Department of Informatics
Technical University of Munich

Master's Thesis in Mechanical Engineering

Real-Time and Multi-Modal 3D Object Detection on the Providentia++ Test Stretch

Echtzeitfähige und multimodale 3D-Objekterkennung auf der Providentia++ Teststrecke

Supervisor	Prof. Dr.-Ing. habil. Alois C. Knoll
Advisor	M.Sc. Walter Zimmer
Author	Maximilian Fortkord
Date	December 15, 2021 in Garching

Disclaimer

I confirm that this Master's Thesis is my own work and I have documented all sources and materials used.

M. Fortkord

Garching, December 15, 2021 (Maximilian Fortkord)

Abstract

In dynamic road environments, vehicle sensors may be influenced in an adverse manner by environmental factors as well as through interaction with other traffic participants. As a result of this, a wide-ranging scene understanding is required in order for vehicles to be operated in a safe and proactive manner. Within the Providentia++ project, an Intelligent Infrastructure System (IIS) is established in order to take advantage of high-fidelity data streaming from multiple sensor modalities such as camera and LiDAR sensors. Accurate vehicle locations, velocities and identifying data are determined on the basis of real-time, multi-modal 3D object detection methods. Abstracted information is then mapped to a digital twin and made available to downstream users for further processing and visualization. Use cases such as autonomous vehicles may be initiated based on the increased situational awareness of individual vehicles as a result of this. For the utilization of time-dependent effects in perception, tracking and prediction of detected objects, an exact synchronization of fused sensor modalities is necessary. In a first step, the synchronization of camera and LiDAR sensors to a common timestamp is to be implemented as part of this thesis. An Ouster OS-1 64-layer LiDAR is enabled to send trigger signals for camera image data acquisition to cameras located within the a networked system in ROS. Camera-LiDAR sensor pair calibration is carried out using calibrated using target-based and targetless calibration methods as well as measurements carried out using GPS and IMU data. Based on output calibration data, projection matrices and extrinsic matrices for sensors for sensors located on the Providentia infrastructure are calculated. The validity of these projection matrices is shown in exemplary data frames for LiDAR and camera data. Inside the fused multi-modality object detection pipeline based on CLOCs implemented in previous work, a more accurate 2D object detector YOLOR, is implemented. The YOLOR detector is integrated into the CLOCs fusion architecture using ROS in order to be deployable on the Providentia++ infrastructure.

Declaration of Consent, Open Source

Hereby I, Maximilian Fortkord, born on September 4, 1995 in Stuttgart, make the software I developed during my Master's Thesis available to the Chair of Robotics, Artificial Intelligence and Real-time Systems under the terms of the license below.

Garching, November 14, 2021

Maximilian Fortkord, B.Sc.

Acronyms

AP	Average Precision
BEV	Bird’s Eye View
CLOCs	Camera-LiDAR Object Candidates
CNN	Convolutional Neural Net
DFU	Data Fusion Unit
DL	Deep Learning
FOV	Field of View
GPS	Global Positioning System
IMU	Internal Measurement Unit
IoU	Intersection over Union
MLP	Multilayer Perceptron
NDS	nuScenes Detection Score
NMS	Non-Maximum Suppression
NTP	Network Time Protocol
PTP	Precision Time Protocol
ROI	Region of Interest
SGD	Stochastic Gradient Descent
SSD	Single Shot Detection

Formula Symbols

Formula Symbols	Unit	Description
ϵ	–	Error
γ	°	Yaw angle
I	–	Intrinsic matrix
P	–	Projection matrix
P_i	–	Tensor: i-th detection candidate
R	–	Rotation matrix
T	–	Translation matrix
$T_{i,j}$	–	Tensor: Set of 2D and 3D detections
x	–	Neural network input
y	–	Neural network output

Contents

Chapter 1

Introduction

One of the core challenges of research in the area of autonomous driving is represented by the accurate and robust 3D detection of objects. At the Chair of Robotics, Artificial Intelligence and Real-time Systems at TUM, this topic is investigated within the Providentia++ project. Here, a highway- and urban-based public road test stretch are equipped with radar, LiDAR and camera sensors in order to capture complex and densely populated traffic scenes. In the first chapter, the topic of this thesis is introduced in the scope of intelligent infrastructure systems being developed in support of automated and autonomous traffic participants. First, in Section 1.1, the motivation for research carried out within the Providentia++ project is described. Then, in Section 1.2, the purpose of research carried out as part of this thesis is detailed. Lastly, in Section 1.3, the structure of this thesis is explained.

1.1 Motivation

As the development of automotive sensors, object detection algorithms and trajectory planning continuously progresses, more and more vehicles are likely to be upgraded to higher degrees of autonomous driving. Current challenges of automotive sensors such as object occlusion due to environmental conditions and other road users, as well as erroneous detection results due to damaged or faulty sensors are therefore of great interest in terms of research. Today, a variety of studies agree that the benefits of autonomous driving such as higher safety for vehicle occupants, more comfort and beneficial effects on the environment e.g. through reduction of traffic and congestion are of significant interest [55]. Bearing in mind the error potential of automotive vehicle sensors, the augmentation of vehicle sensor data and information gathered using an external infrastructure is necessary. Such an intelligent transportation system (ITS) consists of perception sensors such as radar, LiDAR, and camera sensors that feed identifying data on detected objects and obstacles to local units, whereby a digital twin of the live traffic is created. Current velocities, positions and estimated trajectories of vehicles in a pre-defined road sector may then be sent via 5G technology to vehicles located on a certain stretch of the ITS track. This information may then provide a higher level of safety, reduction of uncertainties in e.g. trajectory planning, as well as the verification of sensor data created by autonomous vehicles travelling on the test-stretch themselves [28].

Within the Providentia++ project, such an ITS has been established and is being further developed including downstream uses of data. Examples for these uses are the provision of high-quality data for autonomous driving datasets, establishment of safe and proven standards for object detection. These use cases and applications are based on anonymized data made publicly available.

1.2 Purpose

Within the Providentia++ project, previous research has established a LiDAR-only 3D detection pipeline, as well as a fused camera-LiDAR 3D object detection pipeline. Based on this work, a number of new components are implemented and developed as part of this thesis. Here, the main requirements of a successfully implemented solution for 3D object detection are the real-time capability and accuracy of the inference pipeline. In the scope of this thesis, the object detection pipeline is considered to be running in real-time, if it is able to process sensor data and output detection results at a frequency of $30Hz$ or higher. Hereby, the increased number of object classes that are to be detected in phase two of the Providentia++ project, such as bicycles, pedestrians and other small objects found in an urban environment represent a challenge to be met as part of this thesis.

Due to smaller object sizes and a larger number of varied poses and orientations, detection of these objects is likely to be more difficult than in a highway-only setting. Previously, the direction of travel was restricted to mainly a single forward direction as well as lane change maneuvers.

1.3 Structure

The structure of this thesis is divided into several smaller topics and work packages which are implemented in the following order. First, a literature review regarding real-time and multi-modal 3D object detection algorithms is carried out. Time-synchronization using PTP-1588 is then enabled for Basler camera sensors. A software-trigger for starting Ouster LiDAR data recording is then set up. The main focus of this thesis lies on the improvement of the multi-modal and real-time 3D object detection methods. In order for the execution of a object detection pipeline, camera and LiDAR sensors are then calibrated intrinsically and extrinsically using target-based and targetless calibration.

Traffic scene data recorded on the newly added sensor stations of the Providentia++ urban test stretch is annotated using the proAnno labeling tool based on [81]. 2D object detection inference is tested using YOLOR on videos collected on the Providentia++ infrastructure. A training of CenterPoint on the approximately 350 available P++ LiDAR frames is then to be carried out. The CLOCs fusion network for 3D camera and LiDAR object detection is then adjusted in order to process image data using YOLOR-ROS and input in form of the proFusion dataset.

Chapter 2

State of the Art

In this chapter, the necessary theoretical background for this thesis is provided alongside relevant related works. In Section 2.1, an overview of the Providentia++ project is given. Then, in Section 2.2., the main sensors used for perceptive tasks in autonomous driving are listed. In Section 2.3, state-of-the-art 2D and 3D object detection methods are highlighted. In Section 2.4 an overview of multi-modal sensor fusion with a focus on multi-modal approaches is given. Section 2.5 contains an overview and comparison of relevant autonomous driving datasets, followed by current data augmentation methods. Last, in Section 2.7, common evaluation metrics for 2D/3D object detection are listed.

2.1 The Providentia++ Project

Partly due to the continuous development of automotive sensors, object detection models and data-processing algorithms, more and more vehicles are likely to be upgraded to a higher degree of autonomy in the following years. Current challenges regarding automotive sensors include adverse environmental conditions and object occlusion due to other road-users. Today, it is suggested by some studies, that the benefits of autonomous driving such as higher safety for vehicle occupants, more comfort and beneficial effects for the environment e.g. through reduction of traffic and congestion are of significant interest to the public [55]. In consideration of potential detection and operating errors of automotive vehicle sensors, a supportive augmentation of vehicle sensor data may be necessary. Next to internal sensors present in modern automated vehicles, information can be gathered and processed using external infrastructures. External infrastructure such as this is referred to as an intelligent transportation system (ITS) and is generally made up of perception sensors such as camera, LiDAR and radar.

Within the Providentia++ project, such an ITS has been established and is being further developed. Identification data on detected objects and on-road obstacles is transmitted to a local server, whereby a digital twin of live traffic is created. Current velocities, positions and estimated trajectories of vehicles in a pre-defined road sector may also be sent via 5G communication technology to vehicles located on a specified part of the ITS track. A higher level of safety, reduction of uncertainties in e.g. trajectory planning, as well as the verification of sensor data created by autonomous vehicles travelling on the test-stretch themselves can be attained as a result of this [28]. Further possible use-cases of anonymized data collected on ITS infrastructure include the supply of high-quality autonomous driving datasets and establishment and testing of safety-critical standards for object detection in the autonomous driving context. Until early 2021, the Providentia++ test track on which sensor infrastruc-

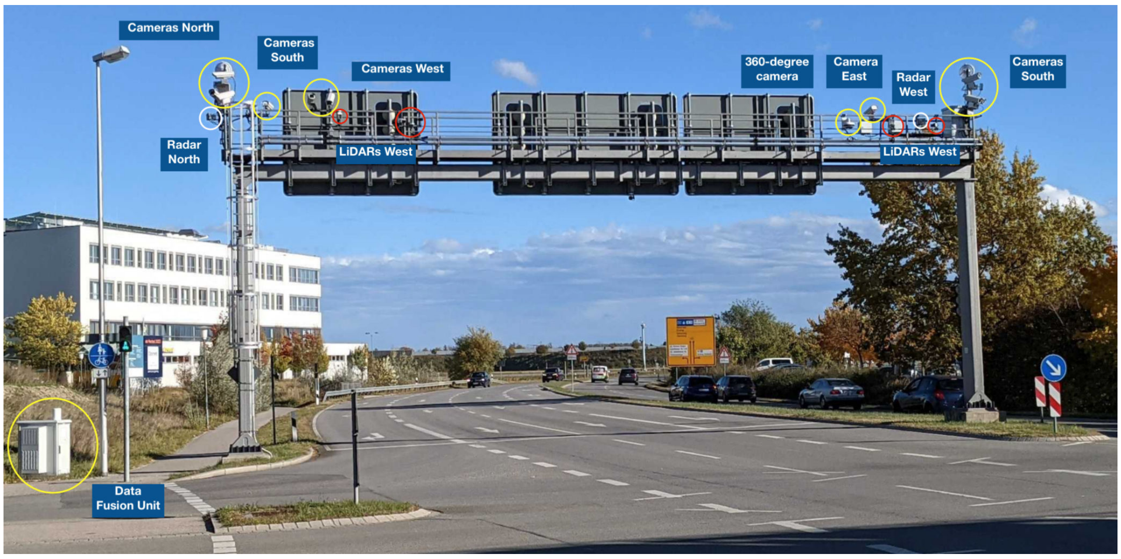

Figure 2.1: Overview of sensors placed on the s110 sensor station in Garching-Hochbrück

ture had been previously installed and operated, was mainly located on a two kilometer long stretch of the A9 Highway close to Garching. Here, a total of three sensor stations are placed on highway gantry bridges and included a total of 12 cameras, two event-based cameras, two LiDARs as well as 12 automotive radar sensors.

In the first half of 2021, the previously built Providentia [28] test track was extended into a more urban area leading up into the industrial area of Garching. The inclusion of more challenging road-user interactions and automotive scenarios are among the main motivating factors for this extension. In a highway setting, vehicles generally move in a single direction with interspersed lane changes. This means that objects and road-users such as cars, busses, trucks are generally seen from a similar perspective and feature a similar on-road orientation. As a result of this, object detection algorithms are enabled to learn features in rapid manner. Within a more urban setting, additional road-users such as pedestrians, bicycles and other smaller vehicles with abnormal dimensions may be encountered. Some challenges introduced by these traffic-participants are small object sizes, which may lead to occlusions through larger objects. In addition, the detection of these newly introduced classes may become more difficult due to smaller LiDAR cross-sections and a lower number of representative pixels in camera images. Dimensions of bounding boxes may be significantly different from other object classes: while bicycles may be as long as 1.5 meters, bicycle cross-section viewed from the front is significantly smaller. Pedestrians may vary in height, pose and therefore bounding box dimensions as well as side ratios in a similar manner. For pedestrians, small changes in pose may seem to lead to a large difference in the pose perceived by an object detection pipeline [65]. As part of current and future research within the Providentia++ project, these challenges are investigated.

In Figure 2.3, the newly added sensor stations located near the industrial area of Garching can be seen. On sensor station s110, it may be of special interest to note that different types of LiDAR-sensors have been placed. Two Ouster OS-1 64 layer LiDARs, as well as three Valeo ScalaGen2 LiDARs with overlapping fields of view are selected for this site. An exemplary overview of different sensor types placed on the s110 sensor station may be found in Figure 2.1. Next to sensors installed on the traffic gantry bridge, a Data Fusion Unit (DFU) may be seen on the left hand side in this image. Data collected from individual sensors is pooled in

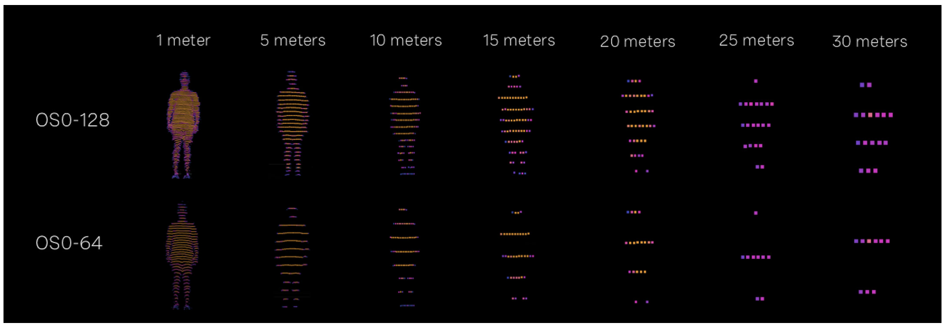

Figure 2.2: Ouster LiDAR pedestrian detection [41]

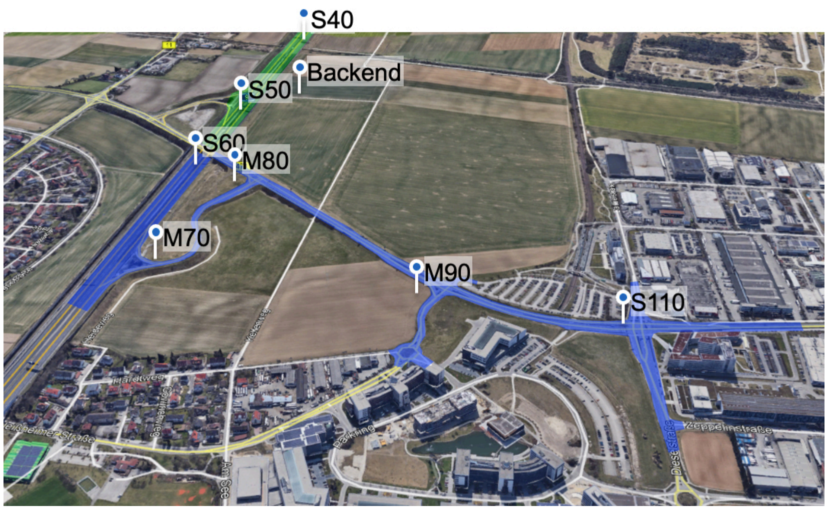

Figure 2.3: BEV-image of Providentia++ sensor stations. Newly added sensor stations are colored blue [57]

each DFU before object detection is carried out and identifying data of detected objects is passed further downstream.

2.2 Autonomous Driving Sensors

As of today, a number of automotive sensor types are available for the determination of position, orientation and class of vehicles and objects. In the context of autonomous driving, the three main types of sensors used are camera, LiDAR and radar sensors. In the following section, the functionality of the different sensor types is described with a focus on camera and LiDAR sensors. Additionally, performance aspects of the individual sensors are compared in regard to the perception task and requirements of autonomous driving.

2.2.1 Camera Sensors

The functionality of camera sensors such as CMOS-type (Complimentary Metal-Oxide Semiconductor) sensors is based on the capture of electromagnetic waves reflected by the surrounding environment. Image data in digital camera sensors usually consists of a red, green and blue color channel (RGB) per pixel. Camera images generally contain a high degree of information in the two-dimensional space on object colors, sizes and textures, while lacking depth information. This detailed information may especially be useful in the context of detecting road markings, traffic signs or distinguishing signals such as traffic lights.

Environmental factors such as adverse lighting conditions and occlusions due to precipitation such as rain, snow and fog or other objects may signiciantly reduce the robustness and performance of camera sensors [31]. Some camera types such as stereo-cameras are able to compensate this deficit by using a SECOND camera sensor placed next to the primary sensor. Due to mass production of CMOS-based camera sensors from a variety of manufacturers, camera costs are quite low compared to LiDAR sensors and widely available. A high frequency of data acquisiton e.g. 50-60 Hz is common for most CMOS-type camera sensors, while the Field of View (FOV) may show a notable difference among different camera types.

2.2.2 LiDAR Sensors

In general, LiDAR sensors include an emission source used to produce electromagnetic waves at a wavelength above the visible spectrum, e.g. at 905 nm ± 10 nm for the Valeo Scala Gen2 LiDAR sensor [59] or 865 nm for the Ouster OS1 64-layer LiDAR [41]. The emission source may be mounted on a rotating axis and swivelled in a circular motion alongside a receiver for a 360 degree FOV. Emmited LiDAR waves are reflected by objects in the surrounding environment of the LiDAR sensor. Similar to radar sensors, the difference between an emitted and received LiDAR pulse is measured in addition to the current angular position in spherical or cylindrical coordinates. Based on this difference, the range, or distance between object and sensor may be measured [65].

In Fig. 2.4, a LiDAR point cloud recorded on an Ouster OS-1 64-layer sensor is shown accumulated over a time frame of one SECOND. The previous locations of moving objects such as a trailer truck within the sensor range is visualized by a lighter red hue.

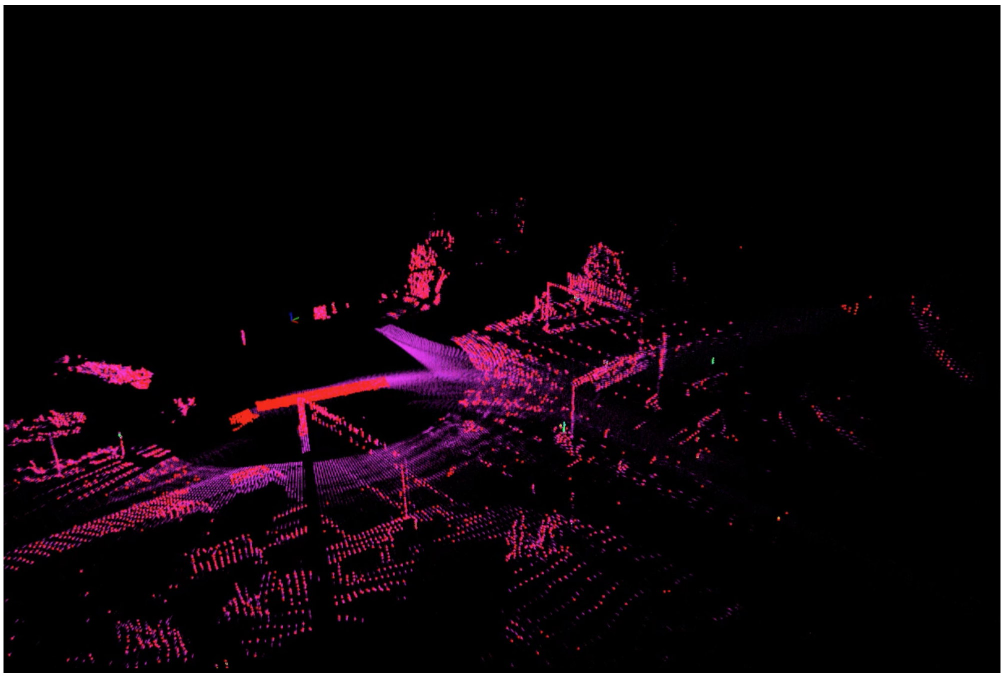

Figure 2.4: Accumulated LiDAR point cloud collected on an Ouster OS-1 64-layer sensor

For LiDAR sensors, a lower rate of data acquisition than e.g. camera sensors is common, approximately in the range of 10-20 Hz. Collected data is mapped into point clouds with accurate relevant geometric and spatial information. While LiDAR data is not susceptible to environmental influences such as bad lighting conditions, the sparsity of LiDAR point cloud data may pose a difficulty for some object detection methods. Precipitation or sensor occlusions may also lower the ability of a LiDAR sensor to perform accurately. Recent trends in LiDAR sensors include a shift away from mechanical solutions towards chip-based approaches as well as towards solid-state LiDARs without mechanical (spinning) components [65]. Spinning components may deteriorate more easily after a certain amount of operating time, possibly leading to degraded data quality and sensor functionality. In addition to standard point cloud data, fixed resolution depth images, as well as signal intensity, ambient, range, and near-infrared images are produced by sensors such as the Ouster OS-1 64-layer LiDAR and output as a full 360 degree image. As a result of this, e.g. object classification can be carried out on e.g. the ambient images. In Fig. **??**, an overview of sensor performance under different environmental conditions and for different tasks is given. Overall, no single sensor is able to fully cover the entire spectrum of listed tasks under all environmental conditions, however, combinations of two or three different sensor types can successfully manage this.

Specification	**Valeo Scala Gen2** [59]	**Ouster OS-1 64-layer** [41]
Vertical Resolution	16 channels	64 channels
Vertical Angular Resolution	0.6°	0.35°– 2.8°
Range	150 m	120 m
Points per SECOND	403,250	1,310,720
Vertical FOV	10°	45°
Horizontal FOV	133°	360°
Accuracy	±3 - 10 cm	0.7 cm @ 0.3-1m 1 cm @ 1-20 m 2 cm @ 20-50m 5 cm @ > 50m
Rotation Rate	25 Hz	10 or 20 Hz

Table 2.1: Key LiDAR sensor specifications

2.3 Object Detection in Autonomous Driving

According to Pendleton et al. [44], automotive vehicle software mainly includes the tasks of perception, planning and control. Within the scope of this thesis, a focus is placed on the task of perception. This includes the collection of data, extraction of information from data and interpretation of knowledge based on information. Object detection as a part of the perception task, consists of two main components, the identification of input objects, also referred to as classification, as well as the localization of objects within sensor data by bounding box encoding. In most settings, object detection is carried out for input data featuring multiple objects. Here, the general approach is to first extract features from input data, such as moving a sliding window across an input image in a rasterized manner. Then, objects are classified within proposed bounding boxes and object position, size and orientation can be refined. Throughout the past decade, as computational costs have sunk and both CPU and GPU computing performance have scaled significantly, the use of Convolutional Neural Nets (CNNs) or more specifically Deep Learning (DL) methods has become increasingly adapted.

