

Hochschule Darmstadt

Fachbereiche Mathematik und
Naturwissenschaften & Informatik

3D-Deep Learning zur Erkennung kleiner Objekte in LiDAR-Daten auf On-Board-Systemen

Abschlussarbeit zur Erlangung des akademischen Grades

Master of Science (M.Sc.)
im Studiengang Data Science

vorgelegt von

Yousef Abdullah

Matrikelnummer: 1125193

Referentin : Prof. Dr. Elke Hergenröther

Korreferent : Prof. Dr. Thomas März

Ausgabedatum : 22.12.2025

Abgabedatum : 05.07.2026

ERKLÄRUNG

Ich versichere hiermit, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die im Literaturverzeichnis angegebenen Quellen benutzt habe.

Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder noch nicht veröffentlichten Quellen entnommen sind, sind als solche kenntlich gemacht.

Die Zeichnungen oder Abbildungen in dieser Arbeit sind von mir selbst erstellt worden oder mit einem entsprechenden Quellennachweis versehen.

Diese Arbeit ist in gleicher oder ähnlicher Form noch bei keiner anderen Prüfungsbehörde eingereicht worden.

Während der Vorbereitung dieser Arbeit habe ich ChatGPT (OpenAI) verwendet, um die Lesbarkeit zu verbessern, Formulierungen zu optimieren und sprachliche Korrekturen vorzunehmen. Nach der Nutzung dieses Tools habe ich sämtliche Inhalte überprüft, bei Bedarf überarbeitet und übernehme die volle Verantwortung für den Inhalt dieser Arbeit.

Darmstadt, 05.07.2026

Yousef Abdullah

ABSTRACT

The robust detection of small objects in Light Detection and Ranging (LiDAR) point clouds represents a fundamental challenge for autonomous driving systems. Particularly at greater distances, the limited spatial resolution of the sensor often provides only few measurement points per object, which complicates the extraction of meaningful features and thus robust object recognition. In the context of the university project Formula Student Autonomous Darmstadt (FASTDA), where an autonomous race car must autonomously navigate a track marked by traffic cones, this problem gains particular significance. Precise and real-time perception of track boundaries is therefore a key prerequisite for safe and reliable vehicle operation.

This thesis investigates the suitability of modern three-dimensional deep learning methods for detecting small and point-sparse objects in sparse LiDAR point clouds. To this end, six state-of-the-art 3D object detectors from different architectural classes: PointPillars (voxel-based), 3DSSD (point-based), PV-RCNN (hybrid), DeLiVoTr (attention-based), LION (sequence-based Mamba architecture), and VoxelNeXt (fully sparse) are fine-tuned via transfer learning on a custom dataset and systematically evaluated.

The results demonstrate that all investigated methods exhibit a pronounced dependency on available point density. In particular, beyond a critical distance, the small number of LiDAR measurement points per object leads to a significant degradation in detection performance. This motivated the investigation of two distinct approaches for improving recognition performance in the critical far-range region. While the integration of geometric shape priors was discarded due to insufficient sequential data, high architectural complexity, and limited effectiveness under extreme sparsity, data-driven point cloud densification via point cloud super-resolution achieves improved recall rates while maintaining real-time processing capability.

The findings demonstrate that fully sparse 3D detection architectures hold significant potential for deployment in safety-critical and resource-constrained autonomous systems. Simultaneously, low point density of small objects in mid-range and far-range distances is identified as a central challenge. Improving the information density of sparse point clouds thus represents a promising avenue for future research.

Keywords: LiDAR, 3D object detection, deep learning, autonomous vehicles, small objects, sparse point clouds, real-time systems, Formula Student.

ZUSAMMENFASSUNG

Die zuverlässige Detektion kleiner Objekte in LiDAR-Punktwolken stellt eine wesentliche Herausforderung für autonome Fahrsysteme dar. Insbesondere bei größeren Entfernungen stehen aufgrund der begrenzten räumlichen Auflösung des Sensors häufig nur wenige Messpunkte pro Objekt zur Verfügung, was die Extraktion aussagekräftiger Merkmale und damit eine robuste Objekterkennung erschwert. Im Kontext des Hochschulprojekts FASTDA, in dem ein autonomes Rennfahrzeug eine durch Verkehrshütchen markierte Rennstrecke selbstständig befahren soll, gewinnt diese Problemstellung besondere Bedeutung. Eine präzise und gleichzeitig echtzeitfähige Wahrnehmung der Streckenbegrenzung stellt hierbei eine zentrale Voraussetzung für einen sicheren und zuverlässigen Fahrzeugbetrieb dar.

Die vorliegende Arbeit untersucht die Eignung moderner dreidimensionaler Deep-Learning-Verfahren zur Detektion kleiner und punktarmer Objekte in spärlichen LiDAR-Punktwolken. Hierzu werden sechs repräsentative 3D-Objektdetektoren unterschiedlicher Architekturklassen: PointPillars (voxelbasiert), 3DSSD (punktbasiert), PV-RCNN (hybrid), DeLiVoTr (aufmerksamkeitsbasiert), LION (sequenziell/Mamba-basiert) und VoxelNeXt (fully sparse) mittels Transfer Learning auf einem eigens erhobenen Datensatz feinjustiert und systematisch evaluiert. Die Ergebnisse zeigen, dass alle untersuchten Verfahren eine deutliche Abhängigkeit von der verfügbaren Punktdichte aufweisen. Insbesondere ab einer kritischen Entfernung führt die geringe Anzahl an LiDAR-Messpunkten pro Objekt zu einer signifikanten Abnahme der Detektionsleistung, was zur Untersuchung von zwei verschiedenen Ansätzen zur Verbesserung der Erkennungsleistung im kritischen Fernbereich motiviert. Während die Integration geometrischer Shape Priors aufgrund unzureichender Sequenzdaten, hoher Architekturkomplexität und eingeschränkter Wirksamkeit bei extremer Punktarmut verworfen wird, führt die datengetriebene Punktverdichtung mittels Super-Resolution hingegen zu einer verbesserten Erkennungsrate bei gleichzeitig weiterhin echtzeitfähiger Verarbeitung. Die gewonnenen Erkenntnisse zeigen, dass vollständig spärliche 3D-Detektionsarchitekturen ein hohes Potenzial für den Einsatz in sicherheitskritischen und ressourcenbeschränkten autonomen Systemen besitzen. Gleichzeitig wird die geringe Punktdichte kleiner Objekte im mittleren und größeren Entfernungsbereich als zentrale Herausforderung identifiziert. Die Verbesserung der Informationsdichte spärlicher Punktwolken stellt daher einen vielversprechenden Ansatzpunkt für zukünftige Forschungsarbeiten dar.

Schlüsselwörter: LiDAR, 3D-Objektdetektion, Deep Learning, autonome Fahrzeuge, kleine Objekte, spärliche Punktwolken, Echtzeitsysteme, Formula Student.

INHALTSVERZEICHNIS

1	Einleitung	1
1.1	Motivation	1
1.2	Ziel der Arbeit	2
1.3	Gliederung	3
2	Theoretische Grundlagen	4
2.1	LiDAR-Sensortechnologie und Punktwolken	4
2.2	Datensatz	6
2.2.1	nuScenes-Datensatz	6
2.2.2	Eigener Datensatz	7
2.3	Deep-Learning-Architekturen für die 3D-Detektion	8
2.3.1	Punktbasierte Architekturen	9
2.3.2	Voxelbasierte Architekturen	9
2.3.3	Projektionsbasierte Architekturen	10
2.3.4	Hybride Architekturen	10
2.3.5	Anker-basierte und ankerfreie Detektoren	11
2.4	Open-Source-Frameworks für die 3D-Objekterkennung	11
2.5	Evaluationsmetriken	12
2.5.1	Detektionsqualität	12
2.5.2	Laufzeiteffizienz	15
3	Methodik und Ergebnisse	16
3.1	Auswahl der Modelle und methodische Vorgehensweise	16
3.1.1	Modelladaptierung und Trainingssetup	16
3.1.2	Vorstellungsstruktur der Modelle	17
3.2	PointPillars	17
3.2.1	Architektur	17
3.2.2	Warum PointPillars für kleine Objekte?	18
3.2.3	Konfiguration und Hyperparameter	18
3.2.4	Ergebnisse	19
3.2.5	Diskussion und Limitationen	20
3.3	3DSSD	22
3.3.1	Architektur	22
3.3.2	Warum 3DSSD für kleine Objekte?	23
3.3.3	Konfiguration und Hyperparameter	23
3.3.4	Ergebnisse	24
3.3.5	Diskussion und Limitationen	25
3.4	PV-RCNN	27
3.4.1	Architektur	27
3.4.2	Warum PV-RCNN für kleine Objekte?	28
3.4.3	Konfiguration und Hyperparameter	29
3.4.4	Ergebnisse	30
3.4.5	Diskussion und Limitationen	30
3.5	DeLiVoTr	32

3.5.1	Architektur	32
3.5.2	Warum DeLiVoTr für kleine Objekte?	33
3.5.3	Konfiguration und Hyperparameter	33
3.5.4	Ergebnisse	33
3.5.5	Diskussion und Limitationen	34
3.6	LION	36
3.6.1	Architektur	36
3.6.2	Warum LION für kleine Objekte?	37
3.6.3	Konfiguration und Hyperparameter	39
3.6.4	Ergebnisse	39
3.6.5	Diskussion und Limitationen	39
3.7	VoxelNeXt	41
3.7.1	Architektur	41
3.7.2	Warum VoxelNeXt für kleine Objekte?	42
3.7.3	Konfiguration und Hyperparameter	43
3.7.4	Ergebnisse	43
3.7.5	Diskussion	44
3.7.6	Fehleranalyse	45
3.8	Gesamtbewertung der untersuchten Modelle	46
4	Ansätze zur Verbesserung der Objekterkennung kleiner LiDAR- Objekte im Fernbereich	50
4.1	Analyse der Datensatzverteilung	50
4.2	Untersuchte Lösungsansätze	52
4.2.1	Shape Priors als Netzwerkerweiterung	52
4.2.2	Punktverdichtung durch Super-Resolution	55
5	Diskussion	59
5.1	Architekturspezifische Erkenntnisse	59
5.2	Empfehlung für das FASTDA-Projekt und kritische Reflexion	60
6	Fazit	62
7	Zusammenfassung und Ausblick	64
	Literatur	66

ABBILDUNGSVERZEICHNIS

Abbildung 1.1	Das im Rahmen des FASTDA-Projekts eingesetzte Formula -Student-Rennfahrzeug der Hochschule Darmstadt bei der Durchfahrt einer durch Pylonen markierten Strecke.	1
Abbildung 2.1	Beispiel einer farbkodierten LiDAR-Punktwolke (Seitenansicht). Die Farbkodierung visualisiert die Distanz zum Sensor: Nahe Punkte erscheinen rot, mittlere Distanzen grün und weit entfernte Punkte blau. Die konzentrischen Bögen im unteren Bereich entsprechen den Scanringen auf der Bodenebene, während die kleinen vertikalen Cluster in der Mitte Verkehrshütchen und die ausgedehnten Strukturen links und rechts Gebäudefassaden darstellen. Der in der BEV als schwarzer Kreis sichtbare sensorbedingte blinde Fleck nimmt mit steigender Montagehöhe zu. Gemäß den Richtlinien des FASTDA-Handbuchs ist eine Sensormontagehöhe zwischen 1,0 m und 1,5 m vorgeschrieben, was zwangsläufig zu einem entsprechend großen blinden Fleck führt.	5
Abbildung 2.2	Im FASTDA-Projekt verwendete Verkehrshütchen (eigene Darstellung).	5
Abbildung 2.3	Übersicht über Deep-Learning-Architekturen für die 3D-Objekterkennung in LiDAR-Daten. Die Punktwolke wird zunächst in verschiedene Repräsentationen überführt, anschließend durch ein Backbone verarbeitet und ein Detektionskopf erzeugt die finalen 3D-Bounding-Boxes [21].	8
Abbildung 3.1	Architektur von PV-RCNN. Die Punktwolke wird zunächst voxelbasiert verarbeitet, anschließend über Key-points mit mehrskaligen Voxel-Features angereichert und schließlich mittels RoI-Grid-Pooling für die Bounding -Box-Verfeinerung genutzt. Quelle: nach Shi u. a. [39].	27
Abbildung 3.2	Architekturübersicht des DeLiVoTr-Modells. Die Eingabepunktwolke wird zunächst dynamisch voxelisiert und anschließend durch den Encoder verarbeitet, der aus <i>Intra-Region</i> - und <i>Inter-Region</i> -Transformer-Schichten besteht. Der Decoder erzeugt schließlich die finalen 3D-Objektvorhersagen. Abbildung nach [38].	32

Abbildung 3.3	Architektur des LION-Modells nach [17]. (a) Struktur eines LION-Blocks mit vier LION-Layern, Voxel-Merging- und Expanding-Operationen sowie 3D Spatial Feature Descriptors. (b) Voxel-Merging für Downsampling und Voxel-Expanding für Upsampling. (c) Aufbau eines LION-Layers mit Mamba-Operator. (d) Details des 3D Spatial Feature Descriptors.	36
Abbildung 3.4	Recall von LION für die Klasse <i>small_cone</i> nach Distanzbereich. Blau kennzeichnet Distanzbereiche mit zuverlässiger Erkennung, Rot kennzeichnet Bereiche mit eingeschränkter bzw. fehlender Detektionsleistung.	41
Abbildung 3.5	Qualitative Ergebnisse von VoxelNeXt für die Klasse <i>small_cone</i> (Frame 25, 129.330 Punkte gesamt). Die Verkehrshütchen sind halbkreisförmig angeordnet. Der schwarze Kreis in der Mitte der Bird's-Eye-View entspricht dem blinden Fleck des LiDAR-Sensors (OS-2-128), in dem keine Messpunkte erfasst werden. Grüne Bounding Boxes kennzeichnen korrekt erkannte Hütchen (<i>True Positives</i> , TP = 20), orange Boxes markieren übersehene Hütchen (<i>False Negatives</i> , FN = 4), rote Boxes stehen für Fehldetektionen (<i>False Positives</i> , FP = 3). Im Hintergrund sind Gebäudestrukturen als großflächige Punktwolken sichtbar.	45
Abbildung 3.6	Recall von VoxelNeXt für die Klasse <i>small_cone</i> nach Distanzbereich. Blau kennzeichnet Distanzbereiche mit zuverlässiger Erkennung, Rot kennzeichnet Bereiche mit eingeschränkter bzw. fehlender Detektionsleistung.	46
Abbildung 3.7	Gesamtvergleich der evaluierten Modelle anhand normalisierter Metriken (Radar Chart). Die Achsen zeigen Precision, Recall, F1-Score, AP@0.5, Low Latency und Speed (FPS), jeweils auf einen Wertebereich von 0–100 normalisiert. Dabei ist zu beachten, dass Latenz und FPS gegenläufige Qualitätsindikatoren sind: Eine niedrige Latenz (ms) bedeutet schnellere Verarbeitung und ist daher wünschenswert, während eine hohe FPS-Zahl ebenfalls auf eine bessere Laufzeiteffizienz hindeutet. Um beide Metriken einheitlich darstellbar zu machen, wurde die Latenzachse invertiert – hohe Werte auf der Low Latency-Achse entsprechen also einer <i>niedrigen</i> tatsächlichen Latenz in ms – während die Speed-Achse (FPS) direkt normalisiert wurde. Somit entsprechen größere Flächen im Radar Chart einer besseren Gesamtleistung. LION und VoxelNeXt dominieren bei der Detektionsqualität, während VoxelNeXt als einziges Modell zugleich eine hohe Laufzeiteffizienz aufweist.	48

Abbildung 3.8	Recall von LION (5,4 FPS) und VoxelNeXt (55,0 FPS) für die Klasse <code>small_cone</code> nach Distanzbereich. Im Nahbereich bis 10 m erfüllen beide Modelle die Mindestanforderung von 80 % deutlich — LION erreicht bis zu 97,6 %, VoxelNeXt bis zu 93,2 %. Im Bereich 10–15 m nimmt die Erkennungsleistung deutlich ab (LION: 60,0 %, VoxelNeXt: 64,8 %). Ab 15 m bricht sie drastisch ein (LION: 0,0 %, VoxelNeXt: 20,0 %), was mit der abnehmenden Punktdichte des Ouster OS2-128-Sensors korreliert. Die grün gestrichelte Linie markiert den Zielwert von 90 %, die orange gepunktete Linie die Mindestanforderung von 80 %.	49
Abbildung 4.1	Durchschnittliche Punktzahl pro Kegel in Abhängigkeit von der Entfernung. Die gestrichelten Linien markieren den Referenzwert guter Erkennungsbedingungen bei etwa 8 Punkten sowie den empirischen Mindestwert für VoxelNeXt bei etwa 3 Punkten.	51

Abbildung 4.2	Gesamtarchitektur des JS3C-Net. Die Pipeline nimmt eine spärliche, unvollständige Single-Sweep-Punktwolke ($N \times 3$) als Eingabe und verarbeitet diese zunächst durch ein SparseConv U-Net, das durch effiziente Submanifold Sparse Convolutions eine kompakte Punktfeature -Repräsentation $\mathbf{F}_{enc} \in \mathbb{R}^{N \times F}$ erzeugt. Auf Basis dieser Encoding-Features generiert MLP_1 ein Shape Embedding $\mathbf{F}_{SE} \in \mathbb{R}^{N \times D^e}$, welches die geometrischen Formeigenschaften jedes Punktes in einem kompakten Raum kodiert. Parallel dazu transformiert MLP_2 die initialen Encoding-Features, die anschließend elementweise mit $\mathbf{F}_{SE}$ summiert und durch MLP_3 zu den finalen Punktfeatures $\mathbf{F}_{out} \in \mathbb{R}^{N \times C}$ verarbeitet werden, welche die semantischen Klassenwahrscheinlichkeiten für die Semantic Segmentation repräsentieren. Im Anschluss fließen $\mathbf{F}_{out}$ in das Semantic Scene Completion (SSC)-Modul, das durch Voxelisierung und mehrere dichte Faltungsblöcke mit Skip-Connections die sparse Eingabe zu einer vollständigen 3D- Szenenrepräsentation vervollständigt. Das Point-Voxel Interaction (PVI)-Modul führt schließlich die feingranularen, aber unvollständigen Punktfeatures aus dem Shape Embedding mit den groben, aber vollständigen Voxelfeatures des SSC-Moduls zusammen, indem es über einen k -Nearest-Neighbor-Mechanismus und ein Graph Convolutional Network die räumlichen und semantischen Beziehungen zwischen Punkten und Voxeln modelliert und so eine verfeinerte Szenenkompletion erzeugt. Hervorzuheben ist, dass das SSC- und das PVI-Modul ausschließlich während des Trainings aktiv sind und zur Inferenzzeit verworfen werden (gestrichelte rote Linie), sodass kein zusätzlicher Rechenaufwand für die Semantic Segmentation entsteht. Abbildung adaptiert von [42].	54
Abbildung 4.3	Interpolation durch gewichtete Summe der Nachbarpixel. Das leere Gitter repräsentiert freie Bereiche ohne LiDAR-Punkte.	55
Abbildung 4.4	Recall pro Meter, Baseline vs. Super-Resolution für VoxelNeXt (Klasse small_cone). Im FOKUS-Bereich (10–15 m) erzielt Super Resolution die größten Verbesserungen, insbesondere bei 13 m mit +13,3 %. Die grauen Balken zeigen die Anzahl der Ground-Truth-Kegel pro Distanzbereich (rechte Achse).	57

TABELLENVERZEICHNIS

Tabelle 2.1	Kennzahlen des nuScenes-Datensatzes	6
Tabelle 2.2	Technische Spezifikationen des Ouster OS2-128	7
Tabelle 2.3	Kennzahlen des eigenen Datensatzes	8
Tabelle 3.1	Überblick der evaluierten Modelle	16
Tabelle 3.2	Optimale Konfiguration von PointPillars	19
Tabelle 3.3	Detektions- und Laufzeitmetriken PointPillars	20
Tabelle 3.4	Trainingsparameter der 3DSSD-Konfigurationen	24
Tabelle 3.5	Detektions- und Laufzeitmetriken 3DSSD	25
Tabelle 3.6	Konfiguration des PV-RCNN-Modells	29
Tabelle 3.7	Detektions- und Laufzeitmetriken PV-RCNN	30
Tabelle 3.8	Konfiguration des DeLiVoTr-Modells	34
Tabelle 3.9	Detektions- und Laufzeitmetriken DeLiVoTr	34
Tabelle 3.10	Konfiguration des LION-Modells	39
Tabelle 3.11	Detektions- und Laufzeitmetriken LION	39
Tabelle 3.12	Konfiguration des VoxelNeXt-Modells	43
Tabelle 3.13	Detektions- und Laufzeitmetriken VoxelNeXt	44
Tabelle 3.14	Distanzanalyse VoxelNeXt	45
Tabelle 3.15	Vergleich der evaluierten Modelle für die Klasse small_cone. 47	
Tabelle 4.1	Verteilung der annotierten Kegel nach Distanzbereich. 50	
Tabelle 4.2	Vergleich Baseline vs. Super-Resolution für VoxelNeXt im FOKUS-Bereich.	58

ABKÜRZUNGSVERZEICHNIS

AP	Average Precision
YOLO	You Only Look Once
BEV	Bird's Eye View
CVAT	Computer Vision Annotation Tool
DL	Deep Learning
FASTDA	Formula Student Autonomous Darmstadt
FPS	Frames per Second
GPU	Graphics Processing Unit
IoU	Intersection over Union
PVRCNN	Point-Voxel Region-based Convolutional Neural Network
DeLiVoTr	Deep and Lightweight Voxel Transformer
LiDAR	Light Detection and Ranging
LION	Linear Group RNN
mAP	Mean Average Precision
IMU	Inertial Measurement Unit
3D	dreidimensional
ToF	Time of Flight
CNN	Convolutional Neural Network
GPS	Global Positioning System
3DSSD	3D Single-Stage Detector
2D	zweidimensional
NMS	Non-Maximum Suppression
TP	True Positive
FP	False Positive
FN	False Negative
TN	True Negative
GAN	Generative Adversarial Network

EINLEITUNG

1.1 MOTIVATION

Im Rahmen des Hochschulprojekts FASTDA wird ein autonomes Fahrsystem (Driverless-System) für ein Formula-Student-Rennfahrzeug entwickelt (vgl. Abbildung 1.1), das eine durch Pylonen (Verkehrshütchen) markierte Strecke eigenständig befahren soll. Die zuverlässige Detektion und genaue Klassifizierung dieser Pylonen stellt dabei eine grundlegende Voraussetzung für die nachfolgende Trajektorienplanung sowie den sicheren Betrieb des Fahrzeugs dar.



Abbildung 1.1: Das im Rahmen des FASTDA-Projekts eingesetzte Formula -Student-Rennfahrzeug der Hochschule Darmstadt bei der Durchfahrt einer durch Pylonen markierten Strecke.

Zur Echtzeit-Objektdetektion in autonomen Fahrsystemen werden kamerabasierte Deep-Learning-Modelle wie You Only Look Once (YOLO) häufig eingesetzt, da sie eine hohe Detektionsgenauigkeit bei gleichzeitig geringer Inferenzzeit ermöglichen [1, 2]. Dennoch weisen kamerabasierte Systeme grundlegende Einschränkungen auf: Da sie die dreidimensionale Welt auf eine zweidimensionale Bildebene projizieren, stehen keine direkten Tiefeninformationen zur Verfügung. Zudem sind ihre Ergebnisse stark von variierenden Licht- und Wetterbedingungen abhängig [2, 3]. In sicherheitskritischen Anwendungen, in denen eine präzise räumliche Lokalisierung und Kollisionsvermeidung essenziell sind, stellen diese Einschränkungen einen wesentlichen Nachteil dar [4-6].

Aufgrund der fehlenden direkten Tiefeninformation und der starken Sensitivität gegenüber Licht- und Wetterbedingungen bei kamerabasierten Systemen wächst das Interesse an LiDAR-Technologien, die eine direkte dreidimensionale Erfassung der Umgebung und präzise Distanzmessungen unabhängig von den Umgebungsbedingungen ermöglichen. Dadurch können kleine und sicherheitsrelevante Objekte unter variierenden Licht- und Wetterbedingungen zuverlässiger erkannt und klassifiziert werden [7, 8].

Der Einsatz von LiDAR-Sensoren bringt jedoch eigene Herausforderungen mit sich. Mit zunehmender Entfernung vom Sensor nimmt die Punktdichte der erzeugten Punktwolken deutlich ab, sodass kleine Objekte häufig nur durch eine sehr geringe Anzahl an 3D-Punkten repräsentiert werden, was ihre zuverlässige Merkmalsextraktion und Detektion erheblich erschwert. Darüber hinaus erzeugen moderne Sensoren typischerweise zwischen 60.000 und 80.000 Messpunkte pro Frame, deren effiziente Verarbeitung unter Echtzeitbedingungen auf ressourcenbeschränkten On-Board-Systemen eine zentrale Herausforderung darstellt [3, 7]. Vor diesem Hintergrund adressiert diese Arbeit den Zielkonflikt zwischen Detektionsgenauigkeit und Recheneffizienz bei der LiDAR-basierten 3D-Objekterkennung. Der Fokus liegt auf der Analyse und dem Vergleich moderner 3D-Deep-Learning-Ansätze hinsichtlich ihrer Eignung, kleine Objekte trotz geringer Punktdichte zuverlässig zu erkennen und dabei gleichzeitig Echtzeitanforderungen zu erfüllen. Die daraus abgeleiteten Ziele und Forschungsfragen werden in Abschnitt 1.2 präzisiert.

