vora, A. H. Lang, B. Helou, and O. Beijbom. *PointPainting: Sequential Fusion for 3D Object Detection*. 2020. arXiv: 1911.10150 [cs.CV].

[Wan+20] Y. Wang, A. Fathi, A. Kundu, D. Ross, C. Pantofaru, T. Funkhouser, and J. Solomon. *Pillar-based Object Detection for Autonomous Driving*. 2020. arXiv: 2007.10323 [cs.CV].

[Wan+21] Z. Wang, Z. Zhao, Z. Jin, Z. Che, J. Tang, C. Shen, and Y. Peng. "Multi-Stage Fusion for Multi-Class 3D Lidar Detection." In: *2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*. 2021, pp. 3113–3121. DOI: 10.1109/ICCVW54120.2021.00347.

[Wan+23] H. Wang, C. Shi, S. Shi, M. Lei, S. Wang, D. He, B. Schiele, and L. Wang. *DSVT: Dynamic Sparse Voxel Transformer with Rotated Sets*. 2023. arXiv: 2301.06051 [cs.CV].

[WBL22] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao. *YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors*. 2022. arXiv: 2207.02696 [cs.CV].

[WJ19] Z. Wang and K. Jia. *Frustum ConvNet: Sliding Frustums to Aggregate Local Point-Wise Features for Amodal 3D Object Detection*. 2019. arXiv: 1903.01864 [cs.CV].

[WR17] H. Wang and B. Raj. *On the Origin of Deep Learning*. 2017. arXiv: 1702.07800 [cs.LG].

[XAJ18] D. Xu, D. Anguelov, and A. Jain. *PointFusion: Deep Sensor Fusion for 3D Bounding Box Estimation*. 2018. arXiv: 1711.10871 [cs.CV].

[Xu+21] J. Xu, R. Zhang, J. Dou, Y. Zhu, J. Sun, and S. Pu. *RPVNet: A Deep and Efficient Range-Point-Voxel Fusion Network for LiDAR Point Cloud Segmentation*. 2021. arXiv: 2103.12978 [cs.CV].

[Yan+21] Z. Yang, Y. Zhou, Z. Chen, and J. Ngiam. *3D-MAN: 3D Multi-frame Attention Network for Object Detection*. 2021. arXiv: 2103.16054 [cs.CV].

[Yan+23] J. Yan, Y. Liu, J. Sun, F. Jia, S. Li, T. Wang, and X. Zhang. *Cross Modal Transformer: Towards Fast and Robust 3D Object Detection*. 2023. arXiv: 2301.01283 [cs.CV].

[Yif+19] W. Yifan, S. Wu, H. Huang, D. Cohen-Or, and O. Sorkine-Hornung. *Patch-based Progressive 3D Point Set Upsampling*. 2019. arXiv: 1811.11286 [cs.CV].

[Yin+20] J. Yin, J. Shen, C. Guan, D. Zhou, and R. Yang. *LiDAR-based Online 3D Video Object Detection with Graph-based Message Passing and Spatiotemporal Transformer Attention*. 2020. arXiv: 2004.01389 [cs.CV].

[YML18] Y. Yan, Y. Mao, and B. Li. "SECOND: Sparsely Embedded Convolutional Detection." In: *Sensors* 18 (Oct. 2018), p. 3337. DOI: 10.3390/s18103337.

[Yos+20] O. Yoshihiko, T. Yuki, W. Shuhei, and O. Masaki. "Multiobjective Tree-structured Parzen Estimator for Computationally Expensive Optimization Problems." In: *The Genetic and Evolutionary Computation Conference (GECCO2020)*. 2020.

[Yu+18] L. Yu, X. Li, C.-W. Fu, D. Cohen-Or, and P.-A. Heng. *PU-Net: Point Cloud Upsampling Network*. 2018. arXiv: 1801.06761 [cs.CV].

[Yua+20] Z. Yuan, X. Song, L. Bai, W. Zhou, Z. Wang, and W. Ouyang. *Temporal-Channel Transformer for 3D Lidar-Based Video Object Detection in Autonomous Driving*. 2020. arXiv: 2011.13628 [cs.CV].

[YZK21a] T. Yin, X. Zhou, and P. Krähenbühl. *Center-based 3D Object Detection and Tracking*. 2021. arXiv: 2006.11275 [cs.CV].

[YZK21b] T. Yin, X. Zhou, and P. Krähenbühl. *Multimodal Virtual Point 3D Detection*. 2021. arXiv: 2111.06881 [cs.CV].

[Zam+21] G. Zamanakos, L. Tsochatzidis, A. Amanatiadis, and I. Pratikakis. "A comprehensive survey of LIDAR-based 3D object detection methods with deep learning for autonomous driving." In: *Computers Graphics* 99 (2021), pp. 153–181. ISSN: 0097-8493. DOI: https://doi.org/10.1016/j.cag.2021.07.003.

[ZCH22] Y. Zhang, J. Chen, and D. Huang. *CAT-Det: Contrastively Augmented Transformer for Multi-modal 3D Object Detection*. 2022. arXiv: 2204.00325 [cs.CV].

[Zha+17] X. Zhang, X. Zhou, M. Lin, and J. Sun. *ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices*. 2017. arXiv: 1707.01083 [cs.CV].

[Zha+20] Z. Zhang, J. Gao, J. Mao, Y. Liu, D. Anguelov, and C. Li. *STINet: Spatio-Temporal-Interactive Network for Pedestrian Detection and Trajectory Prediction*. 2020. arXiv: 2005.04255 [cs.CV].

[Zha+23a] J. Zhan, T. Liu, R. Li, J. Zhang, Z. Zhang, and Y. Chen. *Real-Aug: Realistic Scene Synthesis for LiDAR Augmentation in 3D Object Detection*. 2023. arXiv: 2305.12853 [cs.CV].

[Zha+23b] Y. Zhang, B. Kang, B. Hooi, S. Yan, and J. Feng. *Deep Long-Tailed Learning: A Survey*. 2023. arXiv: 2110.04596 [cs.CV].

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

[Zhu+21] X. Zhu, W. Su, L. Lu, B. Li, X. Wang, and J. Dai. *Deformable DETR: Deformable Transformers for End-to-End Object Detection*. 2021. arXiv: 2010.04159 [cs.CV].

[Zhu+23] Z. Zhu, Y. Zhang, H. Chen, Y. Dong, S. Zhao, W. Ding, J. Zhong, and S. Zheng. *Understanding the Robustness of 3D Object Detection with Bird's-Eye-View Representations in Autonomous Driving*. 2023. arXiv: 2303.17297 [cs.CV].

[Zim+23a] W. Zimmer, J. Birkner, M. Brucker, H. T. Nguyen, S. Petrovski, B. Wang, and A. C. Knoll. *InfraDet3D: Multi-Modal 3D Object Detection based on Roadside Infrastructure Camera and LiDAR Sensors*. 2023. arXiv: 2305.00314 [cs.CV].

[Zim+23b] W. Zimmer, C. Creß, H. T. Nguyen, and A. C. Knoll. *A9 Intersection Dataset: All You Need for Urban 3D Camera-LiDAR Roadside Perception*. 2023. arXiv: 2306.09266 [cs.CV].

[ZT17] Y. Zhou and O. Tuzel. *VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection*. 2017. arXiv: 1711.06396 [cs.CV].