Performance aspect	LiDAR	Radar	Camera
Privacy-safe data	🟢	🟢	🔴
Detailed geometric information	🟢	⚪	🔴
Accurate velocity measurement	🟢	🟢	⚪
Accurate distance measurement	🟢	🟢	🔴
Performance in bad lighting conditions (glare, nighttime, shadows)	🟢	🟢	🔴
Performance in bad weather conditions (rain, fog, snow)	⚪	🟢	🔴
Ability to differentiate textures and colors	⚪	🔴	🟢

Table 2.2: Autonomous driving sensors performance aspects based on [42].
(Tendencies: green = positive, grey=neutral, red=negative)

Fully connected layers, in which all neurons of one layer are connected to all neurons in the previous layer are computationally expensive. Due to the lower number of connections posessed by convolutional layers in CNN, it was possible for computational costs e.g. for high-resolution input images to be significantly reduced.

Object detection methods are generally seperated into single-stage and two-stage approaches. In single-stage methods such as SSD [35] or YOLO [47], detection candidates are directly generated in a single step. Instead, in two-stage methods such as R-CNN [19] or Faster-RCNN [49], candidate bounding boxes are extracted in a first step, followed by the refinement of initially proposed bounding boxes in a SECOND step. While single-stage methods are generally considered to be faster than two-stage methods, they are also often less precise. On the contrary, two-stage methods are generally slower than single-stage methods, but more accurate. In the following section, different approaches to 2D and 3D object detection will be presented alongside state-of-the-art single-modality and multimodality methods. For introduction purposes of fused 3D camera-LiDAR object detectors, an overview of camera-only 2D methods as well as LiDAR-only 3D detection methods is given. Depending on the external requirements, available datasets and end-user requirements, the approach and structure of these object detection methods, may significantly differ from one to the other. While some approaches may set new benchmarks in real-time speed, others display significant advances in accuracy, scaling or transfer learning properties, as well as robust learning strategies in order to prevent over- or underfitting and automate parameter and weight optimization.

2.3.1 2D Methods

In this section, 2D object detection methods are introduced. While humans generally have a strong ability to recognize certain visual patterns, in the past decade, as the use of CNNs has grown more and more popular, these networks rapidly started to outperform people in visual detection tasks such as on the ImageNet dataset [10]. In these types of networks, feature maps are extracted and output at every convolutional layer. In R-CNN [20], region proposals are created for an input image. Each proposed region is warped to a standard square shape before being run through an AlexNet [29] type backbone. A support vector machine is then used for simplification of the detected object candidates. Due to the large number of forward passes that the CNN needs to carry out for each region proposal, R-CNN is relatively slow. Also, in addition to the CNN, the classification head and bounding box regression have to be trained, which further slow down the network.

While Fast R-CNN[18] introduced a unified framework for the extractor, as well as classification and regression head, leading to increased speed, the region proposal component was still considered to be the bottleneck. Therefore, in Faster-RCNN [48], a region proposal network is implemented, consisting of a fully convolutional network placed above the features generated by the CNN. Bounding boxes estimated to be an object are then input into the Fast RCNN for classification and bounding box regression. Next to the convolutional layer approach, image segmentation and more exact semantic segmentation offers a possible approach to object detection. For segmentation, pixel or instance level annotations are created. For example, in Mask R-CNN [22] object detection is carried out at a pixel-level through an additional branch in which a binary mask is used to decide whether individual pixels belong to an object. Current state-of-the-art approaches for 2D object detection methods include YOLOX [14] and YOLOR [61]. YOLOR is an abbreviation of You Only Learn One Representation - Unified Networks for Multiple Tasks. It is characterized by a 3.8 % higher accuracy than PP-YOLOv2 and 88 % higher accuracy than Scaled YOLOv4 [60]. In the paper, a unified network is proposed in order to utilize implicit and explicit knowledge together, similar to the way some learning takes place subconsciously in the human brain. Here, explicit knowledge is linked to earlier network layers, while implicit knowledge representation is linked to extracted features or deeper layers. The unified network can be used to perform kernel space alignment, prediction refinement, and multi-task learning.

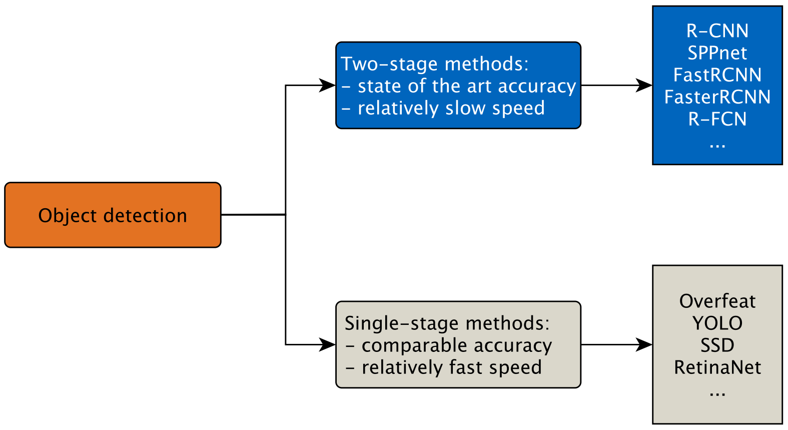

Figure 2.5: Scheme of detector types

In recent years, two-stage methods have been matched and even outperformed in terms of accuracy by single-stage methods such as YOLO-based networks. Most recent, YOLOX [14] and YOLOR [61] are two such state-of-the-art approaches for 2D object detection methods. Other than previous versions of the YOLO-type architecture, an anchor-free detection approach is featured in YOLOX. In addition, a decoupled detection head is used in order to avoid performance issues between the classification and bounding box regression tasks. As a result of these changes, performance in terms of accuracy and speed is increased. On the MS COCO dataset, an increase in accuracy of 1.8 % is achieved for YOLOX-L5 compared to YOLOv5L. Small model versions such as YOLOX-tiny have been shown to outperform comparable networks such as YOLOv4-tiny by an AP of 10 % [14]. In Fig. 2.6 below, the detached detection heads for YOLOX can be seen.

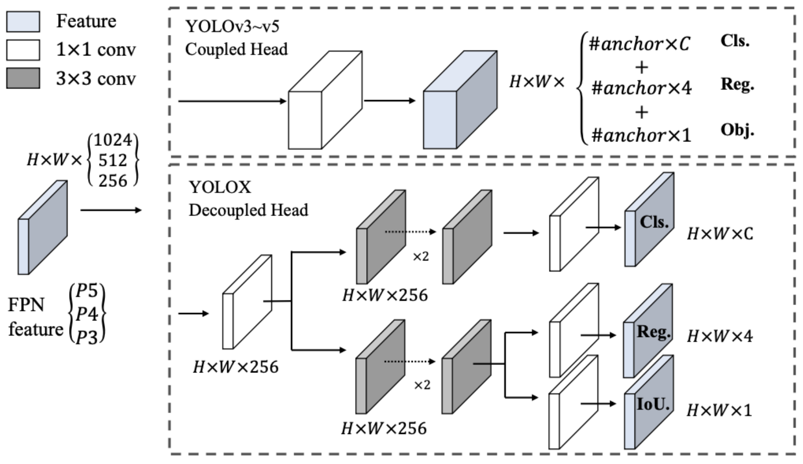

Figure 2.6: Detached detection heads of YOLOX [14]

While detailed color and texture information are generally contained in image data, monocular camera-only methods may lack depth information necessary for accurate 3D localization in autonomous driving tasks. Recently, more attention has been placed on pseudo-LiDAR methods containing a different method of data representation: Pseudo-LiDAR refers to the conversion of a previously estimated depth map of stereo or monocular images to a 3D point cloud. The hereby created pseudo-LiDAR point clouds can then be used as input to commonly available 3D LiDAR object detectors. Pseudo-Lidar point clouds have been shown to contain a greater number of points than LiDAR point clouds. As suggested by Wang et al. [64], the apparent mitigation of the performance discrepancy in 3D object detection performance from stereo images can be attributed to the data representation in pseudo-LiDAR. Based on experiments into this issue, the previously found discrepancy is not attributed to the performance of depth detection algorithms themselves [65].

2.3.2 3D Methods

Although a large number of neural network designs has been implemented for 2D object detectors in the past, 2D object detectors are affected by a number of performance limitations in the sense that they cannot directly be used for 3D object detection. As the basis for accurate 3D object detection, information such as object orientation, 3D location and bounding box size, as well as depth information is necessary in general. Due to the sparsity of point cloud data, some information on the detailed geometry of objects is lost in addition to color and texture information, while distance-information is preserved [67]. Due to the focus of this thesis fused camera-LiDAR object detection, first LiDAR-only methods are introduced, followed by fused camera-LiDAR methods. LiDAR-only methods are mainly seperated into voxel, point-based and projection (view) based methods. For voxel-based methods, the 3D space surrounding the LiDAR-sensor is divided into evenly distributed grid-based sections referred to as voxels. Inside each voxel, points are grouped, sampled and stored in a sparse data representation. Due to the sparsity of point cloud data, information on the detailed geometry of objects may be lost in addition to available color or texture information while range information is preserved [67]. Well-known voxel-based LiDAR-only methods include VoxelNet [80], SECOND [70], PointPillars [32] and TANet [36] as well as CIA-SSD [**zheng2020ciassd**].

Two methods that exist for voxelization are hard and dynamic voxelization. Dynamic voxelization preserves raw point cloud input or voxel information [66]), since not only a pre-defined number of points can be considered for feature extraction. In addition, the need for padding voxels to a specified size such as in VoxelNet, is no longer required when using dynamic voxelization. In VoxelNet [80], an approach using feature extraction and bounding box estimation is combined in a single stage. Unlike previous approaches, point cloud input data is rasterized in voxels of equal size through a proposed voxel feature encoder instead of using hand-made features. Within the voxel feature encoder, points are grouped, sampled randomly and point-wise features are stacked to create a voxel-based feature set. Next, a region proposal network for which output object detections are generated is introduced. The end-to-end trainable approach proposed in VoxelNet is shown to outperform comparable LiDAR-only methods on KITTI at the date of publication.

The approach proposed in SECOND [70] is based on the use of sparse convolutions for representation of otherwise dense 3D convolutions. With an increasing number of voxels in point cloud representation, an increasing computational cost is noted. In VoxelNet, this process is partially sped up due to extraction of discriminative features [65]. However, the approach proposed in SECOND overcomes this due to a use of sparse representation within the network. A further benefit suggested, is that because ointcloud data can directly be transformed, data augmentation can be applied more easily. In SECOND, a ground-truth database for detected objects is set up and used during training. In addition, an angle loss regression is proposed, which is shown to produce higher accuracy performance in regressing bounding box orientation. Overall, a speed-up of factor four is achieved during training on KITTI when compared to VoxelNet.

One of the challenges in 3D object detection using point clouds is the correct determination of 3D bounding box orientation. In practice, the orientation of objects may deviate significantly from default bounding box anchor orientations, which may lead to reduced performance in object detectors. In Centerpoint [73], it is proposed that 3D objects are considered as points instead. Due to the treatment as rotation-invariance of points, 3D object centers can be more easily aligned with object bounding boxes. A so-called keypoint detector is used for identification of object centers. Characteristic object properties such as size, orientation, and velocity are attributed to each keypoint and regressed. In a SECOND stage, estimates are refined through the use of point features. The detection head is implemented in the form of a heat map in Bird's Eye View (BEV). Center locations of objects are displayed as local peaks in the heat map. Furthermore, the sparse resulting map is passed to a 2D detector. The regression head is focused on height above ground, 3D size, yaw and location refinement. Both VoxelNet and PointPillars are available as backbones for Centerpoint.

In CIA-SSD [**zheng2020ciassd**], a single-stage, anchor based approach with a multi-task head is proposed. While other approaches have previously suggested separate localization and classification, in CIA-SSD these tasks are combined. A lightweight module is introduced for fusion of low spatial features and high level semantic features. Furthermore, a confidence-aware rectification module is introduced, leading to an overall improvement in the consistency between confidence and localization. Redundant predictions removed by an Intersection over Union (IoU)-weighted Non-Maximum Suppression (NMS), leading to a moderate AP of 80.28 % and frame-rate of larger than 32 FPS on the KITTI 3D test set. Methods such as PointPillars [32] or TANet [36] are outperformed in terms of accuracy.

In point-based methods, raw point clouds are used as input for 3D object detectors. Downsampling may be employed when using point-based object detectors in order to reduce the computational load of raw point cloud processing. [65]. Prominent examples for point-based LiDAR-only methods include PointRCNN [53], as well as 3DSSD [71]. Oftentimes, these methods are based on PointNet [45] or its successor PointNet++ [46].

With 3D-SSD, a point-based, 3D single stage object detector is introduced [71] using an anchor. A fast inference speed of 25 FPS and high accuracy of 79.57 % on the moderate KITTI test set are achieved despite the use of raw point cloud input data instead of voxels. In 3D-SSD, sample key points are used to extract features based on abstraction modules. The extraction of raw point features from key point features may lead to slower inference. Instead, a point sampling strategy including a feature distance metric for key point sampling is proposed in addition to the Euclidean distance. Since 3D-SSD employs an anchor-free approach, a shift is then added to candidate points for improved center predictions. Based on these improvements, an inference speed of 25 Hz is reached using 3D-SSD.

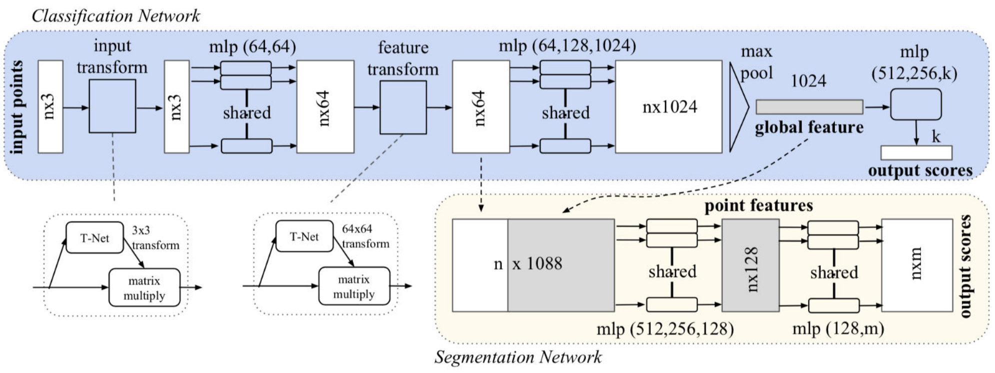

Figure 2.7: PointNet architecture overview [45]

In PointNet [45], input and feature transformations are carried out directly on unordered input data in the form of point clouds. In regard to point cloud input data, the idea of permutation invariance is considered in the paper. While some functions used for data representation are reliant on dimensionality, or the number of input dimensions, symmetric functions can be applied in order to map data in a permutation invariant manner. A similar requirement is invariance in regard to transformations carried out on an object such as rotation or translation. Here, a multi-layer perceptron is applied in order to map a set of input points. Once points are encoded in a feature space of a higher dimension, a global feature vector is created. Max-pooling is then applied in order to map multiple inpust to a single output value. After max-pooling, classification scores are calculated and output. An additional segmentation network is used to provide point-based features. Global point cloud features are calculated from concatenated classification and segmentation features. A downside to the approach suggested in PointNet is that locally, structures may not be fully captured.

Published on the basis of PointNet, PointNet++ [46] introduces measures to solve the issue of local structures not being fully captured. PointNet++ can be characterized as a two-stage, anchor-free model using a multi-task loss. Local features or structures in point cloud data are captured by recursive application of PointNet. The learning process of the model during training is constructed in hierarchical manner. Set abstractions, consisting of a sampling and grouping process for features followed by an application of PointNet can be applied in order for classifcation and segmentation. Multi-scale grouping as well as multi-resolution grouping are investigated as examples of density adaptive layers. In general, point cloud density is shown to have a strong influence on the implemented methods.

Projection (view) based methods are characterized by a projection of point cloud data into a 2D representation. For example, a projection into a BEV perspective is common for autonomous driving tasks. Conventional 2D CNNs can then be used to extract features from projected data. While some information may be lost due to the projection, a higher computational efficiency can often times be achieved using 2D CNNs [65]. Some examples for point-voxel-based LiDAR-only methods include STD[72], PVCNN [37], PV-RCNN[52] as well as SA-SSD[21].

In SA-SSD [21], a single-stage detector, 3D convolution is applied to voxels inside a backbone network, while 2D convolutions are used for refinement and regression. The backbone network is tasked with extracting sparse 3D convolutional features. Output of the backbone is passed to an auxiliary network at different stages where feature interpolation is carried out for point-wise supervision. In addition, the backbone output is passed to a detection network in which 2D convolutional features are further reshaped before being output as classification and bounding box scores. PSWarp or feature map warping is applied to these scores for alignment. As a result, an inference speed of up to 25 Hz similar to two-stage detectors is achieved.

Another single-stage detector for point cloud-based object detection is represented by SE-SSD [78]. In this approach, a student and teacher Single Shot Detection (SSD) are tasked with 3D bounding box prediction. Input received by the student consists of hard targets as well as input augmented by a custom shape-aware data augmentation. A custom orientation-aware distance loss function (DIoU) is defined in order to supervise the student based on hard targets. Student predictions and transformed teacher predictions, also referred to as soft targets are matched based on IoU before application of a consistency loss in combination with bounding box and classification scores.

Camera-only methods for 3D object detection are generally split into monocular and stereo camera image approaches. As introduced in the previous section on camera sensors, depth information is not contained in monocular images. After 2D prediction of candidate bounding boxes, oftentimes a neural network is used to predict missing depth information in order to project 3D bounding boxes. In other approaches such as Cheng et al., 3D object detection proposals are placed in 3D space based on an intersection with the ground plane, since objects should be located on the ground, followed by a refinement of the projected bounding box based on size and location priors.[7, 65].

In PointPillars [32], point cloud data is mapped to voxel-pased pillars. Each 4D point

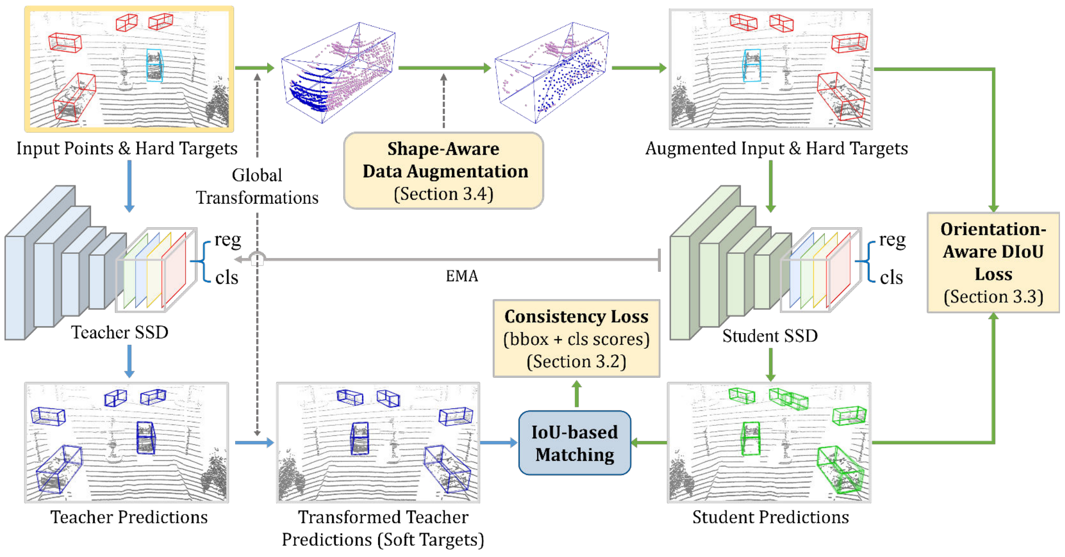
Input Points & Hard Targets
Global Transformations
Shape-Aware Data Augmentation (Section 3.4)
Augmented Input & Hard Targets
Orientation-Aware DIoU Loss (Section 3.3)
reg
cls
Teacher SSD
EMA
Student SSD
reg
cls
Consistency Loss (bbox + cls scores) (Section 3.2)
IoU-based Matching
Teacher Predictions
Transformed Teacher Predictions (Soft Targets)
Student Predictions

Figure 2.8: SE-SSD pipeline [78]

containing information on 3D position and reflectance is mapped to a 9D vector containing distance measurements to the center of a pillar and the mean of the pillar. Furthermore, a feature encoder is used for the conversion of point cloud data into a 2D pseudo-image. A backbone in the form of a 2D CNN is then used to extract features from the pseudo-images. In a last step, a SSD is used for 2D bounding box regression before class prediction results are output. Due to the high inference speed of 42 Hz achieved by PointPillars, the architecture is introduced in more detail in the related works section.

Built on the success of PointPillars, a new approach is introduced in PointPainting [50]. The network architecture is made up of three stages. First semantic segmentation is carried out for a calculation of pixel-level segmentation scores on input images. LiDAR point clouds are then labeled with the previously calculated segmentation masks. The painted point clouds in addition to the x,y,z coordinates and intensity contain class scores. In a third step, a LiDAR-based 3D object detector such as PointPillars is applied to the painted point clouds. In comparison to the PointPillars baseline, only a small latency of 0.75 ms is added in Pointpainting. At the same time, an increase in mAP for the pedestrian and cyclist classes part of the moderate KITTI 3D validation benchmark of 4.57% and 2.84% respectively, is achieved.

Based on previous implementations such as PointPillars and Painted PointPillars, in Fusion Painting: Multimodal Fusion with Adaptive Attention for 3D Object Detection [67] a novel approach is introduced. In FusionPainting, a multimodal fusion of RGB images and LiDAR point clouds is proposed at a semantic level in order to increase 3D object detection performance. Unlike PointPainting, FusionPainting features a separate painting module for the 2D image input data and 3D point cloud input data. A 2D and 3D segmentor are applied to input data before applying the segmented masks (painting). Painted 2D and 3D data is passed to a novel Adaptive Attention Module before the painted point cloud data is input into a 3D object detector. Using this approach, a higher detection performance is achieved over lidar-only methods and methods in which lidar-data is painted with semantic segmentation. On the nuScenes 3D detection benchmark, a 12.3% increase in the nuScenes Detection Score (NDS) score is achieved by FusionPainting in comparison to the previously introduced painted PointPillars. In comparison to CenterPoint [73], a 3.1% increase in NDS score is achieved.

In PV-RCNN elements of a voxel-based and PointNet-based approach are combined. A number of key points are sampled from point cloud input data in order to encode a whole scene. At the same time, voxels are created from the raw point cloud and encoded to key points. After key points are encoded, a refinement is carried out. As the influence of foreground key points should be larger than that of background keypoints, predicted key point weighting is carried out. In a final step, key point features are aggregated to Region of Interest (ROI) grid points for 3D bounding box refinement [53]. In PV-RCNN++ [52], an improvement of +4% mAP and +6 FPS on Waymo One is achieved due to a novel proposed sectorized proposalcentric keypoint sampling strategy. Key points are concentrated to be around 3D proposals for more effective encoding of features as well as scene refinement. In addition, a more efficient local feature aggregation method is introduced in which local point cloud features are considered. In Fig. 2.9, an overview of the architecture used in PV-RCNN is displayed.