1.2 ZIEL DER ARBEIT

Ziel dieser Arbeit ist die systematische Analyse und der Vergleich verschiedener 3D-Deep-Learning-Ansätze zur Detektion von Verkehrshütchen in dreidimensionalen LiDAR-Punktwolken. Dabei werden vortrainierte Modelle aus unterschiedlichen Architekturkategorien einschließlich punkt-, voxel-, hybrid- und aufmerksamkeitsbasierter Verfahren auf einem eigens erhobenen Datensatz mittels Fine-Tuning angepasst und anschließend evaluiert. Ein besonderer Fokus liegt auf der Untersuchung des Zielkonflikts zwischen Erkennungsgenauigkeit und Rechenaufwand, insbesondere im Hinblick auf den Einsatz in ressourcenbeschränkten On-Board-Systemen autonomer Rennfahrzeuge. Aus dieser Zielsetzung ergeben sich die folgenden Forschungsfragen:

1. Wie zuverlässig können verschiedene 3D-Deep-Learning-Ansätze kleine, punktarme Objekte wie Verkehrshütchen in LiDAR-Punktwolken unter Echtzeitbedingungen detektieren?
2. Welches Verhältnis zwischen Erkennungsleistung und Rechenaufwand weisen diese Modelle beim Einsatz auf ressourcenbeschränkten On-Board-Systemen autonomer Rennfahrzeuge auf?

1.3 GLIEDERUNG

Die Arbeit gliedert sich in sieben Kapitel, die jeweils einen spezifischen Aspekt der Untersuchung und ihrer Ergebnisse behandeln:

- **Kapitel 2: Theoretische Grundlagen.** Dieses Kapitel legt die theoretischen Grundlagen dar, die für das Verständnis der in dieser Arbeit untersuchten Ansätze unerlässlich sind. Es werden die LiDAR- Sensortechnologie und Punktwolkenrepräsentationen, die verwendeten Datensätze, relevante Deep-Learning-Architekturen für die 3D-Objekterkennung sowie die im Verlauf der Arbeit eingesetzten Evaluationsmetriken vorgestellt.
- **Kapitel 3: Methodik und Ergebnisse.** Dieses Kapitel beschreibt die Methodik und evaluiert sechs 3D-Objektdetektoren aus unterschiedlichen Architekturkategorien PointPillars, 3DSSD, PV-RCNN, DeLiVoTr, LION und VoxelNeXt hinsichtlich ihrer Detektionsleistung und Laufzeiteffizienz bei der Erkennung kleiner, punktarmer Objekte auf ressourcenbeschränkten On-Board-Systemen.
- **Kapitel 4: Ansätze zur Verbesserung der Objekterkennung kleiner LiDAR-Objekte im Fernbereich.** Dieses Kapitel untersucht zwei Ansätze zur Verbesserung der Erkennungsleistung im kritischen Fernbereich (10–15 m): die Integration geometrischer Shape Priors als Netzwerkerweiterung sowie die datengetriebene Punktverdichtung mittels Super-Resolution. Letztere wird implementiert und gegenüber der Baseline evaluiert.
- **Kapitel 5: Diskussion.** Dieses Kapitel fasst die Ergebnisse aller evaluierten Modelle zusammen und leitet daraus architekturspezifische Erkenntnisse ab. Dabei werden zentrale Entwurfsparmeter wie die Voxelgröße, Downsampling-Strategien sowie Komplexitätsmetriken kritisch analysiert und hinsichtlich ihres Einflusses auf die Detektionsleistung kleiner Objekte bewertet. Abschließend wird eine datenbasierte Empfehlung für den praktischen Einsatz im FASTDA-Projekt formuliert.
- **Kapitel 6: Fazit.** Dieses Kapitel beantwortet die beiden in Abschnitt 1.2 formulierten Forschungsfragen auf Basis der erzielten Ergebnisse und ordnet den Zielkonflikt zwischen Detektionsgenauigkeit und Laufzeiteffizienz kritisch ein.
- **Kapitel 7: Zusammenfassung und Ausblick.** Die Arbeit schließt mit einer Zusammenfassung der zentralen Erkenntnisse und ihrer Beiträge zum Forschungsfeld, gefolgt von Ansatzpunkten für zukünftige Arbeiten, darunter die Erweiterung des Datensatzes, lernbasierte Punktwolken-Vervollständigung, Hardware-Optimierungen sowie Sensor-Fusion- Ansätze.

Dieses Kapitel legt die theoretischen Grundlagen dar, die für das Verständnis der in dieser Arbeit untersuchten Ansätze unerlässlich sind. Zunächst wird die Funktionsweise von LiDAR-Sensoren erläutert und die daraus resultierende Punktwolkenrepräsentation vorgestellt. Anschließend werden die verschiedenen Datenrepräsentationen für die 3D-Objekterkennung beschrieben, gefolgt von einem Überblick über die relevanten Deep-Learning-Architekturen. Abschließend werden die Evaluationsmetriken definiert, anhand derer die Modelle in Kapitel 3 bewertet werden.

2.1 LIDAR-SENSORTECHNOLOGIE UND PUNKTWOLKEN

LiDAR (*Light Detection and Ranging*) ist ein aktives Fernerkundungsverfahren, das Entfernungen durch Messung der Laufzeit reflektierter Laserimpulse bestimmt [9, 10]. Die Distanz d zum reflektierenden Objekt ergibt sich nach dem Time-of-Flight-Prinzip (ToF) gemäß:

$$d = \frac{c \cdot \Delta t}{2}, \quad (2.1)$$

wobei $c \approx 3 \cdot 10^8$ die Lichtgeschwindigkeit und Δt die gemessene Laufzeit bezeichnet. Der Faktor 2 berücksichtigt den doppelten Signalweg zwischen Sensor und Objekt. Da der Sensor gleichzeitig den Azimutwinkel θ und den Elevationswinkel (auch Höhenwinkel genannt) ϕ erfasst, lassen sich die kartesischen Koordinaten jedes Messpunktes berechnen:

$$x = d \cos(\phi) \cos(\theta), \quad (2.2)$$

$$y = d \cos(\phi) \sin(\theta), \quad (2.3)$$

$$z = d \sin(\phi). \quad (2.4)$$

In dieser Arbeit wird der *Ouster OS-2* mit 128 Kanälen eingesetzt, der pro Frame typischerweise zwischen 60.000 und 70.000 Messpunkte erzeugt und eine Reichweite von bis zu 200 m abdeckt. Gegenüber kamerabasierten Systemen bietet LiDAR entscheidende Vorteile: direkte Tiefenmessung, Unabhängigkeit von Lichtverhältnissen sowie Robustheit gegenüber wechselnden Wetterbedingungen [11].

Eine vollständige LiDAR-Messung liefert eine Punktwolke $\mathcal{P} = \{\mathbf{p}_i\}_{i=1}^N$, wobei jeder Punkt durch

$$\mathbf{p}_i = (x_i, y_i, z_i, r_i)^\top \quad (2.5)$$

beschrieben wird. Hierbei repräsentieren (x_i, y_i, z_i) die räumlichen Koordinaten und r_i die Reflexionsintensität des Messpunktes [9]. Abbildung 2.1 zeigt eine typische farbkodierte LiDAR-Punktwolke, wie sie auch im Rahmen des FASTDA-Projekts aufgenommen wurde.

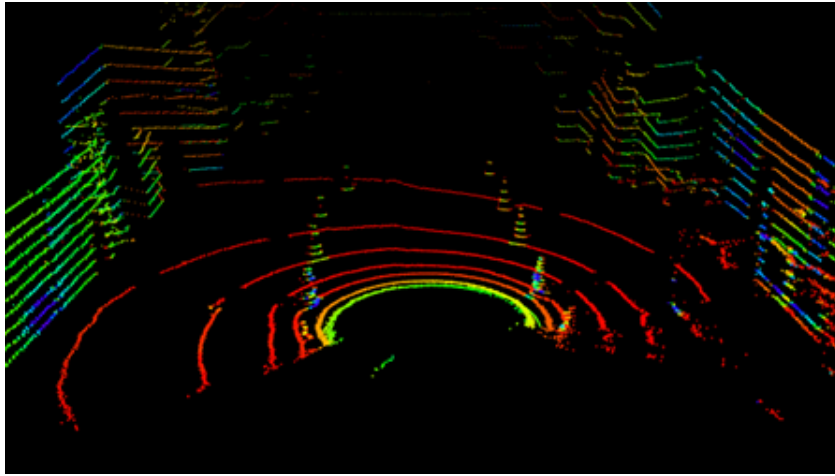


Abbildung 2.1: Beispiel einer farbkodierten LiDAR-Punktwolke (Seitenansicht). Die Farbkodierung visualisiert die Distanz zum Sensor: Nahe Punkte erscheinen rot, mittlere Distanzen grün und weit entfernte Punkte blau. Die konzentrischen Bögen im unteren Bereich entsprechen den Scanringen auf der Bodenebene, während die kleinen vertikalen Cluster in der Mitte Verkehrshütchen und die ausgedehnten Strukturen links und rechts Gebäudefassaden darstellen. Der in der BEV als schwarzer Kreis sichtbare sensorbedingte blinde Fleck nimmt mit steigender Montagehöhe zu. Gemäß den Richtlinien des FASTDA-Handbuchs ist eine Sensormontagehöhe zwischen 1,0 m und 1,5 m vorgeschrieben, was zwangsläufig zu einem entsprechend großen blinden Fleck führt.

Die unregelmäßige Verteilung und abnehmende Punktdichte mit zunehmender Entfernung stellen eine zentrale Herausforderung für 3D-Deep-Learning-Verfahren dar [11]. Besonders kleine Objekte wie Verkehrshütchen (vgl. Abbildung 2.2) werden bereits im Nahbereich nur durch wenige Punkte repräsentiert. Ab einer Distanz von 15 m bestehen solche Objekte häufig aus weniger als fünf Punkten, was eine zuverlässige Merkmalsextraktion erheblich erschwert. Diese Punktarmut motiviert den Einsatz spezialisierter Architekturen, die auch bei geringer Punktdichte robust detektieren können [11].

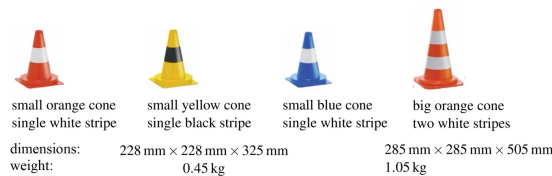


Abbildung 2.2: Im FASTDA-Projekt verwendete Verkehrshütchen (eigene Darstellung).

2.2 DATENSATZ

2.2.1 nuScenes-Datensatz

Als Ausgangspunkt für das Training der in dieser Arbeit untersuchten Modelle werden die vortrainierten Gewichte des öffentlich zugänglichen nuScenes-Datensatzes [12] verwendet. Dieser multimodale Datensatz wurde speziell für die Entwicklung autonomer Fahrsysteme konzipiert und umfasst neben LiDAR-Daten auch synchronisierte Kamera-, Radar- sowie GPS/IMU-Informationen. Mit 1.000 aufgezeichneten Fahrszenen in urbaner Umgebung und einer Gesamtanzahl von etwa 1,4 Millionen annotierten 3D-Bounding-Boxen stellt nuScenes eine der umfangreichsten Ressourcen für die 3D-Objekterkennung dar [12].

Tabelle 2.1: Kennzahlen des nuScenes-Datensatzes für die Klasse *traffic_cone* [12].

Kennzahl	Anzahl
Trainingsframes	12.366
Annotierte Instanzen (Train)	82.362
Validierungs-/Testframes	2.520
Annotierte Instanzen (Test)	15.597

Ein entscheidendes Merkmal für die vorliegende Arbeit ist die im Datensatz enthaltene Klasse *traffic_cone*. Im Gegensatz zu vielen anderen öffentlichen Datensätzen wie KITTI [13] bietet nuScenes eine eigenständige und detaillierte Annotation dieser Objekte [12]. Dies ermöglicht ein gezieltes Transfer Learning: Die auf nuScenes vortrainierten Modelle – darunter PointPillars [14], 3DSSD [15], VoxelNeXt [16] und LION [17] – haben bereits grundlegende geometrische Merkmale von Verkehrshütchen erlernt, die eine hohe Ähnlichkeit mit den im Rahmen des FASTDA-Projekts verwendeten Hütchen aufweisen.

Das Training erfolgt in einem zweistufigen Verfahren: Zunächst werden die vortrainierten Gewichte aus nuScenes als Initialisierung übernommen. Anschließend erfolgt die Feinjustierung (Fine-Tuning) auf den eigenen domänenspezifischen Datensatz, der mit dem *Ouster OS2-128* auf einer Teststrecke des FASTDA-Projekts aufgenommen wurde. Tabelle 2.1 gibt einen Überblick über die relevanten Kennzahlen des nuScenes-Datensatzes für die Klasse *traffic_cone*.

Durch die Verwendung der auf nuScenes vortrainierten Gewichte können die Modelle von einem bereits erlernten Merkmalsraum profitieren, bevor sie auf den deutlich kleineren eigenen Datensatz mit 169 Frames und 1.691 annotierten Instanzen feinjustiert werden. Dieses Transfer-Learning-Verfahren zielt darauf ab, die Generalisierungsfähigkeit der Modelle zu erhöhen und den Bedarf an domänenspezifischen Trainingsdaten zu reduzieren [12].

2.2.2 Eigener Datensatz

Öffentlich verfügbare Datensätze wie nuScenes [12] enthalten zwar Annotationen von Verkehrshütchen, jedoch weisen die dort abgebildeten Objekte eine andere geometrische Form und Größe auf als die im Rahmen des FASTDA-Projekts verwendeten Hütchen. Da direkte Transferleistungen auf die spezifische Anwendungsdomäne aufgrund dieser Abweichungen nur eingeschränkt möglich sind, wurde im Rahmen dieser Arbeit ein eigener Datensatz unter rennstreckentypischen Bedingungen erhoben und annotiert.

Die Aufnahmen erfolgten mit dem LiDAR-Sensor *Ouster OS2-128* auf einem abgesperrten Außengelände. Um die Robustheit der später trainierten Modelle gegenüber unterschiedlichen Einsatzszenarien zu erhöhen, wurden die Daten unter variierenden Bedingungen hinsichtlich Sensorhöhe (0,5 m und 1,0 m), Bodenbeschaffenheit (trocken und nass) sowie Aufnahmewinkel erfasst. Die technischen Spezifikationen des Sensors sind in Tabelle 2.2 zusammengefasst.

Tabelle 2.2: Technische Spezifikationen des verwendeten Ouster OS2-128 LiDAR-Sensors.

Eigenschaft	Wert
Modell	Ouster OS2-128
Vertikale Kanäle	128
Horizontale Auflösung	0,2 ° (bei 10 Hz)
Reichweite	bis zu 200 m
Scanrate	10 Hz (10 Frames/Sekunde)
Punkte pro Frame	ca. 60.000 – 70.000
Ausgangsformat	PCD (Point Cloud Data)
Montagehöhe	0,5 m und 1,0 m

Die Annotation der LiDAR-Daten erfolgte mit Computer Vision Annotation Tool (CVAT). Der fertig annotierte Datensatz wurde anschließend in das KITTI-Format konvertiert, um eine direkte Kompatibilität mit den verwendeten Frameworks MMDetection3D [18] und OpenPCDet [19] zu gewährleisten. Tabelle 2.3 gibt einen Überblick über die zentralen Kennzahlen des erstellten Datensatzes.

Der Datensatz weist eine deutliche Klassenunbalance zwischen *small_cone* mit 1.642 Instanzen und *large_cone* mit lediglich 49 Instanzen auf. Aufgrund der geringen Anzahl von *large_cone*-Instanzen ist eine aussagekräftige Evaluierung der Detektionsleistung für diese Klasse nicht möglich. Daher wird im Rahmen dieser Arbeit der Fokus auf die Klasse *small_cone* gelegt, während *large_cone* bei der quantitativen Bewertung der untersuchten 3D-Deep-Learning-Modelle nicht berücksichtigt wird.

Tabelle 2.3: Kennzahlen des im Rahmen dieser Arbeit erhobenen und annotierten Datensatzes.

Kennzahl	Anzahl
Gesamtanzahl Frames	169
Instanzen <i>small_cone</i>	1.642
Instanzen <i>large_cone</i>	49
Gesamtanzahl Instanzen	1.691
Trainingsframes	135 (80 %)
Testframes	34 (20 %)

2.3 DEEP-LEARNING-ARCHITEKTUREN FÜR DIE 3D-DETEKTION

Deep-Learning-Verfahren für die 3D-Detektion in LiDAR-Punktwolken lassen sich in der Literatur vor allem in punktbasierte, voxelbasierte und projektionsbasierte Architekturen einteilen [20, 21]. Diese Einteilung ergibt sich aus der Frage, wie die unstrukturierte und sparse Punktwolke in eine für neuronale Netze verarbeitbare Repräsentation überführt wird [21, 22]. Für autonome Systeme ist diese Repräsentationswahl besonders wichtig, da sie sowohl die Detektionsgenauigkeit als auch Speicherbedarf, Laufzeit und Echtzeitfähigkeit auf On-Board-Systemen beeinflusst [21, 23]. Abbildung 2.3 gibt einen schematischen Überblick über die Architekturkategorien und deren typischen Verarbeitungspipeline.

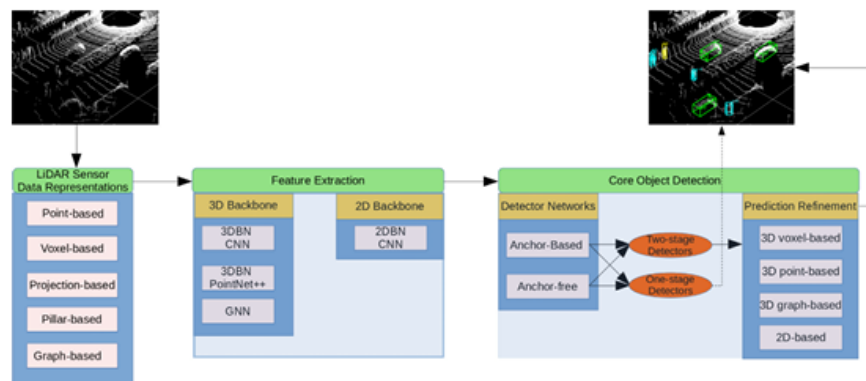


Abbildung 2.3: Übersicht über Deep-Learning-Architekturen für die 3D-Objekterkennung in LiDAR-Daten. Die Punktwolke wird zunächst in verschiedene Repräsentationen überführt, anschließend durch ein Backbone verarbeitet und ein Detektionskopf erzeugt die finalen 3D-Bounding-Boxes [21].

Die nachfolgenden Abschnitte erläutern die Architekturkategorien im Detail und diskutieren ihre jeweiligen Vor- und Nachteile im Hinblick auf die Erkennung kleiner Objekte wie Verkehrshütchen.

2.3.1 Punktbasierte Architekturen

Punktbasierte Architekturen verarbeiten LiDAR-Punktwolken direkt als ungeordnete Punktmengen und verzichten damit auf eine vorherige Umwandlung in Voxel oder zweidimensionale Projektionen. Dadurch bleiben die ursprünglichen geometrischen Informationen der Punktwolke erhalten, sodass lokale Strukturen ohne Quantisierungsverluste beschrieben werden können [20, 24]. Die Merkmalsextraktion erfolgt typischerweise durch die Aggregation lokaler Nachbarschaften, wodurch räumliche Beziehungen zwischen benachbarten Punkten modelliert werden. Dieser Ansatz eignet sich insbesondere für kleine Objekte, da bereits wenige vorhandene Messpunkte unmittelbar in die Merkmalsbildung einfließen und nicht durch eine Diskretisierung abgeschwächt werden [20, 21]. Im Gegensatz dazu ist der erforderliche Rechenaufwand vergleichsweise hoch. Grund dafür ist, dass Nachbarschaftssuchen sowie Sampling- und Gruppierungsoperationen für eine große Anzahl von Punkten durchgeführt werden müssen. Zudem reagieren punktbasierte Verfahren empfindlich auf ungleichmäßige Punktdichten und stark ausgedünnte Punktwolken, wie sie insbesondere bei großen Objektdistanzen auftreten [20, 22]. Insgesamt bieten punktbasierte Architekturen eine hohe geometrische Detailtreue und eignen sich daher grundsätzlich für die Detektion kleiner LiDAR-Objekte. Ihre praktische Einsetzbarkeit auf ressourcenbeschränkten On-Board-Systemen hängt jedoch wesentlich von effizienten Verfahren zur Nachbarschaftsaggregation und einer reduzierten Rechenkomplexität ab.

2.3.2 Voxelbasierte Architekturen

Voxelbasierte Verfahren zerlegen den dreidimensionalen Raum in reguläre Gitterzellen, sogenannte Voxel, und aggregieren die darin enthaltenen Punkte zu einer strukturierten Zwischenrepräsentation [25, 26]. Diese Diskretisierung ermöglicht den Einsatz effizienter Faltungsoperationen und bildet die Grundlage vieler leistungsfähiger LiDAR-Detektoren [25, 26]. Ein bedeutendes Beispiel ist VoxelNet, das die Punktwolke in gleichmäßig verteilte 3D-Voxel aufteilt, innerhalb jedes Voxels Merkmale extrahiert und die resultierende volumetrische Darstellung für die Detektion weiterverarbeitet [25].

Der wesentliche Vorteil voxelbasierter Ansätze besteht in der guten Balance zwischen Genauigkeit und Effizienz [21, 26]. Ein grundlegendes Problem der Voxelisierung sind jedoch Quantisierungseffekte, durch die feine geometrische Details verloren gehen können besonders bei kleinen Objekten [21, 23]. Die Voxelgröße ist daher ein zentraler Entwurfsparemeter: Kleine Voxel bewahren mehr Details, erhöhen jedoch den Rechen- und Speicherbedarf, während große Voxel die Effizienz verbessern, jedoch insbesondere bei kleinen Objekten zu Informationsverlusten führen können [21, 26].

2.3.3 Projektionsbasierte Architekturen

Projektionsbasierte Verfahren transformieren die 3D-Punktwolke in eine reguläre 2D-Darstellung, etwa als Bird's-Eye-View oder Range Image [21, 22]. Durch diese Abbildung können etablierte 2D-CNN-Architekturen genutzt werden, was die Berechnung effizient macht und eine einfache Integration in echtzeitfähige Systeme unterstützt [21]. Der Preis dieser Effizienz ist jedoch ein Informationsverlust, da die dreidimensionale Struktur entlang mindestens einer Raumdimension komprimiert wird [21, 22].

Für die Detektion kleiner Objekte wie Verkehrshütchen kann dies besonders kritisch sein, weil solche Objekte nur wenig LiDAR-Punkte erzeugen und ihre Projektion in ein 2D-Raster dazu führen könnte, dass geometrische entscheidende Strukturen verloren gehen oder vollständig durch Hintergrundrauschen überlagert werden [20, 21]. Erschwerend kommt hinzu, dass die charakteristische Kegelform aus der Vogelperspektive lediglich als kleiner Kreis erscheint, wodurch die zur Objektunterscheidung notwendige diskriminative Information stark reduziert wird. Aufgrund der variierenden Punktdichte weisen Hütchen zudem unterschiedliche Repräsentationen im 2D-Raster auf. Bei sehr geringer Punktdichte von drei oder weniger Punkten wird eine zuverlässige Trennung vom Hintergrundrauschen nahezu unmöglich [21, 23]. Dennoch bleiben projektionsbasierte Ansätze für On-Board-Systeme relevant, wenn geringe Latenz wichtiger ist als maximale geometrische Detailtreue [21].

2.3.4 Hybride Architekturen

Neuere Arbeiten kombinieren mehrere Repräsentationen, um die jeweiligen Stärken punkt-, voxel- und projektionsbasierter Ansätze zusammenzuführen [20, 26]. Hybride Architekturen koppeln beispielsweise voxelbasierte Merkmalsextraktion mit punktwiser Verfeinerung oder integrieren Projektionsmerkmale als zusätzliche Kontextquelle [20, 26]. Ziel ist es, die Effizienz strukturierter Repräsentationen mit der Detailtreue roher Punktdaten zu verbinden [21, 26].

Gerade für kleine Objekte sind solche hybriden Ansätze wissenschaftlich besonders relevant, da sie lokale Details bewahren und zugleich globalen Szenenkontext modellieren können [20, 21]. Für den praktischen Einsatz stellt sich daher die Frage, ob die erhöhte Komplexität solcher Ansätze mit einem konkreten Genauigkeitsgewinn einhergeht [21, 23]. Deshalb wird in vielen aktuellen Arbeiten nicht nur die Detektionsleistung, sondern auch das Verhältnis zwischen Genauigkeit, Inferenzzeit und Speicherbedarf analysiert [21, 23].