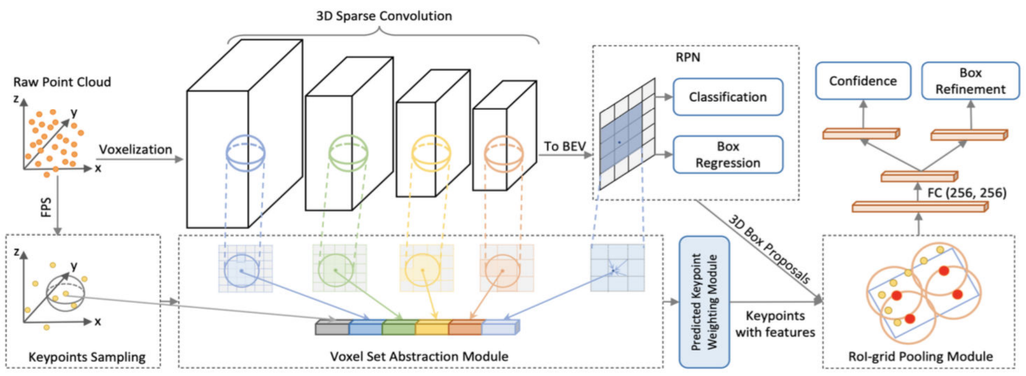

Figure 2.9: PV-RCNN architecture [52]

2.4 Multimodal Sensor Fusion

As introduced in the previous section on sensor modalities for object detection in autonomous driving, the benefits of individual sensors such camera, radar or LiDAR may vary depending on environmental conditions and intrinsic sensor characteristics. The use of multiple sensors or modalities in order to increase detection robustness and accuracy through redundancy is therefore being considered increasingly. For the full exploitation of individual sensor strengths, data captured by different sensors and sensor types first needs to be consolidated in a common frame of reference. This combination of data from different sensors is generally referred to as sensor fusion. In case different sensor types are used, it is referred to as multimodal sensor fusion [44].

Sensor fusion is generally divided into early and late fusion approaches. Early fusion is a term used to describe e.g. pixel-level correspondence between sensor data such as data from two identical camera sensors. Late fusion is characterized by the fusion of representative features extracted from the individual object detection pipelines of different sensors. Until recently, object detection algorithms based on fused LiDAR-camera systems have not significantly outperformed e.g. LiDAR-only methods evaluated on autonomous driving datasets such as KITTI or Nuscenes [2, 15]. Methods such as CLOCs [65] aim to change this by using a fusion approach in which object detection candidates are determined individually in the camera and LiDAR detection pipelines and fused in a late step. A near real-time latency may be considered as a benefit of the CLOCs approach.

Late fusion on the other hand is characterized by the fusion of representative features extracted from individual object detection pipelines of different sensors. Next to CLOCs, which is described in more detail in the section on related works, the approach suggested in Fast and Accurate 3D Object Detection for LiDAR-Camera-Based Autonomous Vehicles Using One Shared Voxel-Based Backbone [66] may be interesting for fused camera-LiDAR based 3D object detection. Unlike region-based or point-fusion based approaches, here a so-called Point Feature Fusion Module is used in order to extract pointwise features from RGB image data which is then fused with pointwise features from voxelized LiDAR point cloud data. As the premise for this approach, it is considered that object detection backbones for different sensors consume memory and may be slow. A Voxel Feature Encoder (VFE) and 3D-backbone are used alongside a detection head for 3D bounding box classification and regression. As a result of this, an inference speed of 17.8 FPS is achieved on the KITTI dataset.

In ROIFusion proposed by Chen et al [6], a deep fusion approach is used to fuse camera and LiDAR data in a lightweight network, achieving state-of-the-art results on the KITTI 3D dataset. In a first step, a fused keypoint layer containing sparse 3D keypoint estimations for point and pixel-based regions of interest is created. Then, 3D ROI in point cloud data are combined with 2D ROI from image data instead of dense point-wise features for keypoints used in other approaches. Subsequently, deep fusion is carried out on extracted regions of interest at a reduced computational cost when compared to dense pointwise fusion methods.

In autonomous driving research regarding object detection, different sensor combinations such as camera-radar and camera-LiDAR fusion are investigated. In camera-radar fusion, the accurate determination of distance and velocity measured using radar sensors is combined with the detail-rich information of camera images. While this type of sensor fusion is investigated by some works such as [4], a majority of research today seems to be focused on camera-LiDAR approaches. Here, the sparse, but spatial-information rich LiDAR-data is combined with detail-rich camera image data is combined. Although radar and LiDAR data are both stored in the form of sparse point clouds, LiDAR point clouds tend to be much denser than radar and in automotive applications have a larger FOV. A consideration that may need be taken into account for real-time applications is the additional computational overhead inherent with the use of fused sensor approaches. Further challenges in multimodal sensor fusion are presented e.g. by data acquisition points that may be different between individual sensors in case sensors are not properly synchronized.

Camera-LiDAR-based object detection algorithms include point-fusion methods and region-fusion approaches. Some commonly known point-fusion based methods include 3D-CVF[74], a two-stage, anchor based detector using voxels and a projection of RGB-data into BEV-images. These images are in turn fed into a conventional CNN. Another example for point-fusion methods is represented by MVX-Net[54], while commonly used methods based on region-fusion are for example MV3D [8], AVOD[30], MMF[33].

In MV3D [8], a camera-LiDAR approach is suggested using a multi-view representation. The proposed network is split in two, consisting of a proposal network for 3D objects, as well as a network tasked with fusion of the extracted multi-view features. A LiDAR BEV is used for generation of anchor boxes within the first proposal subnetwork. Features are linked to regions from which they are extracted, allowing for a deep fusion approach using multi-view based feature maps. Different from other object detection approaches, 3D box regression is carried out in MV3D in consideration of object orientation. Bounding boxes for objects detected in the proposal network are then projected into an front view image for classification.

MMF, or Multi-Task Multi-Sensor Fusion for 3D Object Detection is an anchor-free detector based on two-stage fused methods such as MV3D [33]. In comparison to previous related works, MMF includes a depth estimation for an approach that can be considered similar to pseudo-LiDAR point cloud creation. A point-based as well as ROI-based feature fusion is implemented for mapping of LIDAR points with multi-scale features, while a ROI-based feature fusion is used for projection of 3D bounding boxes into BEV images as well as the camera image.

For fused object detection methods such as those based on camera-LiDAR sensor data, a number of different possible solutions exist. One possibility consists of, a seperate object detection pipeline for each sensor type. In a last step, at a feature level object detection candidates are fused in order to provide multi-modal 3D object detection. One example for this type of multi-modal architecture is represented by Camera-LiDAR Object Candidates (CLOCs) [43]. Here, a two-dimensional object detection network is tasked with feature extraction from image data and a 3D network is tasked with feature extraction of pointcloud data. For the 2D neural network, e.g. a YOLO variant can be used, for the 3D point cloud based neural network, an approach such as PointPillars. The benefit of to this type of network is that the object detector of each sensor modality can be optimized and fine-tuned on an individual basis. However, there may perhaps be less control of the combined result. Due to the late fusion approach employed in CLOCs, detection candidates are combined before NMS which means that all potential single modality detection candidates are usable. More specifically, 2D-detections and 3D-detections act as input for a new tensor containing IoU, 2D-, 3D-detection confidence and normalized distance. Geometric and semantic consistency are ensured as the IoU of 2D bounding boxes and projected 3D bounding box corners have to be zero. For semantic consistency, only 2D and 3D detections of the same class are associated.

Next to CLOCs, an interesting approach for LiDAR is represented by the One Shared Voxel-Based Backbone [66]. The premise is that separate object detection backbones for different sensors consume memory and may be slow. Instead, point features of raw RGB-images are combined with point cloud features leading to an elimination of the image backbone here. A novel pointwise fusion between point cloud and RGB images is introduced, whereby a pair of pointwise features is extracted from raw RGB image data and point cloud data. Fused pointwise features are used as input to a 3D neural network, where a detection head is tasked with 3D bounding box classification and regression. Hereby, a framerate of 17.8 FPS is achieved on KITTI, with a 3D object detection performance of 88.04 %, 77.60 %, 76.23 % for the easy, moderate and hard category, respectively using a TITAN RTX GPU.

In Multimodal Virtual Point 3D Detection [73], 2D instance segmentation is carried out for input image data. The resulting semantic segmentation mask is used to identify LiDAR points attributed to individual objects in the foreground. Pixels located inside each object bounding box are sampled randomly to prepare locations of virtual LiDAR points. In order to assign a location to these virtual locations inside each bounding box, distance to the sensor origin is estimated via an interpolation of LiDAR points previously attributed to an object bounding box. Using nearest-neighbor depth interpolation between pixels and projected LiDAR points, virtual points are sampled and individually mapped to a 3D location. Resulting point clouds are more dense than input point clouds and may be passed on to standard 3D object detection networks. Due to multimodal virtual points, an increase in accuracy compared to the CenterPoint baseline of 6.6 % mAP is observed on nuScenes. It is suggested, that especially the sparsity of point clouds such as those of small distant objects like pedestrians or cyclists is improved due virtual points.

2.5 Autonomous Driving Datasets

In the following section, an overview of current autonomous driving datasets is given. While an increasing number of datasets are being published for the autonomous driving context, there is a large variety in the collected dataset size, application area and level of annotation. In KITTI [15], recorded data scenes are divided into three categories depending on the number of movable objects such as vehicles into easy, moderate and hard scenes. Data is recorded in an urban environment where up to 15 cars and 30 pedestrians are visible in each frame. Benchmarks on 2D and 3D detection, as well as tracking have been used as one of the main points of comparison for object detection performance in the autonomous driving context. Originally released in 2019 with perception data and labels for 1950 segments, the Waymo Open Dataset was extended in March 2021 by the addition of more than 100k segments featuring object trajectories and map data. Within the perception dataset, synchronized camera and LiDAR sensor data is included recorded across a number of American metropolitan areas. Four classes are labeled: vehicles, pedestrians, cyclists as well as signs. Annual challenges hosted by Waymo have recently been expanded with the real-time 3D object detection challenge being of great interest [56].

In 2019, the nuScenes [2] dataset was released by Motional, formerly known as nuTonomy. This dataset includes 1000 scenes with a length of 20 SECONDs each, collected in the dense urban hubs that are Boston and Singapure. At the time, the nuScenes dataset was one of the first autonomous driving datasets to include data from five radar sensors placed on the data recording vehicle. In addition, six cameras and a 360 degree LiDAR sensor provide further data alongside GPS and IMU readings. Overall, this resulted in approximately 7 times more data annotations than KITTI. While sensor recording rates are higher, keyframes are annotated at 2 Hz. For accumulation of a larger number of LiDAR sweeps, point cloud labels can be interpolated based on these keyframes. In 2020, the dataset was extended by nuScenes lidarseg in which 32 classes are labeled within point cloud segmentation annotations. Here, approx. 1.4 Billion LiDAR points are included across 850 training scenes and 150 testing scenes. Further refinement was presented in 2021 with the nuImages dataset, in which 93 thousand short video clips with 13 frames spaced out at 2 Hz are included. In NuImages, 93k 2D instance masks are annotated as well as 2D object bounding boxes [2, 3].

As of today, a number of publicly available datasets for autonomous driving include semantic segmentation image data. Examples include the SemanticKITTI dataset [1] in which 28 classes are labeled, including stationary objects such as ground elements, building structures, vegetation, as well traffic objects such as vehicles or pedestrians. Here, 360 degrees of LiDAR sensors sweeps are annotated on a point-based level. For some datasets such as Apolloscapes, Berkley Deep Drive, Cityscapes and Mapillary Vistas, the main emphasis is placed on camera-based object detection. Other recently published datasets include the A2D2 dataset published by AUDI [17] or the PixSet [11] dataset.

In the A2D2 [17] dataset, more than 40k data frames are included for point cloud and RGB-image data. A total of 38 classes are annotated for semantic segmentation images including road-surfaces, vehicles and vegetation. Next to 3D bounding box annotations for 14 classes such as cars or pedestrians, notably point cloud segmentation is available through the fusion of pixel-level semantic segmentation data and LiDAR point cloud data.

In the PixSet [11] dataset created by Leddar Tech, a novel approach is chosen for LiDAR data collection. On the basis of solid-state flash LiDAR, full-waveform data is collected and annotated with 3D bounding boxes as part of PixSet. LiDAR is collected on digital or mechan-

Dataset	Published	PC frames	RGB images	Classes
Caltech Pedestrian Dataset [12]	2009	-	250k	1
KITTI [15]	2012	15.4k	15k	8
Cityscapes [9]	2016	-	25k	30
Mapillary Vistas [40]	2017	-	25k	37
Berkley Deep Drive [75]	2018	-	100k	10
Argoverse [5]	2019	44k	490k	15
Apolloscape [23]	2019	80k	90k	8
Lyft Level 5 [26]	2019	323k	46k	9
nuScenes [3]	2019	400k	1.5M	23
Waymo [56]	2019	200k	1M	4
A2D2 [17]	2020	41k	41k	38
PixSet [11]	2021	29k	29k	22

Table 2.3: Overview of autonomous driving data sets for object detection (PC=point cloud, number of classes for detection, not necessarily segmentation)

ical spinning LiDAR sensors in the form of 3D point clouds. This however, is not the raw data form of LiDAR data but processed full-waveform data. Points can therefore be expressed as the peak of a wave function for an emitted LiDAR wave for which the intensity distribution is measured. While this type of LiDAR has already been more extensively used in aerospace applications, it is considered a relatively new concept in autonomous driving applications. Overall, PixSet consists of almost 29 k data frames distributed across 97 scenes. In addition to flash-LiDAR, camera and radar data is collected and associated with relevant IMU data of the sensor platform.

In terms of dataset size, Mapillary vistas is larger than Cityscapes by a factor of 5. Data is collected across all continents at street level and annotated with semantic annotations at a pixel and instance level. In the table below, an overview of different autonomous driving data sets is given. The focus of comparison is placed on the number of frames containing point cloud data, RGB image data as well as the number of annotated classes for object detection. The Lyft dataset [26] contains a large amount of data collected on vehicles operated as part of Lyft's autonomous vehicle fleet in the United States. Within the Lyft dataset, the nuScenes data format is used. As of 2020, the perception level 5 dataset has been published [27], increasing the total number of 3D bounding box annotations as well as 30 k LiDAR point clouds.

In the recently published ONCE dataset [39], data is collected using vehicles equipped with seven cameras as well as a LIDAR sensor. The dataset name, ONCE, is an abbreviation of one million scenes, since the dataset contains one million LiDAR frames as well as seven million camera images. Data is recorded in both urban and suburban areas, and 15 k frames are fully labelled for five classes including a car, bus, truck, pedestrian and cyclist class. For ONCE, a benchmark is published on its validation set for common object detectors such as PointPillars. Across individual scenes and frames of described autonomous driving datasets, the number of represented object classes, dimensions, visibility and orientation may vary greatly. While some datasets are aimed at X statistical distribution, it may be difficult to ensure a high quality or consistency of included data. Imbalances in data distribution across samples or scenes can result in adverse effects on object detector training and inference later on. This is due to networks not being able to generalize on previously unseen data properly. Commonly seen examples for these imbalances are e.g. the over-representation of individual classes such as cars with a simultaneous under-representation of e.g. trucks. In order to circumvent the effects on training, focal loss [34] introduces a method of dealing with foreground-background imbalance. Weighted loss functions may also be a step in achieving better object detector training.

2.6 Data Augmentation

In the previous section, common challenges in regard to dataset composition and data distribution across different sample frames, scenes and datasets are named. An approach to deal with challenges such as these is represented by data augmentation. In a basic sense, data augmentation is used to increase the ability of object detection networks to generalize on previously unseen data. Common data augmentation methods are e.g. rotation, translation, the addition of Gaussian noise, as well as warping. Before or during training of the object detection network, these methods are applied to training data. As a result of this, features extracted from input data are found in a more robust way. In addition, since labelling costs for LiDAR data are generally high, data augmentation reduces dataset requirements in terms of size.

Recently, data augmentation has received more attention in regard to multimodal object detection. Since individual perception sensors for autonomous driving are assumed to possess strengths and weaknesses, a combination of several of these sensors such as LiDAR sensors and cameras should in theory be able to produce more informative data or descriptions and therefore better performance results than e.g. a LiDAR-only object detector. Since this does not appear to be the case for a number of current camera-LiDAR object detection methods, a multi-modality data augmentation, Multi-mOdality Cut and pAste (MoCa) is proposed by Zhang et al. [77]. Due to the difficulty in ensured consistency between sensor modalities for data augmentation, multi-modal object detection methods are seen using fewer types of data augmentation than single-modality methods. In order to deal with this, a transformation flow is introduced as part of MoCa. With respect to object occlusions and physical validity, measures are included in MoCa. This is due to the fact that points in space may be simultaneously occupied by multiple objects in multi-modal detection methods. Through application of these proposed methods, the mAP of a PointPainting-based camera-LiDAR approach is improved by 1.5% on the nuScenes validation dataset and almost 10% for a MVXNet based approach on KITTI [77].

Other promising approaches are represented by e.g. LiDAR-Aug [13] or PointAugmenting[62]. In LiDAR-Aug, it is criticized that object occlusions are not necessarily taken into consideration in other LiDAR-based data augmentation approaches. Global augmentation such as rotation, translation, flipping of data and and ground-truth augmentation in the form of copying and pasting of data from one frame to another. In ground-truth augmentation, occlusion between objects and background is not taken into account. As a result, this type of data augmentation cannot be applied to projection-based 3D object detectors such as MV3D. Therefore, a novel approach is suggested in LiDAR-Aug for pose generation while avoiding collision between augmented objects and ground truth. Here, rendering is used for proper foreground-background composition. Due to this consideration of object occlusions due to data augmentation placement, a higher mAP is demonstrated on KITTI. Improved results are shown on different backbones e.g. PointPillars, SECOND, PV-RCNN.[13]

In PointAugmenting, a cross-modal data augmentation for 3D object detection is presented in which an augmentation of point-wise CNN-features is carried out. With the use of the proposed PointAugment method, virtual objects are pasted into images as well as point clouds, outperforming LiDAR-only baselines. The core element of this approach lies in the assumption that segmentation features are more useful than segmentation scores for fusion with LiDAR points. It is criticized that the simple process of copying and pasting ground-truth boxes in data augmentation for multi-modal object detection destroys the consistency between camera images and LiDAR point clouds. Therefore, the fusion of CNN features from 2D object detection with LIDAR points is proposed in order for 3D object detection. The process includes the voxelization of camera and LiDAR features and a backbone for 3D bounding box generation [62].

2.7 Evaluation Metrics in 2D/3D Object Detection

In autonomous driving research, some commonly defined metrics are used to assess the accuracy and performance of neural networks used for object detection. The IoU is used to represent the overlap between a candidate bounding box and its corresponding ground truth bounding box. The IoU threshold is generally set at a value of greater than 0.5 and may depend on the training and evaluation datasets used. When an object class is predicted correctly with an IoU higher than the set threshold for positive results, it is considered to be a true result [44, 65].

The number of true positive results to the number of all positive results is expressed by the true positive (TP). Other metrics calculated in a similar manner to the number of total positive or negative results are the true negative (TN), false positive (FP) and false negative (FN). Based on these terms of true/false negative and positive results, two key terms may be defined: precision and recall. The two terms are defined as:

$$Precision = \frac{TP}{TP + FP} \tag{2.1}$$

$$Recall = \frac{TP}{TP + FN} \tag{2.2}$$

By calculating the precision as a function of the recall $p(r)$, an integral from $r = 0$ to $r = 1$ can be used to output the average precision (AP), essentially calculating the area underneath the precision-recall curve[44]. For a single object class, while the mean average precision (mAP) introduced by Salton and McGill [51] is used to measure accuracy averaged across multiple object classes. Due to the nature of the recall and precision definitions, some shortcomings are sometimes criticized. For example, the localization and or orientation accuracy of objects is not measured. Some datasets or research projects choose to define custom accuracy metrics, however. Within the nuScenes dataset or object detection challenge, such a custom accuracy metric, known as nuScenes Detection Score (NDS) is calculated [3].

The average precision AP is calculated using the precision p and recall r:

$$AP = \int_0^1 p(r)\,dr \tag{2.3}$$

Mean average precision mAP is calculated by:

$$mAP = \frac{\sum_{q=1}^{Q} AP\,(q)}{Q} \tag{2.4}$$

where the number of queries is represented by Q. The NDS is calculated as:

$$NDS = \frac{1}{10} 5\, mAP + (1 - min(1, mTP)) \tag{2.5}$$

where mTP corresponds to the five significant mean values mATE, mASE, mAOE, mAVE and mAAE defined in the nuScenes detection benchmark [3].

Chapter 3

Related Work

In the following section, the 2D and 3D based approaches used for the development of a comprehensive real-time and 3D multi-modal object detection pipeline are described. The focus of the method is placed on YOLOR for image-based 2D detection, CenterPoint for LiDAR-based 3D detection and CLOCs for multimodal 3D detection. In addition, PointPillars is introduced as a backbone candidate for CenterPoint.

3.1 YOLOR

In You Only Learn One Representation: Unified Network for Multiple Tasks (YOLOR) [61], a fast and accurate image-based 2D detector is introduced. A unified network is proposed in order to utilize implicit and explicit knowledge together, similar to the way some learning tasks take place consciously or subconsciously in the human brain. Explicit knowledge is linked to earlier neural network layers, close to the input layer, while implicit knowledge representation may be gained in later stages or layers e.g. related to extracted features. Multiple tasks can be pursued by the unified network, potentially including semantic segmentation and keypoint detection in the future. Misalignments in kernel spaces e.g. for multi-task neural networks are reduced in YOLOR through an integrated kernel space alignment component. Also included in the YOLOR architecture are a prediction refinement and multi-task learning. In comparison to previous state of the art YOLO variants and networks such as Scaled YOLOv4 [60], YOLOR achieves a similar average precision on test data of 55.4 % compared to an AP of 55.5 % for Scaled YOLOv4 while displaying a significantly higher frame-rate of 30 FPS compared to 16 FPS for Scaled YOLOv4 [61]. In Image 3.1, a comparison of accuracy and speed performance between YOLOR and other state of the art 2D object detection networks may is provided.