2.3.5 Anker-basierte und ankerfreie Detektoren

Im Bereich des Computer Vision haben sich für die 3D-Objekterkennung zwei grundlegende Strategien zur Vorhersage von Bounding-Boxen etabliert: ankerbasierte und ankerfreie Ansätze [27].

Ankerbasierte Ansätze bilden eine Menge vordefinierter Bounding-Box-Vorlagen (Anchors), die als Referenz für die Objekterkennung und Lokalisierung dienen. Diese Anchors repräsentieren typische Objektgrößen und -orientierungen und werden an jeder Position der Feature-Map beziehungsweise des Pseudo-Bildes platziert. Für jede Position existieren mehrere Anchors mit unterschiedlichen Abmessungen und Ausrichtungen, um verschiedene Objektgrößen und -formen abzudecken. Anstatt die Bounding-Box eines Objekts vollständig vorherzusagen, lernt das neuronale Netzwerk lediglich Korrekturwerte (Regressionsoffsets) für die Position ($\Delta x, \Delta y, \Delta z$), die Dimensionen ($\Delta w, \Delta l, \Delta h$) sowie die Orientierung ($\Delta \theta$). Dadurch wird jeder Anchor möglichst präzise an die tatsächliche Objektbox angepasst. Zusätzlich schätzt das Netzwerk für jeden Anchor die Wahrscheinlichkeit, dass dieser ein Objekt enthält, sowie dessen Klasse. Während des Trainings werden die Anchors den Ground-Truth-Bounding-Boxen anhand der Intersection over Union (IoU) zugeordnet. Anchors mit einer ausreichend hohen Überlappung gelten als positive Trainingsbeispiele, während Anchors mit geringer Überlappung als Hintergrund klassifiziert oder vom Training ausgeschlossen werden. Da an jeder Position mehrere Anchors erzeugt werden, entstehen zahlreiche überlappende Vorhersagen. Diese werden mithilfe der Non-Maximum Suppression (NMS) gefiltert, sodass nur die Bounding-Boxen mit der höchsten Vorhersagewahrscheinlichkeit als endgültige Detektionsergebnisse verbleiben [28, 29].

Ankerfreie Ansätze verzichten vollständig auf vordefinierte Templates. Stattdessen sagt das Netzwerk für jeden Punkt bzw. jede Position direkt die Existenz eines Objekts sowie dessen vollständige Bounding-Box-Parameter (Zentrum, Größe, Rotation) vorher. Dadurch entfällt die manuelle Definition und Abstimmung von Ankergrößen, was insbesondere für Objektklassen mit stark heterogenen oder im Vorfeld schwer abschätzbaren Größenverhältnissen vorteilhaft sein kann. Diese Eigenschaften haben dazu geführt, dass sich aktuelle Architekturen zunehmend von ankerbasierten zu ankerfreien Designs entwickeln.

2.4 OPEN-SOURCE-FRAMEWORKS FÜR DIE 3D-OBJEKTERKENNUNG

Open-Source-Frameworks bilden eine wesentliche Grundlage für die Forschung im Bereich der 3D-Objekterkennung und kooperativen Wahrnehmung [30]. Im Rahmen dieser Arbeit kommen zwei etablierte Frameworks zum Einsatz:

1. **MMDetection3D** [18]: ist ein modular aufgebautes Framework für die 3D-Objekterkennung, das sowohl uni- als auch multimodale Verfahren unterstützt. Es integriert etablierte Benchmark-Datensätze wie nuSc-

nes, KITTI und Waymo Open Dataset und stellt eine Vielzahl vortrainierter Modelle für unterschiedliche Architekturen bereit. Aufgrund seiner konsistenten und erweiterbaren Architektur ermöglicht es die effiziente Integration neuer Modelle sowie Trainingspipelines und hat sich als De-facto-Standardframework in der Forschung zur 3D-Objekterkennung etabliert.

2. **OpenPCDet** [19]: Diese PyTorch-basierte Toolbox wurde speziell für die LiDAR-basierte 3D-Objekterkennung konzipiert. Im Mittelpunkt steht ein klar strukturiertes, komponentenbasiertes Design, das eine einfache Konfiguration und Reproduzierbarkeit von Experimenten ermöglicht. OpenPCDet unterstützt eine Reihe leistungsstarker Detektionsarchitekturen und legt besonderen Wert auf eine saubere Trennung zwischen Datenpipeline, Modellarchitektur und Trainingslogik, was die Entwicklung und den Vergleich neuer Ansätze erheblich erleichtert.

Ein wesentlicher Vorteil beider Frameworks liegt in der Verfügbarkeit vortrainierter Modellgewichte. Für die in dieser Arbeit betrachteten Architekturen – PointPillars, 3DSSD, PV-RCNN und DeLiVoTr stehen Checkpoints zur Verfügung, die auf nuScenes [12] bzw. KITTI [31] vortrainiert wurden und als Initialisierung für das anschließende Fine-Tuning auf dem eigenen Datensatz dienen. MMDetection3D wird dabei für PointPillars und PV-RCNN eingesetzt, OpenPCDet für 3DSSD und DeLiVoTr.

2.5 EVALUATIONSMETRIKEN

Zur Bewertung der untersuchten Modelle werden zwei Kategorien von Metriken herangezogen: Detektionsqualität und Laufzeiteffizienz [11].

2.5.1 Detektionsqualität

Im Bereich der 3D-Objekterkennung quantifizieren Evaluationsmetriken die Qualität der erkannten Objekte, indem sie Aspekte wie Genauigkeit und Vollständigkeit von Algorithmen bei der Objekterkennung zu messen [30, 32]. Die Grundlage aller Detektionsmetriken bilden die folgenden vier Größen, die aus dem Vergleich zwischen vorhergesagten und tatsächlichen Objektpositionen ermittelt werden:

- **True Positive (TP):** Ein True Positive tritt auf, wenn das Modell ein real existierendes Objekt korrekt erkennt und eine 3D Bounding Box an der richtigen Position vorhersagt [32].
- **True Negative (TN):** Ein True Negative tritt auf, wenn das Modell korrekt identifiziert, dass kein Objekt vorhanden ist. Es wird keine 3D Bounding Box für einen leeren Bereich der LiDAR-Punktwolke vorhergesagt [32].

- **False Positive (FP):** Das Modell erzeugt ein False Positive, wenn es ein Objekt erkennt, das in der Realität nicht vorhanden ist und platziert eine Bounding Box ohne reales Objekt. Auch bekannt als Fehler Typ I [32].
- **False Negative (FN):** Ein False Negative tritt auf, wenn das Modell ein real existierendes Objekt in der Punktwolke nicht erkennt und keine Bounding Box vorhersagt, obwohl das Objekt tatsächlich vorhanden ist. Dies betrifft besonders kleine Objekte mit geringer Punktdichte. Auch bezeichnet als Fehler Typ II [32].

Es ist zu beachten, dass im Kontext der 3D-Objekterkennung kein echtes True Negative (TN) existiert, da es in einer Punktwolke unendlich viele Raumbereiche gibt, die kein Objekt enthalten und somit nicht detektiert werden sollten.

Die obengenannten Definitionen erfordern die Feststellung, was eine korrekte Erkennung und eine falsche Erkennung sind. Eine gängige Methode ist die Verwendung der **3D-Intersection over Union (3D-IoU)** [30, 32, 33]. Sie misst das überlappende **Volumen** zwischen der vorhergesagten 3D-Bounding-Box $\mathbf{B}_{pred}$ und der Ground-Truth-3D-Bounding-Box $\mathbf{B}_{gt}$, geteilt durch das Volumen der Vereinigung beider Boxen:

$$IoU = \frac{\text{Schnittvolumen}}{\text{Vereinigungsvolumen}} = \frac{|B_{pred} \cap B_{gt}|}{|B_{pred} \cup B_{gt}|} \quad (2.6)$$

Durch den Vergleich der IoU mit einem gegebenen Schwellenwert t können Detektionen als korrekt oder falsch klassifiziert werden:

- Wenn $IoU \geq t$, gilt die Detektion als **korrekt** (True Positive).
- Wenn $IoU < t$, gilt die Detektion als **falsch** (False Positive).
- Wenn für eine Ground-Truth-3D-Bounding-Box $\mathbf{B}_{gt}$ keine vorhergesagte 3D-Bounding-Box $\mathbf{B}_{pred}$ mit $IoU \geq t$ existiert, wird die Detektion als **nicht erkannt** (False Negative) gewertet.

In der 3D-Objekterkennung wird der Schwellenwert t typischerweise in Abhängigkeit von der Objektgröße gewählt. Für große Objekte wie **Fahrzeuge** wird häufig $t = 0,7$ verwendet, während für kleine Objekte wie **Fußgänger** oder **Radfahrer** ein niedrigerer Schwellenwert von $t = 0,5$ üblich ist, da diese aufgrund ihrer geringen Punktdichte in der Punktwolke schwieriger zu lokalisieren sind [30–32].

Da die wahren Negativen (TN) in Objekterkennungs-Frameworks nicht verwendet werden, verzichtet man auf Metriken, die auf TN basieren, wie TPR, FPR und ROC-Kurven [32, 34]. Stattdessen basiert die Bewertung von Objekterkennungsmethoden hauptsächlich auf den Konzepten **Precision** und **Recall**, die wie folgt definiert sind [32]:

$$\text{Precision} = \frac{TP}{TP + FP} \quad (2.7)$$

$$\text{Recall} = \frac{TP}{TP + FN} \quad (2.8)$$

Precision beschreibt die Fähigkeit eines Modells, nur relevante Objekte zu identifizieren also den Anteil der korrekt vorhergesagten Objekte an allen vom Modell vorhergesagten Objekten. **Recall** hingegen beschreibt die Fähigkeit eines Modells, alle tatsächlich vorhandenen Objekte zu finden also den Anteil der korrekt erkannten Objekte an allen Ground-Truth-Bounding-Boxes.

Präzision und Recall bewegen sich oft in entgegengesetzte Richtungen. Wenn sich das eine verbessert, kann das andere sinken. Dieser Trade-off ist eine häufige Herausforderung bei Machine-Learning-Aufgaben.

Ein Modell mit hoher Präzision sagt etwas nur dann als positiv voraus, wenn es sich sicher ist. Das reduziert Fehllalarme, kann aber echte positive Fälle übersehen, was den Recall senkt. Ein Modell, das versucht, jeden positiven Fall zu erfassen, erhöht den Recall, riskiert aber mehr Fehllalarme, was die Präzision senkt [32].

Dieser Trade-off wird durch die Wahl des **Konfidenzschwellenwerts** beeinflusst: Ein niedrigerer Schwellenwert führt zu mehr positiven Vorhersagen, was den Recall erhöht, aber die Precision senkt. Ein höherer Schwellenwert hat den gegenteiligen Effekt. Daher wird der **F1-Score** als harmonisches Mittel beider Metriken verwendet, um beide Aspekte gleichzeitig zu berücksichtigen:

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \quad (2.9)$$

Durch die Variation des Konfidenzschwellenwerts erhält man verschiedene Kombinationen von Precision und Recall, die zu einer **Precision-Recall-Kurve** (PR-Kurve) zusammengefasst werden können [30, 32].

Die zentrale Metrik für die Bewertung von 3D-Objektdetektoren ist die **Average Precision** (AP), die die Fläche unter der Precision-Recall-Kurve berechnet [12]:

$$AP = \int_0^1 P(r) dr \quad (2.10)$$

wobei $P(r)$ die Precision beim Recall r darstellt. In der Praxis wird die Fläche üblicherweise durch numerische Integration der PR-Kurve geschätzt [30].

Ein vorhergesagtes Objekt gilt dabei als korrekt erkannt, wenn die Überlappung zwischen vorhergesagter und tatsächlicher 3D-Bounding-Box – gemessen durch die IoU – einen definierten Schwellenwert überschreitet. In dieser Arbeit wird ein Schwellenwert von $\text{IoU} \geq 0,5$ verwendet, was als $\text{AP}@0.5$ bezeichnet wird.

RECALL ALS SICHERHEITSRELEVANTE BEWERTUNGSMETRIK

Im Kontext sicherheitskritischer autonomer Fahrsysteme besitzen die beiden Fehlerarten False Positive und False Negative eine grundlegend unterschied-

liche Relevanz. Ein False Positive, also eine Fehldetektion, bei der ein nicht existierendes Verkehrshütchen erkannt wird, kann durch nachgelagerte Verfahren wie Multi-Frame-Tracking oder Plausibilitätsprüfungen gegenüber der bekannten Streckengeometrie häufig erkannt und korrigiert werden.

Ein False Negative hingegen, also ein nicht erkanntes Verkehrshütchen, führt dazu, dass ein tatsächlich vorhandenes Hindernis nicht in die weitere Verarbeitungskette eingeht. Infolgedessen kann das Objekt von nachgelagerten Modulen wie der Trajektorienplanung oder Kollisionsvermeidung nicht berücksichtigt werden. False Negatives besitzen daher eine deutlich höhere sicherheitsrelevante Bedeutung, da sie das Risiko von Fehlentscheidungen und potenziellen Kollisionen erhöhen.

Aus diesem Grund wird in der vorliegenden Arbeit dem Recall eine besondere Bedeutung beigemessen. Da Recall als Anteil der korrekt erkannten Objekte an allen tatsächlich vorhandenen Objekten definiert ist (vgl. Gleichung 2.8), beschreibt diese Metrik unmittelbar, welcher Anteil der real vorhandenen Verkehrshütchen vom System erkannt wird. Sie eignet sich daher besonders zur Beurteilung der sicherheitsrelevanten Detektionsleistung.

Da eine isolierte Optimierung des Recalls jedoch zu einer erhöhten Anzahl von Fehldetektionen führen kann, wird für die Gesamtbewertung und den Modellvergleich zusätzlich der F1-Score herangezogen, der Precision und Recall gleichermaßen berücksichtigt

2.5.2 Laufzeiteffizienz

Neben der Detektionsqualität ist die Laufzeiteffizienz eine entscheidende Anforderung für den Einsatz auf ressourcenbeschränkten On-Board-Systemen. Die folgenden Metriken werden zur Bewertung der Effizienz herangezogen:

- **Latenz (ms):** Die Latenz beschreibt die durchschnittliche Verarbeitungszeit pro Frame. Eine geringe Latenz ist für den sicheren Fahrzeugbetrieb unerlässlich, da jede Verzögerung in der Verarbeitungskette die Reaktionsfähigkeit des Gesamtsystems direkt beeinträchtigt [35].
- **Frames per Second (FPS):** Die Anzahl der verarbeiteten Frames pro Sekunde. Als Mindestanforderung für Echtzeitfähigkeit wird in dieser Arbeit eine Verarbeitungsrate von mindestens 10 FPS definiert. Diese Grenze ergibt sich unmittelbar aus der Abtastrate des verwendeten LiDAR-Sensors (Ouster OS2-128), der 10 Punktwolken pro Sekunde liefert [36]. Ein 3D-Objektdetektor muss daher in der Lage sein, jeden Frame innerhalb von 100 ms zu verarbeiten, um eine Echtzeitverarbeitung ohne Frame-Verluste zu gewährleisten [37].
- **GPU-Auslastung (GB):** Der Speicherbedarf des Modells auf der GPU, der den Einsatz auf ressourcenbeschränkter Hardware limitiert.
- **Modellgröße (MB):** Der Speicherbedarf des trainierten Modells, relevant für den Einsatz auf eingebetteten Systemen.

METHODIK UND ERGEBNISSE

3.1 AUSWAHL DER MODELLE UND METHODISCHE VORGEHENSWEISE

Im Rahmen dieser Arbeit werden verschiedene dreidimensional (3D)-Deep-Learning-Modelle aus unterschiedlichen Architekturkategorien untersucht und miteinander verglichen. Die Auswahl der Modelle folgt zwei zentralen Kriterien: Erstens einer hohen Erkennungsgenauigkeit für kleine, punktar-me Objekte und zweitens einer niedrigen Inferenzzeit für den Echtzeiteinsatz auf ressourcenbeschränkten On-Board-Systemen. Tabelle 3.1 gibt einen Überblick über die evaluierten Ansätze.

Tabelle 3.1: Überblick der evaluierten Modelle.

Modell	Kategorie	Referenz
PointPillars	Voxelbasiert, ankerbasiert	[14]
3DSSD	Punktbasiert, ankerfrei	[15]
DeLiVoTr	Aufmerksamkeitsbasiert	[38]
PV-RCNN	Hybrid (Punkt + Voxel)	[39]
VoxelNeXt	Vollständig spärlich (fully sparse)	[16]
LION	Sequenziell (Mamba)	[17]

3.1.1 Modelladaptierung und Trainingssetup

Alle in dieser Arbeit evaluierten Modelle wurden mittels Fine-Tuning an den in Abschnitt 2.2.2 beschriebenen Datensatz angepasst. Aufgrund der geringen Anzahl von `large_cone`-Instanzen ist eine aussagekräftige quantitative Evaluierung für diese Klasse nicht möglich. Darüber hinaus würde die Einbeziehung dieser Klasse aggregierte Metriken wie die Mean Average Precision (mAP) verzerren, da bereits einzelne Fehlklassifikationen aufgrund der geringen Grundgesamtheit einen unverhältnismäßig großen Einfluss auf den Gesamtwert hätten. Daher wird die Klasse `large_cone` ausgeschlossen, sodass der Fokus ausschließlich auf der Klasse `small_cone` liegt.

Für jedes untersuchte Modell wurden im Vorfeld systematische Hyperparameteranpassungen durchgeführt. Ziel dieser Optimierung war es, für jedes Modell die beste Balance zwischen Detektionsgenauigkeit und Echtzeiteffizienz zu finden. Im Folgenden wird ausschließlich die jeweilige Modellversion präsentiert und diskutiert, die diesen Zielkonflikt optimal auflöst. Das Training aller Modelle erfolgte auf einem Linux-System mit einer NVIDIA RTX 6000 Ada GPU und 49GB VRAM. Als Evaluationsmetriken kommen die in Abschnitt 2.5 beschriebenen Maße zum Einsatz.

3.1.2 *Vorstellungsstruktur der Modelle*

Jedes Modell wird nach einer einheitlichen Gliederung präsentiert, um eine vergleichbare Darstellung zu gewährleisten:

1. **Architekturbeschreibung:** Vorstellung des grundlegenden Funktionsprinzips und der charakteristischen Eigenschaften des Ansatzes.
2. **Eignung für kleine Objekte:** Begründung, warum das Modell besonders für die Erkennung kleiner, punktarmer Objekte wie Verkehrshütchen geeignet ist.
3. **Konfiguration und Hyperparameter:** Darstellung der gewählten Modellkonfiguration sowie der wichtigsten Hyperparameter.
4. **Ergebnisse:** Präsentation der Detektionsleistung sowie der Laufzeitmetriken.
5. **Diskussion und Limitationen:** Kritische Einordnung der erzielten Leistung sowie Benennung von Schwächen und Einschränkungen des jeweiligen Ansatzes.

3.2 POINTPILLARS

3.2.1 *Architektur*

PointPillars [14] ist ein voxelbasierter, ankerbasierter 3D-Objektdetektor, der Punktwolken effizient in sogenannte Säulen (*Pillars*) umwandelt. Im Gegensatz zu klassischen voxelbasierten Ansätzen, die den dreidimensionalen Raum in ein reguläres 3D-Gitter unterteilen, aggregiert PointPillars alle Punkte entlang der z-Achse in vertikale Säulen. Dadurch entsteht eine zweidimensionale Pseudobild-Repräsentation, die anschließend mit einem effizienten 2D-CNN verarbeitet werden kann [14].

Die Architektur besteht aus drei Hauptkomponenten:

1. **Pillar Feature Net:** Zunächst wird die LiDAR-Punktwolke in der x-y-Ebene diskretisiert, wodurch eine Menge vertikaler Säulen (*Pillars*) entsteht. Anschließend werden die Punkte innerhalb jeder Säule um zusätzliche geometrische Merkmale ergänzt, die ihre relative Lage zum Mittelpunkt der Säule sowie zum Mittelwert aller Punkte innerhalb derselben Säule beschreiben. Diese erweiterten Punktmerkmale werden anschließend durch ein vereinfachtes PointNet [24] (Pillar Feature Network) verarbeitet und mittels Max-Pooling zu einem Säulenmerkmalsvektor aggregiert. Abschließend werden die resultierenden Merkmalsvektoren entsprechend ihrer ursprünglichen räumlichen Position auf einem zweidimensionalen Raster angeordnet, wodurch ein Pseudo-Bild entsteht, das als Eingabe für nachfolgende 2D-Faltungsnetzwerke dient.

2. **Backbone (2D-CNN):** Die erzeugte Pseudobild-Repräsentation wird durch ein zweiteiliges 2D-CNN-Backbone verarbeitet: Ein Top-Down-Netzwerk extrahiert Merkmale auf verschiedenen Skalen, indem es die räumliche Auflösung schrittweise verringert. Anschließend werden diese Merkmale durch ein zweites Netzwerk wieder auf eine gemeinsame Auflösung hochskaliert (Upsampling) und zu einer einzigen Merkmalsrepräsentation zusammengeführt.
3. **Detection Head (SSD):** Ein Single-Shot-Detektor [40] sagt auf Basis der extrahierten Merkmale für jeden Anker (siehe Unterabschnitt 2.3.5) die Klassenwahrscheinlichkeit sowie Regressionsoffsets (Δx , Δy , Δz für die Position, Δw , Δl , Δh für die Größe und $\Delta \theta$ für die Rotation) vorher, die als Korrekturwerte zur Anpassung des Ankers an die tatsächliche Objektbox dienen. Abschließend wird mittels Non-Maximum Suppression (NMS) die finale Vorhersagemenge bestimmt, indem redundante, überlappende Boxen entfernt werden.

3.2.2 Warum PointPillars für kleine Objekte?

PointPillars eignet sich aus mehreren Gründen für die Erkennung kleiner Objekte wie Verkehrshütchen:

- **Effizienz durch 2D-Pseudobild-Repräsentation:** Durch die Transformation der Punktwolke in eine zweidimensionale Pseudobild-Repräsentation können etablierte 2D-CNN-Architekturen genutzt werden. Dies ermöglicht eine Verarbeitungsgeschwindigkeit von bis zu 62 FPS [14].
- **Robuste Merkmalsextraktion bei geringer Punktdichte:** Das Pillar Feature Net verarbeitet jeden einzelnen Punkt innerhalb einer Säule separat, was eine zuverlässige Merkmalsextraktion auch bei wenigen Punkten pro Objekt ermöglicht.
- **Keine Diskretisierung in der Höhe:** Da eine Säule in z-Richtung unbegrenzt ausgedehnt ist, entfällt die Notwendigkeit, einen Hyperparameter für die Höhen-Auflösung festzulegen. Dies ist insbesondere für kleine Objekte wie Verkehrshütchen relevant, da bei voxelbasierten Ansätzen eine ungünstige Wahl der Höhen-Bins zu einer unzureichenden Repräsentation solcher Objekte führen kann.

Diese hohe Effizienz ist für den Echtzeiteinsatz auf ressourcenbeschränkten On-Board-Systemen autonomer Rennfahrzeuge von zentraler Bedeutung.

3.2.3 Konfiguration und Hyperparameter

Da Verkehrshütchen kleine und punktarmer Objekte darstellen, war eine gezielte Anpassung der Modellparameter erforderlich, um eine optimale Balance zwischen Detektionsleistung und Echtzeiteffizienz zu erreichen. Die Größe der räumlichen Säulen (Voxelgröße in der x-y-Ebene) bestimmt einen

zentralen Kompromiss im Modelldesign. Kleinere Säulen ermöglichen eine feinere räumliche Auflösung und verbessern die Erfassung der Objektgeometrie kleiner Objekte, da mehr Säulen pro Objekt zur Verfügung stehen. Gleichzeitig steigt jedoch die Anzahl nicht-leerer Säulen, wodurch das Pseudo-Bild größer wird und die Inferenzgeschwindigkeit abnimmt. Größere Säulen beschleunigen die Verarbeitung, gehen jedoch insbesondere bei kleinen Objekten mit Genauigkeitsverlusten einher.

Ein weiterer zentraler Hyperparameter ist die Ankergröße (siehe Unterabschnitt 2.3.5). Da die Standard-Ankergrößen von PointPillars auf Fahrzeuge, Fußgänger und Radfahrer ausgelegt sind und damit deutlich größer als Verkehrshütchen ausfallen, würde die IoU-basierte Zuordnung zu Ground-Truth-Boxen erheblich erschwert und zu einer hohen Anzahl an False Negatives führen. Daher wurde die Ankergröße `small_cone` an die realen Abmessungen der Verkehrshütchen angepasst, wodurch die Anzahl an False Negatives deutlich reduziert wird.

Im Rahmen einer systematischen Hyperparametersuche wurden mehrere Konfigurationen getestet und hinsichtlich ihrer Performance verglichen. Die beste Gesamtleistung wurde mit der datensatzspezifisch angepassten Variante erzielt (siehe Tabelle 3.2).

Tabelle 3.2: Optimale Konfiguration von PointPillars.