In order for a modelling of implicit knowledge in a unified, multi-purpose network such as YOLOR, the objective functions of conventional and unified networks are introduced. In conventional networks, the objective function can be expressed by:

$$\begin{aligned} &y = f_\theta(x) + \epsilon \\ &\quad \text{minimize } \epsilon \end{aligned} \tag{3.1}$$

With the observation x, parameter set of a neural network θ, neural network operation f_θ and task target y. Then, a minimization of ϵ is carried out to reduce the difference between f_θ and network target y. In other words, a similar solution space is found based on different observations. For general purpose neural networks, ϵ may need to be relaxed or modelled in order to arrive at the solution to different simultaneous tasks. In unified networks, explicit

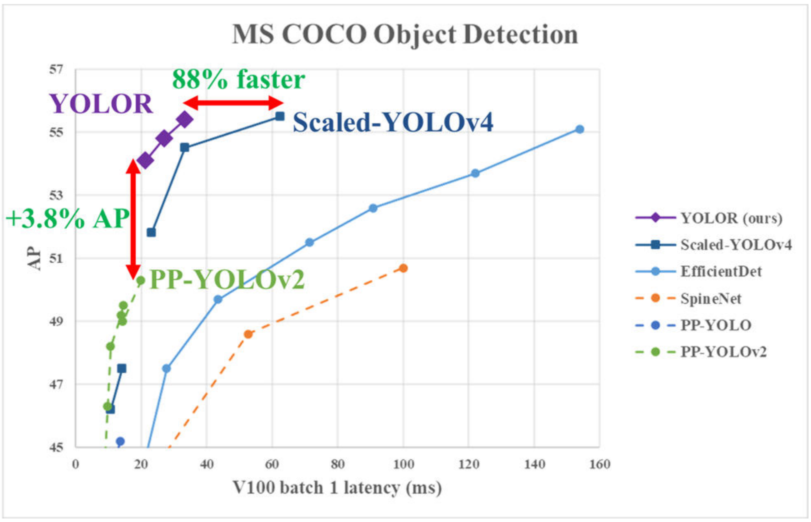

Figure 3.1: 2D Object detector latency vs. Average Precision (AP) on the MS COCO dataset [25]

knowledge is utilized in addition to implicit knowledge in order to model the error term. With the use of the implicit error ϵ_{im} and explicit error ϵ_{ex} resulting from an observation x and latent code z the equation is formulated as:

$$\begin{aligned} y &= f_\theta(x) + \epsilon + g_\phi(\epsilon_{ex}(x), \epsilon_{im}(z)) \\ &\text{minimize } \epsilon + g_\phi(\epsilon_{ex}(x), \epsilon_{im}(z)) \end{aligned} \tag{3.2}$$

Here, g_ϕ is used to describe a task-specific operation for information adaptation. With some methods capable of integrating explicit knowledge in the neural network operation f_θ, the equation can be rewritten:

$$y = f_\theta(x) \star g_\phi(z) \tag{3.3}$$

Thereby, the $\star$ operator is used as a placeholder for possible methods introduced in YOLOR for implicit knowledge modeling based on addition, multiplication and concatenation. A vector, matrix or tensor z is used as the basic representation of implicit knowledge. The application of a weight matrix $W * z$ may then result in linear combination or non-linearization of a implicit knowledge representation. Matrix factorization $zT * c$ is introduced as the third type of implicit knowledge modeling, whereby several vector bases are used in combination with a coefficient c. Constraints such as sparse or non-negative constraints can then be applied to c for further refinement. As the observation x is not affected by implicit knowledge, regardless how complex the implicit model g_ϕ is, it can be reduced to a set of constant tensor before inference. The result is a low impact on computational complexity.

3.2 PointPillars

In PointPillars [32], a highly efficient approach for object detection from LiDAR point clouds is proposed. Based on PointNets [45], in which point cloud data is represented in vertical columns referred to as pillars, features are extracted. Following data input in the form of point clouds, the main components of PointPillars include a Pillar Feature Net, CNN backbone, Detection Head before an output of predictions. Inside the Pillar Feature Net, point clouds are first mapped to pillars and stacked. Learned features are used to generate a 2D pseudo image which is used as input for a 2D CNN. The (SSD) detection is then used to

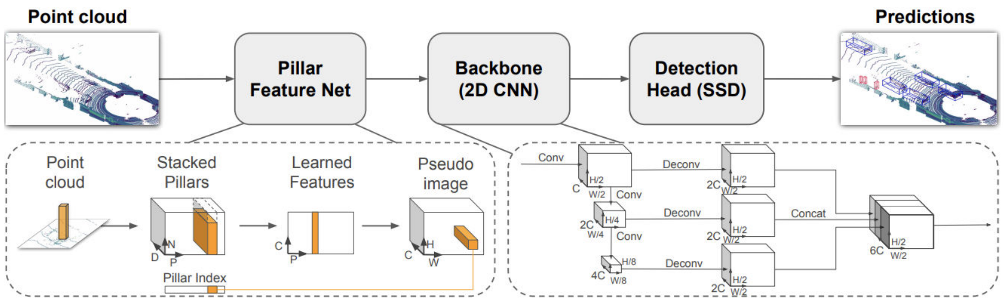

Figure 3.2: Object detection pipeline in PointPillars [32]

predict 3D bounding boxes based on features generated by the CNN backbone. In Fig. 3.2, an overview of this architecture is shown. Until recently, the PointPillars LiDAR-only method has outperformed LiDAR-only methods such as VoxelNet and SECOND [70] in terms of speed and accuracy. While 4.4 Hz and a 3D mAP of 49.05% is achieved on KITTI for VoxelNet and 20 Hz, with a 3D mAP of 56.69% on KITTI for SECOND, PointPillars is shown to achieve a 42 Hz frame-rate and 59.20% mAP on KITTI 3D. In addition, the accuracy and inference speed displayed by PointPillars has outranked a number of multi-modal object detection algorithms in the KITTI 3D benchmark.

After discretization of the input point cloud into a grid of pillars on the x-y plane, initial point representation of (x,y,z,r=reflectance) is augmented with x_c, y_c, z_c, p. Here, the distance to the arithmetic mean of all points in the pillar is denoted by the subscript c and the offset from the pillar x,y-center by the subscript p. The resulting augmented point vector/description D is 9-dimensional. As a result of the point cloud sparsity, most of the space contained in the pillars is empty. A dense tensor representation of the data is achieved using a limit on the number of non-empty pillars per sample P and the number of points per pillar N in the shape of (D,P,N). In case too much or too little data is contained in a pillar in order to fit into the dense tensor, random sampling or zero-padding are employed, respectively. Following the linear layer (1x1 convolution) for each point, Batch-Norm and ReLU are applied similar to PointNet, creating a tensor of shape (C,P,N). A channel-wise max operation leads to an output tensor of shape (C,P). Features are then mapped to the original pillar locations creating a pseudo-image. The pseudo-image has a size of (C,H,W) where H is used to represent the height and W the width of the canvas.

Inside the backbone, a top-down network and an upsampling network. Features of decreasing size are provided by the top-down network, while the upsampling network is used to concatenate extracted top-down features. The top-down network is formally described as containing blocks of shape (S, L, F), containing stride S, number of L 3x3 2D convolutional layers and number of F output channels - followed by BatchNorm and ReLU. Inside the upsampling network, blocks of shape Up(Sin, Sout, F) including initial and final stride S_{in}, S_{out} as well as final features F are used before BatchNorm and ReLU are applied. Inside the detection head, a Single Shot Detector (SSD) is used for 3D object detection. Here 2D IoU is used for calculating the overlap between object detections and ground truth bounding boxes. Object height and elevation are introduced as additional regression targets. Inside PointPillars, weights are initialized randomly following a uniform distribution. The encoder network has $C = 64$ output features.

In PointPillars, the loss functions are defined by the 3D location (x,y,z), 3D dimension (width, length, height) as well as 3D rotation (yaw, angle) which correspond to those functions implemented in SECOND [70]. The localization regression residuals between ground truth and anchors are defined by:

$$
\begin{aligned}
&\Delta x = \frac{x^{gt} - x^a}{d^a}, \Delta y = \frac{y^{gt} - y^a}{d^a}, \Delta z = \frac{z^{gt} - z^a}{d^a} \\
&\Delta w = \log \frac{w^{gt}}{w^a}, \Delta l = \log \frac{l^{gt}}{l^a}, \Delta h = \log \frac{h^{gt}}{l^a} \\
&\Delta \theta = \sin(\theta^{gt} - \theta^{h^a})
\end{aligned}
\tag{3.4}
$$

where x^{gt} and x^a are used to represent ground truth as well as anchor bounding boxes with $d^a = \sqrt{(w^a)^2 + (l^a)^2}$. The localization loss can then be expressed as:

$$
\mathcal{L}_{loc} = \sum_{b \in (x,y,z,w,l,h,\theta)} \text{SmoothL1}\,(\Delta b) \tag{3.5}
$$

In order to enable the network to learn object headings, a softmax classification loss is applied to the discretized directions $\mathcal{L}_{dir}$ since flipped boxes cannot be distinguished by the angle localization loss. For the object classification loss, the focal loss is expressed by:

$$
\mathcal{L}_{cls} = -\alpha_a (1 - p^a)^\gamma \log p^a \tag{3.6}
$$

where p^a represents the class probability of an anchor. The total loss function is:

$$
\mathcal{L} = \frac{1}{N_{pos}} (\beta_{loc} \mathcal{L}_{loc} + \beta_{cls} \mathcal{L}_{cls} + \beta_{dir} \mathcal{L}_{dir}) \tag{3.7}
$$

with $\alpha = 0.25$ and $\gamma = 2$ as in the original paper. N_{pos} is used to describe the number of positive anchors, while $\beta_{loc} = 2$, $\beta_{cls} = 1$ and $\beta_{dir} = 0.2$.

3.3 CenterPoint

With CenterPoint [73], an approach is introduced on the basis that an improved estimation of object orientation can be achieved when considering object centers as points. A keypoint detector is used for initial object center detection based on camera and LiDAR input data. As part of a first stage, keypoint detection is carried out followed by the regression of key bounding box attributes such as 3D object orientation, size and location. Confidence scores and object locations are then refined in a second stage as part of CenterPoint. With the methods implemented in CenterPoint, a mAPH of 72.8 % is achieved at 17.5 FPS on the Waymo 3D detection test set. In the nuScenes detection benchmark, a mAP of 67.1 % is achieved alongside a NDS of 0.714, outperforming a majority of related methods at the date of publication. In Fig. 3.4, an overview of the CenterPoint framework is given, including keypoint detection and the regression of 3D bounding box attributes in the first and second stages.

In a number of object detectors, bounding box anchors are pre-defined with a limited number of initial orientations. Objects oriented in an axis-aligned manner may also generally be well aligned with default anchor definitions, while objects diverging from this orientation due to road direction or vehicle maneuvers may result in misalignment and low IoU-scores. In Fig. 3.3, such a mismatch between predicted object bounding boxes and ground truth in anchor-based object detectors is shown. The possible mismatch between available anchor-based bounding boxes and object orientation is an issue that is circumvented by the approach suggested in CenterPoint, the premise of which is that object centers may be considered as

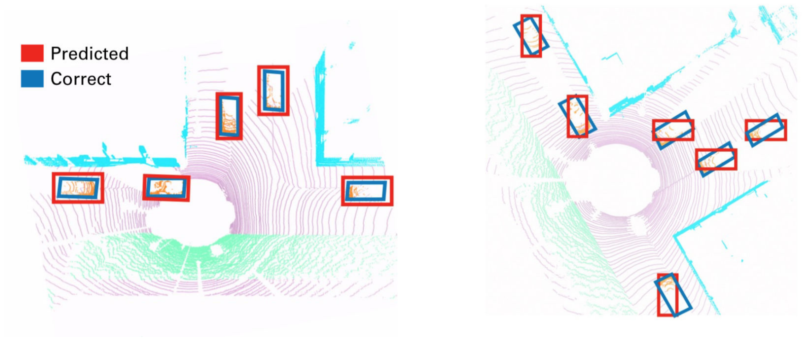

Figure 3.3: Mismatch between predicted vs. ground truth bounding box in anchor-based detectors [73]

rotation-invariant points. As a result, there is no need for definition of anchors and the overhead of regression for box orientation is reduced in the network.

Following the voxelization of point cloud input data, a BEV feature map is created from point clouds in CenterPoint. In the first stage, object centers and 3D boxes are calculated. A heatmap of objects is set up and the height above ground, bounding box size, rotation angles, subvoxel refinement are regressed. This is achieved using input processing similar to that employed for 2D images in 2D CenterNet [79], whereby a BEV heatmap is output. Within the heatmap, local maxima are attributed to object centers. In CenterPoint, LiDAR input data is fed into a 3D backbone such as VoxelNet [80] or PointPillars [32]. The output of the 3D backbone can be described as a BEV feature map $M \in R^{WxLxF}$. Focal loss is used in combination with a 2D Gaussian in the map view centered on ground truth bounding box projections during training. For each class K, a two-dimensional $w \times h$ heatmap $\hat{Y} \in [0,1]^{w\,h\,K}$ is thereby predicted. Convolutional blocks applied to the backbone are each made up of a 3x3 convolution, followed by Batch Norm and ReLU.

In the second stage, sequential Multilayer Perceptron (MLP)-blocks are used to output confidence and bounding box information. Each MLP block is made up of a fully connected layer, Batch Norm, ReLU loss function and Dropout. Since a number of features cannot be inferred with a high degree of accuracy from center-based features alone, a second stage is introduced in CenterPoint for object location and orientation refinement. Due to the CenterPoint architecture, other novel models besides VoxelNet or PointPillars can be implemented in the framework. In the heatmap, false positives have to be removed. Inference speed is decreased as a result of the two-stage approach to 57 ms and information may be lost during voxelization. VoxelNet tends to be more accurate, but slower.

Further modifications to the CenterPoint architecture are investigated such as the misalignment between localization accuracy and classification confidence voxelization performance in terms of speed. An IoU-aware confidence rectification module is suggested for the mitigation of this misalignment. For this purpose, an additional regression head tasked with prediction of IoU-scores between ground truth and object detection boxes is introduced. Dynamic voxelization is implemented on GPU as part of CenterPoint in order to increase the voxelization speed. As CPU-based methods are challenged in terms of real-time processing speed especially for multi-sweep input used for denser pointcoud accumulation, dynamic voxelization is considered a significant improvement. With the implemented dynamic voxelization, a speed-up of factor 10 is achieved, reducing voxelization per frame from approximately 50 ms down to 3 ms.

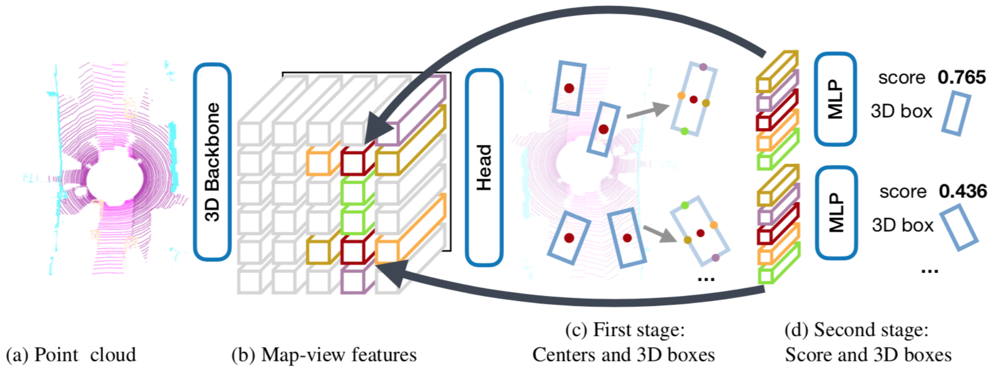

Figure 3.4: Overview of CenterPoint framework [73]

3.4 CLOCs

In Camera-LiDAR Object Candidates Fusion for 3D Object Detection (CLOCs) [43], a late fusion approach for 3D object detection using camera and LiDAR data is proposed. As such, the object detector of each sensor modality can be chosen and fine-tuned on an individual basis. However, the fused detection candidates rely heavily on the quality of the individual 2D and 3D object detections. Due to the late fusion approach employed in CLOCS, detection candidates from 2D and 3D detection pipeline are combined before non-maximum suppression. On the joint detection candidates, a 2D CNN is applied and from the resulting sparse tensor, a probabilistic score map is calculated. On the KITTI test set, improved results are achieved in comparison to single-modality state-of-the-art methods. For example a 1.8 % increase in 3D easy AP on KITTI 3D dataset is achieved by the fused camera-LiDAR detection of CLOCS_PVCas over PV-RCNN as well as a 3.04 % easy 3D AP increase in comparison to SECOND + Cascade R-CNN. Due to the lightweight model architecture, a latency of approximately 3 ms per frame is achieved.

In CLOCs, semantic and geometric consistency between 2D and 3D detections are considered based on learned probabilistic dependencies. As a basis for geometric consistency, it is assumed that accurate detections provided by 2D and 3D detectors should mean that 2D and 3D bounding boxes correspond in the image plane. The IoU between 2D bounding box and projected 3D bounding box is therefore used as a measure for geometric consistency, contributing to overall improved fusion performance/results. Although object detectors may output a number of classes during training, only detections belonging to the same class are considered during fusion as a measure of semantic consistency. Detection thresholds are set to low values at this stage and only increased before final output.

In order for both 2D and 3D object detection candidates to be used as input for the fusion network proposed in CLOCs, 2D and 3D object detection candidates are encoded. The set of 2D bounding boxes $\mathbf{p^{2D}}$ for candidate object detections in each image is expressed as a tensor consisting of k detections, where the i_{th} detection is expressed as:

$$\mathbf{p_i^{2D}} = \{[x_{i1}, y_{i1}, x_{i2}, y_{i2}], s_i^{2D}\} \tag{3.8}$$

In this expression, x_{i1-2} and y_{i1-2} are used to describe the pixel coordinates of the top left and bottom right 2D bounding box corners. The prediction confidence is expressed by s_i^{2D}. The set of 3D bounding boxes $\mathbf{p^{3D}}$ for candidate object detections in each LiDAR point cloud is structured similar to the KITTI type description consisting of a 7-digit vector containing 3D

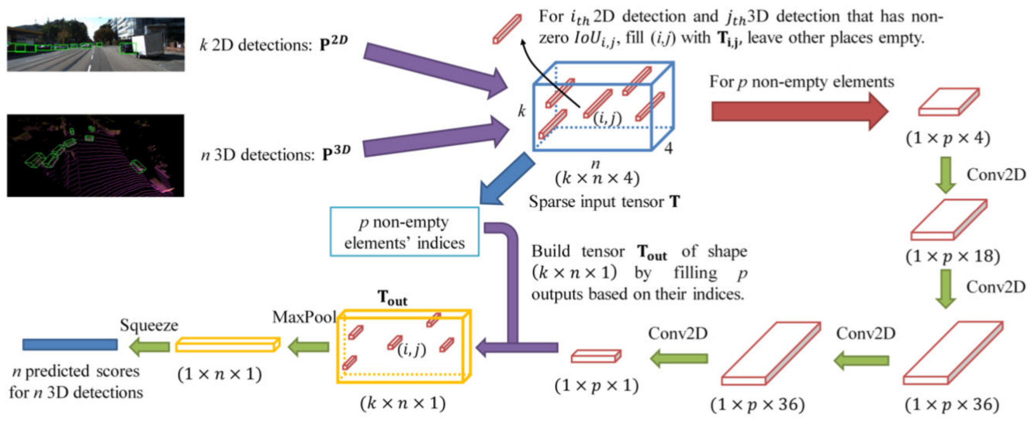
k 2D detections: $\mathbf{P^{2D}}$
n 3D detections: $\mathbf{P^{3D}}$
For i_{th} 2D detection and j_{th} 3D detection that has non-zero $IoU_{i,j}$, fill (i,j) with $\mathbf{T_{i,j}}$, leave other places empty.
(i,j)
k
n
4
$(k \times n \times 4)$
Sparse input tensor $\mathbf{T}$
For p non-empty elements
$(1 \times p \times 4)$
Conv2D
$(1 \times p \times 18)$
Conv2D
p non-empty elements' indices
Build tensor $\mathbf{T_{out}}$ of shape $(k \times n \times 1)$ by filling p outputs based on their indices.
$\mathbf{T_{out}}$
MaxPool
Squeeze
n predicted scores for n 3D detections
$(1 \times n \times 1)$
$(k \times n \times 1)$
$(1 \times p \times 1)$
Conv2D
$(1 \times p \times 36)$
Conv2D
$(1 \times p \times 36)$

Figure 3.5: Overview of CLOCs fusion framework [43]

dimensions (height, width, length), 3D locations (x,y,z) and 3D orientation (yaw angle) for n detections [15]. The i_{th} detection is therefore expressed as:

$$\mathbf{p_i^{3D}} = \{[h_i, w_i, l_i, x_i, y_i, z_i, \theta_i], s_i^{3D}\} \tag{3.9}$$

where s_i^{3D} is used to describe the prediction confidence. A $k \times n \times 4$ tensor $\mathbf{T}$ is then used to express the set of k 2D detection candidates and n 3D detection candidates. A single element of the tensor $\mathbf{T_{i,j}}$ is therefore expressed as:

$$\mathbf{T_{i,j}} = \{IoU_{i,j}, s_i^{2D}, s_j^{3D}, d_j\} \tag{3.10}$$

where the $IoU_{i,j}$ between 2D detection i and 3D detection j is given. Confidence scores for 2D and 3D detections are given by s_i^{2D} and s_j^{3D} with d_j being used to represent the characteristic difference between LiDAR and j_{th} 3D bounding box. Due to the geometric consistency measures introduced as part of CLOCs, detections with a zero- IoU are removed. In case the projection of a detected 3D bounding box has no intersection with the corresponding 2D detection, the 3D detection is kept for possible further use. $IoU_{k,j}$ and s_k^{2D} are set to -1 instead of 0, which allows for a consideration separate from other cases with low IoU and s^{2D}. In Fig. 3.5, an overview of the CLOCs framework may be seen. The main steps include the input of 2D and 3D detection candidates into a combined, sparse tensor which is processed using a 2D CNN. Score predictions are output for 3D object detections after maxpooling and mapping detection candidate results to a probability score map.

The fusion network that is part of CLOCs is made up of a sequence of $1\,x\,1$ 2D convolutional layers. The 2D convolution operator $Conv2D(c_{in}, c_{out}, \mathbf{k}, \mathbf{s})$ is used to describe the structure, where c_{in} and c_{out} refer to the number input and output channels, respectively. Kernel size and stride are represented by the **k** and **s**. The values contained in the four convolutional layers are Conv2D(4, 18, (1,1), 1), Conv2D(18, 36, (1,1), 1), Conv2D(36, 36, (1,1), 1) and Conv2D(36, 1, (1,1), 1) where ReLU is applied after the first three layers and resulting in a $1 \times p \times 1$ tensor with p non-empty elements. Based on the indices of the non-empty elements (i, j), a tensor T_{out} of shape $k \times n \times 1$ can be built by filling p outputs based on the indices (i, j) and putting negative infinity elsewhere. Lastly, this tensor is mapped to the desired learning targets, a probability score map of size 1 n, through maxpooling in the first dimension.

A cross-entropy loss function is used for target classification, while focal loss is employed due to the class imbalance between foreground and background. Training of the fusion network is carried out using Stochastic Gradient Descent (SGD) and an Adam optimizer with scheduled learning rate decay.

Chapter 4

Solution Approach

In this chapter, the basis for implemented software components and experiments carried out as part of this thesis is described. First, the topic of timestamp and trigger-based synchronization is introduced. The process of multi-modal sensor calibration is then described for camera-LiDAR calibration. Changes made to the 2D object detection pipeline and implementation in ROS are then listed.

4.1 The IEEE PTP-1588 Standard

In order to enable a high degree of synchronization and time-measurement accuracy across different sensors distributed across the Providentia++ infrastructure, a common timestamping format is defined. The Precision Time Protocol (PTP) as defined in the IEEE PTP-1588 standard [24] regarding precise clock synchronization, is a network protocol enabling services in a packet-based distributed network. Using PTP, a sub-microsecond accuracy can be achieved in addition to sub-nanosecond time transfer accuracy in suitably designed networks. In general, network hardware components and sensors connected to a networked system can be setup in such a way that the sensor-internal hardware clock is synchronized with a master- or grandmaster clock of the system. These masterclocks are characterized by a high degree of accuracy and oftentimes directly linked to or based on atomic time sources. Once synchronized using the PTP-1588 standard, a sensor-internal clock receives regular updates to assure that its internal clock remains within a specified tolerance and offset in comparison to the linked masterclock. In Fig. 4.1, a schematic overview of different stratum levels is given. Atomic clocks found for example in GPS satellites are attributed to stratum 0. Such a Global Navigation Satellite System, which contains a highly accurate relativistic clock, may continuously send pulsed time information to downstream receivers. Further downstream, at stratum 1, oftentimes Network Time Protocol (NTP)-timeservers can be found with a lower accuracy. End-users and local network services can be connected to e.g. stratum 1 time sources in a simple manner.

While sensors may locally measure time in seconds or nanoseconds since startup, a global or system-wide time-format is generally desired in a distributed system. In the context of NTP and PTP, the UNIX time format is utilized. The UNIX system is characterized by its starting date, January 1st, 1970 at 00:00:00 hours, which acts as a zero starting point from which point on, timestamps are counted upwards sequentially. For example, the date of December 15th, 2021 at 12:00:00 p.m. is represented by the number of 1639569600000 milliseconds.

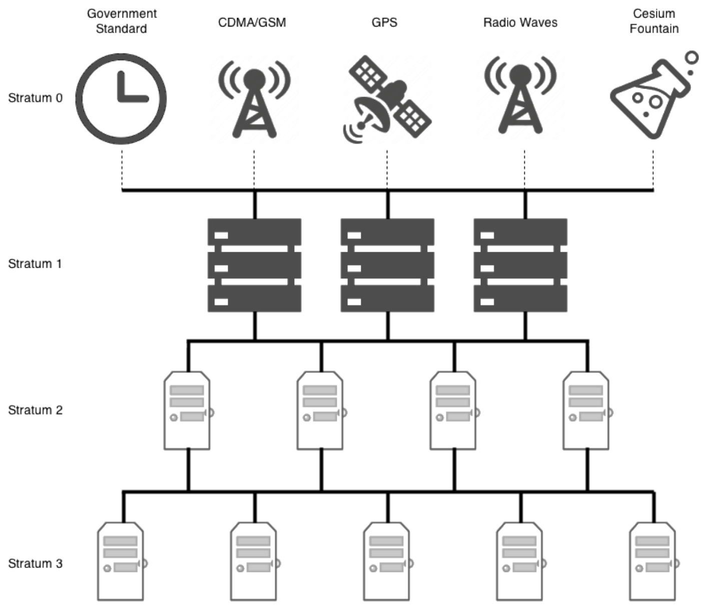

Figure 4.1: Stratum levels for precision time servers [58]

4.2 Sensor Synchronization

Several methods of timestamping exist, for example sensors may register and imprint on a data package the point in time when a data packet was recorded, fully sent or fully received. Although these inaccuracies may initially consist of (sub)-millisecond differences, inaccuracies can accumulate over a large amount of time and large number of sensors in a network. A complicated measurement and rectification of downstream timestamp-offsets can therefore be reduced by ensuring highly accurate timestamping and data processing in the first place. In Fig. 4.2, an applied overview of the PTP-timestamping service within the Providentia++ architecture is shown. A NTP client in the Providentia++ backend receives time signals from an internet-based NTP server. The NTP clock acts as a server of the local PTP grandmaster setup at the backend. Real-time clocks are adjusted based on the PTP gradnmaster using a PTP slave running of the DFU backend. As a result, a single point of reference is used for clock synchronization. Locally distributed PTP masters are then used to feed sensors equipped with PTP support In order to assure that data stemming from individual sensors is closely aligned time-wise, the timestamps of data packets received can be checked by a stream-synchronizer. However, a successful stream synchronization requires a simultaneous initial triggering of sensors.

In a dynamic context where sensors are mounted on a moving platform for data recording such as in the PixSet dataset [11], time synchronization and calibration are prerequisite for downstream annotation or detection. As the sensor platform on which data is recorded is moving, offsets in sensor data have to be compensated for the ego-vehicle motion. In order to achieve time synchronization, a signal generated by a Ouster OS1 64-layer LiDAR is used to trigger other sensors such as cameras mounted on the sensor rig. Sensors such as radar or IMU are excluded from this trigger-based synchronization due to their high inherent framerates. A high accuracy time server in the form of a GPS module is used in PixSet in order to continously output accurate timestamps to the Ouster LiDAR. Additional comparison of sensor-internal timestamps and the trigger signal is used to track and minimize potential offsets in recorded time stamps. LiDAR sensors such as the Ouster LiDAR measure data while

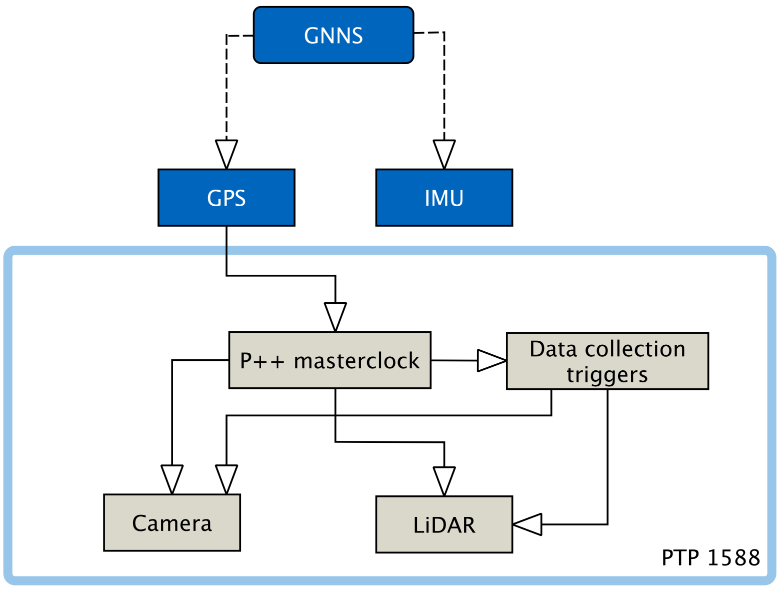

Figure 4.2: Architecture layout for PTP-1588 network standard

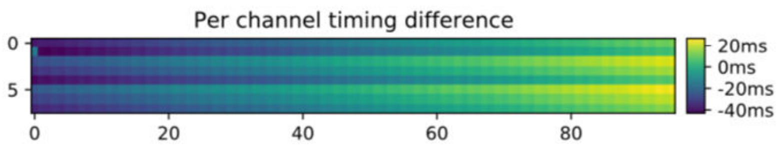

Figure 4.3: Per channel timing difference for Ouster 64-layer LiDAR [11]

LiDAR receiver and emitter complete a rotation around the vertical sensor axis. Although LiDAR data recorded in a single 360 degree sweep is attributed a single timestamp, a time-difference between the first and last scanning line of a rotation is noted. In the figure below, this offset is visualized since the same LiDAR sensor is used in the Providentia++ project for LiDAR data collection.

In the Providentia++ project, different sensor modalities such as camera, LiDAR and radar sensors are used for a variety of detection tasks. Individual sensors are manufactured by different companies and contain a variety of sensing hardware. Due to this, camera sensors may record data at up to 50 Hz, while LiDAR sensors are limited to a range of 10-20 Hz and radar sensors to approximately 13 Hz. In order to receive synchronized data frames that can be used as input for object detection pipelines, sensors need to be individually set up as part of the system. In general, no data frames should go to waste as a result of constrained bandwidth or due to not keeping up a set frame collection pace. In addition, LiDAR sensors are placed at individually different angles and in an orientation not necessarily lining up with camera sensors, with which the data should be fused for increased object detection robustness. Due to the intrinsic software and hardware parameters, in this thesis, the experimental focus of synchronization is placed on Basler type Gig-E cameras as well as the Ouster OS1 64-layer LiDAR.

The PTP-based synchronization of the internal clock of the LiDAR sensor with the NTP-timeserver is carried out automatically during launch of the sensor in approximately 10-15 seconds. For the Basler camera used here, this process may take a significantly longer amount of time. This is due to the non-monotonous optimizer inside the camera network interface: The offset between Masterclock and sensor-internal clock is carried out by the optimizer until an accuracy threshold is crossed. Here, this synchronization process of the camera is limited to 60 seconds for a threshold of 0.1 millisecond. Depending on the requirements of downstream applications, this threshold can be set higher or lower at any point before camera startup. In the figure below, a representation of this offset between master and slave time source is shown.

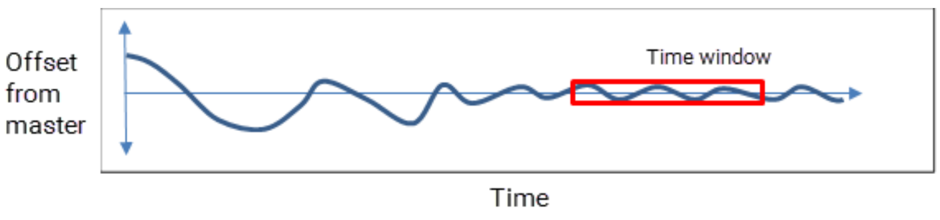

Figure 4.4: Sensor-internal offset from masterclocks

In order for the synchronization of a camera-LiDAR sensor pair, in addition to the synchronized timestamp, an initial trigger can be set up to start data recording. Here, due to the nature of the source code in camera and LiDAR, an implementation is chosen in which the LiDAR acts as a trigger for a camera sensor. When the LiDAR sweep reaches a specified position in its circular motion, a trigger signal is sent to the linked camera sensor in order to trigger the initial camera shutter. Ideally, the trigger should be activated at the point in time in which the LiDAR sweep phase angle faces the same direction as the central camera FOV. Due to the distance between camera and LiDAR sensor in the sensor-station network, the latency of image and point cloud messages has to be measured and input into the synchronization launch file in order to adjust the trigger time of the LiDAR. Thereby, a high degree of accuracy can be achieved even for multiple cameras.

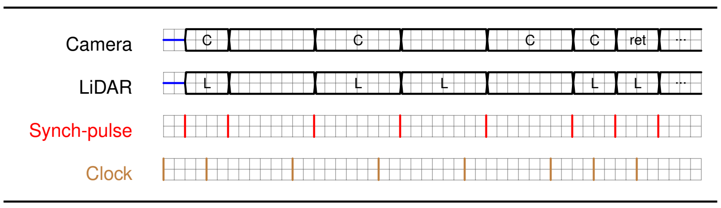

Figure 4.5: Test timing diagram for a LiDAR and camera sensor

4.3 Camera-LiDAR Calibration

In order for accurate data labeling, sensors on which data is recorded must first be calibrated. Especially for fused object detection approaches such as camera-LiDAR systems a high accuracy is important in order to fully utilize observed and extracted features. Sensor calibration can be divided into intrinsic (internal) calibration and extrinsic (external) calibration. Intrinsic sensor calibration is usually carried out initally by the sensor manufacturer. For camera sensors, an intrinsic calibration can be carried out in a simple manner, using two-dimensional reference shapes such as a checkerboard. For LiDAR-sensors, the intrinsic calibration process may not be as easy and may involve more complex and three-dimensional shapes.

For camera-LIDAR-calibration, target-based and targetless calibration methods exist. In addition, self-calibration methods based on neural networks such as LCCNet [38] have been proposed in recent years. For target-based methods, a reference shape such as a black and white checkerboard or other patterns such as ARUCO markers can be used. In this thesis, both types of calibration approach are tested. However, due to the distributed placement of camera and LiDAR sensors across several locations and in a height of at least 6 meters above ground, a targetless approach is considered to be more desirable than a target-based approach. In the following section, the targetless calibration method introduced in more detail in the following section.

In [76], an approach based on pixel-level feature extraction is used for camera-LiDAR calibration. Target-based methods are described as using a checkerboard or similar calibration pattern which is extracted in camera and LiDAR data. Following this, optimization functions such as the least-squares methods is tasked with minimizing distances between extracted features from the different sensors. The authors attribute a high level of noise to spinning LiDAR which may degrade the performance of this calibration approach. Instead, a targetless method is implemented as part of this approach. Features are extracted for indoor and outdoor scenes using edge and plane based features instead of geometric shapes. With this approach, an accuracy level comparable to target-based methods is achieved.

In PixSet [11], camera and LiDAR sensors mounted on a moving platform are calibrated using a chessboard placed in view of each of the sensors. The 3D corners of the calibration board are extracted from camera and LiDAR data using an iterative closest point method for a number of transformation matrices. Averages are calculated for the set of transformation matrices and solved using a perspective n-point method. Based on the projection of LiDAR data into a camera image, the accuracy of the calibration method used is then visualized. Unlike the data recording approach proposed in PixSet [11], the infrastructure on which sensors are mounted as part of the Providentia++ project, is stationary, with the exception of small movements due to wind forces on the highway gantry bridges. However, the comprehensive approach to sensor synchronization can be used as the basis for a similar implementation in the Providentia++ project.

The targetless camera-lidar calibration method utilized here is based on the livox calibration repository [76]. Originally, this tool is introduced for high-resolution Livox-type non-repeating LiDAR sensors. With some modifications, the data collected on spinning LiDAR sensors such as the Valeo Scala Gen2 or Ouster OS1 LiDAR can be used as input as well. In order for edge-detection and feature extraction algorithms included in this tool to perform well, the input data recorded on camera and LiDAR data for calibration should be recorded in an environment with clearly visible, straight edges. Organic clutter such as trees and bushes,

Figure 4.6: Cropped LiDAR-point cloud recorded at the s110 sensor station

as well as moving objects should be avoided. In using this targetless calibration approach, some of the main challenges are:

point cloud sparsity: point clouds may not be dense enough for accurate feature detection and optimization between camera and LiDAR feature vectors. In an outdoor context, the closest stationary objects on which features can potentially be detected are further than 15-20 meters away from the sensor. An accumulation of multiple LiDAR-sweeps and mapping to a common frame, may improve the point cloud density and contain additional objects for edge and feature detection. Some examples for common LiDAR odometry and mapping algorithms are [63, 68].

Noise and intrinsic calibration: For LiDAR sensors, an evaluation of the distortion caused by inaccurate intrinsic calibration or changes in data collection quality is difficult without a reference-sensor of known quality. Depending on sensor quality, a significant variance in data quality can be observed for different LiDAR sensors. While some sensors feature a high resolution in a large near range, they may be outperformed significantly at longer distances or specific parameters such as a larger azimuth. Since denoising or averaging significantly shift initial detection positions necessary for calibration, noisy data is especially detrimental.

Initial extrinsic calculation accuracy: In case Global Positioning System (GPS) and Internal Measurement Unit (IMU) measurements are used for the estimation of an initial 3D location and orientation, measurements may be inconsistent, e.g. in the satellite link quality, due to interference from metal structures as well as inconsistent placement of measurement equipment on the sensors that are to be calibrated.

Optimizer convergence: Optimization algorithms oftentimes allow for the specification of an initial value, in this case an initial extrinsic matrix consisting of translation and rotation values. However, due to the specific structure of the solver/algorithm, an accurate initial value may not always lead to a better convergence or globally accurate result.

In the implementation that is part of this thesis, in a first step, the point cloud density is increased for a recorded scene. The fast-lio mapping algorithm [69] is used to accumulate and store point clouds collected over an arbitrary amount of data frames. LiDAR sensors such as the Ouster LiDAR collect data within a 360-degree view. For fusion of a single camera and LiDAR sensor, this is not required due to the significantly lower FOV of cameras employed in most Providentia cameras. For example, the Basler Gig-E type camera has an approximately 70-degree FOV. The previously accumulated point cloud is therefore cropped as can be seen in Fig. **??**. Due to the reduction of the total number of points contained in a point cloud as a result of this cropping, a longer scene can be considered for calibration without exceeding the limitations of the calibration algorithm. In Fig. 4.8, the camera feature extraction is shown side by side with the initial reprojection of point cloud data into the camera view. Sensor positions are determined with the help of a GPS-RTK device with centimeter-accuracy for fixed-point measurements and a 9-DOF IMU.

Camera and LiDAR sensors inherently use different coordinate-systems. The Ouster-LiDAR used for calibration measurements here is oriented in a right-hand-side coordinate system coordinate system, with the x axis facing in the road direction, the y axis oriented

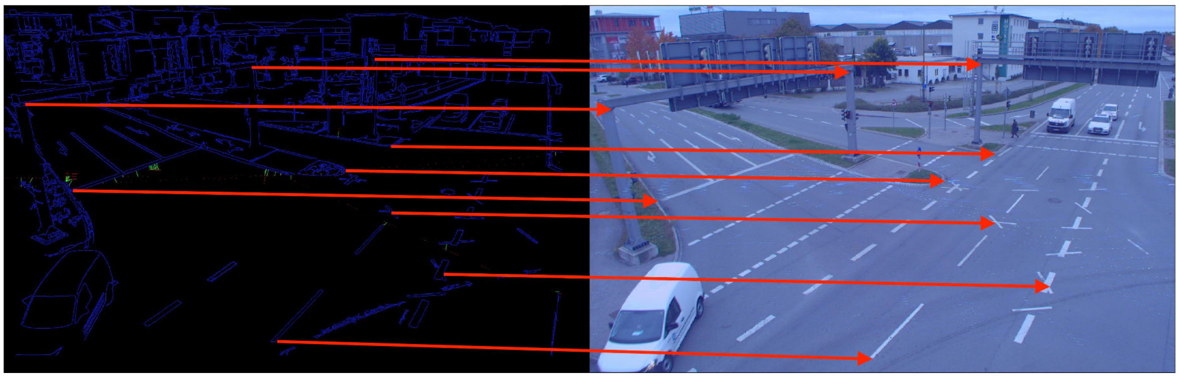
Figure 4.7: Camera image feature extraction for 8 mm Basler Gig-E camera on s110 sensor station

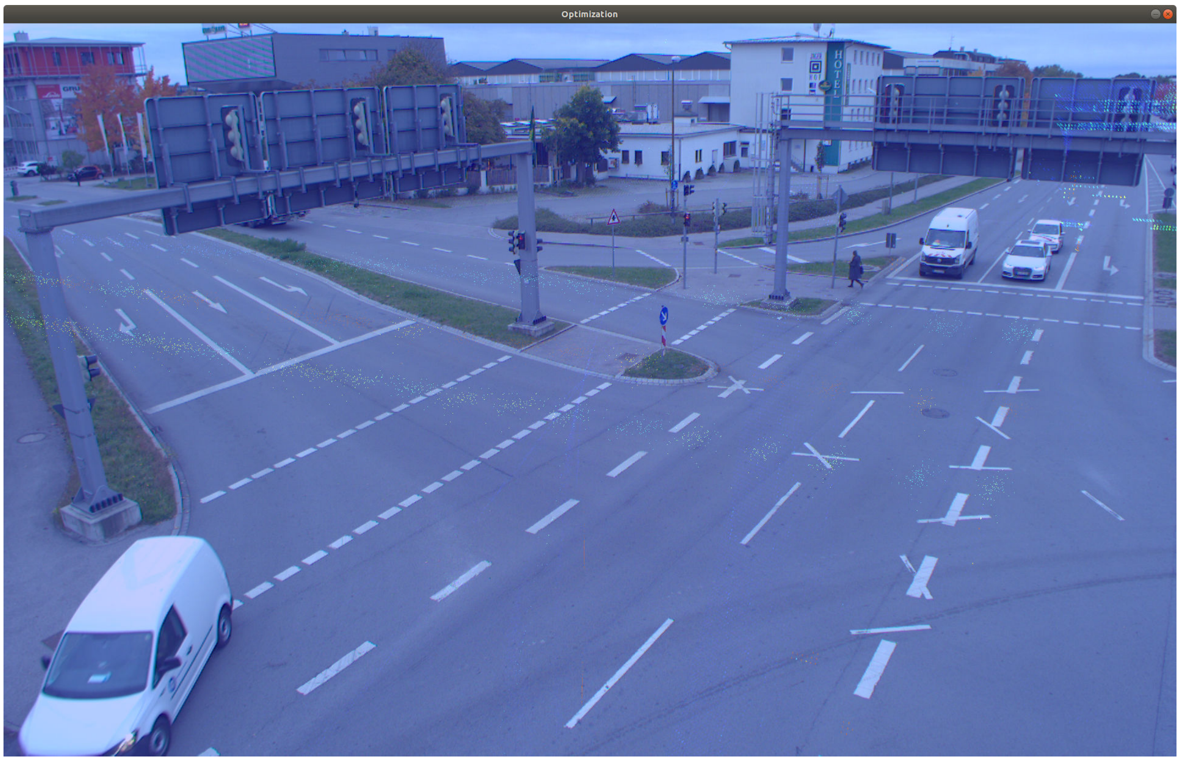

Figure 4.8: Initial LiDAR data projection during targetless calibration for s110 sensor station

laterally to the road direction and z axis directed in the upwards direction normal to the x-y road plane. In the camera sensor, axes are oriented differently. The x direction is oriented in a negative x direction, the y axis is oriented in the negative z axis direction and the z axis is oriented in the negative y direction. Therefore, a transformation has to be carried out in order for an accurate projection of camera data into LiDAR data and vise-versa. In Fig. 4.9, the camera and LiDAR sensor-based coordinate systems are shown.

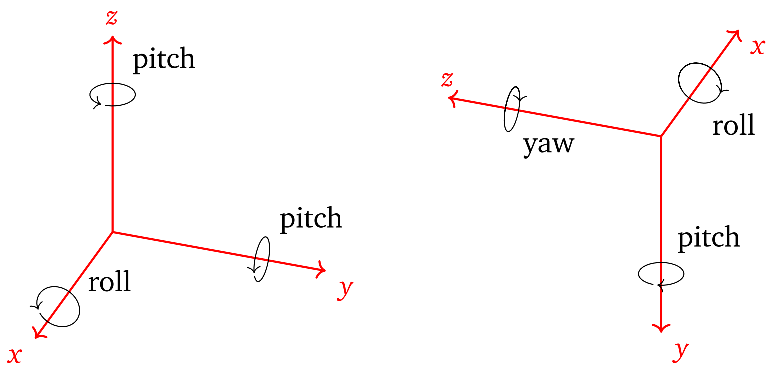

Figure 4.9: Coordinate systems of camera sensor with image plane x-y (right) into LiDAR sensor with road-plane x-y (left)

The extrinsic matrix E is shown to consist of a 3x3 rotation matrix R in the upper left and a 3x1 translation vector t consisting of values for each of the three cartesian axes. The 3x4 projection matrix P is calculated as the product of the 3x3 intrinsic matrix I and 4x4 extrinsic matrix E where the matrix X is reshaped before multiplication to a 3x4 matrix:

$$E = \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & 0 & t_1 \\ 0 & 1 & 0 & t_2 \\ 0 & 0 & 1 & t_3 \end{pmatrix} = \begin{pmatrix} 1400 & 0 & 967 \\ 0 & 1403 & 581 \\ 0 & 0 & 1.0 \end{pmatrix} \times \begin{pmatrix} 1 & 0 & 0 & t_1 \\ 0 & 1 & 0 & t_2 \\ 0 & 0 & 1 & t_3 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

The proejction matrix including the transformation from the camera coordinate system into the LiDAR coordinate system R_{init} can therefore be calculated by:

$$P = R_z * R_y * R_x * R_{init} \tag{4.1}$$

$$R_{init} = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & -1 & 0 \end{pmatrix}$$

4.4 Data Annotation

As part of the Providentia++ project, a comprehensive autonomous driving dataset is being created including labelled camera and LiDAR data recorded at a different the Providentia++

test stretch. In this thesis, some camera and LiDAR-frames are labelled for the Provid21 dataset. For annotation, the proAnno labeling tool is used, which is based on the 3D Bounding Box Annotation Tool 3D-BAT [81]. In Fig. 4.10 below, a BEV perspective of a annotated LiDAR-point cloud is shown as it is being labelled in the proAnno tool. Object direction of travel is marked by the darker shade of the corresponding bounding boxes. Based on the previously calculated projection matrix for the s110 sensor station, data frames can be annotated in the future. In general, an annotation bounding box is first placed and aligned with an object in the camera image. Next to values such as 3D size and and position, the 3D object orientation such as pitch, roll and yaw can then be adjusted in the LiDAR coordinate system.