Parameter	Wert
Point Cloud Range	[0, $\pm 39,68$, 69,12] m
Voxelgröße	[0,16, 0,16, 4,0] m
Ankergröße (small_cone)	[0,228, 0,228, 0,325] m
Score-Threshold	0,15
Epochen	80
Lernrate	0,001
Batch-Größe	4

3.2.4 Ergebnisse

Tabelle 3.3 fasst die zentralen Detektions- und Laufzeitmetriken des PointPillars -Modells für die Klasse `small_cone` mit $\text{IoU} = 0,5$ und $\text{Score-Threshold} = 0,15$ zusammen. Die Tabelle umfasst sowohl klassische Detektionsmetriken (TP, FP, FN, Precision, Recall, F1-Score, AP) als auch Laufzeit- und Ressourcenkennzahlen (Latenz, FPS, Modellgröße), die gemeinsam die Eignung des Modells für den Echtzeiteinsatz auf dem Rennfahrzeug bewerten.

Tabelle 3.3: Detektions- und Laufzeitmetriken des PointPillars-Modells für die Klasse *small_cone*.

Metrik	Wert
TP	233/396
FP	148
FN	163/396
Precision (%)	61,2
Recall (%)	58,8
F1-Score (%)	60,0
AP@0.5 (%)	50,7
Avg. Latenz	22,66 ms
Avg. FPS	48,3
Modellgröße	55,9 MB

3.2.5 Diskussion und Limitationen

Die in Tabelle 3.3 dargestellten Ergebnisse zeigen, dass PointPillars grundsätzlich in der Lage ist, kleine Objekte wie Verkehrshütchen zu erkennen. Mit einem AP@0,5 von 50,7 % und einem Recall von 58,8 % werden 233 von 396 Instanzen im Testdatensatz korrekt erkannt (TP), während 163 Hütchen übersehen werden (FN), was einem Anteil von rund 41 % entspricht. Diese hohe Anzahl an False Negatives lässt sich auf die im Folgenden diskutierten Limitationen des voxelbasierten Ansatzes zurückführen, insbesondere auf den Informationsverlust durch die Aggregation der Punkte entlang der z-Achse bei der Transformation in das 2D-Pseudobild.

Hinsichtlich der Laufzeit zeigt PointPillars eine überzeugende Leistung: Mit einer mittleren Latenz von 22,66 ms und 48,3 FPS liegt das Modell deutlich unter der maximal verfügbaren Verarbeitungszeit von 100 ms bei einer Sensorrate von 10 Hz. Diese Echtzeitfähigkeit resultiert direkt aus der Transformation der Punktwolke in eine 2D-Pseudobild-Repräsentation, welche die Verwendung effizienter 2D-CNN-Architekturen anstelle aufwändiger 3D-Faltungsoperationen ermöglicht. Der VRAM-Bedarf von 2,1 GB sowie die kompakte Modellgröße von 55,9 MB sind auch für ressourcenbeschränkte Embedded-Systeme problemlos handhabbar.

Trotz dieser Stärken weist der voxelbasierte Ansatz grundlegende Limitationen auf:

- Durch die Aggregation aller Punkte einer Säule entlang der z-Achse gehen sowohl die charakteristische Höhenausdehnung als auch die feine Form der Objekte verloren, was die hohe Anzahl an False Negatives erklärt.

- Ab einer Distanz von 15 m ist eine zuverlässige Erkennung aufgrund der mit zunehmender Entfernung vom LiDAR-Sensor abnehmenden Punktdichte kaum noch möglich.
- Das Zusammenspiel von Anker und NMS erfordert klar definierte Objektgrößen und stabile Feature-Repräsentationen Anforderungen, die kleine, punktarme Objekte wie Verkehrshütchen in der Praxis häufig nicht erfüllen [27].

Diese Einschränkungen motivieren die Untersuchung alternativer Architekturen, die geometrische Form sowie Höheninformation besser erhalten. Im Gegensatz zu PointPillars verzichtet das im Folgenden vorgestellte Modell 3D Single-Stage Detector (3DSSD) vollständig auf eine Voxelisierung und arbeitet direkt auf der Rohpunktcloud, wodurch feingranulare geometrische Details erhalten bleiben. Zudem verwendet 3DSSD einen ankerfreien Detektionskopf, der die aufwändige manuelle Ankerkonfiguration überflüssig macht und speziell für die effiziente Erkennung kleiner Objekte optimiert ist.

3.3 3DSSD

3.3.1 Architektur

3DSSD [15] ist ein punktbasierter, ankerfreier 3D-Objektdetektor, der im Gegensatz zu zweistufigen punktbasierten Methoden verzichtet er auf rechenintensive Feature-Propagation-Layer und ein zweistufiges Refinement, was eine deutlich schnellere Inferenz ermöglicht. Die Architektur gliedert sich in drei Kernkomponenten:

1. **Backbone mit Fusion Sampling:** Das Backbone extrahiert über mehrere Set-Abstraction-Schichten hierarchische Merkmale aus der Rohpunkt- wolke. Die zentrale Erweiterung das Fusion-Sampling-Verfahren (FS), das zwei Sampling-Strategien kombiniert:
 - **Distance based Furthest-Point-Sampling (D-FPS)** wählt Punkte iterativ rein anhand ihres räumlichen Abstands aus, ohne semantische Information zu berücksichtigen. Das kann dazu führen, dass Hintergrundpunkte (z.,B. Bodenpunkte) aufgrund ihrer starken Überrepräsentation in der Punktwolke bei der Auswahl dominieren.
 - **Feature based Furthest-Point-Sampling** erweitert das klassische Furthest -Point-Sampling um ein gewichtetes Abstandsmaß aus räumlicher Distanz und Merkmalsdistanz, definiert als

$$C(A, B) = \lambda L_d(A, B) + L_f(A, B), \quad (3.1)$$

wobei $L_d(A, B)$ den euklidischen Abstand der XYZ-Koordinaten, $L_f(A, B)$ den Abstand der Merkmalsvektoren und λ einen Gewichtungsfaktor bezeichnet. Dadurch werden viele unwichtige Hintergrundpunkte verworfen, während Objektpunkte auch von entfernten oder kleinen Instanzen mit höherer Wahrscheinlichkeit erhalten bleiben.

Dabei kombiniert das Fusion Sampling unterschiedliche Sampling- Strategien, um sowohl geometrische als auch merkmalsbasierte Informationen bei der Auswahl repräsentativer Punkte zu berücksichtigen [15].

2. **Candidate Generation (CG) Schicht:** Die CG-Schicht ist eine spezielle Set-Abstraction-Schicht zur Erzeugung von Objektkandidaten. Zunächst werden nur die **F-FPS**-Punkte als initiale Zentren genutzt und durch eine gelernte Verschiebungsoperation (Shifting) näher an die tatsächlichen Objektzentren verschoben. Die so entstandenen **Kandidatenpunkte** bilden die Zentren, für die anschließend umliegende Punkte aus der gesamten repräsentativen Punktmenge (F-FPS und D-FPS) aggregiert werden. Dadurch entstehen zentrumsnahe, kontextreiche Merkmale, die im nachfolgenden Prädiktionskopf für Klassifikation und Regression verwendet werden [15].

3. **Ankerfreier Prädiktionskopf:** Die in der CG-Schicht erzeugten Kandidatenpunkte und ihre Merkmale bilden die Eingabe für den Detektionskopf, der ankerfrei konzipiert ist. Für jeden Kandidatenpunkt werden direkt die Parameter der finalen 3D-Bounding-Box vorhergesagt:

- die Distanz zum Objektzentrum,
- die Objektgröße (Länge, Breite, Höhe),
- die Orientierung des Objekts.

Eine *3D-Zentrumszuweisungsstrategie* weist jedem Kandidatenpunkt während des Trainings ein Klassifikationslabel zu, das von seiner Nähe zu den Oberflächen der zugehörigen Objektinstanz abhängt, Punkte näher am Objektzentrum erhalten höhere Werte. Aus diesen Vorhersagen ergeben sich die finalen Detektionsergebnisse direkt, ohne ein nachgeschaltetes Refinement-Modul. Die Gesamtverlustfunktion setzt sich aus drei Anteilen zusammen: Klassifikationsverlust, Regressionsverlust (für Distanz, Größe und Orientierung) sowie Verschiebungsverlust (Shifting Loss) für die Zentrumsverschiebung [15].

3.3.2 Warum 3DSSD für kleine Objekte?

Der punktbasierte Ansatz von 3DSSD bietet für die Erkennung kleiner, punktarmer Objekte wie Verkehrshütchen mehrere Vorteile:

- **Direkte Verarbeitung der Punktwolke:** 3DSSD arbeitet direkt auf der gesamten LiDAR-Punktwolke ohne Voxelisierung. Feingranulare geometrische Details gehen dadurch nicht durch Diskretisierungsartefakte verloren, was insbesondere für kleine Objekte mit wenigen Punkten entscheidend ist.
- **Adaptive Merkmalsextraktion:** Das Backbone von 3DSSD extrahiert über mehrere Set-Abstraction-Schichten hierarchisch Merkmale, indem für jeden Punkt umliegende Nachbarpunkte gruppiert und über MLP-Schichten verarbeitet werden [15]. Dadurch können auch bei den wenigen Punkten kleiner Objekte wie Verkehrshütchen lokale geometrische Strukturen erfasst werden.
- **Effizientes Sampling:** Die Kombination aus geometrischem und Feature-basiertem Sampling reduziert die Punktzahl, ohne relevante Informationen für kleine Objekte zu verlieren [15].

3.3.3 Konfiguration und Hyperparameter

Im Rahmen der Evaluation werden zwei Konfigurationen des punktbasierten 3DSSD-Modells untersucht, die sich in der Anzahl der nach dem Downsampling verbleibenden 3D-Punkte unterscheiden. Die Eingangspunktwolke umfasst ursprünglich circa 60 000–70 000 Punkte, von denen das *24k-Modell*

24 000 Punkte und das *48k-Modell* 48 000 Punkte nach Downsampling beibehalten. Im originalen 3DSSD-Paper [15] wurden lediglich 12 000 Punkte verwendet, was für die dort betrachteten Objektklassen insbesondere Fahrzeuge ausreichend war, da diese eine deutlich größere räumliche Ausdehnung besitzen als Verkehrshütchen. Für die vorliegende Aufgabe erwies sich diese Konfiguration als ungeeignet: Mit 12 000 Punkten erreichte das Modell lediglich einen AP@0.5 von 32,9 %, da die geringe Punktdichte nicht ausreicht, um die kleinen und kompakten Hütchen zuverlässig zu repräsentieren.

Tabelle 3.4 fasst die zentralen Trainingsparameter der beiden Konfigurationen zusammen.

Tabelle 3.4: Trainingsparameter der untersuchten 3DSSD-Konfigurationen.

Parameter	24k-Modell	48k-Modell
Gesampelte Punkte	24.000	48.000
Epochen	80	80
Lernrate	0,002	0,002
Batch-Größe	4	4
Optimierer	AdamW	AdamW
Score-Threshold	0,15	0,15
IoU-Threshold (NMS)	0,4	0,4

Beide Modelle wurden ausgehend von einem auf dem KITTI-Datensatz [31] vortrainierten Checkpoint initialisiert und über 80 Epochen trainiert. Die Lernrate wurde mittels Step-Decay bei den Epochen 60 und 75 reduziert. Zur Erhöhung der Robustheit kamen folgende Datenaugmentationstechniken zum Einsatz: zufälliges horizontales Spiegeln mit einer Wahrscheinlichkeit von 50 %, globale Rotation im Bereich $[-45^\circ, +45^\circ]$ sowie Skalierungsjitterung im Bereich $[0,95; 1,05]$.

3.3.4 Ergebnisse

Tabelle 3.5 stellt die zentralen Detektions- und Laufzeitmetriken der beiden Konfigurationen für die Klasse *small_cone* gegenüber.

Das 24k-Modell erreicht mit einem Recall von 73,98 % eine deutlich höhere Erkennungsrate als das 48k-Modell mit 55,8 %. Es detektiert 72 zusätzliche Verkehrshütchen korrekt (293 gegenüber 221 TP). Diese höhere Sensitivität geht jedoch mit einer erhöhten Anzahl an FP einher (133 gegenüber 86), was in einer geringeren Precision resultiert (68,77 % gegenüber 71,98 %).

Hinsichtlich der Laufzeit überzeugt das 24k-Modell: Mit einer mittleren Latenz von 45,40 ms und 22,03 FPS ist es deutlich schneller als das 48k-Modell (71,30 ms, 14,03 FPS). Der geringere GPU-Speicherbedarf von 270 MB gegenüber 344 MB macht das 24k-Modell zudem besser für ressourcenbeschränkte On-Board-Systeme geeignet.

Tabelle 3.5: Detektions- und Laufzeitmetriken der untersuchten 3DSSD-Konfigurationen für die Klasse *small_cone* (Score-Threshold = 0,15, IoU = 0,5).

Metrik	24k-Modell	48k-Modell
TP	293/396	221/396
FP	133	86
FN	103/396	175/396
Precision (%)	68,77	71,98
Recall (%)	73,98	55,8
F1-Score (%)	71,9	62,86
AP@0.5 (%)	50,74	40,20
Avg. Latenz (ms)	45,40	71,30
Avg. FPS	22,03	14,03
GPU-Auslastung (MB)	270	344

3.3.5 Diskussion und Limitationen

Die Ergebnisse zeigen, dass eine stärkere Reduktion der Punktzahl beim untersuchten 3DSSD-Modell nicht zwangsläufig zu einer Verschlechterung der Detektionsleistung führt. Das 24k-Modell erzielt mit einem Recall von 73,98 % gegenüber 55,8 % beim 48k-Modell eine deutlich höhere Erkennungsrate. Auch der F1-Score fällt mit 71,9 % höher aus als beim 48k-Modell (62,86 %). Darüber hinaus erreicht das 24k-Modell eine AP@0.5 von 50,74 %, während das 48k-Modell lediglich 40,20 % erzielt. Dies deutet darauf hin, dass die höhere Punktzahl des 48k-Modells nicht automatisch zu einer besseren Merkmalsrepräsentation führt. Stattdessen könnte das stärkere Downsampling des 24k-Modells redundante Hintergrundinformationen reduzieren und dadurch die Generalisierungsfähigkeit verbessern.

Besonders relevant ist der Unterschied im Recall. Das 24k-Modell erkennt 293 von 396 Verkehrshütchen korrekt, während das 48k-Modell lediglich 221 von 396 Objekten detektiert. Für autonome Fahrsysteme ist diese Eigenschaft von besonderer Bedeutung, da nicht erkannte Objekte potenziell sicherheitskritische Situationen verursachen können. Zwar weist das 24k-Modell mit 133 False Positives mehr Fehldetektionen auf als das 48k-Modell mit 86 False Positives, dennoch überwiegt der Vorteil der deutlich höheren Erkennungsrate. Die Ergebnisse verdeutlichen damit den Zielkonflikt zwischen Recall und Precision. Während das 48k-Modell mit einer Precision von 72,08 % eine etwas höhere Vorhersagegenauigkeit erreicht, erzielt das 24k-Modell mit 68,77 % die bessere Balance zwischen Erkennungsrate und Präzision, was sich im höheren F1-Score widerspiegelt.

Neben der höheren Detektionsleistung weist das 24k-Modell auch Vorteile hinsichtlich der Recheneffizienz auf. Die mittlere Latenz reduziert sich von 71,30 ms beim 48k-Modell auf 45,40 ms, was einer Verringerung um rund 36 % entspricht. Gleichzeitig steigt die Verarbeitungsrate von 14,03 FPS auf

22,03 FPS. Auch der GPU-Speicherbedarf fällt mit 270 MB gegenüber 344 MB geringer aus. Damit bietet das 24k-Modell sowohl eine bessere Detektionsleistung als auch eine höhere Effizienz und erscheint daher für ressourcenbeschränkte Echtzeitanwendungen besonders geeignet.

Trotz dieser Stärken weist der punktbasierte Ansatz von 3DSSD grundlegende Limitationen auf:

1. **Verlust relevanter Objektpunkte durch Downsampling**

3DSSD reduziert die Anzahl der zu verarbeitenden Punkte durch hierarchische Sampling-Verfahren. Da kleine Verkehrshütchen insbesondere in größeren Entfernungen häufig nur durch wenige LiDAR-Punkte repräsentiert werden, können während dieses Prozesses wichtige Objektpunkte verloren gehen. Dies erschwert die Bildung einer aussagekräftigen Merkmalsrepräsentation und kann zu Fehldetektionen oder nicht erkannten Objekten führen.

2. **Begrenzte Kontextinformationen durch die Single-Stage-Architektur**

Zur Reduzierung der Rechenzeit verzichtet 3DSSD auf eine nachgelagerte Verfeinerungsstufe. Dadurch stehen dem Modell weniger globale Kontextinformationen zur Verfügung als bei zweistufigen Verfahren wie *PV-RCNN*. Insbesondere kleine Verkehrshütchen können dadurch leichter mit Hintergrundstrukturen verwechselt werden.

3. **Höherer Rechenaufwand gegenüber voxelbasierten Verfahren**

Obwohl 3DSSD effizienter als viele ältere punktbasierte Ansätze arbeitet, erfordern die Sampling-, Grouping- und Nachbarschaftsoperationen weiterhin einen vergleichsweise hohen Rechenaufwand. Dadurch liegen die Inferenzzeiten typischerweise über denen voxelbasierter Verfahren wie *PointPillars*. Für Echtzeitanwendungen mit begrenzten Hardware-Ressourcen kann dies eine Einschränkung darstellen.

Insgesamt wird deutlich, dass 3DSSD trotz seiner guten Detektionsleistung und hohen Effizienz bei der Erkennung kleiner Objekte durch die begrenzte Punktdichte entfernter Objekte sowie die fehlende Verfeinerungsstufe eingeschränkt wird. Diese Beobachtungen motivieren die Untersuchung von **PV-RCNN**, einer hybriden Architektur, die voxelbasierte Kontextinformationen mit einer punktbasierten Verfeinerung kombiniert. Ziel ist es, die Vorteile beider Repräsentationen zu vereinen und dadurch eine robustere Erkennung kleiner Verkehrshütchen zu ermöglichen.

3.4 PV-RCNN

Nach der Analyse der punkt- und voxelbasierten Ansätze zeigt sich ein grundlegender Zielkonflikt: Punktbasierte Methoden wie 3DSSD erfassen feine geometrische Details, sind jedoch rechenintensiv, während voxelbasierte Methoden wie PointPillars effizienter sind, aber durch die Voxelisierung wichtige Höhen- und Forminformationen kleiner Objekte verlieren [11]. Um diese Nachteile zu überwinden, werden hybride Architekturen verfolgt, die die Stärken beider Ansätze kombinieren [39].

3.4.1 Architektur

PV-RCNN (Point-Voxel Region-based Convolutional Neural Network) ist ein zweistufiges Verfahren zur 3D-Objekterkennung, das die Vorteile voxelbasierter und punktbasierter Repräsentationen innerhalb einer gemeinsamen Architektur kombiniert. Ziel ist es, die hohe Effizienz voxelbasierter Verfahren mit der präzisen Lokalisierungsfähigkeit punktbasierter Ansätze zu verbinden [39].

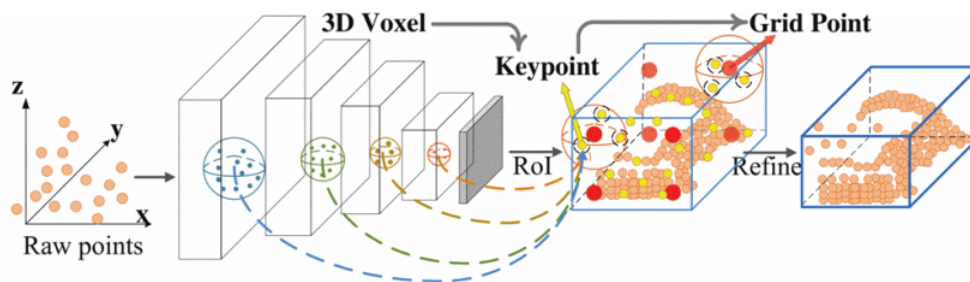


Abbildung 3.1: Architektur von PV-RCNN. Die Punktwolke wird zunächst voxelbasiert verarbeitet, anschließend über Keypoints mit mehrskaligen Voxel-Features angereichert und schließlich mittels RoI-Grid Pooling für die Bounding-Box-Verfeinerung genutzt. Quelle: nach Shi u. a. [39].

Wie in Abbildung 3.1 dargestellt, besteht PV-RCNN aus drei aufeinander aufbauenden Verarbeitungsschritten: (1) voxelbasierte Merkmalsextraktion und Proposal-Generierung, (2) Voxel-to-Keypoint Scene Encoding sowie (3) RoI-Grid Pooling mit anschließender Bounding-Box-Verfeinerung.

Die Eingabe des Modells ist eine LiDAR-Punktwolke, die aus einer Vielzahl dreidimensionaler Punkte besteht und die räumliche Struktur der Umgebung beschreibt. Da Punktwolken unstrukturiert vorliegen und somit nicht direkt mit klassischen Faltungsoperationen verarbeitet werden können, wird die Punktwolke zunächst voxelisiert. Hierbei wird der Raum in kleine dreidimensionale Gitterzellen unterteilt, wodurch eine regelmäßige Repräsentation entsteht. Die erzeugten Voxeldaten werden anschließend durch einen Sparse-Convolution-Backbone verarbeitet, der mehrskalige Merkmale der Szene extrahiert. Durch mehrere Verarbeitungsebenen entstehen Feature-Repräsentationen unterschiedlicher Auflösung, die sowohl lokale als auch globale Kontextinformationen enthalten. Gleichzeitig werden die 3D-Feature-

Volumes entlang der Höhenachse projiziert, um eine Bird's-Eye-View-(BEV)-Merkmalskarte zu erzeugen. Diese stellt die Szene aus der Vogelperspektive dar und dient als Grundlage für die Erzeugung erster Objektvorschläge [39].

Auf Basis der extrahierten Backbone-Features generiert die erste Detektionsstufe eine Menge grober 3D-Bounding-Boxen, sogenannte Region Proposals. Diese enthalten bereits erste Schätzungen für Position, Größe und Orientierung potenzieller Objekte, sind jedoch noch nicht präzise genug für die finale Detektion.

Anschließend erfolgt das sogenannte Voxel-to-Keypoint Scene Encoding, welches den zentralen Bestandteil von PV-RCNN darstellt. Hierfür werden zunächst repräsentative Punkte aus der ursprünglichen Punktwolke mittels Farthest Point Sampling ausgewählt. Diese sogenannten Keypoints besitzen die exakten räumlichen Positionen der ursprünglichen Punkte und dienen als Verbindung zwischen der punktbasierten und der voxelbasierten Repräsentation. Um die Keypoints mit semantischen Informationen anzureichern, aggregiert das Voxel Set Abstraction (VSA)-Modul mehrskalige Merkmale aus verschiedenen Backbone-Ebenen sowie Informationen aus den Rohpunkten und der BEV-Merkmalskarte. Dadurch erhält jeder Keypoint sowohl präzise geometrische Informationen als auch semantisch aussagekräftige Merkmale der Umgebung [39].

Im nächsten Schritt werden die erzeugten Region Proposals detailliert analysiert. Hierzu wird innerhalb jeder vorgeschlagenen Bounding-Box ein regelmäßiges dreidimensionales Gitter erzeugt. Dieses Verfahren wird als RoI-Grid Pooling bezeichnet. Für jeden Gitterpunkt werden benachbarte Keypoints gesucht und deren Merkmale aggregiert. Auf diese Weise entsteht eine objektspezifische Merkmalsrepräsentation, welche die räumliche Struktur des potenziellen Objekts detailliert beschreibt [39].

Die aggregierten Gittermerkmale werden anschließend an einen Refinement Head übergeben. In dieser zweiten Detektionsstufe werden die zunächst groben Objektvorschläge hinsichtlich ihrer Position, Größe und Orientierung verfeinert. Zusätzlich berechnet das Netzwerk eine Konfidenzbewertung für jedes erkannte Objekt. Das Ergebnis sind präzise lokalisierte 3D-Bounding-Boxen mit zugehörigen Klassifikationswerten [39].

Zusammenfassend kombiniert PV-RCNN die Effizienz eines voxelbasierten Backbones mit der geometrischen Genauigkeit punktbasierter Repräsentationen. Durch die Übertragung mehrskaliger Voxel-Features auf repräsentative Keypoints sowie die objektspezifische Merkmalsextraktion mittels RoI-Grid Pooling erreicht das Verfahren eine hohe Detektionsgenauigkeit bei gleichzeitig effizienter Verarbeitung großer LiDAR-Punktwolken.

3.4.2 Warum PV-RCNN für kleine Objekte?

PV-RCNN besitzt mehrere architektonische Eigenschaften, die besonders für die Erkennung kleiner, punktarmer Objekte wie Verkehrshütchen (Traffic Cones) vorteilhaft sind.

- **Kompensation des Quantisierungsverlusts durch Rohpunkt-Features:** Kleine Objekte bestehen häufig nur aus wenigen LiDAR-Punkten, die bei rein voxelbasierten Verfahren durch die Voxelisierung an räumlicher Genauigkeit verlieren können. Im Voxel-to-Keypoint Scene Encoding werden Keypoints direkt mittels Farthest Point Sampling aus den Rohpunkten gewählt, und ihre Merkmale werden zusätzlich durch Rohpunkt-Features ergänzt. Dadurch bleiben die exakten Punktpositionen erhalten, was gerade bei Objekten mit wenigen Messpunkten entscheidend ist [39].
- **Erhalt der Höheninformation:** Die initiale Proposal-Generierung stützt sich, ähnlich wie bei PointPillars, auf eine Bird's-Eye-View- Repräsentation, bei der die Höhendimension komprimiert wird. Für die anschließende Verfeinerung bleibt die Höheninformation hingegen erhalten, da die Voxel Set Abstraction die Keypoint-Features direkt aus den mehrskaligen 3D-Voxel-Merkmalen extrahiert, ohne dass diese zuvor auf die BEV-Ebene reduziert werden. Dies ist insbesondere für die charakteristische, schmale und hohe Form eines Verkehrshütchens von Bedeutung [39].
- **Kontextreiche Verfeinerung durch RoI-Grid Pooling:** Anstatt eine Bounding-Box durch ein einzelnes gepooltes Merkmal zu beschreiben, aggregiert das RoI-Grid-Pooling die Merkmale benachbarter Keypoints über mehrere, sich überlappende lokale Nachbarschaften auf ein reguläres $6 \times 6 \times 6$ Gitter. Dadurch erhält auch ein punktarmes Hütchen über die umgebenden Keypoints reichhaltigen Kontext, der die Klassifikation und Lokalisierung stabilisiert, während klassisches RoI-Pooling bei wenigen Punkten pro Region viele leere Merkmale erzeugen würde [39].

3.4.3 Konfiguration und Hyperparameter

Tabelle 3.6 fasst die zentralen Trainingsparameter des PV-RCNN-Modells zusammen.

Tabelle 3.6: Konfiguration und Hyperparameter des PV-RCNN-Modells.

Parameter	Wert
Voxelgröße	[0,05, 0,05, 0,1] m
Point Cloud Range	[0, -40, -2, 70,4, 40, 1] m
Ankergröße (small_cone)	[0,22, 0,22, 0,40] m
Ankergröße (large_cone)	[0,35, 0,35, 0,70] m
Epochen	100
Lernrate	0,001
Augmentation	Rotation, Skalierung, Spiegeln

3.4.4 Ergebnisse

Tabelle 3.7 fasst die zentralen Detektions- und Laufzeitmetriken des PV-RCNN-Modells zusammen.

Tabelle 3.7: Detektions- und Laufzeitmetriken des PV-RCNN-Modells für die Klasse *small_cone* (Score-Threshold = 0,2, IoU = 0,5).

Metrik	Wert
TP	257/396
FP	193
FN	139/396
Precision (%)	57,1
Recall (%)	64,9
F1-Score (%)	60,7
AP@0.5 (%)	51,5
Avg. Latenz	97,71 ms
Avg. FPS	10,2
GPU-Auslastung	3,0 GB

3.4.5 Diskussion und Limitationen

Die Detektionsmetriken von PV-RCNN (vgl. Tabelle 3.7) zeigen Schwächen in zwei Richtungen gleichzeitig: Das Modell übersieht einen erheblichen Anteil der vorhandenen Objekte und erzeugt zugleich viele falsche Detektionen.

- Mit einem Recall von 64,9 % bleiben 139 der 396 Verkehrshütchen unentdeckt (FN). Ein möglicher Grund hierfür liegt im Voxel-to-Keypoint Scene Encoding: Da das Farthest Point Sampling die Keypoints global über die gesamte Szene verteilt, ohne dabei gezielt kleine, punktarmer Objekte zu priorisieren, könnten Hütchen mit wenigen Rohpunkten mitunter nur unzureichend durch Keypoints repräsentiert werden. In solchen Fällen würde der Refinement-Stufe vermutlich nicht genug geometrische Information zur Verfügung stehen, um ein tatsächlich vorhandenes Hütchen zuverlässig zu bestätigen.
- Gleichzeitig fällt die Precision mit 57,1 % erheblich ab; die hohe Anzahl an False Positives (193) deutet auf eine überaggressive Detektion hin. Ein möglicher Grund hierfür liegt in der zweistufigen Proposal-Architektur: Die RPN-Stufe generiert ihre initialen Vorschläge auf Basis einer BEV-Merkmalkarte, die durch das Downsampling im Voxel-Backbone bereits einen Teil der feinen geometrischen Struktur kleiner Objekte verlieren könnte. Für Verkehrshütchen könnte dies zu ungenau lokalisierten oder fälschlich generierten Proposals führen, die in

der nachfolgenden Refinement-Stufe nicht zuverlässig genug verworfen werden.

- Diese gegenläufigen Fehlerquellen zu wenige bestätigte Detektionen einerseits, zu viele falsche Vorschläge andererseits führen in Summe zu einem moderaten F1-Score von 60,7 % und verdeutlichen, dass sowohl die Keypoint-Repräsentation als auch die Proposal-Generierung für punktarmer, kleine Objekte nicht ausreichend sensitiv kalibriert sind.
- Mit 97,71 ms Latenz und 10,2 FPS erfüllt PV-RCNN zudem nur knapp die Mindestanforderung an Echtzeitfähigkeit und bietet kaum Reserven für zusätzliche Rechenlasten im Gesamtsystem.

Zusammenfassend zeigt sich, dass der hybride Ansatz von PV-RCNN die Kombination aus voxelbasierter Effizienz und punktbasierter Detailtreue bei kompakten, punktarmeren Objekten wie Verkehrshütchen nicht zur vollen Wirkung kommt, da sowohl die Proposal-Generierung als auch die Keypoint-Repräsentation strukturell auf größere, punktreichere Objekte ausgelegt sind. PV-RCNN erweist sich damit für die Detektion kleiner Objekte als nicht geeignet. Diese Einschränkungen motivieren die Untersuchung **aufmerksamkeitsbasierter Architekturen**. Im Gegensatz zu PV-RCNN, das durch Downsampling der Feature-Maps bereits nach wenigen Faltungsschichten feine geometrische Details verliert, verzichtet das folgende Modell DeLiVoTr vollständig auf Downsampling. Durch den Selbstaufmerksamkeitsmechanismus kann **DeLiVoTr** ein großes rezeptives Feld bei gleichbleibender räumlicher Auflösung erreichen eine Eigenschaft, die für die Erkennung kleiner, punktarmer Objekte entscheidend ist.

3.5 DELIVOTR

Während voxelbasierte Ansätze wie PointPillars durch die Aggregation von Höheninformationen und die Verwendung von 2D-Detektionsköpfen effizient sind, geht mit der Voxelisierung und dem Downsampling der Feature-Maps feine geometrische Information kleiner Objekte verloren [11]. Punkt-basierte Methoden wie 3DSSD vermeiden zwar diese Quantisierungsverluste, sind jedoch rechenintensiv im Vergleich zu voxelbasierte Ansätze. Eine vielversprechende Alternative stellen aufmerksamkeitsbasierte Architekturen dar, die durch den Selbstaufmerksamkeitsmechanismus ein großes rezeptives Feld bei gleichbleibender Feature-Map-Auflösung erreichen können [38].

3.5.1 Architektur

DeLiVoTr [38] ist ein aufmerksamkeitsbasierter 3D-Objektdetektor, der speziell für die Erkennung kleiner Objekte in LiDAR-Punktwolken entwickelt wurde. Im Gegensatz zu klassischen voxelbasierten Ansätzen verzichtet DeLiVoTr vollständig auf ein Downsampling der Feature-Maps. Abbildung 3.2 gibt einen Überblick über die Gesamtarchitektur, die aus drei Hauptkomponenten besteht.

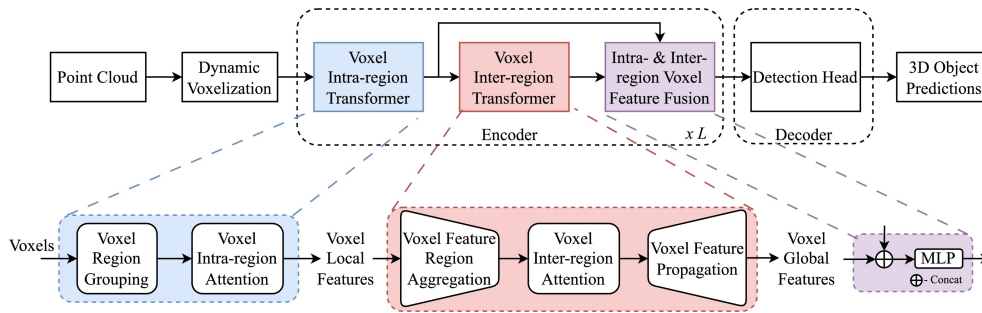


Abbildung 3.2: Architekturübersicht des DeLiVoTr-Modells. Die Eingabepunktwolke wird zunächst dynamisch voxelisiert und anschließend durch den Encoder verarbeitet, der aus *Intra-Region*- und *Inter-Region*-Transformer-Schichten besteht. Der Decoder erzeugt schließlich die finalen 3D-Objektvorhersagen. Abbildung nach [38].

- 1. Voxelisierung:** Die Punktwolke wird zunächst mittels dynamischer Voxelisierung in ein regelmäßiges Gitter aus Voxeln unterteilt, wobei jeder Punkt einer Voxelzelle zugeordnet wird. Im Gegensatz zur harten Voxelisierung werden dabei keine Punkte aufgrund fester Begrenzungen der Voxel- oder Punktzahl verworfen, wodurch Informationsverluste minimiert werden. Dies ist besonders für kleine Objekte wie Verkehrshütchen vorteilhaft, da diese häufig nur durch wenige LiDAR-Punkte repräsentiert werden und jeder zusätzliche Punkt für eine zuverlässige Erkennung wichtig ist [38].

2. **DeLiVoTr-Encoder:** Der Encoder besteht wie in Abbildung 3.2 (unten) dargestellt aus mehreren Schichten mit zwei zentralen Mechanismen:

- *Intra-Region Transformer:* Die Voxel werden in nicht überlappende Regionen ($15 \times 15 \times 1$ Voxel) unterteilt. Innerhalb jeder Region werden die lokalen Beziehungen zwischen den Voxeln berechnet.
- *Inter-Region Transformer:* Um globale Kontextinformationen zu erfassen, werden die Merkmale jeder Region aggregiert und die Wechselwirkungen zwischen verschiedenen Regionen modelliert. Dies ermöglicht ein großes rezeptives Feld ohne Downsampling [38].

3. **Decoder:** Die semantisch reichen Voxelmerkmale des Encoders werden in eine BEV-Feature-Map gerastert. Ein CenterPoint-Detektionskopf sagt daraufhin klassenspezifische Heatmaps, Subvoxel-Positionsverfeinerung, Objektgröße und Orientierung vorher [38].

Ein zentrales Element des Encoders ist der *DeLiVoTr-Block*, der eine DeLight-Transformation mit einer Expand-and-Reduce-Strategie verwendet, um die Breite und Tiefe des Netzwerks effizient zu variieren [38].

3.5.2 Warum DeLiVoTr für kleine Objekte?

DeLiVoTr adressiert gezielt die Herausforderungen der Erkennung kleiner, punktarmer Objekte:

- **Kein Downsampling der Feature-Maps:** Ein Verkehrshütchen, das bei einer Voxelgröße von 0,05 m nur 3×3 Voxel belegt, wäre nach zweimaligem Downsampling nicht mehr erkennbar. DeLiVoTr behält die ursprüngliche Auflösung bei [38].
- **Globaler Kontext durch Inter-Region Attention:** Der Inter-Region Transformer ermöglicht es, auch weit entfernte Voxel miteinander in Beziehung zu setzen, was für die Erkennung punktarmer Hütchen im Fernbereich entscheidend sein kann [38].

3.5.3 Konfiguration und Hyperparameter

Tabelle 3.8 zeigt die zentralen Konfigurationsparameter des DeLiVoTr-Modells.

3.5.4 Ergebnisse

Tabelle 3.9 fasst die zentralen Detektions- und Laufzeitmetriken des DeLiVoTr-Modells zusammen. Die Evaluation wurde mit einem Score-Threshold von 0,5 und IoU von 0,5 durchgeführt.

Tabelle 3.8: Konfiguration und Hyperparameter des DeLiVoTr-Modells.

Parameter	Wert
Voxelgröße	[0,05, 0,05, 0,1] m
Point Cloud Range	[0, -36, -2, 72, 36, 1] m
Regionsform (Intra-Region)	15 × 15 × 1
Modellvariante	DeLiVoTr_small (4,08 M Parameter)
Epochen	80
Lernrate	$4,5 \times 10^{-3}$ (zyklisch)
Batch-Größe	2
Optimierer	AdamW

Tabelle 3.9: Detektions- und Laufzeitmetriken des DeLiVoTr-Modells für die Klasse *small_cone* (Score-Threshold = 0,5, IoU = 0,5).

Metrik	Wert
TP	235/396
FP	16
FN	161/396
Precision (%)	93,6
Recall (%)	59,3
F1-Score (%)	72,6
AP@0.5 (%)	50,4
Avg. Latenz	213 ms
Avg. FPS	4,7
GPU-Auslastung	33 GB
Modellgröße	16,6 MB

3.5.5 Diskussion und Limitationen

Die Detektionsmetriken von DeLiVoTr (vgl. Tabelle 3.9) zeigen ein deutlich asymmetrisches Fehlerbild: Während das Modell nur wenige Fehldetektionen erzeugt, wird ein erheblicher Anteil der vorhandenen Verkehrshütchen nicht erkannt.

- Mit einer Precision von 93,6 % und lediglich 16 Fehldetektionen (FP) erreicht DeLiVoTr eine sehr hohe Vorhersagegenauigkeit. Das Modell klassifiziert nur selten Hintergrundstrukturen fälschlicherweise als Verkehrshütchen. Dies deutet darauf hin, dass die durchgängige Verarbeitung ohne Downsampling eine detaillierte Merkmalsrepräsentation ermöglicht und dadurch Fehlalarme reduziert werden.
- Demgegenüber fällt der Recall mit 59,3 % vergleichsweise niedrig aus. Insgesamt werden 161 der 396 vorhandenen Verkehrshütchen nicht er-

kannt. Für eine sicherheitskritische Anwendung wie das autonome FASTDA-Rennfahrzeug ist dies kritisch, da ein übersehenes Hütchen unmittelbar zu einer Fehleinschätzung der Streckenbegrenzung und damit potenziell zu einer Kollision führen kann.

- Die Ergebnisse zeigen zudem eine deutliche Distanzabhängigkeit. Im Nahbereich (< 10 m) erreicht DeLiVoTr einen F1-Score von 79,5 %, während der Recall im Entfernungsbereich von 10 bis 20 m auf 40,8 % absinkt. Da Verkehrshütchen mit zunehmender Entfernung durch immer weniger LiDAR-Punkte beschrieben werden, stehen dem Modell weniger geometrische Informationen für die Merkmalsextraktion zur Verfügung. Die sinkende Punktdichte stellt daher eine grundlegende Herausforderung für die Erkennung kleiner Objekte dar.
- Hinsichtlich der Laufzeiteigenschaften erreicht DeLiVoTr lediglich 4,7 FPS bei einer durchschnittlichen Inferenzzeit von 213 ms. Gleichzeitig beträgt der Speicherbedarf während der Inferenz etwa 33 GB VRAM. Damit werden die Anforderungen an eine Echtzeitverarbeitung im FASTDA-Fahrzeug nicht erfüllt. Der hohe Ressourcenbedarf ist auf die transformerbasierte Architektur zurückzuführen, deren Selbstaufmerksamkeitsmechanismen einen erheblichen Rechen- und Speicheraufwand verursachen.

Zusammenfassend zeigt DeLiVoTr, dass der Verzicht auf Downsampling für die Erkennung kleiner Objekte grundsätzlich vorteilhaft sein kann, da feingranulare räumliche Informationen erhalten bleiben. Gleichzeitig führt dieser Ansatz jedoch zu einer vergleichsweise hohen Rechen- und Speicherkomplexität. Darüber hinaus bleibt die Erkennungsrate insbesondere bei größeren Entfernungen begrenzt. Für einen praktischen Einsatz in ressourcenbeschränkten Echtzeitsystemen ist das Modell daher nur eingeschränkt geeignet.

Diese Limitationen motivieren die Untersuchung von **LION**. Durch den Einsatz linear skalierender Mamba-Blöcke und speziell entwickelter 3D Spatial Feature Descriptoren verfolgt LION das Ziel, die für kleine und punktarmer Objekte wichtigen räumlichen Informationen zu erhalten und gleichzeitig den Rechenaufwand gegenüber transformerbasierten Ansätzen zu reduzieren.

3.6 LION

Nach der Analyse der bisher betrachteten Architekturen zeigt sich, dass keine der evaluierten Methoden den Zielkonflikt zwischen Detektionsgenauigkeit und Echtzeiteffizienz vollständig auflösen kann. Eine vielversprechende Alternative ist Linear Group RNN (LION), der mit linearer Komplexität skaliert und potenziell hohe Genauigkeit bei moderatem Rechenaufwand bietet [17].

3.6.1 Architektur

LION [17] ist ein sequenzieller 3D-Objektdetektor, der auf dem Mamba-Operator basiert, einem linearen RNN-Modell zur effizienten Verarbeitung langer Sequenzen mit linearer statt quadratischer Komplexität [41]. Im Gegensatz zu Transformer-basierten Ansätzen skaliert LION daher linear mit der Anzahl der Voxel. Die Architektur ist in Abbildung 3.3 dargestellt.

Eine zentrale Herausforderung bei der Anwendung linearer RNNs (Mamba) auf 3D-Punktwolken ist der Verlust räumlicher Informationen beim Flattening der Voxel-Features in eine 1D-Sequenz. Zwei im 3D-Raum benachbarte Voxel können in der 1D-Sequenz weit voneinander entfernt sein. Um dies zu adressieren, integriert LION **3D Spatial Feature Descriptoren**, die lokale räumliche Informationen durch eine 3D-Submanifold-Convolution extrahieren und so die fehlende 3D-Positionswahrnehmung der linearen RNNs kompensieren [17].

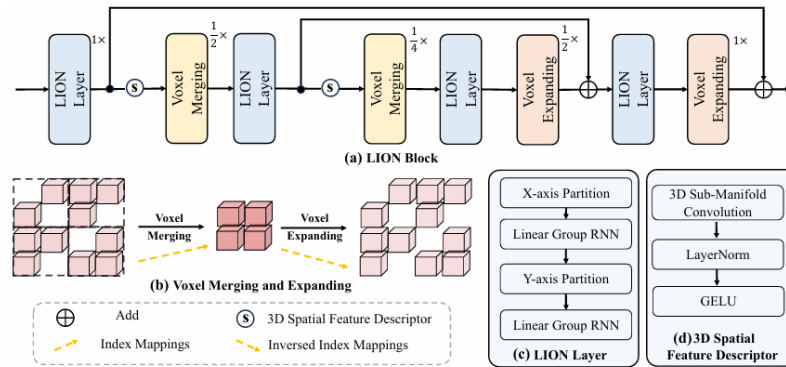


Abbildung 3.3: Architektur des LION-Modells nach [17]. (a) Struktur eines LION-Blocks mit vier LION-Layern, Voxel-Merging- und Expanding-Operationen sowie 3D Spatial Feature Descriptors. (b) Voxel-Merging für Downsampling und Voxel-Expanding für Upsampling. (c) Aufbau eines LION-Layers mit Mamba-Operator. (d) Details des 3D Spatial Feature Descriptors.

1. **Voxelisierung.** Die Eingabepunktwolke wird zunächst in ein spärliches 3D-Voxelgitter überführt. Im Gegensatz zur harten Voxelisierung, die eine feste Anzahl von Punkten pro Voxel erzwingt, verwendet LION eine dynamische Voxelisierung, bei der alle Punkte vollständig auf ihre entsprechenden Voxel abgebildet werden, ohne Punkte zu verwerfen [17].
2. **LION-Backbone mit 3D Spatial Feature Descriptor.** Das Backbone besteht aus mehreren hierarchisch angeordneten LION-Blöcken. Wie in Abbildung 3.3 (a) dargestellt, umfasst ein LION-Block vier LION-Layer, zwei Voxel-Merging-Operationen, zwei Voxel-Expanding-Operationen sowie zwei 3D Spatial Feature Descriptoren. Die Feature-Map durchläuft dabei drei Auflösungsstufen:

- $1\times$ – ursprüngliche Auflösung (H, W, D)
- $\frac{1}{2}\times$ – halbe Auflösung $(\frac{H}{2}, \frac{W}{2}, \frac{D}{2})$
- $\frac{1}{4}\times$ – viertel Auflösung $(\frac{H}{4}, \frac{W}{4}, \frac{D}{4})$

Die Voxel-Merging-Operation (Abbildung 3.3 (b)) führt ein räumliches Downsampling durch, indem benachbarte Voxel zu größeren Einheiten zusammengefasst werden. Die Voxel-Expanding-Operation realisiert das Upsampling zurück zur ursprünglichen Auflösung. Jeder LION-Layer (Abbildung 3.3 (c)) nutzt den Mamba-Operator als zentrales Sequenzmodell, ergänzt durch eine Group Linear Layer-Struktur zur effizienten Merkmalsverarbeitung.

Der 3D Spatial Feature Descriptor (Abbildung 3.3 (d)) besteht aus einer 3D-Submanifold-Convolution, einer LayerNorm und einer GELU-Aktivierung. Er wird vor der Voxel-Merging-Operation platziert, um den Informationsverlust während des Downsamplings zu minimieren. Durch die Extraktion lokaler räumlicher Beziehungen kompensiert der Descriptor die fehlende 3D-Positionswahrnehmung der linearen RNNs und ist insbesondere für die Erkennung kleiner, punktarmer Objekte wie Verkehrshütchen von Bedeutung [17].

3. **TransFusion-Detektionskopf.** Der TransFusion-Head ist ein aufmerksamkeitsbasierter Detektionskopf, der gelernte Objektabfragen mit den Backbone-Merkmalen verfeinert. Für die reine LiDAR-basierte Detektion in dieser Arbeit wird der TransFusion-Head in seiner unimodalen Konfiguration eingesetzt, um die finalen 3D-Bounding-Boxes sowie die Klassenswahrscheinlichkeiten vorherzusagen [17].

3.6.2 Warum LION für kleine Objekte?

Um kleine Objekte wie Verkehrshütchen in LiDAR-Punktwolken zuverlässig zu erkennen, ist eine feine Voxelauflösung erforderlich. Dies führt jedoch zu einer großen Anzahl an Voxeln, was den Rechen- und Speicheraufwand klassischer Aufmerksamkeitsmechanismen erheblich erhöht: Da beim Self-Attention jedes Voxel mit allen anderen k Voxeln verglichen wird, wächst die

Komplexität quadratisch mit $\mathcal{O}(k^2)$. LION adressiert diese Herausforderung durch mehrere gezielte Designentscheidungen:

- **Lineare Komplexität durch sequenzielle Verarbeitung:** Anstelle eines Aufmerksamkeitsmechanismus verwendet LION lineare Recurrent Neural Networks (RNNs), die Voxelmerkmale sequenziell zunächst entlang der X-Achse und anschließend entlang der Y-Achse verarbeiten. Da jedes Voxel nur vom vorherigen Zustand abhängt, wächst der Rechenaufwand linear mit der Anzahl der Voxel – ein entscheidender Vorteil für die Echtzeit-Inferenz bei hoher räumlicher Auflösung [17].
- **Erhalt räumlicher Information durch 3D Spatial Feature Descriptor:** Für kleine Objekte wie Verkehrshütchen ist die lokale 3D-Geometrie wichtiger als globaler Kontext. Da lineare RNNs beim Flatten der Voxel in 1D-Sequenzen räumliche Nachbarschaftsinformation verlieren können, ergänzt LION jeden Block mit einem 3D Spatial Feature Descriptor, der auf 3D-Submanifold-Faltungen basiert. Dieser erfasst explizit die feine räumliche Struktur der lokalen Voxelumgebung und kompensiert so den typischen Informationsverlust linearer RNNs [17].
- **Multi-Skalen-Merkmale durch Voxel Merging und Expanding:** Um Merkmale auf verschiedenen Skalen zu erfassen, nutzt LION Voxel Merging für das räumliche Downsampling sowie Voxel Expanding für das Upsampling der Feature-Maps. Dies ermöglicht dem Netzwerk, sowohl feine geometrische Details als auch größere Strukturen zu modellieren [17].
- **Voxel Generation zur Verdichtung der Feature-Map:** Eine besondere Stärke von LION ist die Voxel Generation: Zunächst werden die Vordergrund-Voxel anhand ihrer Relevanz identifiziert (Top- m Voxel). Um diese werden leere Voxel platziert (Diffusion), die anschließend durch den auto-regressiven Mechanismus des linearen RNN mit sinnvollen Merkmalen befüllt werden. Dies verdichtet die Feature-Map im Bereich relevanter Objekte und verbessert die Erkennung punktarmer Objekte wie Verkehrshütchen erheblich [17].