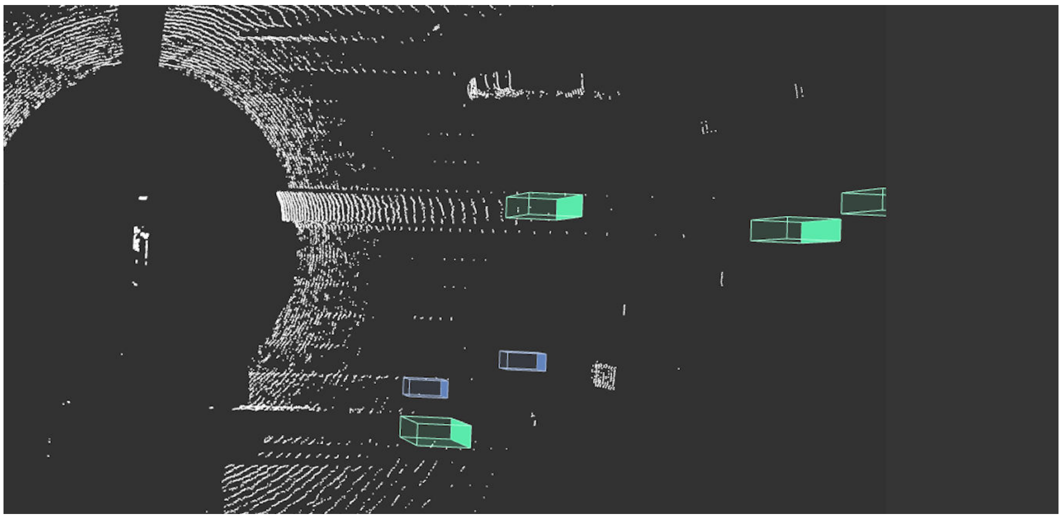

Figure 4.10: LiDAR point cloud data labelled with 3D bounding boxes in proAnno tool

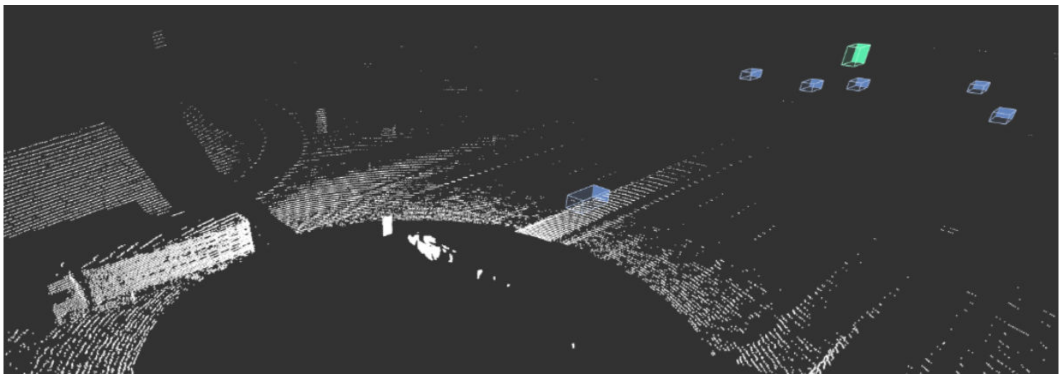

Figure 4.11: Truck in LiDAR point cloud data labelled in proAnno tool

4.5 2D Object Detection

After a comprehensive literature review of state of the art 2D object detectors, as well as initial testing of other architectures such as YOLOX [14], the YOLOR [61] architecture is chosen to be implemented as part of this thesis. In order for the 2D object detector to be used on the live Providentia++ infrastructure, the fused 3D object detection archtitecture is implemented in ROS. In the image below, the YOLOR-ROS pipeline in which YOLOR is

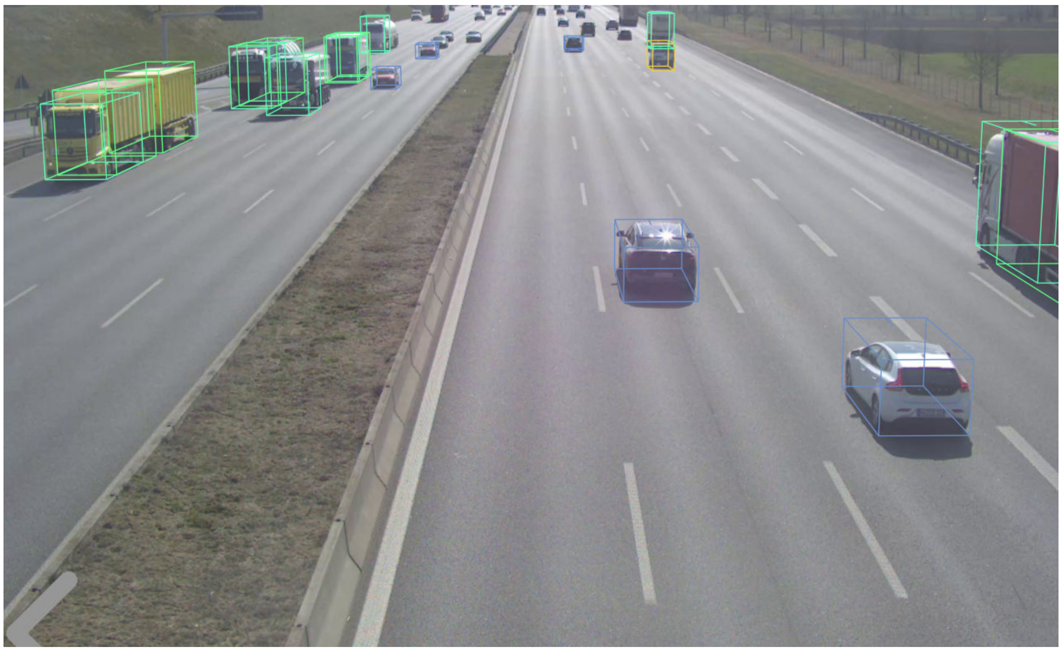

Figure 4.12: Camera image annotation with 3D bounding boxes in proAnno tool

implemented is shown. Input data is read from a rosbag in the form of raw 2D RGB-images and published to the *image_raw* topic. Inside the YOLOR-ROS node, object detection is performed on received images. Two topics are published by the yolor-ros node, the first being a list of 2D bounding boxes (detections) and the second being the *image_raw* topic. These two topics may then be used as input for the CLOCs fused 3D object detection model. As a basis, Darknet ROS-messages are selected, which can be modified to include additional output values. Bounding box messages include x,y-position, image width and height, id, center distance object class and detection probability.

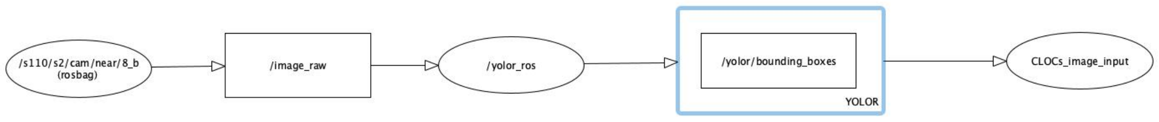

Figure 4.13: YOLOR-ROS implementation with camera-image input and bounding-box + labelled image output

4.6 3D Object Detection

In the existing CenterPoint architecture with a PointPillars backbone for 3D detection, adjustments are made in order to train and evaluate the CLOCs fusion network. Configuration files are specified in order to use labelled camera and LiDAR data collected on the Providentia++ infrastructure. In order to integrate 2D YOLOR-ROS in the CLOCs fusion network, some changes to the inference pipeline are made, which was previously based on YOLOv4. For example, in YOLOR, no anchors are defined. In order for object detection to be performed using CLOCs, camera images and LiDAR point clouds have to be synchronized to a good degree. Although the YOLOR-ROS inference is generally much faster than inference of Centerpoint, input images have to be provided in time. In case object detection probability is lower than a threshold of 0.3, the candidate bounding box is disregarded. Due to the structure of CLOCs, each class was previously implemented in an individual detection head, in other words, the fusion network is run for each class. Other than KITTI or Waymo, where a limited number of object classes such as pedestrians, cars and trucks is annotated, data collected for Provid21 includes up to 9 classes, including special vehicles such as ambulances or police vehicles. For Providentia++ sensor stations located near the A9 highway, it may make sense to exclude some object classes such as bicycles from optimization during training. On sensor stations located near the Garching industrial area however, a significantly larger number of pedestrians and cyclists is present, making the need to include these object classes in model training and weight optimization clear.

Chapter 5

Experimental Details

In this chapter, the experimental results of implemented 2D and 3D object detectors as well as necessary preprocessing steps are described. In addition, previously introduced synchronisation and calibration methods are listed with an implementation focus on the s110 sensor station. All neural networks are trained using a Nvidia GeForce RTX 3090 GPU with 24GB of RAM as well as an AMD EPYC 7282 16-Core CPU.

5.1 Experiments

First, the two-dimensional object detectors YOLOX and YOLOR are retrained an evaluated on Providentia camera image data. Then, the implemented YOLOR-ROS pipeline is tested for inference of RGB input images from a rosbag file. Next, the 3D object detector Centerpoint is trained on pointcloud data collected on the Providentia infrastructure. In a final step, the multi-modal fused inference for 3D object detection is carried out using the YOLOR-CenterPoint-CLOCs architecture. The inference pipeline is then deployed to the s110 sensor station for an exemplary test of inference on the live system.

5.1.1 YOLOX

Training using YOLOX is initially carried out for 300 epochs. Pre-trained weights from MS COCO are selected and used for each YOLOX experiment. Pre-trained weights trained on MS COCO are used for each YOLOX experiment. Then, the two YOLOX-variants are retrained on the current Providentia camera dataset including approximately 800 images as of November 2021. Since the scenes that are part of this dataset are recorded on the A9 highway close to Garching, mainly vehicle traffic and therefore objects such as cars, trucks and vans may be seen. In MS COCO, 80 object classes are initially present. In the proFusion dataset, the number of classes is significantly lower and consists of the car, truck, van, trailer, pedestrian, motorcycle, bicycle and special vehicle classes. Annotations that are part of the proFusion camera dataset are converted to the COCO format. A 60:20:20 split is used for the ratio of training, validation and test data on this data set. Training and test are carried out using –fp16. While a batchsize of four is used during training, inference batchsize is equal to one.

YOLOX variants are named according to the model size or number of parameters ranging from YOLOX-s to YOLOX-x. Lightweight models with a significantly smaller amount of parameters are also available. While these models may be large enough to learn a good representation of a small dataset such as the one used here, these models are rather used in mobile applications. Therefore, several models ranging from YOLOX-tiny to YOLOX-x are trained and evaluated in order to compare inference speed and accuracy to YOLOR. Main configuration parameters for different YOLOX variants are the model depth and width, which range from a depth of 0.33 and width of 0.5 for YOLOX-s to a depth of 1.33 and width of 1.25 for YOLOX-x. In YOLOX, an anchor-free approach is used. In YOLOX, support for acceleration using TensorRT is available. The input size is 640x640 pixels, which is significantly smaller than for YOLOR. As part of YOLOX, several types of data augmentation are available. During preprocessing for training, mixup and mosaic augmentation are carried out. In addition to classic data augmentation methods such as rotation or translation, multi-scale training is used whereby a range for scaling input images to a larger and smaller size is carried out. Alternatively, multi-scale training can also be set to randomly generate a scale range. For a further speed-up, RAM-caching is used during training of YOLOX.

5.1.2 YOLOR

Similar to YOLOX, training is carried out over 300 epochs for YOLOR using weights pre-trained on MS COCO. In YOLOR, image input size is initially set to 1280 pixels. Input images large than this size are scaled down in accordance to the ratio of their input dimensions. Due to the filter sizes used in the default YOLOR configuration, input sizes can be multiples of 64. While inference is carried out at a batchsize by default, training in YOLOR is carried out for a larger batchsize such as a batchsize of 32, which is not possible within the scope of this thesis due to memory constraints. In order for YOLOR to be applied to the proFusion dataset, annotation bounding boxes and labels have to be transformed to the YOLO annotation. Here only the object id and bounding box height and width as well as x and y position are stored in a list format. YOLOR model types are named in a scheme of YOLOR-xn, where x represents a letter such as D,E,W,P and n represents a number 5 through 7. Three different YOLOR models are trained and tested within this thesis, of which the highest reported inference for a batchsize of 1 in the original paper is reported to be 76 FPS, and the slowest 34 FPS. Due to the real-time constraint of approximately 30 FPS that is present in this project, any YOLOR model will fulfill this constraint for default settings. In a direct comparison, the YOLOR-D6 model has been shown to outperform the YOLOv4-CSP-P7 model in terms of accuracy.

5.1.3 Centerpoint

CenterPoint is trained on 350 labelled pointclouds collected on the highway portion of the Providentia test stretch. A PointPillars backbone with 64x64 filters is chosen alongside a number of four input features. As part of the architecture, two Pillar Feature Layers are selected. Voxel sizes are configured to a size of (0.2, 0.2, 8) while the pointcloud range for PointPillars is limited to approximately 51 meters in x and y direction as well as 3 meters vertically in order to focus on the area below the horizon of the LiDAR sensor due to its placement on an elevated position. Within the main part of the network, sequential convolutions are applied each followed by Batch Norm and ReLU as part of the Region Proposal Network. For downsampling in the RPN, the number of filters is set to multiples of 64 up to 256 with a stride of 2. For upsampling, a constant filter size of 128 with strides ranging from 0.5 to 2. For training, three classes are selected for detection: car, van, and trailer. Depending on the configuration setup as part of CLOCs, seperate detection heads are used for each class. During initial training, the number of LiDAR sweeps is set to 1 in order to determine a baseline. For later testing, the number of sweeps is increased in order to evaluate the effect of higher

Model	AP_val	AP_test conf=0.001	AP_test conf=0.25	latency	param
YOLOx-tiny	37.98	**21.2**	14.4	**11.87 ms**	5.06 M
YOLOx-s	**38.55**	19.6	**16.6**	13.91 ms	9.0 M
YOLOx-m	28.93	16.1	14.6	18.38 ms	25.3 M
YOLOx-x	32.98	17.7	15.8	27.77 ms	99.1 M

Table 5.1: Training and inference results of YOLOX variants re-trained on proFusion camera dataset

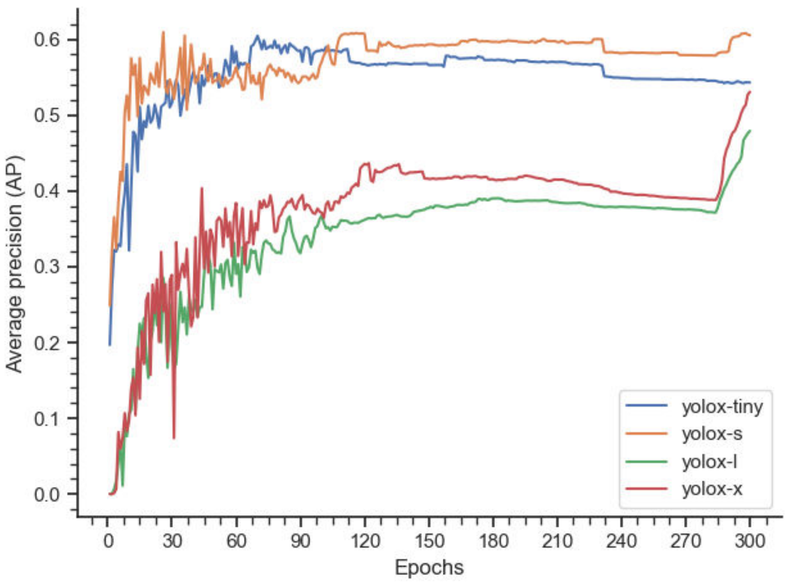

Figure 5.1

Figure 5.2: Average precision results (AP-50) of YOLOX variants re-trained on proFusion camera dataset

density pointclouds on performance and accuracy. For detections, a maximum number of candidate bounding boxes and voxels are defined in order to limit computational cost. For testing, a voxel size of 0.2x0.2 is set due to the best-practice tradeoff between accuracy and performance. For each class, ground-truth augmentation is carried out using a number of sample groups specified before training. In addition, input data is augmented before training using noisy rotations and translations. For 2D object detection, two state of the art neural networks are implemented as part of this thesis. The two models are YOLOX and YOLOR [source]). Each of these two models is considered to be among the fastest and most accurate 2D detectors. Both models are pre-trained on the MS COCO [source] dataset and then re-trained on the Providentia camera dataset collected on the A9 highway north of Munich. Variations of different model sizes are trained under similar conditions over the course of 300 epochs and with a 60:20:20 ratio (train:val:test). Models are then evaluated (inference) and compared.

In the table below, results of four different yolox variants are summarized.

Training and test (inference) are carried out using –fp16. Training batchsize=4, inference batchsize=1. **YOLOR**

In Tab. **??**, the average precision values for three different YOLOR-training sessions are shown. Average precision is evaluated once for a test-confidence of 0.01 as well as 0.25. Here, it can be seen that with an increasing number of model parameters, the latency strongly increases. Due to the input size of 1200 pixels, input images of size 1920x1200 are scaled down

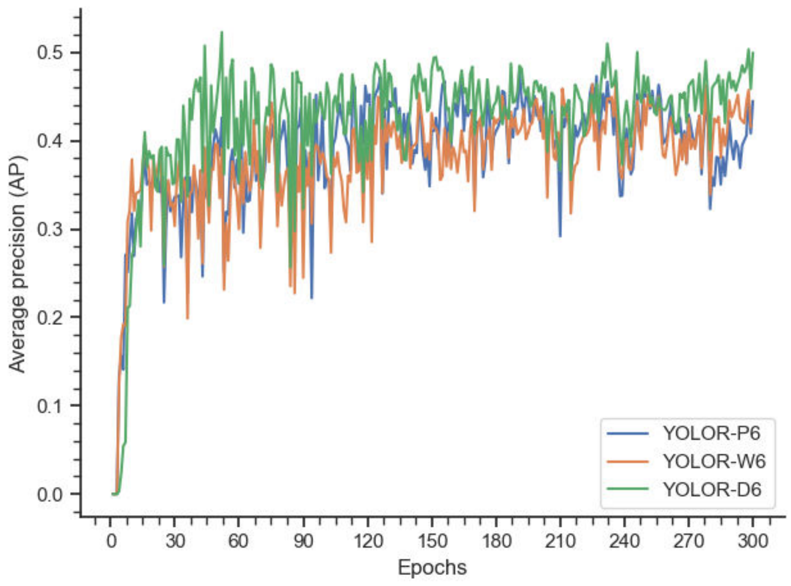

Figure 5.3

Figure 5.4: Average validation precision results (AP-50) of YOLOR variants re-trained on proFusion camera dataset

in order to fully make use of the set of optimized hyperparameters and learning schedules published in YOLOR. After training each of the models on 300 epochs, it can be seen that some loss functions such as the classification and box loss are not fully converged. Therefore, an extended training of different YOLOR models is carried out for a number of 450 epochs. In Fig. 5.6, some more detailed metrics are displayed for training (upper row) and validation (lower row).

5.2 Results

For the YOLOR-P6 model, 81.56 Gflops are allocated, while for the YOLOR-W6 model variant, 113.47 Gflops are allocated. Due to the different model structure of the YOLOR-D6 variant, no such information is present. For testing inference speed of YOLOR, due to square image input and a filter size of 64, image sizes with equal side lengths and dimensions as a multiple of 64 have to be provided. Due to the image size of 1920x1200 pixels inherent to camera images that are recorded on cameras for the proFusion dataset, the largest possible square ratio is 1152x1152 pixels. As a result, the YOLOR-P6 model tested on batchsize 1 for a test image size of 1152x1152 is evaluated to take 25.4 ms for inference, 1.5 ms for NMS and 28.4 ms in total. The YOLOR-W6 model is evaluated in the same manner for a result of 25.1 ms for inference, 1.1 ms for NMS and 26.2 ms in total. Due to the different image sizes used for training as well as testing for YOLOR and YOLOX, a direct comparison of inference speed is not straightforward. Instead an additional evaluation of inference speed for the ROS-implementation of the 2D detectors may be more conclusive.

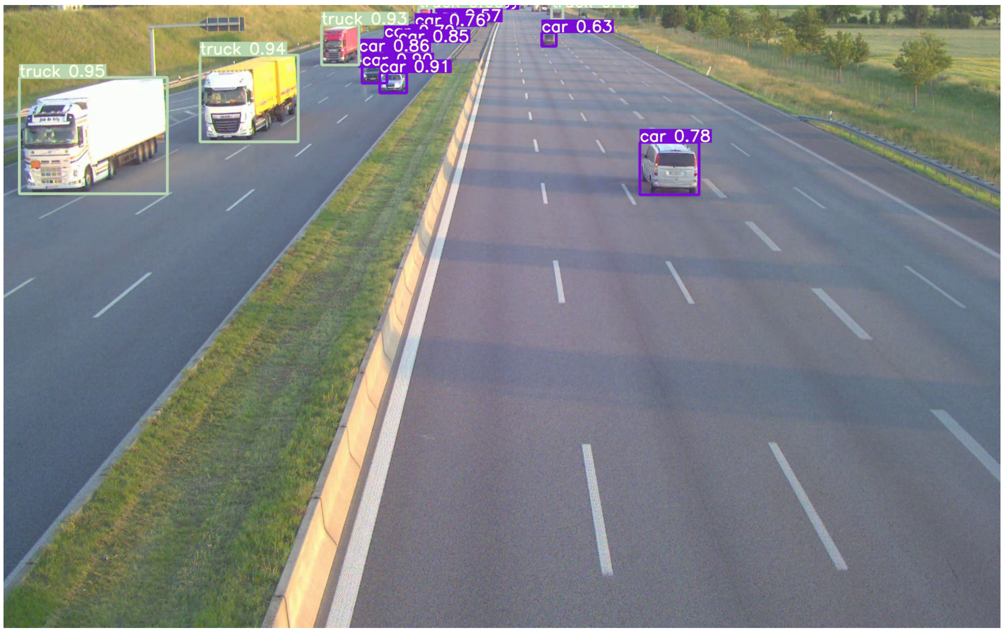

Figure 5.5: YOLOR-D6 inference output at the s50 sensor station for conf=0.25, IoU-threshold=0.5 using weights pre-trained on MS COCO

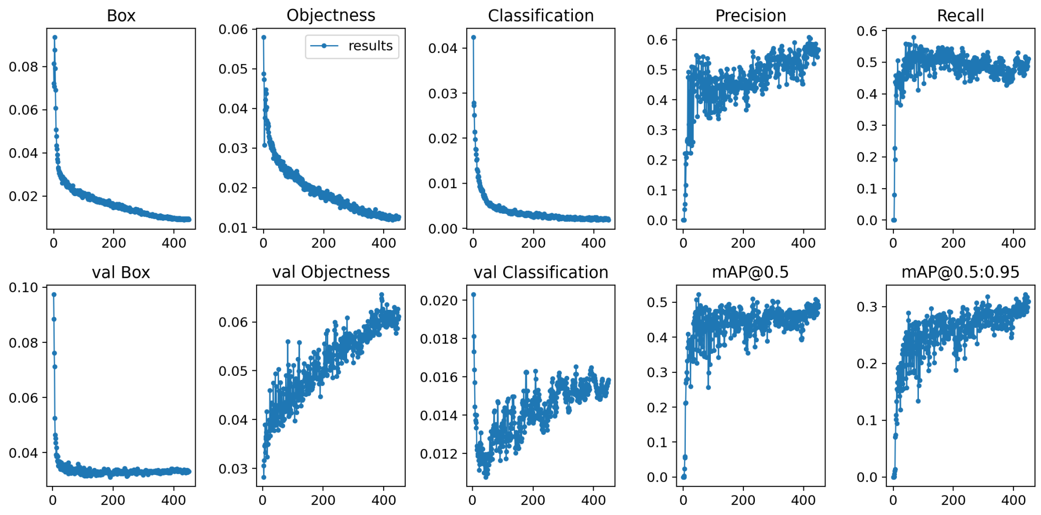

Figure 5.6: YOLOR-D64 detailed training metrics over 450 epochs on proFusion image dataset

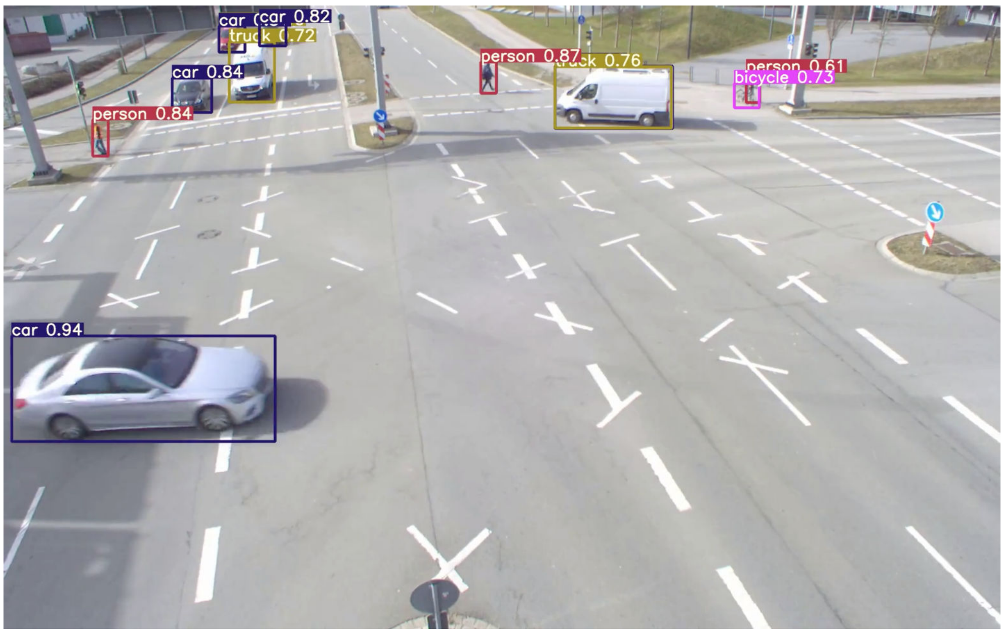

Figure 5.7: YOLOR-D6 inference output at the s110 sensor station for conf=0.25, IoU-threshold=0.5 using weights pre-trained on MS COCO

Chapter 6

Evaluation

Both the YOLOX and YOLOR networks previously are re-trained on the Providentia camera dataset. The proFusion dataset is currently made up of approximately 800 images in a 1920x1200 format. Across these images, 14,502 annotations in the form of 2D bounding boxes are included, resulting in an average of more than 17 annotations per image across 9 classes. In the image below, a distribution of the annotations belonging to individual classes may be found. The large class imbalance between cars and other object classes can be seen. While a significant number of cars is included in the current dataset, other object classes are underrepresented. While this large ratio of cars to other object classes is to be expected in a highway setting without pedestrian or urban traffic such as bicycles, the data imbalances in the current dataset needs to be considered in regard to effects on the training of object detectors. Emergency vehicles such as fire and police trucks are included in the special vehicle category but are also underrepresented.