3.6.3 Konfiguration und Hyperparameter

Tabelle 3.10 fasst die zentralen Konfigurationsparameter des LION-Modells zusammen.

Tabelle 3.10: Konfiguration und Hyperparameter des LION-Modells.

Parameter	Wert
Voxelgröße	[0,075, 0,075, 0,1] m
Point Cloud Range	[−36, −36, −2, 36, 36, 1] m
Backbone	LION _{3D} BackboneOneStride (Mamba)
Detektionskopf	TransFusion Head
Epochen	50
Lernrate	0,001
Batch-Größe	4
Optimierer	AdamW

3.6.4 Ergebnisse

Tabelle 3.11 zeigt die zentralen Detektions- und Laufzeitmetriken für die Klasse `small_cone`.

Tabelle 3.11: Detektions- und Laufzeitmetriken des LION-Modells für die Klasse `small_cone` (IoU = 0,5, *Score – Threshold* = 0,20).

Metrik	Wert
TP	335/396
FP	29
FN	61/396
Precision (%)	92,0
Recall (%)	84,6
F1-Score (%)	88,2
AP@0.5 (%)	83,2
Avg. Latenz	185 ms
Avg. FPS	5,4
GPU-Auslastung	~3,5 GB
Modellgröße	178 MB

3.6.5 Diskussion und Limitationen

Die Ergebnisse des LION-Modells (vgl. Tabelle 3.11) zeigen eine insgesamt hohe Leistungsfähigkeit bei der Erkennung von Verkehrshütchen. Mit ei-

nem Recall von 84,6% werden 335 der insgesamt 396 vorhandenen Objekte korrekt erkannt. Gleichzeitig erreicht das Modell eine Precision von 92,0% sodass nur 29 Fehldetektionen auftreten. Die Kombination aus hoher Precision und hohem Recall führt zu einem F1-Score von 88,2% der den besten Kompromiss zwischen Erkennungsrate und Zuverlässigkeit der Vorhersagen darstellt.

Besonders hervorzuheben ist die AP@0.5 von 83,2%. Im Vergleich zu den zuvor untersuchten Modellen deutet dieser Wert darauf hin, dass LION die Positionen und Ausdehnungen der Verkehrshütchen über verschiedene Konfidenzschwellen hinweg deutlich zuverlässiger bestimmt. Dies spricht für eine robuste Merkmalsextraktion auch bei kleinen und punktarmeren Objekten.

Eine mögliche Erklärung für die hohe Detektionsleistung liegt in der Architektur von LION. Durch die Verwendung von Mamba-basierten Modelle wird eine effiziente Modellierung räumlicher Zusammenhänge ermöglicht, ohne auf die speicherintensive Selbstaufmerksamkeit klassischer Transformer zurückzugreifen. Gleichzeitig erlauben die eingesetzten 3D Spatial Feature Descriptoren die Erhaltung lokaler geometrischer Informationen, die insbesondere für kleine Objekte mit wenigen LiDAR-Punkten von Bedeutung sind.

Trotz dieser herausragenden Detektionsleistung zeigt LION zwei wesentliche Limitationen:

- **Laufzeiteffizienz:** Trotz des vergleichsweise geringen Speicherbedarfs von lediglich etwa 3,5 GB VRAM erreicht LION nur 5,4 FPS bei einer durchschnittlichen Latenz von 185 ms. Damit wird die für das FASTDA-Projekt definierte Echtzeitanforderung von mindestens 10 FPS nicht erfüllt. Eine mögliche Ursache liegt im generell hohen Rechenaufwand der 3D-LiDAR-Verarbeitung. Insbesondere die Voxelisierung der Punktwolke, die Merkmalsextraktion im 3D-Backbone sowie die nachgelagerte Bounding-Box-Generierung tragen wesentlich zur Inferenzzeit bei. Für einen produktiven Einsatz wären daher zusätzliche Optimierungsmaßnahmen wie Quantisierung oder eine TensorRT-basierte Beschleunigung erforderlich.
- **Distanzabhängigkeit:** Die Fehleranalyse zeigt eine klare Abhängigkeit von der Entfernung (vgl. Abbildung 3.5). Im Nahbereich bis 10 m erreicht LION einen Recall von 97,6 % bzw. 96,8 %, während er im Bereich 10–15 m auf 60,0 % sinkt und jenseits von 15 m auf 0,0 % abfällt. Dies ist auf die stark abnehmende Punktdichte mit zunehmender Distanz zurückzuführen [11]. Ab etwa 15 m bestehen Verkehrshütchen typischerweise aus weniger als fünf Punkten, was eine robuste Merkmalsextraktion unmöglich macht.

Da LION die höchste Detektionsleistung erzielt, die definierten Echtzeitanforderungen jedoch weiterhin verfehlt, stellt sich die Frage, ob ein vollständig sparsamer Ansatz beide Ziele – hohe Detektionsgenauigkeit und Echtzeitfähigkeit – gleichzeitig erreichen kann. Diese Fragestellung motiviert die Untersuchung von VoxelNeXt [16].

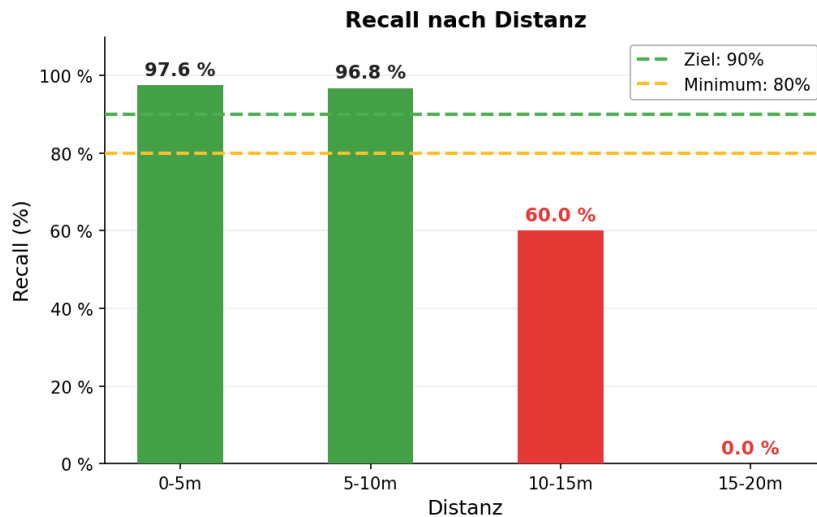


Abbildung 3.4: Recall von LION für die Klasse *small_cone* nach Distanzbereich. Blau kennzeichnet Distanzbereiche mit zuverlässiger Erkennung, Rot kennzeichnet Bereiche mit eingeschränkter bzw. fehlender Detektionsleistung.

3.7 VOXELNEXT

Die Analyse der bisher betrachteten Ansätze zeigt, dass alle evaluierten Modelle die spärliche 3D-Punktwolke auf eine 2D-Darstellung projizieren, um etablierte 2D-Detektionsköpfe nutzen zu können. Dieser Zwischenschritt führt jedoch zu einem Verlust relevanter Höheninformationen, die insbesondere für die Erkennung kleiner Objekte wie Verkehrshütchen von großer Bedeutung sind. Die vergleichsweise schwachen Detektionsergebnisse der bisherigen Modelle lassen darauf schließen, dass die Projektion in die 2D-Ebene die Erfassung feiner räumlicher Merkmale erschwert. Daraus ergibt sich die Motivation, Ansätze zu untersuchen, die auf eine solche Projektion verzichten und die 3D-Struktur der Punktwolke vollständig erhalten.

Mit VoxelNeXt [16] wird daher ein vollständig sparsamer 3D-Detektor betrachtet, der ohne Zwischenrepräsentationen in Form von Bird's-Eye-View- oder Bildprojektionen auskommt.

3.7.1 Architektur

VoxelNeXt [16] ist ein vollständig sparsamer 3D-Objektdetektor, der Objekte direkt aus den erzeugten Voxel-Features vorhersagt. Im Gegensatz zu vielen bisherigen Ansätzen verzichtet das Modell auf die Umwandlung der sparsamen 3D-Repräsentation in dichte 2D-Feature-Maps sowie auf vordefinierte Anker oder Objektzentren. Stattdessen erfolgt die Objekterkennung direkt auf den relevanten Voxeln der Punktwolke.

Der Verarbeitungsablauf besteht aus drei Hauptschritten:

1. Extraktion mehrskaliger Voxel-Features

Zunächst wird die Eingabepunktwolke voxelisiert und durch ein sparsames 3D-CNN verarbeitet. Das Backbone erweitert die üblichen vier Feature-Ebenen mit den Strides $\{1, 2, 4, 8\}$ um zwei weitere Downsampling-Stufen mit den Strides $\{16, 32\}$. Dadurch vergrößert sich das rezeptive Feld erheblich, sodass auch weiter entfernte Kontextinformationen in die Merkmalsrepräsentation einfließen können. Die Feature-Maps der letzten drei Ebenen $\{F_4, F_5, F_6\}$ werden anschließend zusammengeführt und bilden eine mehrskalige Repräsentation der Szene [16].

2. Sparse Height Compression

Die erzeugten 3D-Voxel-Features werden anschließend entlang der Höhenachse (z-Achse) auf die Grundebene projiziert. Dabei werden Voxel mit identischer Position in der Vogelperspektive zusammengefasst und ihre Merkmale addiert. Im Gegensatz zu herkömmlichen Verfahren bleibt die Repräsentation dabei vollständig sparsam, wodurch Speicherbedarf und Rechenaufwand deutlich reduziert werden [16].

3. Direkte Objekterkennung auf sparsamen Voxeln

Auf Basis der komprimierten Voxel-Features sagt der Detection Head zunächst für jedes aktive Voxel Klassenscores voraus. Anschließend werden mittels eines sparsamen Max-Poolings nur die lokal stärksten Voxel ausgewählt. Diese sogenannten Query-Voxel dienen als Ausgangspunkt für die direkte Regression der Bounding Boxes. Dabei werden Position, Größe, Höhe und Orientierung des jeweiligen Objekts vorhergesagt. Da nur relevante Voxel betrachtet werden, kann VoxelNeXt auf eine klassische Non-Maximum Suppression (NMS) verzichten [16].

Zur weiteren Effizienzsteigerung verwendet VoxelNeXt zusätzlich *Spatially Voxel Pruning*. Während der Downsampling-Schritte werden Voxel mit geringer Feature-Aktivierung schrittweise entfernt, sodass nur informative Bereiche der Szene weiterverarbeitet werden. Dadurch sinkt der Rechenaufwand deutlich, ohne die Detektionsleistung wesentlich zu beeinträchtigen [16].

3.7.2 Warum VoxelNeXt für kleine Objekte?

VoxelNeXt adressiert gezielt die Limitationen der bisher evaluierten Modelle:

- **Keine Projektion auf dichte 2D-Feature-Maps:** Die Objekterkennung erfolgt direkt auf den sparsamen Voxel-Features. Dadurch bleiben feine geometrische Informationen der Punktwolke erhalten, die für kleine Objekte wichtig sind [16].
- **Direkte Vorhersage aus Voxel-Features:** VoxelNeXt nutzt sogenannte Query-Voxel zur Bounding-Box-Regression. Da die Vorhersage direkt auf den vorhandenen LiDAR-Punkten basiert, können auch Objekte mit wenigen Punkten erkannt werden [16].

- **Großes rezeptives Feld durch zusätzliche Downsampling-Stufen:** Das Problem fehlender globaler Kontextinformation bei sparsamen Netzwerken wird durch zusätzliche Downsampling-Stufen gelöst, ohne komplexe Aufmerksamkeitsmechanismen oder Transformerarchitekturen einzusetzen. Dies ermöglicht eine robuste Erkennung auch kleiner Objekte bei moderatem Rechenaufwand [16].
- **Hohe Effizienz durch vollständig sparse Verarbeitung:** Da ausschließlich nicht-leere Voxel verarbeitet werden, ist VoxelNeXt deutlich effizienter als Modelle mit dichten Detektionsköpfen. Dies spiegelt sich in einer hohen Inferenzgeschwindigkeit wider, die den Echtzeiteinsatz auf ressourcenbeschränkten On-Board-Systemen ermöglicht.

3.7.3 Konfiguration und Hyperparameter

Tabelle 3.12 fasst die zentralen Konfigurationsparameter des VoxelNeXt-Modells zusammen. Als Startpunkt diente ein auf dem nuScenes-Datensatz vortrainierter Checkpoint, der für die Klasse `traffic_cone` bereits einen AP von 80,7% erzielt. Das Modell wurde anschließend über 50 Epochen auf den eigenen Datensatz feinjustiert.

Tabelle 3.12: Konfiguration und Hyperparameter des VoxelNeXt-Modells.

Parameter	Wert
Voxelgröße	[0,075, 0,075, 0,1] m
Point Cloud Range	[-36, -36, -2, 36, 36, 1] m
Backbone	Sparse CNN (Strides {2, 4, 8, 16, 32})
Detektionskopf	Sparse Prediction Head (NMS-frei)
Epochen	50
Lernrate	0,001
Batch-Größe	4
Optimierer	AdamW

3.7.4 Ergebnisse

Tabelle 3.13 fasst die zentralen Detektions- und Laufzeitmetriken des VoxelNeXt-Modells für die Klasse `small_cone` zusammen. Die Evaluation erfolgte auf dem Validierungsset mit 34 Frames und 396 Ground-Truth-Instanzen.

Tabelle 3.13: Detektions- und Laufzeitmetriken des VoxelNeXt-Modells für die Klasse *small_cone* (IoU = 0,5, Score-Threshold = 0,5).

Metrik	Wert
TP	335 / 396
FP	41
FN	61 / 396
Precision (%)	89,1
Recall (%)	84,6
F1-Score (%)	86,8
AP@0.5 (%)	79,6
Avg. Latenz	18,2 ms
Avg. FPS	55,0
GPU-Auslastung	~2,5 GB

3.7.5 Diskussion

Die quantitative Auswertung in Tabelle 3.13 zeigt, dass VoxelNeXt insgesamt eine ausgewogene Detektionsleistung für die Klasse *small_cone* erreicht. Mit einer Precision von 89,1 % und einem Recall von 84,6 % resultiert ein F1-Score von 86,8 %, was auf eine robuste Balance zwischen Genauigkeit und Vollständigkeit der Vorhersagen hinweist.

Konkret werden 335 von 396 Instanzen korrekt erkannt, während 61 Objekte übersehen werden (False Negatives). Eine detaillierte Analyse dieser Fehlerquellen erfolgt in Unterabschnitt 3.7.6. Gleichzeitig fällt die Anzahl der Fehlalarme mit 41 False Positives vergleichsweise gering aus, was auf eine effektive Unterdrückung von Rauschen und bodennahen Strukturartefakten schließen lässt.

Die Ergebnisse bestätigen damit die zuvor beschriebenen architektonischen Vorteile von VoxelNeXt, insbesondere die direkte voxelbasierte Vorhersage und die Erhaltung feiner geometrischer Strukturen, welche sich positiv auf die Detektion kleiner und schwer separierbarer Objekte auswirken.

Besonders hervorzuheben ist die Laufzeiteffizienz: Mit einer mittleren Latenz von 18,2 ms und 55,0 FPS ist VoxelNeXt das mit Abstand schnellste Modell aller evaluierten Ansätze. Die Verarbeitungsrate liegt dabei mit 55,0 FPS um den Faktor 5,5 über der LiDAR-Abtastrate von 10 Hz und erfüllt die Echtzeitanforderungen somit deutlich. Der entscheidende Effizienzgewinn gegenüber LION (185 ms) und DeLiVoTr (213 ms) ergibt sich dabei wegen der vollständig sparsamen Verarbeitung: Während LION den Mamba-Operator sequenziell auf alle Voxel anwendet und DeLiVoTr eine quadratische Komplexität $\mathcal{O}(k^2)$ im Selbstaufmerksamkeitsmechanismus aufweist, skaliert VoxelNeXt linear mit der Anzahl nicht-leerer Voxel und profitiert dabei von hochoptimierten Sparse-Convolution-Bibliotheken wie SpConv, die eine direkte GPU-Beschleunigung ermöglichen.

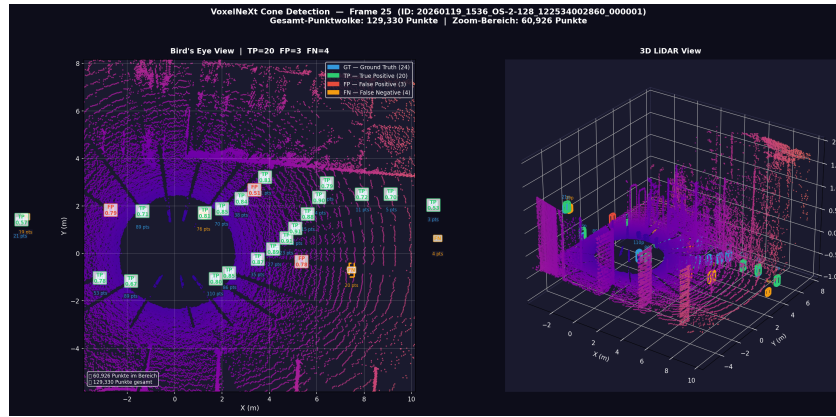


Abbildung 3.5: Qualitative Ergebnisse von VoxelNeXt für die Klasse *small_cone* (Frame 25, 129.330 Punkte gesamt). Die Verkehrshütchen sind halbkreisförmig angeordnet. Der schwarze Kreis in der Mitte der Bird's-Eye-View entspricht dem blinden Fleck des LiDAR-Sensors (OS-2-128), in dem keine Messpunkte erfasst werden. Grüne Bounding Boxes kennzeichnen korrekt erkannte Hütchen (*True Positives*, TP=20), orange Boxes markieren übersehene Hütchen (*False Negatives*, FN=4), rote Boxes stehen für Fehldetektionen (*False Positives*, FP=3). Im Hintergrund sind Gebäudestrukturen als großflächige Punktwolken sichtbar.

VoxelNeXt ist damit das einzige Modell, das sowohl die Detektions- als auch die Echtzeitanforderungen vollständig erfüllt nicht durch einen Kompromiss zwischen beiden Zielen, sondern weil die vollständig sparsame Architektur beide Anforderungen strukturell begünstigt.

3.7.6 Fehleranalyse

Tabelle 3.14 und Abbildung 3.6 zeigen die Detektionsleistung von VoxelNeXt in Abhängigkeit von der Distanz zum Sensor.

Distanz	Recall	Verpasste Cones	Punkte/Cone	Bewertung
0–5 m	93,2 %	6	15–25	Ausgezeichnet
5–10 m	92,3 %	15	8–15	Ausgezeichnet
10–15 m	64,8 %	36	4–8	Verbesserungsbedarf
15–20 m	20,0 %	4	2–4	Kritisch
> 20 m	–	0	< 2	Keine Daten

Tabelle 3.14: Recall von VoxelNext für die Klasse *small_cone* nach Distanzbereich. *Verpasste Cones* entspricht den False Negatives (FN); *Punkte/Cone* gibt die typische Anzahl an LiDAR-Punkten des Ouster OS2-128 je Distanzbereich an.

Die Analyse zeigt eine deutliche Abhängigkeit der Detektionsleistung von der Distanz. Im Nahbereich bis 10 m liefert VoxelNeXt eine zuverlässige Erkennung: Mit nur 6 verpassten Cones bei 15–25 LiDAR-Punkten pro Objekt erreicht das Modell im Bereich 0–5 m einen Recall von 93,2 %. Im Be-

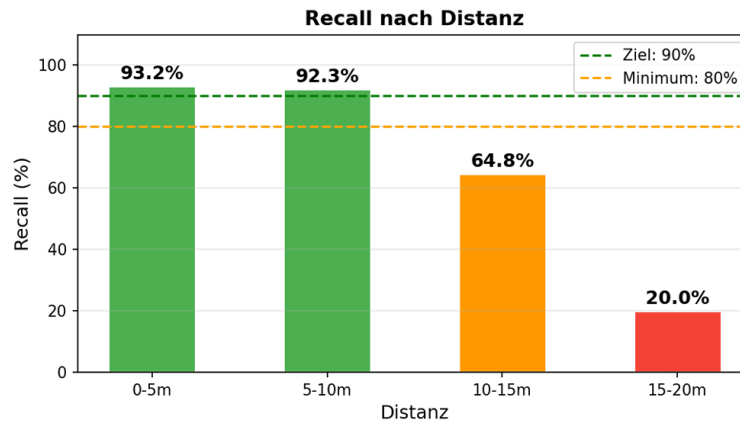


Abbildung 3.6: Recall von VoxelNeXt für die Klasse *small_cone* nach Distanzbereich. Blau kennzeichnet Distanzbereiche mit zuverlässiger Erkennung, Rot kennzeichnet Bereiche mit eingeschränkter bzw. fehlender Detektionsleistung.

reich 5–10 m bleiben bei 8–15 Punkten pro Cone lediglich 15 Instanzen unerkannt, was einem Recall von 92,3 % entspricht. Beide Bereiche übertreffen die definierte Mindestanforderung von 80 % deutlich, wie Abbildung 3.6 veranschaulicht. Mit zunehmender Distanz verschlechtert sich die Detektionsleistung erheblich, was direkt mit der abnehmenden Punktdichte korreliert. Im Bereich 10–15 m stehen dem Modell nur noch 4–8 Punkte pro Cone zur Verfügung, wodurch 36 Instanzen nicht erkannt werden und der Recall auf 64,8 % sinkt deutlich unterhalb der Mindestanforderung. Im Bereich 15–20 m werden Verkehrshütchen lediglich durch 2–4 LiDAR-Punkte repräsentiert; der Recall fällt auf 20,0 %, wobei 4 Cones unerkannt bleiben, was eine zuverlässige Merkmalsextraktion stark einschränkt [11]. Jenseits von 20 m liegen keine auswertbaren Detektionen vor, sodass für diesen Distanzbereich keine belastbare Aussage getroffen werden kann. Zusammenfassend zeigt VoxelNeXt im Nahbereich bis 10 m eine robuste Detektionsleistung mit Recall-Werten über 92 %, die die Mindestanforderung klar erfüllen. Ab 10 m nimmt die Erkennungsleistung mit sinkender Punktdichte deutlich ab und erreicht im Bereich 15–20 m nur noch 20,0 %, was für einen zuverlässigen Einsatz im FASTDA-Projekt nicht ausreichend ist.

3.8 GESAMTBEWERTUNG DER UNTERSUCHTEN MODELLE

Die Einzelanalysen der untersuchten 3D-Deep-Learning-Modelle zeigen, dass jede Architekturkategorie (voxelbasiert, punktbasiert, hybrid sowie aufmerksamkeitsbasiert) spezifische Stärken und Schwächen bei der Erkennung kleiner, punktarmer Objekte aufweist. Der folgende Gesamtvergleich führt diese individuellen Eigenschaften in einer gemeinsamen Betrachtung zusammen. Dadurch wird der zentrale Zielkonflikt zwischen Detektionsqualität und Laufzeiteffizienz quantitativ sichtbar gemacht, wie Tabelle 3.15 und Abbildung 3.7 zeigen. Mit einem F1-Score von 88,2 % erreicht LION die höchste

Tabelle 3.15: Vergleich der evaluierten Modelle für die Klasse `small_cone`.

Modell	Precision (%)	Recall (%)	F1 (%)	AP@0.5 (%)	FPS	Echtzeit
PointPillars	61,2	58,8	60,0	50,7	48,3	✓
3DSSD-24k	68,77	73,98	71,9	50,7	22,0	✓
PV-RCNN	57,1	64,9	60,7	51,5	10,2	✓
DeLiVoTr	93,6	59,3	72,6	50,4	4,7	×
LION	92,0	84,6	88,2	83,2	5,4	×
VoxelNeXt	89,1	84,6	86,8	79,6	55,0	✓

Anmerkung: ✓ = Echtzeitanforderung erfüllt (≥ 10 FPS), × = nicht erfüllt.

Detektionsqualität aller evaluierten Modelle. Entscheidend dafür ist die Voxel Generation, die die Feature-Map gezielt im Bereich relevanter Objekte verdichtet und so die Erkennung punktarmer Objekte wie Verkehrshütchen erst zuverlässig ermöglicht, gefolgt von VoxelNeXt mit 86,8 %, der durch seine durchgehend sparsame Feature-Verarbeitung die feine geometrische Information kleiner Objekte bewahrt.

Beide Modelle erzielen eine deutlich höhere Erkennungsrate als die übrigen Ansätze, die sich im Bereich von 60–72 % bewegen. DeLiVoTr weist mit 93,6 % zwar die höchste Precision auf, wird jedoch durch einen Recall von lediglich 59,3 % auf einen F1-Score von 72,6 % begrenzt. Obwohl beide Modelle eine vergleichbare Detektionsleistung erzielen, liegt der entscheidende Unterschied in der Laufzeit: VoxelNeXt übertrifft mit 55,0 FPS die Echtzeitanforderung von mindestens 10 FPS deutlich, während LION mit 5,4 FPS und DeLiVoTr mit 4,7 FPS für den Echtzeitbetrieb ausscheiden. Dieser Zielkonflikt zwischen Detektionsqualität und Laufzeiteffizienz wird in Abbildung 3.7 anhand eines normierten Radar Charts veranschaulicht.

Diese Analyse zeigt, dass VoxelNeXt als einziges Modell sowohl die Echtzeitanforderungen als auch eine hohe Detektionsleistung erfüllt und sich damit als geeignetste Lösung für den unmittelbaren Einsatz im FASTDA-Projekt erweist. LION erzielt zwar die beste Detektionsqualität aller evaluierten Modelle, scheidet jedoch aufgrund seiner Laufzeit für ressourcenbeschränkte On-Board-Systeme aus.

Beide Modelle teilen jedoch eine zentrale Schwäche: Die Erkennungsleistung nimmt ab einer Entfernung von etwa 15 m deutlich ab, bedingt durch die sinkende Punktdichte des LiDAR-Sensors. Bei höheren Renngeschwindigkeiten stellt dies ein kritisches Sicherheitsproblem dar 3.8.

Das folgende Kapitel widmet sich daher gezielten Optimierungsansätzen zur Verbesserung der Detektion kleiner LiDAR-Objekte im Fernbereich. Ziel ist es, die distanzabhängige Leistungsabnahme der untersuchten Modelle zu reduzieren und die Erkennungsreichweite insbesondere für Verkehrshütchen in Entfernungen von mehr als 10 m zu erhöhen.

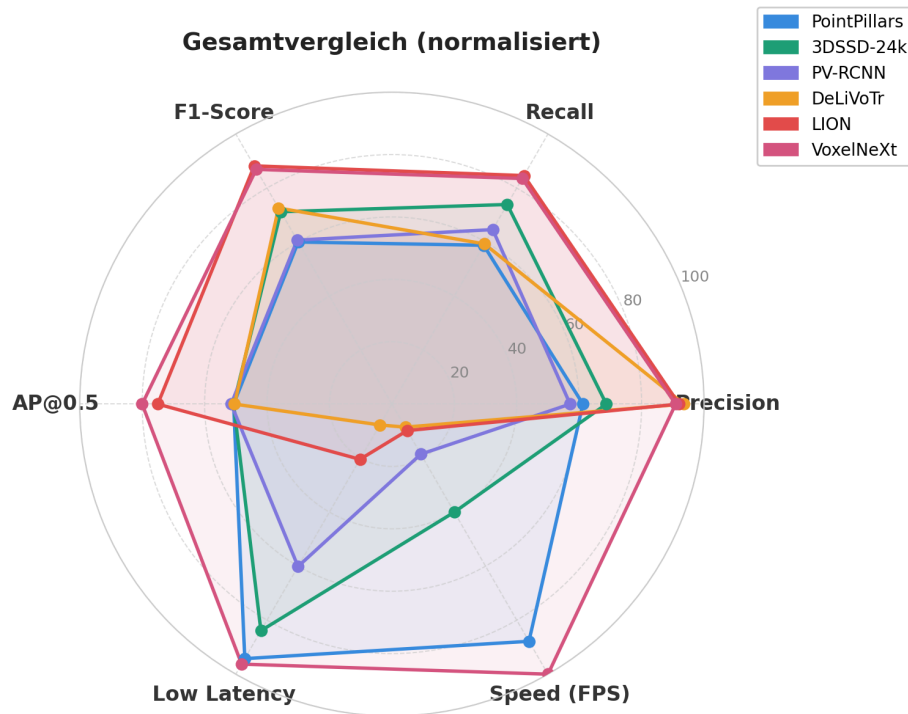


Abb.: Gesamtvergleich aller evaluierten Modelle. Speed und Low Latency: höher = besser (invertiert).

Abbildung 3.7: Gesamtvergleich der evaluierten Modelle anhand normalisierter Metriken (Radar Chart). Die Achsen zeigen Precision, Recall, F1-Score, AP@0.5, Low Latency und Speed (FPS), jeweils auf einen Wertebereich von 0–100 normalisiert. Dabei ist zu beachten, dass Latenz und FPS gegenläufige Qualitätsindikatoren sind: Eine niedrige Latenz (ms) bedeutet schnellere Verarbeitung und ist daher wünschenswert, während eine hohe FPS-Zahl ebenfalls auf eine bessere Laufzeiteffizienz hindeutet. Um beide Metriken einheitlich darstellbar zu machen, wurde die Latenzachse invertiert – hohe Werte auf der Low Latency-Achse entsprechen also einer *niedrigen* tatsächlichen Latenz in ms – während die Speed-Achse (FPS) direkt normalisiert wurde. Somit entsprechen größere Flächen im Radar Chart einer besseren Gesamtleistung. LION und VoxelNeXt dominieren bei der Detektionsqualität, während VoxelNeXt als einziges Modell zugleich eine hohe Laufzeiteffizienz aufweist.

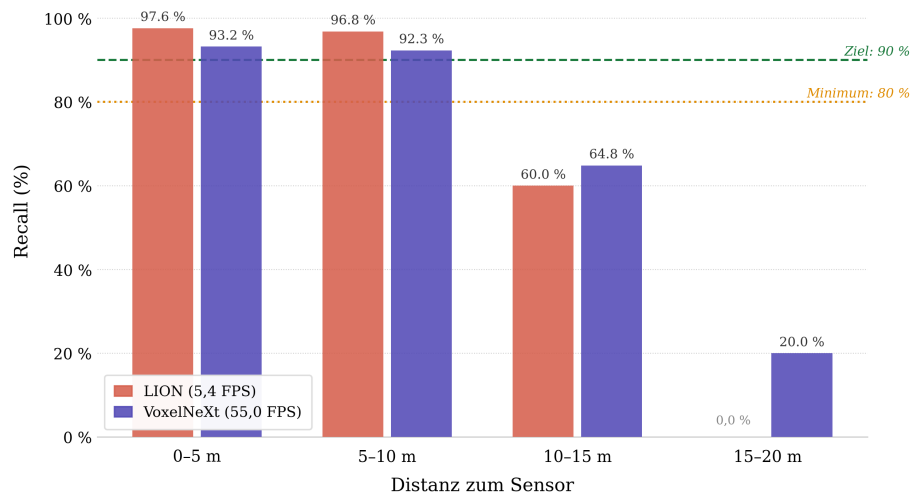


Abbildung 3.8: Recall von LION (5,4 FPS) und VoxelNeXt (55,0 FPS) für die Klasse `small_cone` nach Distanzbereich. Im Nahbereich bis 10 m erfüllen beide Modelle die Mindestanforderung von 80 % deutlich — LION erreicht bis zu 97,6 %, VoxelNeXt bis zu 93,2 %. Im Bereich 10–15 m nimmt die Erkennungsleistung deutlich ab (LION: 60,0 %, VoxelNeXt: 64,8 %). Ab 15 m bricht sie drastisch ein (LION: 0,0 %, VoxelNeXt: 20,0 %), was mit der abnehmenden Punktdichte des Ouster OS2-128-Sensors korreliert. Die grün gestrichelte Linie markiert den Zielwert von 90 %, die orange gepunktete Linie die Mindestanforderung von 80 %.

ANSÄTZE ZUR VERBESSERUNG DER OBJEKTERKENNUNG KLEINER LIDAR-OBJEKTE IM FERNBEREICH

Wie die Gesamtbewertung der untersuchten Modellen in Kapitel 3.8 gezeigt hat, weisen sowohl LION als auch VoxelNeXt eine gemeinsame zentrale Limitation auf: die stark distanzabhängige Erkennungsleistung, die ab einer Entfernung von etwa 10 m deutlich abnimmt. Ursache hierfür ist die abnehmende Punktdichte des Ouster OS2-128-Sensors, der Verkehrshütchen ab ca. 10 m nur noch durch eine geringe Anzahl an LiDAR-Punkten repräsentiert.

Um diese Limitation zu adressieren, wird zunächst der Datensatz hinsichtlich seiner Distanzverteilung systematisch analysiert, bevor anschließend gezielte Optimierungsansätze untersucht werden. Ziel ist es, die Erkennungsreichweite im kritischen Distanzbereich ab 10 m zu verbessern.

4.1 ANALYSE DER DATENSATZVERTEILUNG

Bevor die Optimierungsansätze vorgestellt werden, wird der eigene Datensatz hinsichtlich der Distanzverteilung analysiert, um die Ursache der distanzabhängigen Erkennungsschwäche quantitativ zu motivieren. Tabelle 4.1 zeigt die Verteilung der annotierten Kegel sowie die durchschnittliche Punktzahl pro Kegel nach Distanzbereich für Trainings- und Validierungsdaten.

Tabelle 4.1: Verteilung der annotierten Kegel nach Distanzbereich.

Bereich	Training			Validierung		
	Kegel	Anteil (%)	Ø Punkte	Kegel	Anteil (%)	Ø Punkte
0–5 m	237	19,0	40,6	85	21,5	37,9
5–10 m	614	49,3	8,3	186	47,0	8,4
10–15 m	377	30,3	3,7	120	30,3	3,4
15–20 m	18	1,4	2,2	5	1,3	1,8
Gesamt	1246	100	–	396	100	–

Die Analyse zeigt zwei wesentliche Herausforderungen, die die Detektionsleistung ab 10 m maßgeblich einschränken.

Punktarmut im FOKUS-Bereich. Im Distanzbereich 10–15 m (im Folgenden als *FOKUS-Bereich* bezeichnet) stehen dem Modell durchschnittlich nur noch 3,4 LiDAR-Punkte pro Kegel zur Verfügung, gegenüber 37,9 Punkten im Nahbereich bis 5 m (vgl. Tabelle 4.1, Validierungsdaten). Diese erhebliche Abnahme der Punktdichte auf weniger als ein Zehntel des Nahbereichswertes macht eine zuverlässige Merkmalsextraktion für alle evaluierten Modelle

erheblich schwieriger, da die charakteristische Kegelform bei so wenigen Messpunkten kaum noch rekonstruierbar ist.

Unterrepräsentation im Training. Der FOKUS-Bereich (10–15 m) macht zwar 30,3 % aller Trainingsinstanzen aus, enthält jedoch kaum mehr als 3–4 Punkte pro Objekt. Der Bereich jenseits von 15 m ist mit lediglich 18 Trainingsinstanzen (1,4 %) stark unterrepräsentiert, sodass die Modelle für diesen Bereich keine ausreichende statistische Grundlage erlernen können. Abbildung 4.1 zeigt die durchschnittliche Anzahl der LiDAR-Punkte pro Verkehrskegel in Abhängigkeit von der Entfernung zum Sensor. Mit zunehmender Distanz nimmt die Punktdichte deutlich ab. Während ein Kegel bei einer Entfernung von 1 m im Mittel noch durch 74,2 Punkte repräsentiert wird, sinkt die durchschnittliche Punktzahl bei 3 m bereits auf 30,0 und bei 5 m auf 14,1 Punkte. Ab einer Entfernung von etwa 7 m liegt sie mit durchschnittlich 7,3 Punkten unterhalb des in der Abbildung markierten Referenzwerts von acht Punkten, der als Bereich günstiger Erkennungsbedingungen betrachtet wird. Zwischen 8 und 12 m reduziert sich die Punktzahl weiter von 6,2 auf etwa 3,0 Punkte pro Kegel. Ab ungefähr 13 m wird schließlich auch der für VoxelNeXt angenommene Mindestwert von etwa drei Punkten unterschritten. In diesem Entfernungsbereich stehen somit häufig nicht mehr genügend Messpunkte zur Verfügung, um die geometrische Struktur eines Verkehrskegels zuverlässig abzubilden.

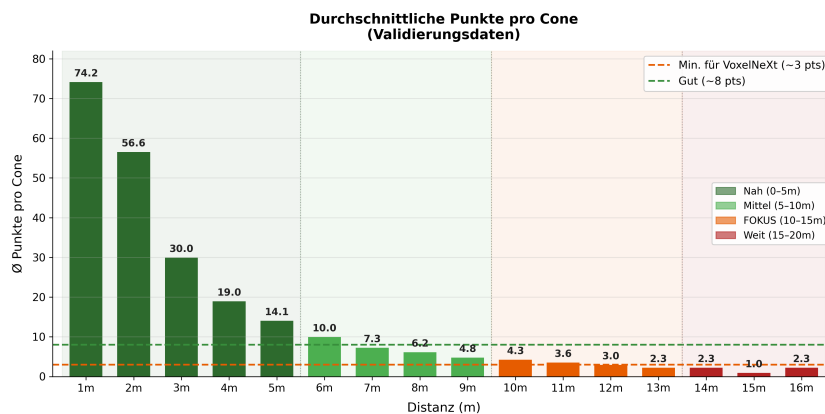


Abbildung 4.1: Durchschnittliche Punktzahl pro Kegel in Abhängigkeit von der Entfernung. Die gestrichelten Linien markieren den Referenzwert guter Erkennungsbedingungen bei etwa 8 Punkten sowie den empirischen Mindestwert für VoxelNeXt bei etwa 3 Punkten.

Diese Analyse motiviert den Einsatz von Methoden zur künstlichen Punktverdichtung, die im folgenden Abschnitt vorgestellt werden. Ziel ist es, die Punktarmut im FOKUS-Bereich gezielt zu kompensieren und dadurch die Erkennungsreichweite von VoxelNeXt über 10 m hinaus zu verbessern. Der Bereich jenseits von 15 m wird aufgrund der stark unterrepräsentierten Datenbasis mit lediglich 1,4 aller Trainingsinstanzen von der Optimierung ausgeschlossen.

4.2 UNTERSUCHTE LÖSUNGSANSÄTZE

Zur Verbesserung der Erkennungsleistung im FOKUS-Bereich werden zwei grundlegende Strategien untersucht: die Erweiterung des Netzwerks durch *Shape Priors* sowie die künstliche Punktverdichtung mittels *Super-Resolution*. Beide Ansätze verfolgen das Ziel, die Punktarmut im Distanzbereich 10–15 m zu kompensieren, unterscheiden sich jedoch grundlegend in ihrer Methodik und Eignung für den vorliegenden Anwendungsfall.

4.2.1 *Shape Priors als Netzwerkerweiterung*

Ein vielversprechender Ansatz zur Verbesserung der Detektion punktarmer Objekte im mittleren Entfernungsbereich (10–15 m) ist die Integration eines Shape-Prior-Moduls. Die Grundidee besteht darin, dem Detektor eine kompakte, im Nahbereich gelernte Repräsentation der typischen Objektgeometrie bereitzustellen. Anstatt die Objektform ausschließlich aus wenigen lidar-Messpunkten zu rekonstruieren, kann das Netzwerk fehlende geometrische Information durch gespeichertes Vorwissen über die typische Kegelform ergänzen. Für geometrisch einfache und rotationssymmetrische Objekte wie Verkehrshütchen erscheint dieser Ansatz prinzipiell attraktiv, da ihre Form durch kompakte Formmodelle gut approximierbar ist.

In der Literatur existieren unterschiedliche Ansätze, um geometrisches Vorwissen in LiDAR-basierte Verfahren zu integrieren. Ein für den vorliegenden Anwendungsfall besonders relevanter Ansatz ist JS3C-Net [42]. Das Verfahren wurde primär für semantische Segmentierung und semantische Szenenvervollständigung entwickelt und zeigt, wie fehlende räumliche Strukturen in spärlichen Punktwolken mithilfe gelernter Kontext- und Forminformationen ergänzt werden können. Das Framework kombiniert semantische Segmentierung mit einem semantische Szenenvervollständigung-Modul, das aus mehreren aufeinanderfolgenden -Frames kontextuelle Forminformationen lernt und fehlende räumliche Strukturen ergänzt (siehe Abbildung 4.2). Besonders relevant für den vorliegenden Anwendungsfall ist, dass bestimmte Hilfsmodule ausschließlich während des Trainings aktiv sind und zur Inferenzzeit verworfen werden können, sodass kein zusätzlicher Rechenaufwand entsteht. Damit verdeutlicht JS3C-Net, dass geometrisches Vorwissen prinzipiell zur Verarbeitung spärlicher und unvollständiger LiDAR-Daten beitragen kann.

Trotz der konzeptionellen Eignung wurde der Shape-Prior-Ansatz im Rahmen dieser Arbeit nach eingehender Analyse aus mehreren, miteinander zusammenhängenden Gründen nicht weiterverfolgt:

- **Fehlende Sequenzdaten als strukturelle Voraussetzung.** JS3C-Net [42] setzt für das Erlernen kontextueller Forminformationen sequenzielle lidar-Frames voraus, da das ssc-Modul zusätzliche Supervisionssignale aus dem zeitlichen Verlauf mehrerer Aufnahmen gewinnt. Der im Rahmen dieser Arbeit erhobene Datensatz besteht jedoch aus statisch annotierten Einzelframes ohne zeitliche Verknüpfung. Eine di-

rekte Übertragung des JS3C-Net-Ansatzes ist damit strukturell nicht möglich, ohne den Datensatz grundlegend neu zu erheben.

- **Architekturkomplexität und Trainingsaufwand.** Die Integration von Shape-Prior-Modulen erfordert tiefgreifende Eingriffe in bestehende Netzwerkarchitekturen, insbesondere in die Feature-Extraktion oder den Detektionskopf von Modellen wie VoxelNeXt und LION. Viele Shape-Prior-basierte Ansätze setzen zudem das Vortrainieren separater Formrepräsentationen voraus, wofür eine ausreichende Menge vollständiger und qualitativ hochwertiger 3D-Objektinstanzen erforderlich ist. Der vorliegende Datensatz mit 169 Frames und 1.642 annotierten Instanzen der Klasse *small_cone* erfüllt diese Anforderung nur eingeschränkt, insbesondere im fokus-Bereich von 10–15 m, in dem pro Objekt durchschnittlich nur 3,4 Punkte vorliegen. Eine robuste Formrepräsentation lässt sich auf dieser Datenbasis daher nicht zuverlässig erlernen.
- **Eingeschränkte Wirksamkeit bei extremer Punktarmut.** Der entscheidende Grund für den Ausschluss liegt in der sehr geringen Informationsdichte der LiDAR-Punktwolken im kritischen Entfernungsbereich. Shape Priors können vorhandene geometrische Informationen ergänzen oder stärker gewichten, jedoch keine zuverlässige Objektstruktur rekonstruieren, wenn im Eingangssignal nur sehr wenige Messpunkte vorhanden sind. Im fokus-Bereich ab etwa 10 m liegen für Verkehrshütchen teilweise weniger als fünf Punkte pro Objekt vor. Unter diesen Bedingungen bleibt fraglich, ob ein zusätzlicher Shape-Prior-Ansatz eine robuste und konsistente Verbesserung der Detektionsleistung ermöglichen würde.

Aufgrund dieser Einschränkungen wird der Shape-Prior-Ansatz im Rahmen dieser Arbeit nicht implementiert. Stattdessen wird im folgenden Abschnitt ein datengetriebener Optimierungsansatz verfolgt, der ohne zusätzliche architektonische Eingriffe auskommt und gezielt die Punktarmut im fokus-Bereich kompensiert.

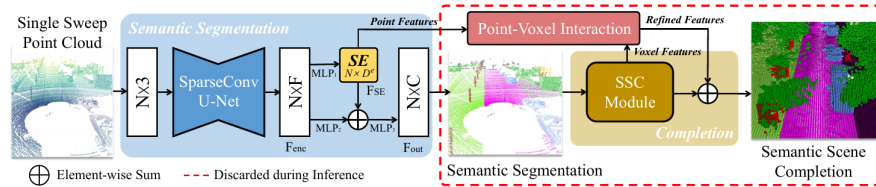


Abbildung 4.2: **Gesamtarchitektur des JS3C-Net.** Die Pipeline nimmt eine spärliche, unvollständige Single-Sweep-Punktwolke ($N \times 3$) als Eingabe und verarbeitet diese zunächst durch ein SparseConv U-Net, das durch effiziente Submanifold Sparse Convolutions eine kompakte Punktfeature -Repräsentation $F_{enc} \in \mathbb{R}^{N \times F}$ erzeugt. Auf Basis dieser Encoding-Features generiert MLP_1 ein Shape Embedding $F_{SE} \in \mathbb{R}^{N \times D^e}$, welches die geometrischen Formeigenschaften jedes Punktes in einem kompakten Raum kodiert. Parallel dazu transformiert MLP_2 die initialen Encoding-Features, die anschließend elementweise mit F_{SE} summiert und durch MLP_3 zu den finalen Punktfeatures $F_{out} \in \mathbb{R}^{N \times C}$ verarbeitet werden, welche die semantischen Klassenwahrscheinlichkeiten für die Semantic Segmentation repräsentieren. Im Anschluss fließen F_{out} in das Semantic Scene Completion (SSC)-Modul, das durch Voxelisierung und mehrere dichte Faltungsblöcke mit Skip-Connections die sparse Eingabe zu einer vollständigen 3D- Szenenrepräsentation vervollständigt. Das Point-Voxel Interaction (PVI)-Modul führt schließlich die feingranularen, aber unvollständigen Punktfeatures aus dem Shape Embedding mit den groben, aber vollständigen Voxelfeatures des SSC-Moduls zusammen, indem es über einen k -Nearest-Neighbor-Mechanismus und ein Graph Convolutional Network die räumlichen und semantischen Beziehungen zwischen Punkten und Voxeln modelliert und so eine verfeinerte Szenenkompletion erzeugt. Hervorzuheben ist, dass das SSC- und das PVI-Modul ausschließlich während des Trainings aktiv sind und zur Inferenzzeit verworfen werden (gestrichelte rote Linie), sodass kein zusätzlicher Rechenaufwand für die Semantic Segmentation entsteht. Abbildung adaptiert von [42].

4.2.2 Punktverdichtung durch Super-Resolution

Als zentraler Optimierungsansatz wird die künstliche Punktverdichtung mittels *Super-Resolution* (SR) untersucht. Ziel ist es, die Informationsdichte der LiDAR-Punktwolke durch synthetisch erzeugte Punkte zu erhöhen, ohne die Architektur des Detektionsmodells zu verändern. Die verdichtete Punktwolke wird als vorverarbeitete Eingabe an die bestehende VoxelNeXt-Pipeline übergeben.

Einen verwandten Ansatz verfolgen [43], die eine Range-Image-basierte Interpolationsmethode zur Punktverdichtung niedrigauflösender LiDAR-Daten vorstellen. Dabei wird die dreidimensionale Punktwolke zunächst in ein zweidimensionales Range Image mit den vier Kanälen (x, y, z , Intensität) überführt. Leere Rasterpositionen ohne gültigen LiDAR-Messpunkt werden anschließend anhand von sechs benachbarten Pixeln $P_1, \dots, P_6$ interpoliert.

Der interpolierte Wert P' ergibt sich aus einer gewichteten Kombination der benachbarten Pixel. Dabei nimmt das Gewicht eines Nachbarpixels mit zunehmender Pixeldistanz d_i exponentiell ab, sodass räumlich näher gelegene Messpunkte einen stärkeren Einfluss auf den synthetisch erzeugten Punkt besitzen. Die grundlegende Gewichtung wird durch

$$W_i = e^{-0,5d_i} \quad (4.1)$$

beschrieben. Der interpolierte Pixelwert ergibt sich anschließend zu

$$P' = \frac{\sum_{i=1}^6 W_i P_i}{\sum_{i=1}^6 W_i}. \quad (4.2)$$

Abbildung 4.3 veranschaulicht die verwendete Nachbarschaft und die relativen Pixelabstände zur zu interpolierenden Rasterposition.

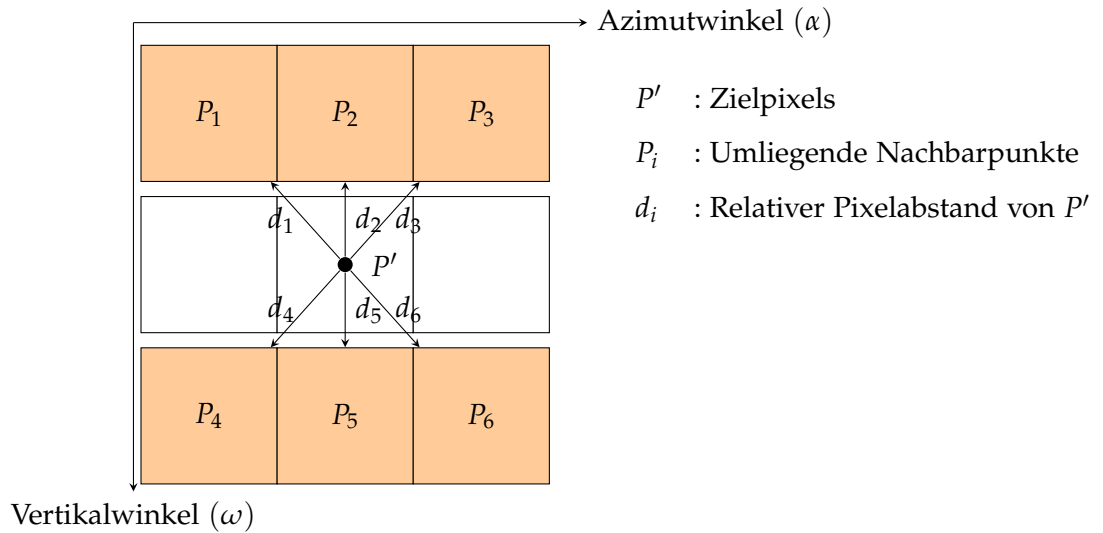


Abbildung 4.3: Interpolation durch gewichtete Summe der Nachbarpixel. Das leere Gitter repräsentiert freie Bereiche ohne LiDAR-Punkte.