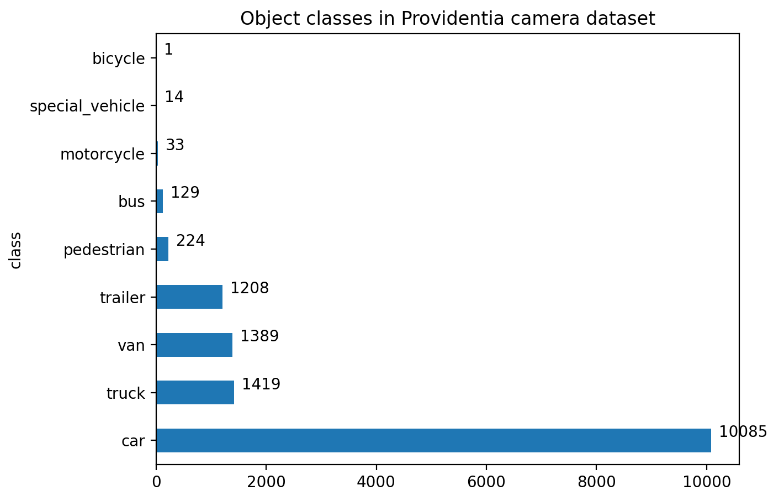

Figure 6.1: Object classes in Provid21 dataset as of October 2021

In the second image below, the distribution of object distance from the camera sensor is shown in x and y direction. Here it can be seen, that some objects are are annotated as far away as 400 meters from the camera sensor, while in a lateral direction, objects stay within the 20 meters of the road surface to either size. As the camera image dataset is extended with more diverse scenes it may be useful to compare the composition of the dataset once again.

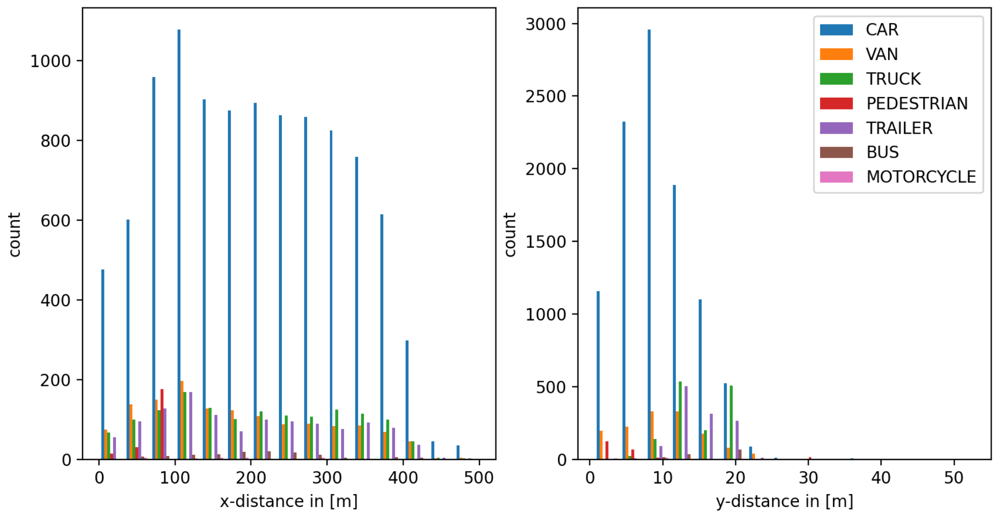

Figure 6.2: Distribution of object distance from sensor in Provid21 dataset as of October 2021 (where y represents lateral distance)

Tot. num. of anns	anns/frame	Sunny	Cloudy	Frames
14 502	17.8	470	338	808

Table 6.1: Annotation and scene descriptions proFusion dataset

param	x-dimension	y-dimension	z-dimension
Mean	4.3m	1.9m	2.4m

Table 6.2: Averaged object sizes in the Provid21 dataset

While the inference speed of YOLOR is not significantly higher than that of the previously implemented YOLOv4 model, the higher accuracy in benchmarks such as KITTI is the reason why this state-of-the-art approach is of interest as part of the CLOCs network.

6.1 Calibration

Target-based calibration point clouds and images facing a black and white checkerboard are recorded in an indoor setting. The Valeo Scala Gen2 LiDAR is primarily used for this purpose. A number of ROS-based calibration tools are evaluated with similar results. Due to noise contained in the input LiDAR point clouds, an accurate calibration is not directly possible. When applying denoising or averaging filters as seen below, observed error levels are significantly reduced. Output accuracy is shown to be below 0.01cm as well as a rotational error of less than five degrees. This is visualized in below in addition to the the error in pixels.

Here, the calibration target is placed at a distance of three to four meters to the sensors. In Fig. 6.5, these poses are visualized. On the Providentia++ infrastructure, sensors are placed in an outdoor environment, in which object detections may be located at distances much farther away from the sensor. As a result, even though a low error estimate is achieved in the indoor target-based test, calibration result quality may not scale to the outdoor context. Due to this, a targetless calibration approach is pursued.

In figure 6.3, an output image of detected checkerboard point extracted from the LiDAR data and mapped to the camera image is shown.

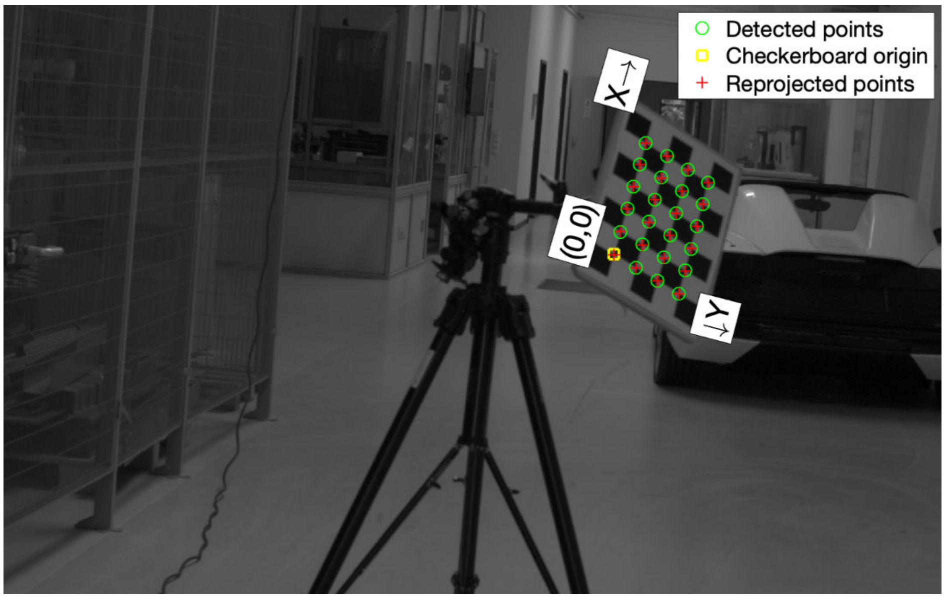

Figure 6.3: Mapping of extracted checkerboard corner points in camera data

For targetless calibration, pointclouds mapped to the LiDAR sensor frame are accumulated over several minutes in order to achieve a dense pointcloud representation. The accumulated pointclouds are cropped in order to reduce the search area for the feature and edge extraction algorithms in the livox camera calibration tool [76]. An initial extrinsic matrix is calculated from GPS and IMU measurements of the involved camera and LiDAR sensor. In the calibration process, features are extracted from image and pointcloud data based on a pixel-level.

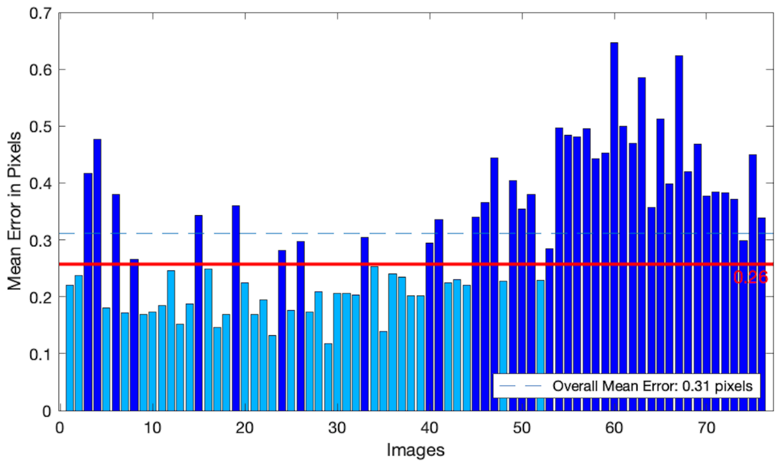

Figure 6.4: Calibration error for target-based camera-LiDAR calibration

The first image of the input rosbag file is extracted and kept as a static image on which a optimization of distances between estimated feature locations and re-projected LiDAR points is shown.

For comparison of results to the targetless calibration method, a manual calculation of the extrinsic and rotation matrices is performed in order to receive a projection matrix for the camera-LiDAR sensor pair. Due to the different coordinate systems in which camera and LiDAR sensor record data, a coordinate transformation is first carried out, after which the relevant rotations are applied. In the figure below, the resulting projection can be seen and has achieved a satisfying result from a visual perspective.

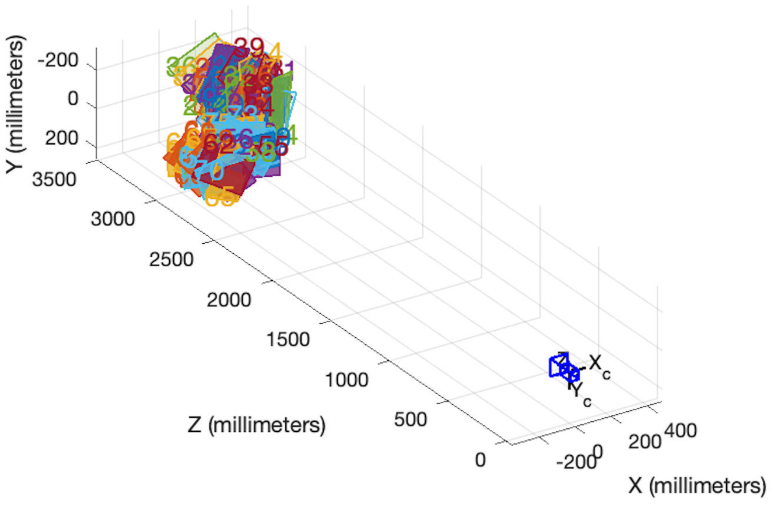

Figure 6.5: Calibration board poses for target-based camera-LiDAR calibration

In the image below, an image of LiDAR reprojection during the calibration process can be seen for targetless calibration.

Based on the calculated extrinsic matrix, the projection matrix for camera sensors is calculated. In the image below, the projection of a 3D bounding box can be seen in an image during annotation process using the proAnno labelling tool. The corresponding LiDAR bounding box is displayed to the right of the camera image in a top-down perspective.

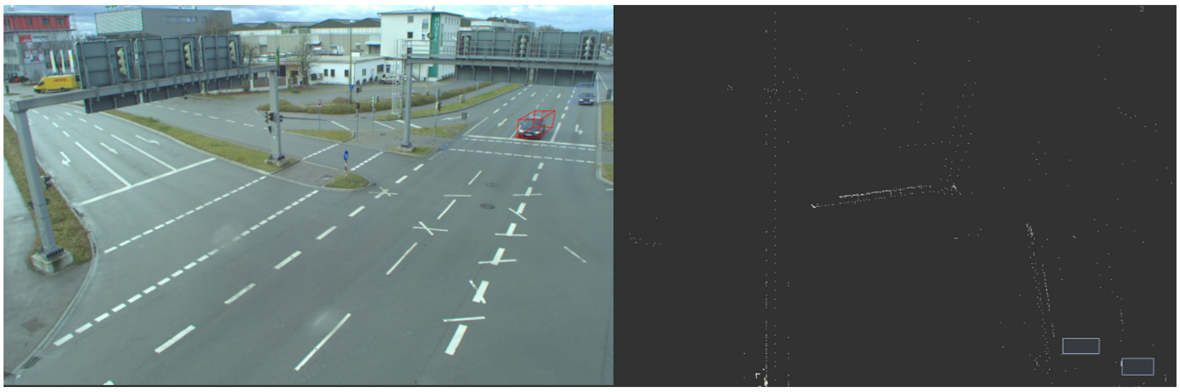

Figure 6.6: Annotation of LiDAR and camera data at the s110 sensor station using proAnno

6.2 2D Object Detection

As introduced in the previous section, some classes such as pedestrians or bicycles annotated as part of the Provid21 dataset are underrepresented. During training of the YOLOX and YOLOR models described in the section on experimental details before, all nine available classes were used for training. In case instances of these underrepresented classes are displayed in image input during testing, the likelihood of a false detection are much larger than for one of the well represented classes such as cars or trucks. Due to this, the overall accuracy of the validation or testing mean Average Precision is expected to be lower than for a training run containing e.g. only the car and truck object classes. In the graphs included in the previous section, it can be seen that a mean Average Precision of approximately 45% is achieved on the Provid21 validation set consisting of 161 images. Although this accuracy is lower than the AP values published for the same YOLOX and YOLOR models trained on MS COCO data, it is a reasonable result. The detection accuracy displayed by YOLOR model variants is higher than that of the YOLOX variants by approximately 4%. Due to this, the choice of the 2D object detector implemented in ROS for live inference is clear.

In the images below, some output images of the YOLOR-ROS pipeline are shown. Input images are recorded by a camera located on the s110 sensor station. Here, within the camera FOV, an intersection including pedestrian and bicycle paths is shown. During inference, it can be seen that object classes such as cars directly facing the camera are detected with a high confidence of more than 96%. Some objects such as pedestrians crossing the street are detected in some frames but not continuously. In addition, a large trailer-truck passing from right to left in the image view is not detected. In order to interpret these missed calibrations, the training dataset and learned features are once again called up. In the Provid21 image dataset used for training, objects are also recorded and annotated from a similar height-wise perspective. However, due to the orientation of the road, objects are largely seen to be travelling into the direction of the camera itsself. As a result, few objects if any are ever labelled or seen from a side-view. Since the models have learned front-view features of objects such as cars and trucks, it is that evident that objects seen from the side in the s110 inference test are not detected as easily. Furthermore, the pillars and signs of one of several gantry bridges partially occlude the view of the top right and left corners in this camera view. In regard to the missed detection of pedestrians in some frames, the low number of labelled pedestrians in the training input data set is referred to once again. Other mitigation methods while continuing to use the Provid21 image dataset include a larger data augmentation of certain ground truth annotations. For example by adding further augmentation such as rotation, translation and scaling of the truck or pedestrian bounding boxes, the different orientations of objects moving in a non-linear path in view of the s110 camera detection can be made more robust.

Due to the large input size of 1920x1200 pixels, the YOLOR models trained on Provid21 data are shown to produce a lower inference speed than that of the original published methods for an input and test size of 1280x1280 pixels. While the highest accuracy of the evaluated YOLOR models, is achieved by the D6 variant, the inference framerate of approximately 14 Hz does not fulfill the real-time requirement of the overall detection pipeline. Due to this, one of the other YOLOR models such as the P6 variant may be chosen. As a result of the lower number of parameters included in this variant, the inference speed of 30 FPS can be guaranteed. In both the YOLOX and YOLOR models, a high accuracy of the vehicle and truck classes can be observed during inference. Within the CLOCs fusion network, 2D and 3D detection candidates are fused. Based on this, an increase detection accuracy likely has a direct influence on fused detection results.

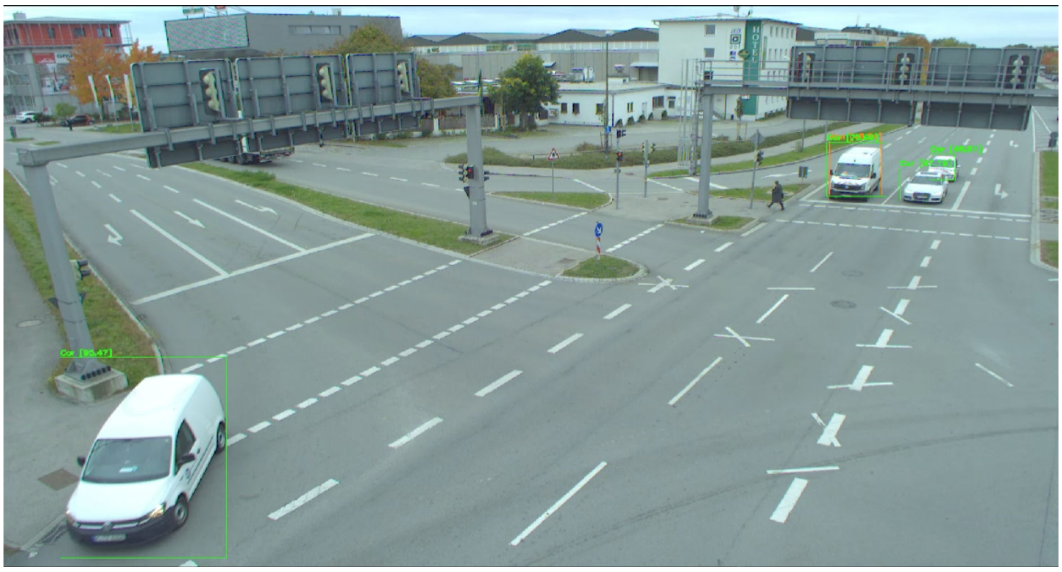

Figure 6.7: Missed detection of a pedestrian during YOLOR inference output for s110 camera

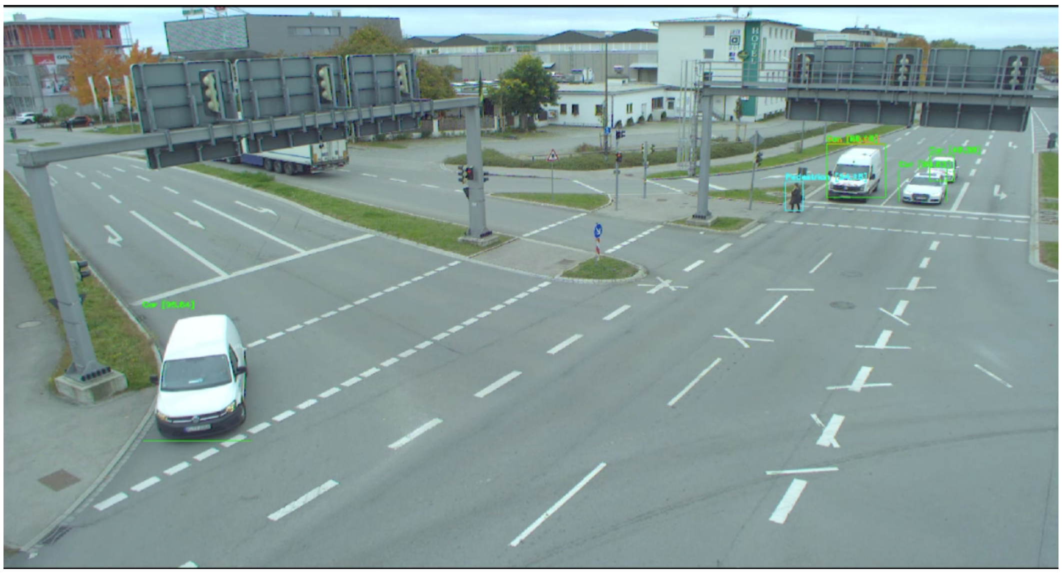

Figure 6.8: Accurate detection of a pedestrian during YOLOR inference output for s110 camera

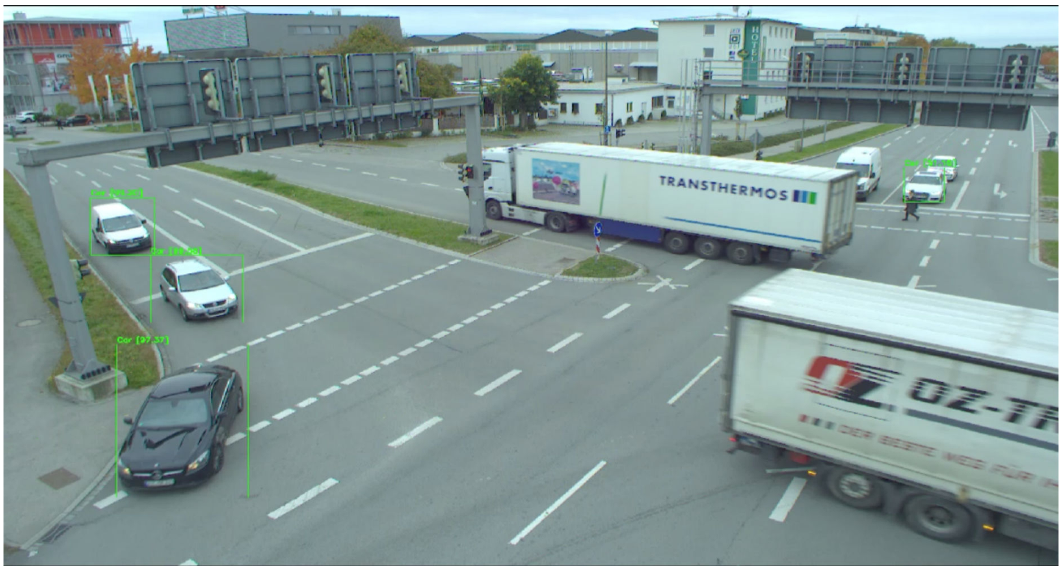

Figure 6.9: Missed detection of a truck detection during YOLOR inference output for s110 camera

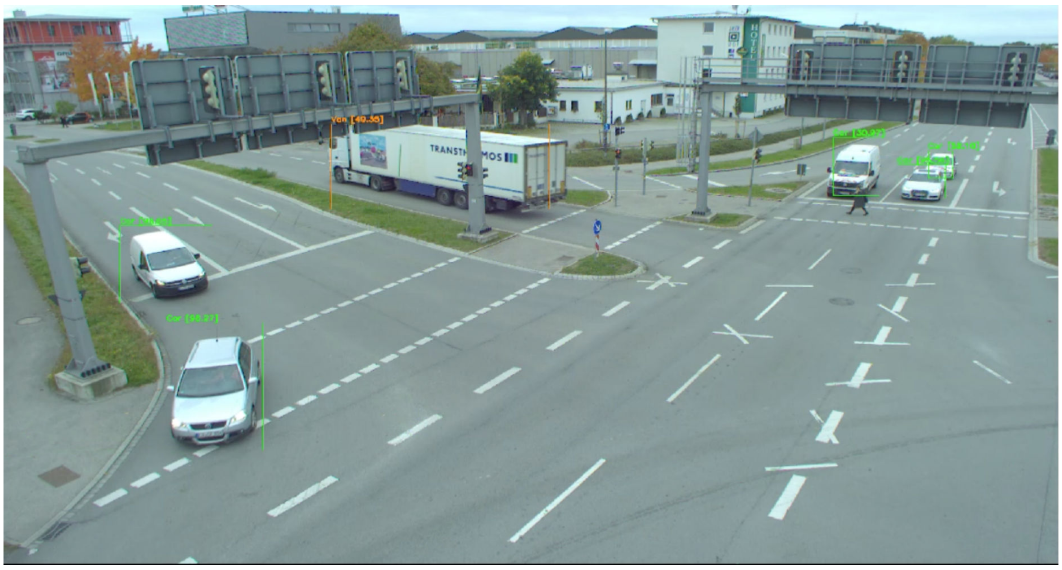

Figure 6.10: Accurate truck detection during YOLOR inference output for s110 camera

Chapter 7

Outlook

As part of the work carried out for this thesis, some approaches were considered but not employed for final implementation due to e.g. unavailable code. However, some of these methods may be of interest in future development of the 3D object detection architecture in Providentia++. In the following paragraphs, some of these promising approaches are highlighted.

- FusionPainting: Introduced by Xu et al. [67], FusionPainting has been published as having achieved a higher accuracy on benchmark datasets than CenterPoint. Due to its architecture being built upon a PointPillars backbone as is the case in this project, FusionPainting may be of interest for the replacement of components within CenterPoint as the 3D object detector.

- Recently, a new high score for CLOCs of 82.28 % mAP for the moderate difficulty in 3D detection performance was achieved on KITTI using CT3D [16, 43] and Cascade-RCNN for CLOCs. Thereby, a new best performance in the KITTI 3D detection leaderboard is achieved. A replacement of the existing Centerpoint (PointPillars) backbone may therefore be of interest in order to achieve higher AP values across all classes.

- Multimodal Virtual Point 3D Detection: In LiDAR data collected on the new P++ sensor stations, smaller object instances such as the pedestrian and bicycle class are more frequently seen. Due to the sparsity in LiDAR points for small or distant objects, the approach suggested in MVP 3D OD [73] may provide a valuable tool in order to deal with this through introduction of virtual LiDAR points and resulting in more dense PCs. One of the advantages of this approach would consist in the fact that the MVP method is developed on top of a CenterPoint or VoxelNet structure as it is already implemented in the P++ infrastructure and can mainly be used for training (so no overhead for inference)

- Fast-Lio2 As introduced in chapter 4, the fast-lio algorithm is used to increase the density of LiDAR point clouds for improved calibration results. An updated version of the algorithm with increased speed (mapping) has been published by the authors. Employing this new version may be of interest e.g. in accumulating multiple sweeps of LiDAR data before publishing on the Providentia infrastructure with low overhead. In addition, the livox-camera calibration package may receive an update to include spinning LiDAR compatibility, removing the need to use fast-lio or similar mapping algorithms [68].

Using the 2D YOLOR object detector implemented in ROS, a first step towards a performance increase for the fused camera-LiDAR based CLOCs detection approach is set up. This is mainly due to the higher accuracy achieved by YOLOR models in comparison to various YOLOv4 models. Further data augmentation or an increase in the dataset size and class distribution used to train the 2D detector is likely to increase detection accuracy through more generalized learned features.

Chapter 8

Summary

In conclusion, as part of this thesis, the synchronization of timestamps for camera-sensors is implemented on the Providentia++ infrastructure using the PTP-1588 industrial standard. Additionally, an Ouster OS-1 64-layer LiDAr is enabled to send trigger signals for camera image data acquisition to cameras located within the same ROS system environment. As a result, more accurate synchronization is enabled for future data recording activities using camera and LiDAR sensors located on the Providentia++ infrastructure.

Camera-LiDAR sensor pairs are calibrated using target-based and targetless calibration methods. Based on output calibration data, projection matrices and extrinsic matrices for sensors at the s110 sensor station are calculated. The validity of these projection matrices is shown in exemplary data frames for LiDAR and camera data.

Inside the fused multi-modality object detection pipeline based on CLOCs implemented in previous work, a more accurate 2D object detector YOLOR, is implemented and tested. The YOLOR detector is integrated into the CLOCs fusion architecture as well as ROS in order to be deployed on the Providentia++ infrastructure. Due to this, detection candidates received by the CLOCs fusion layer are shown to be more accurate than those previously based on a YOLOv4 model.

List of Figures

List of Tables

Bibliography

[1] Behley, J., Garbade, M., Milioto, A., Quenzel, J., Behnke, S., Stachniss, C., and Gall, J. "SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences". In: *Proc. of the IEEE/CVF International Conf. on Computer Vision (ICCV)*. 2019.

[2] Caesar, H., Bankiti, V., Lang, A. H., Vora, S., Liong, V. E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., and Beijbom, O. "nuScenes: A multimodal dataset for autonomous driving". In: *arXiv preprint arXiv:1903.11027* (2019).

[3] Caesar, H., Bankiti, V., Lang, A. H., Vora, S., Liong, V. E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., and Beijbom, O. *nuScenes: A multimodal dataset for autonomous driving*. 2020. arXiv: 1903.11027 [cs.LG].

[4] Chadwick, S., Maddern, W., and Newman, P. *Distant Vehicle Detection Using Radar and Vision*. 2019. arXiv: 1901.10951 [cs.RO].

[5] Chang, M.-F., Lambert, J., Sangkloy, P., Singh, J., Bak, S., Hartnett, A., Wang, D., Carr, P., Lucey, S., Ramanan, D., and Hays, J. *Argoverse: 3D Tracking and Forecasting with Rich Maps*. 2019. arXiv: 1911.02620 [cs.CV].

[6] Chen, C., Fragonara, L. Z., and Tsourdos, A. "RoIFusion: 3D Object Detection From LiDAR and Vision". In: *IEEE Access* 9 (2021), pp. 51710–51721. DOI: 10.1109/ACCESS.2021.3070379.

[7] Chen, X., Kundu, K., Zhang, Z., Ma, H., Fidler, S., and Urtasun, R. "Monocular 3D Object Detection for Autonomous Driving". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 2147–2156. DOI: 10.1109/CVPR.2016.236.

[8] Chen, X., Ma, H., Wan, J., Li, B., and Xia, T. "Multi-view 3D Object Detection Network for Autonomous Driving". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 6526–6534. DOI: 10.1109/CVPR.2017.691.

[9] Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., and Schiele, B. "The Cityscapes Dataset for Semantic Urban Scene Understanding". In: *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[10] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. "Imagenet: A large-scale hierarchical image database". In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[11] Déziel, J.-L., Merriaux, P., Tremblay, F., Lessard, D., Plourde, D., Stanguennec, J., Goulet, P., and Olivier, P. *PixSet : An Opportunity for 3D Computer Vision to Go Beyond Point Clouds With a Full-Waveform LiDAR Dataset*. 2021. arXiv: 2102.12010 [cs.RO].

[12] Dollar, P., Wojek, C., Schiele, B., and Perona, P. "Pedestrian detection: A benchmark". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 304-311. DOI: 10.1109/CVPR.2009.5206631. URL: https://ieeexplore.ieee.org/abstract/document/5206631.

[13] Fang, J., Zuo, X., Zhou, D., Jin, S., Wang, S., and Zhang, L. “LiDAR-Aug: A General Rendering-Based Augmentation Framework for 3D Object Detection”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2021, pp. 4710–4720.

[14] Ge, Z., Liu, S., Wang, F., Li, Z., and Sun, J. *YOLOX: Exceeding YOLO Series in 2021*. 2021. arXiv: 2107.08430 `[cs.CV]`.

[15] Geiger, A., Lenz, P., Stiller, C., and Urtasun, R. “Vision meets Robotics: The KITTI Dataset”. In: *International Journal of Robotics Research (IJRR)* (2013).

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

[17] Geyer, J., Kassahun, Y., Mahmudi, M., Ricou, X., Durgesh, R., Chung, A. S., Hauswald, L., Pham, V. H., Mühlegg, M., Dorn, S., Fernandez, T., Jänicke, M., Mirashi, S., Savani, C., Sturm, M., Vorobiov, O., Oelker, M., Garreis, S., and Schuberth, P. “A2D2: Audi Autonomous Driving Dataset”. In: (2020). arXiv: 2004.06320 `[cs.CV]`. URL: https://www.a2d2.audi.

[18] Girshick, R. *Fast R-CNN*. 2015. arXiv: 1504.08083 `[cs.CV]`.

[19] Girshick, R., Donahue, J., Darrell, T., and Malik, J. “Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation”. In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 580–587. DOI: 10.1109/CVPR.2014.81.

[20] Girshick, R., Donahue, J., Darrell, T., and Malik, J. *Rich feature hierarchies for accurate object detection and semantic segmentation*. 2014. arXiv: 1311.2524 `[cs.CV]`.

[21] He, C., Zeng, H., Huang, J., Hua, X.-S., and Zhang, L. “Structure Aware Single-stage 3D Object Detection from Point Cloud”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2020.

[22] He, K., Gkioxari, G., Dollár, P., and Girshick, R. *Mask R-CNN*. 2018. arXiv: 1703.06870 `[cs.CV]`.

[23] Huang, X., Wang, P., Cheng, X., Zhou, D., Geng, Q., and Yang, R. “The ApolloScape Open Dataset for Autonomous Driving and Its Application”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42.10 (Oct. 2020), pp. 2702–2719. ISSN: 1939-3539. DOI: 10.1109/tpami.2019.2926463. URL: http://dx.doi.org/10.1109/TPAMI.2019.2926463.

[24] “IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems”. In: *IEEE Std 1588-2019 (Revision of IEEE Std 1588-2008)* (2020), pp. 1–499. DOI: 10.1109/IEEESTD.2020.9120376.

[25] Kanjee, R. *YOLOR performance, Accessed on 26.11.2021*. https://medium.com/augmented-startups/is-yolor-better-and-faster-than-yolov4-54812da66cc1. 2019.

[26] Kesten, R., Usman, M., Houston, J., Pandya, T., Nadhamuni, K., Ferreira, A., Yuan, M., Low, B., Jain, A., Ondruska, P., Omari, S., Shah, S., Kulkarni, A., Kazakova, A., Tao, C., Platinsky, L., Jiang, W., and Shet, V. *Lyft Level 5 Perception Dataset 2020*. https://level5.lyft.com/dataset/. 2019.

[27] Kesten, R., Usman, M., Houston, J., Pandya, T., Nadhamuni, K., Ferreira, A., Yuan, M., Low, B., Jain, A., Ondruska, P., Omari, S., Shah, S., Kulkarni, A., Kazakova, A., Tao, C., Platinsky, L., Jiang, W., and Shet, V. *Woven Planet Holdings, Inc. 2019,, Level 5 Perception Dataset 2020*. https://level-5.global/level5/data/. 2019.

[28] Krämmer, A., Schöller, C., Gulati, D., and Knoll, A. “Providentia - A Large Scale Sensing System for the Assistance of Autonomous Vehicles”. In: *Robotics Science and Systems Workshops (RSS Workshops)*. Freiburg, Germany: RSS Foundation, June 2019. URL: https://sites.google.com/view/uad2019/accepted-posters.

[29] Krizhevsky, A. *One weird trick for parallelizing convolutional neural networks*. 2014. arXiv: 1404.5997 [cs.NE].

[30] Ku, J., Mozifian, M., Lee, J., Harakeh, A., and Waslander, S. L. “Joint 3D Proposal Generation and Object Detection from View Aggregation”. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2018, pp. 1–8. DOI: 10.1109/IROS.2018.8594049.

[31] Kuroda, T. *Essential Principles of Image Sensors*. CRC Press, 2017. ISBN: 9781315215419. DOI: 10.1201/b17411.

[32] Lang, A. H., Vora, S., Caesar, H., Zhou, L., Yang, J., and Beijbom, O. “PointPillars: Fast Encoders for Object Detection from Point Clouds”. In: *CVPR*. 2019.

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

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

[35] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. “SSD: Single Shot MultiBox Detector”. In: *Computer Vision – ECCV 2016*. Ed. by Leibe, B., Matas, J., Sebe, N., and Welling, M. Vol. 9905. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2016, pp. 21–37. ISBN: 978-3-319-46447-3. DOI: 10.1007/978-3-319-46448-0_2.

[36] Liu, Z., Zhao, X., Huang, T., Hu, R., Zhou, Y., and Bai, X. *TANet: Robust 3D Object Detection from Point Clouds with Triple Attention*. 2019. arXiv: 1912.05163 [cs.CV].

[37] Liu, Z., Tang, H., Lin, Y., and Han, S. *Point-Voxel CNN for Efficient 3D Deep Learning*. 2019. arXiv: 1907.03739 [cs.CV].

[38] Lv, X., Wang, B., Ye, D., and Wang, S. *LCCNet: LiDAR and Camera Self-Calibration using Cost Volume Network*. 2021. arXiv: 2012.13901 [cs.CV].

[39] Mao, J., Niu, M., Jiang, C., Liang, X., Li, Y., Ye, C., Zhang, W., Li, Z., Yu, J., Xu, C., et al. “One Million Scenes for Autonomous Driving: ONCE Dataset”. In: 2021.

[40] Neuhold, G., Ollmann, T., Bulò, S. R., and Kontschieder, P. “The Mapillary Vistas Dataset for Semantic Understanding of Street Scenes”. In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 5000–5009. DOI: 10.1109/ICCV.2017.534.

[41] *OS1 Mid-Range High-Resolution Imaging Lidar Datasheet*. Revision: 7/1/2021. HARDWARE VERSION: 840-102145-C (Rev C). Ouster Inc. 2021. URL: https://data.ouster.io/downloads/datasheets/datasheet-revd-v2p1-os1.pdf.

[42] Ouster-Inc. *Ouster lidar: the most advanced technology powering the future of safe and efficient mobility*. 2020. URL: https://ouster.com/blog/ouster-lidar-the-most-advanced-technology-powering-the-future-of-safe-and-efficient-mobility/.

[43] Pang, S., Morris, D., and Radha, H. “CLOCs: Camera-LiDAR Object Candidates Fusion for 3D Object Detection”. In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020, pp. 10386–10393. DOI: 10.1109/IROS45743.2020.9341791.

[44] Pendleton, S. D. “Perception, Planning, Control, and Coordination for Autonomous Vehicles”. In: *machines* Machines 2017, 5, 6. (2017). DOI: 10.3390/machines5010006.

[45] Qi, C. R., Su, H., Mo, K., and Guibas, L. J. *PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation*. 2017. arXiv: 1612.00593 [cs.CV].

[46] Qi, C. R., Yi, L., Su, H., and Guibas, L. J. *PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space*. 2017. arXiv: 1706.02413 [cs.CV].

[47] Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. “You Only Look Once: Unified, Real-Time Object Detection”. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 779–788. DOI: 10.1109/CVPR.2016.91.

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

[49] Ren, S., He, K., Girshick, R., and Sun, J. “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.6 (2017), pp. 1137–1149. DOI: 10.1109/TPAMI.2016.2577031.

[50] S. Vora A. H. Lang, B. H. and Beijbom, O. “PointPainting: Sequential Fusion for 3D Object Detection”. In: *CVPR*. 2020.

[51] Salton, G. and McGill, M. *Introduction to Modern Information Retrieval*. International student edition. McGraw-Hill, 1983. ISBN: 9780070544840. URL: https://books.google.se/books?id=7f5TAAAAMAAJ.

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

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

[54] Sindagi, V. A., Zhou, Y., and Tuzel, O. *MVX-Net: Multimodal VoxelNet for 3D Object Detection*. 2019. arXiv: 1904.01649 [cs.CV].

[55] Stange, V., Vollrath, M., and Kuhn, M. *Hochautomatisiertes Fahren im Mischverkehr, Forschungbericht Nr. 71*. 2020. URL: https://udv.de/de/publikationen/forschungsberichte/hochautomatisiertes-fahren-im-mischverkehr.

[56] Sun, P., Kretzschmar, H., Dotiwalla, X., Chouard, A., Patnaik, V., Tsui, P., Guo, J., Zhou, Y., Chai, Y., Caine, B., Vasudevan, V., Han, W., Ngiam, J., Zhao, H., Timofeev, A., Ettinger, S., Krivokon, M., Gao, A., Joshi, A., Zhao, S., Cheng, S., Zhang, Y., Shlens, J., Chen, Z., and Anguelov, D. *Scalability in Perception for Autonomous Driving: Waymo Open Dataset*. 2020. arXiv: 1912.04838 [cs.CV].

[57] *Testrecke-gesamt.jpg*. Chair of Robotics, Artifical Intelligence and Real-time Systems, Department of Informatics, Technical University of Munich. 2021. URL: https://innovation-mobility.com/projekt-providentia/.

[58] Toponce, A. *Real Life NTP, Strata, Accessed on 26.11.2021, created on 05.11.2013*. https://pthree.org/2013/11/05/real-life-ntp/. 2019.

[59] *Valeo SCALA® 3D Laser Scanner (Gen 2) User Manual*. Version 1.1. Valeo S.A. 2020.

[60] Wang, C.-Y., Bochkovskiy, A., and Liao, H.-Y. M. *Scaled-YOLOv4: Scaling Cross Stage Partial Network*. 2021. arXiv: 2011.08036 [cs.CV].

[61] Wang, C.-Y., Yeh, I.-H., and Liao, H.-Y. M. *You Only Learn One Representation: Unified Network for Multiple Tasks*. 2021. arXiv: 2105.04206 [cs.CV].

[62] Wang, C., Ma, C., Zhu, M., and Yang, X. “PointAugmenting: Cross-Modal Augmentation for 3D Object Detection”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2021, pp. 11794–11803.

[63] Wang, H., Wang, C., Chen, C.-L., and Xie, L. *F-LOAM: Fast LiDAR Odometry And Mapping*. 2021. arXiv: 2107.00822 [cs.RO].

[64] Wang, Y., Chao, W.-L., Garg, D., Hariharan, B., Campbell, M., and Weinberger, K. Q. *Pseudo-LiDAR from Visual Depth Estimation: Bridging the Gap in 3D Object Detection for Autonomous Driving*. 2020. arXiv: 1812.07179 [cs.CV].

[65] Wang, Y., Mao, Q., Zhu, H., Zhang, Y., Ji, J., and Zhang, Y. *Multi-Modal 3D Object Detection in Autonomous Driving: a Survey*. 2021. arXiv: 2106.12735 [cs.CV].

[66] Wen, L.-H. and Jo, K.-H. “Fast and Accurate 3D Object Detection for Lidar-Camera-Based Autonomous Vehicles Using One Shared Voxel-Based Backbone”. In: *IEEE Access* 9 (2021), pp. 22080–22089. DOI: 10.1109/ACCESS.2021.3055491.

[67] Xu, S., Zhou, D., Fang, J., Yin, J., Bin, Z., and Zhang, L. *FusionPainting: Multimodal Fusion with Adaptive Attention for 3D Object Detection*. 2021. arXiv: 2106.12449 [cs.CV].

[68] Xu, W., Cai, Y., He, D., Lin, J., and Zhang, F. *FAST-LIO2: Fast Direct LiDAR-inertial Odometry*. 2021. arXiv: 2107.06829 [cs.RO].

[69] Xu, W. and Zhang, F. *FAST-LIO: A Fast, Robust LiDAR-inertial Odometry Package by Tightly-Coupled Iterated Kalman Filter*. 2021. arXiv: 2010.08196 [cs.RO].

[70] Yan, Y., Mao, Y., and Li, B. “SECOND: Sparsely Embedded Convolutional Detection”. In: *Sensors* 18.10 (2018). ISSN: 1424-8220. DOI: 10.3390/s18103337. URL: https://www.mdpi.com/1424-8220/18/10/3337.

[71] Yang, Z., Sun, Y., Liu, S., and Jia, J. *3DSSD: Point-based 3D Single Stage Object Detector*. 2020. arXiv: 2002.10187 [cs.CV].

[72] Yang, Z., Sun, Y., Liu, S., Shen, X., and Jia, J. *STD: Sparse-to-Dense 3D Object Detector for Point Cloud*. 2019. arXiv: 1907.10471 [cs.CV].

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

[74] Yoo, J. H., Kim, Y., Kim, J., and Choi, J. W. “3D-CVF: Generating Joint Camera and LiDAR Features Using Cross-view Spatial Feature Fusion for 3D Object Detection”. In: *Lecture Notes in Computer Science* (2020), pp. 720–736. ISSN: 1611-3349. DOI: 10.1007/978-3-030-58583-9_43. URL: http://dx.doi.org/10.1007/978-3-030-58583-9_43.

[75] Yu, F., Chen, H., Wang, X., Xian, W., Chen, Y., Liu, F., Madhavan, V., and Darrell, T. *BDD100K: A Diverse Driving Dataset for Heterogeneous Multitask Learning*. 2020. arXiv: 1805.04687 [cs.CV].

[76] Yuan, C., Liu, X., Hong, X., and Zhang, F. *Pixel-level Extrinsic Self Calibration of High Resolution LiDAR and Camera in Targetless Environments*. 2021. arXiv: 2103.01627 [cs.RO].

[77] Zhang, W., Wang, Z., and Loy, C. C. *Exploring Data Augmentation for Multi-Modality 3D Object Detection*. 2021. arXiv: 2012.12741 [cs.CV].

[78] Zheng, W., Tang, W., Jiang, L., and Fu, C.-W. *SE-SSD: Self-Ensembling Single-Stage Object Detector From Point Cloud*. 2021. arXiv: 2104.09804 [cs.CV].

[79] Zhou, X., Wang, D., and Krähenbühl, P. *Objects as Points*. 2019. arXiv: 1904.07850 [cs.CV].

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

[81] Zimmer, W., Rangesh, A., and Trivedi, M. *3D BAT: A Semi-Automatic, Web-based 3D Annotation Toolbox for Full-Surround, Multi-Modal Data Streams*. 2019. arXiv: 1905.00525 [cs.CV].