Ungültige Nachbarpixel, beispielsweise Rasterpositionen ohne gültigen Entfernungswert oder mit einem maximalen Messwert, können bei einer herkömmlichen Interpolation zu fehlerhaften synthetischen Punkten führen. Der Ansatz nach [43] berücksichtigt daher zusätzlich einen Gültigkeitskoeffizienten s_i . Für gültige Nachbarpixel gilt $s_i = 1$, während ungültige Messwerte mit $s_i = 0$ aus der Berechnung ausgeschlossen werden.

Darüber hinaus wird die Gewichtung um die relative Entfernung der Nachbarpixel erweitert. Nachbarpunkte mit ähnlichen Entfernungswerten werden stärker gewichtet, da sie mit höherer Wahrscheinlichkeit derselben Objektoberfläche angehören. Die modifizierte Interpolation wird durch

$$P' = \frac{\sum_{i=1}^6 s_i W_i P_i}{\sum_{i=1}^6 s_i W_i} \quad (4.3)$$

mit

$$W_i = e^{-0,5d_i} \left(\frac{2}{1 + e^{(R_i - R_{\min})}} \right) \quad (4.4)$$

beschrieben. Dabei bezeichnet R_i den Entfernungswert des jeweiligen Nachbarpixels und $R_{\min}$ den kleinsten Entfernungswert innerhalb der betrachteten Nachbarschaft. Durch diese kombinierte Gewichtung werden Punkte mit großer räumlicher oder tiefenbezogener Abweichung abgeschwächt, wodurch die Entstehung geometrisch unplausibler Punkte insbesondere an Objektgrenzen reduziert wird.

Auf dem KITTI-Datensatz erzielte die in [43] vorgestellte Methode in Kombination mit PointPillars Verbesserungen der mittleren durchschnittlichen Präzision für Fußgänger, Fahrradfahrer und Fahrzeuge gegenüber der niedrigauflösenden LiDAR-Baseline. Diese Ergebnisse zeigen, dass eine vorverarbeitungs-basierte Punktverdichtung grundsätzlich zur Verbesserung der Detektion kleiner und punktarmer Objekte beitragen kann.

4.2.2.1 Methodik

Der in dieser Arbeit implementierte SR-Ansatz orientiert sich an dem zuvor beschriebenen Prinzip der Range-Image-basierten Punktverdichtung. Die LiDAR -Punktwolke wird zunächst anhand der bekannten Azimut- und Elevationswinkel des Sensors in eine zweidimensionale Range-Image- Repräsentation überführt. Jeder gültige Pixel enthält die kartesischen Koordinaten (x, y, z) sowie den zugehörigen Intensitätswert.

Die Punktverdichtung erfolgt vor der Objektdetektion und unabhängig von vorhergesagten Bounding-Boxes oder Ground-Truth-Annotationen. Der für die Interpolation betrachtete FOKUS-Bereich wird ausschließlich anhand des direkten Abstands zwischen Sensor und erfasstem Objektpunkt definiert und umfasst Messpunkte zwischen 10m und 15m. Diese Einschränkung wurde gewählt, da die Baseline in diesem Entfernungsbereich einen deutlichen Rückgang der Detektionsleistung aufweist, gleichzeitig jedoch noch genügend Messpunkte für eine sinnvolle Interpolation vorhanden sind.

Innerhalb des FOKUS-Bereichs werden leere Rasterpositionen in Elevationsrichtung anhand ihrer gültigen Nachbarpixel interpoliert. Dabei werden sowohl die relative Pixeldistanz als auch die Differenz der Entfernungswerte berücksichtigt. Ungültige Messwerte werden von der Interpolation ausgeschlossen. Die interpolierten Werte der vier Kanäle (x, y, z , Intensität) werden anschließend als zusätzliche dreidimensionale Punkte in die ursprüngliche Punktwolke zurückgeführt.

Das Verfahren wird sowohl während des Trainings als auch während der Inferenz eingesetzt:

1. **Training:** Die Punktverdichtung wird als Bestandteil der Datenvorverarbeitung auf die Trainingsframes angewendet. Dadurch wird VoxelNeXt unmittelbar auf den angereicherten Punktwolken feinjustiert und kann Merkmale aus den zusätzlich interpolierten Punkten erlernen.
2. **Inferenz:** Zur Laufzeit wird dieselbe Vorverarbeitung auf bislang ungesehene LiDAR-Frames angewendet. Die Punktwolke wird zunächst in das Range Image überführt, im Bereich von 10m bis 15m interpoliert und anschließend wieder in eine dreidimensionale Punktwolke transformiert. Erst danach wird die angereicherte Punktwolke an VoxelNeXt übergeben und die Detektion der Verkehrshütchen durchgeführt.

Da die Punktverdichtung ausschließlich als Vorverarbeitungsschritt realisiert wird, sind keine Änderungen am Backbone oder am Detektionskopf von VoxelNeXt erforderlich. Der zusätzliche Rechenaufwand entsteht somit nur durch die Transformation in das Range Image, die Interpolation und die Rücktransformation in die dreidimensionale Punktwolke.

4.2.2.2 Ergebnisse

Tabelle 4.2 zeigt den Vergleich der Detektionsleistung von VoxelNeXt mit und ohne Super-Resolution im FOKUS-Bereich (10–15 m). Abbildung 4.4 veranschaulicht die Verbesserung des Recalls pro Meter.

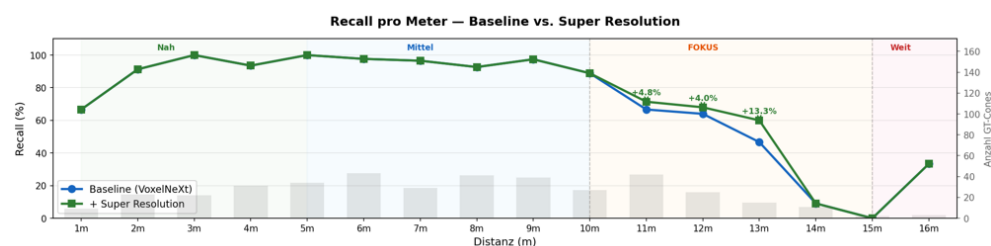


Abbildung 4.4: Recall pro Meter, Baseline vs. Super-Resolution für VoxelNeXt (Klasse `small_cone`). Im FOKUS-Bereich (10–15 m) erzielt Super Resolution die größten Verbesserungen, insbesondere bei 13 m mit +13,3 %. Die grauen Balken zeigen die Anzahl der Ground-Truth-Kegel pro Distanzbereich (rechte Achse).

Tabelle 4.2: Vergleich Baseline vs. Super-Resolution für VoxelNeXt im FOKUS-Bereich.

Methode	Recall 10–15 m	Delta	FP	F1 gesamt	Avg. Latenz	FPS
Baseline	63,3 %	–	22	86,8 %	18,2 ms	55,0
Super Resolution	67,5 %	+4,2 %	21	~87 %	26,42 ms	37,9

Anmerkung: FP = False Positives, F1 gesamt bezieht sich auf alle Distanzbereiche.

Super-Resolution erzielt im FOKUS-Bereich eine Verbesserung des Recalls von 63,3 % auf 67,5 % (+4,2 %), bei gleichzeitig stabiler Anzahl an Fehldetektionen (FP: 22 → 21). Der F1-Score gesamt bleibt mit ~87 % nahezu unverändert, was zeigt, dass die Verbesserung im FOKUS-Bereich nicht auf Kosten der Gesamtleistung erzielt wird. Allerdings steigt die mittlere Latenz von 18,2 ms auf 26,42 ms, was einem Rückgang der Verarbeitungsgeschwindigkeit von 55,0 auf 37,9 FPS entspricht. Da VoxelNeXt damit jedoch weiterhin deutlich über der Mindestanforderung von 10 FPS liegt, bleibt die Echtzeitfähigkeit des Systems gewährleistet.

4.2.2.3 Diskussion und Limitationen

Die Ergebnisse zeigen, dass SR im Bereich 11–13 m die größte Wirkung entfaltet, wo die Punktdichte mit 2,3–3,6 Punkten pro Kegel gerade noch ausreicht, um eine sinnvolle Interpolation durchzuführen:

- Bei 11 m: +4,8 % Recall (3,6 Punkte/Kegel)
- Bei 12 m: +4,0 % Recall (3,0 Punkte/Kegel)
- Bei 13 m: +13,3 % Recall (2,3 Punkte/Kegel)

Jenseits von 14 m hingegen ist die Ausgangspunktdichte mit weniger als 2 Punkten pro Kegel zu gering, sodass SR keine ausreichende geometrische Grundlage für eine sinnvolle Interpolation findet. Dies deckt sich mit der Beobachtung von [43], dass Interpolationsverfahren eine Mindestanzahl an Nachbarpunkten benötigen, um die Objektform zuverlässig zu rekonstruieren. Diese physikalische Grenze verdeutlicht: Super-Resolution kann vorhandene Information verdichten, jedoch keine Information erzeugen, die im Originalsignal nicht vorhanden ist.

Die Ergebnisse aus Kapitel 3 und die untersuchte Optimierungsansätze aus Kapitel 4 zeigen, dass die Detektion kleiner, punktarmer Objekte in LiDAR-Punktwolken nicht allein von der Modellgenauigkeit abhängt, sondern wesentlich durch die gewählte Punktwolkenrepräsentation, die Punktdichte, die Modellkomplexität und die Laufzeiteffizienz bestimmt wird. Insbesondere im Kontext des FASTDA-Projekts ist daher nicht das isoliert genaueste Modell entscheidend, sondern diejenige Architektur, die eine robuste Erkennung kleiner Verkehrshütchen mit den Echtzeitanforderungen eines ressourcenbeschränkten On-Board-Systems verbindet.

In diesem Kapitel werden die erzielten Ergebnisse daher nicht erneut einzeln dargestellt, sondern architekturenspezifisch eingeordnet und kritisch diskutiert. Der Fokus liegt auf der Frage, warum bestimmte Modellklassen im vorliegenden Anwendungsfall besser geeignet sind als andere und welche Entwurfparameter, insbesondere Voxelgröße, Downsampling-Strategien, Punktdichte und Inferenzzeit, die Detektionsleistung maßgeblich beeinflussen. Abschließend wird auf dieser Grundlage eine datenbasierte Empfehlung für den praktischen Einsatz im fastda-Projekt abgeleitet und deren Gültigkeitsbereich kritisch reflektiert.

5.1 ARCHITEKTURSPECIFISCHE ERKENNTNISSE

Aus den Ergebnissen lassen sich folgende architekturübergreifende Erkenntnisse ableiten:

- **Voxelbasierte Ansätze (PointPillars, VoxelNeXt):** Die Voxelgröße ist ein kritischer Entwurfparameter. PointPillars verwendet Pillars mit einer Auflösung von 0,16 m, was für Verkehrshütchen mit einem Durchmesser von ca. 0,23 m zu grob ist. VoxelNeXt hingegen nutzt eine Voxelgröße von 0,075 m und verarbeitet Features durchgehend sparse, wodurch geometrische Details erhalten bleiben. Dies erklärt den deutlichen Leistungsunterschied zwischen den beiden voxelbasierten Ansätzen. Eine Verringerung der Pillargröße könnte die Erkennungsleistung von PointPillars zwar grundsätzlich verbessern, würde jedoch zu einer deutlich höheren räumlichen Auflösung der Pillarrepräsentation führen und damit den Rechen- und Speicheraufwand steigern. VoxelNeXt kann trotz seiner feineren Voxelauflösung eine hohe Effizienz aufrechterhalten, da die Architektur vollständig auf Sparse Convolutions basiert und ausschließlich tatsächlich belegte Voxel verarbeitet. Zudem vermeidet das Modell die in vielen voxelbasierten Verfahren notwendige Sparse-to-Dense-Konvertierung, wodurch zusätzliche Rechenkosten reduziert werden [14, 23].

- **Punktbasierende Ansätze (3DSSD):** Das aggressive Downsampling der typischerweise 60.000–80.000 LiDAR-Punkte pro Frame auf 24.000 Punkte führt gegenüber der 48k-Konfiguration zu einer verbesserten Detektionsleistung. Dies deutet darauf hin, dass eine höhere Punktdichte nicht zwangsläufig zu einer besseren Detektionsleistung führt. Dennoch bleibt 3DSSD mit einem AP@0,5 von 50,7 %, und einem F1-Score von 71,2 %, deutlich hinter VoxelNeXt zurück. Eine mögliche Ursache liegt darin, dass die wiederholten Sampling-Schritte punktbasierter Verfahren bei kleinen Objekten einen Teil der begrenzten geometrischen Informationen entfernen [15].
- **Hybride Ansätze (PV-RCNN):** Die Kombination aus Voxel- und Punktmerkmalen führt nicht automatisch zu einer verbesserten Erkennung kleiner Objekte. PV-RCNN erzeugt mit 193 False Positives die höchste Fehldetektionsrate aller untersuchten Modelle. Die Ergebnisse zeigen, dass die in PV-RCNN eingesetzte hybride Architektur im vorliegenden Anwendungsfall keine Vorteile gegenüber sparsamen voxelbasierten Ansätzen bietet. Eine mögliche Ursache liegt darin, dass die mehrstufige Merkmalsextraktion und Fusion bei kleinen, punktarmeren Objekten nicht genügend zusätzliche Informationen bereitstellt, um eine zuverlässige Detektion dieser Objekte zu gewährleisten [39].
- **Aufmerksamkeitsbasierte Ansätze (DeLiVoTr):** Der Verzicht auf Downsampling ermöglicht eine herausragende Precision von 93,6 %, geht jedoch mit einem inakzeptablen Rechenaufwand von 33 GB VRAM und 213 ms Latenz einher. Die quadratische Komplexität $\mathcal{O}(k^2)$ des Selbstaufmerksamkeitsmechanismus ist der limitierende Faktor.
- **Sequenzielle Ansätze (LION):** Durch den Mamba-Operator skaliert LION linear mit der Anzahl der Voxel $\mathcal{O}(k)$ und erreicht dabei die höchste Detektionsleistung aller Modelle. Der Engpass liegt in der Laufzeit von 185 ms, was dieses Modell für Echtzeitanwendungen ungeeignet macht.

5.2 EMPFEHLUNG FÜR DAS FASTDA-PROJEKT UND KRITISCHE REFLEXION

Auf Basis der in Abschnitt 3.8 präsentierten Ergebnisse, der distanzabhängigen Fehleranalyse sowie der in Abschnitt 4.2 untersuchten Punktverdichtungsansätze ergibt sich eine klare Empfehlung für das FASTDA-Projekt: Die vollständig sparsame Modellarchitektur VoxelNeXt ist das geeignetste Modell für den Einsatz im FASTDA-Projekt, da sie unter allen untersuchten Modellen den besten Kompromiss bietet. Zwar erreicht VoxelNeXt nicht die höchste Detektionsleistung, doch das Zusammenspiel aus niedriger Latenz und guter Detektionsleistung macht es für dieses Projekt am besten geeignet.

Im praxisrelevanten Nahbereich bis 10 m liegt die Erkennungsleistung deutlich über der definierten Mindestanforderung (Recall über 90 %, vgl. Abbildung 3.8). Bei den im FASTDA-Projekt typischen Geschwindigkeiten

von 20 bis 70 km/h ist dieser Erkennungsbereich für einen sicheren Betrieb tendenziell ausreichend, wobei die verfügbare Vorwarnzeit bei höheren Geschwindigkeiten entsprechend abnimmt. Mit zunehmender Distanz nimmt die Erkennungsleistung jedoch erheblich ab, was bei höheren Geschwindigkeiten zu einem kritischen Sicherheitsproblem werden kann. Eine Verbesserung der Erkennungsreichweite sollte daher als Ziel für zukünftige Arbeiten verfolgt werden. Der geringe VRAM-Bedarf und die vollständig sparsame Verarbeitungsarchitektur machen VoxelNeXt zudem für eingebettete GPU-Systeme geeignet.

Für zukünftige Arbeiten, in denen die Echtzeitfähigkeit durch Hardware-Optimierungen wie Quantisierung [44] oder TensorRT-Beschleunigung [45] verbessert werden könnte, LION aufgrund seiner überlegenen Detektionsleistung und seines hohen Recalls im Nahbereich eine vielversprechende Alternative dar. Allerdings muss die aktuelle Latenz vor einem Praxiseinsatz durch geeignete Optimierungsmaßnahmen deutlich reduziert werden.

Diese Empfehlung ist jedoch im Lichte der folgenden methodischen Einschränkungen zu interpretieren.

1. **Begrenzte Datenbasis.** Der eigene Datensatz mit 169 Frames und 1.642 annotierten Instanzen der Klasse `small_cone` ist relativ klein, sodass die Aussagekraft der Ergebnisse auf die spezifischen Aufnahmebedingungen und die verwendete Sensorhöhe beschränkt bleibt. Eine Generalisierung auf andere Streckenbedingungen, Wetterverhältnisse oder Sensorpositionen erfordert eine Erweiterung des Datensatzes. Darüber hinaus ist eine aussagekräftige Klassifikation zwischen `small_cone` und `large_cone` aufgrund der geringen Anzahl an `large_cone`-Instanzen nicht möglich, weshalb diese Klasse von der quantitativen Analyse ausgeschlossen werden musste.
2. **Eingeschränkte Übertragbarkeit der Optimierungsansätze.** Die unzureichende Datenbasis begrenzt auch die Übertragbarkeit der untersuchten Optimierungsansätze: Der Super-Resolution-Ansatz ist in seiner aktuellen Form auf den FOKUS-Bereich (10–15 m) beschränkt und setzt eine Mindestpunktdichte von ca. zwei Punkten pro Objekt voraus, sodass eine Übertragung auf den Fernbereich jenseits von 15 m aufgrund der physikalischen Grenzen der Interpolationsmethode ohne zusätzliche Sensorik oder datenbasierte Verdichtungsverfahren nicht möglich ist. Der zuvor diskutierte Shape-Prior-Ansatz würde eine noch deutlich größere Anzahl annotierter Frames voraussetzen und wurde aus diesem Grund nicht weiterverfolgt.
3. **Fehlende Validierung auf der Zielhardware.** Alle Messungen wurden auf einem stationären Linux-System mit einer NVIDIA RTX 6000 Ada GPU durchgeführt. Die tatsächliche Inferenzzeit auf dem ressourcenbeschränkten On-Board-System des FASTDA-Rennfahrzeugs kann erheblich abweichen und muss in einem nächsten Schritt direkt auf der Zielhardware validiert werden.

FAZIT

Die systematische Analyse verschiedener 3D-Deep-Learning-Modelle zeigt, dass die zuverlässige Erkennung kleiner, punktarmer Objekte in LiDAR-Punktwolken unter Echtzeitbedingungen grundsätzlich realisierbar ist. Die Detektionsleistung hängt jedoch maßgeblich von mehreren Faktoren ab, insbesondere von der gewählten Repräsentationsstrategie der Punktwolke, der Entfernung zum Sensor sowie der zugrunde liegenden Modellarchitektur.

Ziel dieser Arbeit war es, den besten Kompromiss zwischen der Detektionsgenauigkeit kleiner Objekte, insbesondere von Verkehrshütchen, und den Anforderungen an die Echtzeitfähigkeit zu identifizieren. Die Ergebnisse zeigen, dass VoxelNeXt als einziges untersuchtes Modell beide Anforderungen erfüllt. Es ermöglicht eine zuverlässige Erkennung kleiner Objekte und erreicht gleichzeitig mit einer Inferenzzeit von lediglich 18.2 ms pro Frame Echtzeitfähigkeit.

Eine zentrale Limitation besteht jedoch weiterhin für alle untersuchten 3D-Modelle: Die Erkennungsleistung nimmt ab einer Entfernung von etwa 10 Metern deutlich ab. Trotz der implementierten Datenverdichtung durch Super Resolution bleibt die geringe Punktdichte entfernter LiDAR-Messpunkte eine grundlegende Herausforderung für die zuverlässige Detektion kleiner Objekte.

Auf Grundlage der in Kapitel 3 erzielten Ergebnisse lassen sich die in Abschnitt 1.2 formulierten Forschungsfragen wie folgt beantworten:

Forschungsfrage 1: Wie zuverlässig können verschiedene 3D-Deep-Learning Ansätze kleine, punktarmer Objekte wie Verkehrshütchen in LiDAR-Punktwolken unter Echtzeitbedingungen detektieren?

Die Ergebnisse dieser Arbeit zeigen, dass moderne 3D-Deep-Learning-Ansätze grundsätzlich in der Lage sind, kleine und punktarmer Objekte wie Verkehrshütchen zuverlässig zu detektieren. Die Detektionsleistung hängt jedoch sowohl von der gewählten Modellarchitektur als auch von der verfügbaren Punktdichte der LiDAR-Daten ab. Während klassische voxel- und punktbasierte Verfahren wie PointPillars, 3DSSD und PV-RCNN Recall-Werte zwischen 58,8 % und 73,9 % erreichen, erzielen die neueren Architekturen LION und VoxelNeXt mit jeweils 84,6 % Recall die besten Ergebnisse. Dies spiegelt sich ebenfalls in den höchsten F1-Scores von 88,2 % beziehungsweise 86,8 % wider.

Die Analyse der Distanzbereiche verdeutlicht jedoch, dass die Detektionszuverlässigkeit nicht ausschließlich von der Modellarchitektur bestimmt wird. Mit zunehmender Entfernung sinkt die Anzahl der verfügbaren LiDAR-Punkte pro Verkehrshütchen deutlich. Im Nahbereich bis etwa 10 m erreichen die leistungsstärksten Modelle Recall-Werte von über 90 % und ermöglichen damit eine zuverlässige Objekterkennung. Ab einer Entfernung von

etwa 10 m nimmt die Erkennungsleistung jedoch merklich ab. Im Bereich von 10–15 m sinken die Recall-Werte auf rund 60 %, während jenseits von 15 m teilweise nur noch wenige Messpunkte pro Objekt zur Verfügung stehen, wodurch die Detektionsleistung aller Modelle deutlich einbricht.

Daraus lässt sich ableiten, dass die Zuverlässigkeit der Erkennung kleiner Objekte nicht allein durch die Wahl des Detektionsmodells begrenzt wird, sondern maßgeblich von den physikalischen Eigenschaften der LiDAR-Sensorik abhängt. Selbst leistungsfähige Architekturen können fehlende Informationen nicht vollständig kompensieren, wenn ein Objekt nur noch durch wenige Messpunkte repräsentiert wird. Die zentrale Herausforderung liegt daher nicht ausschließlich in der Entwicklung leistungsfähigerer Detektoren, sondern ebenso in der Bereitstellung einer ausreichend informationsreichen Punktwolke.

Zusammenfassend kann die Forschungsfrage dahingehend beantwortet werden, dass moderne 3D-Deep-Learning-Modelle kleine, punktarme Objekte unter Echtzeitbedingungen zuverlässig erkennen können, sofern eine ausreichende Punktdichte vorhanden ist. Die größte Einschränkung entsteht im mittleren und größeren Entfernungsbereich, in dem die sensorbedingt geringe Punktdichte die Detektionsleistung aller untersuchten Ansätze deutlich reduziert.

Forschungsfrage 2: Welches Verhältnis zwischen Erkennungsleistung und Rechenaufwand weisen diese Modelle beim Einsatz auf ressourcenbeschränkten On-Board-Systemen autonomer Rennfahrzeuge auf?

Die Gegenüberstellung der Detektions- und Laufzeitmetriken in Tabelle 3.15 zeigt, dass zwischen Detektionsqualität und Laufzeiteffizienz kein prinzipieller Widerspruch bestehen muss. DeLiVoTr und LION erreichen zwar die Spitzenwerte in der Erkennung, fallen mit Latenzen von 213 ms bzw. 185 ms jedoch durch das vorgegebene Echtzeitlimit von 100 ms. Zudem ist DeLiVoTr mit einem VRAM-Bedarf von 33 GB für eingebettete Systeme ungeeignet. Demgegenüber erreicht VoxelNeXt mit einer Inferenzrate von 55 FPS und einem Speicherbedarf von lediglich 2,5 GB die höchste Recheneffizienz aller untersuchten Modelle. Gleichzeitig bleibt die Detektionsleistung auf einem hohen Niveau. Dieser Vorteil ergibt sich aus der ausschließlichen Verarbeitung belegter Voxel, wodurch die Rechenlast linear mit der Anzahl relevanter Punkte skaliert.

Die zentrale Erkenntnis für diese Forschungsfrage lautet: Nicht die Detektionsgenauigkeit an sich, sondern die Repräsentationsstrategie entscheidet über die Echtzeitfähigkeit eines Modells. Während effiziente voxelbasierte Verfahren wie VoxelNeXt eine hohe Erkennungsleistung mit geringer Rechenlast kombinieren, erkaufen aufmerksamkeits- und sequenzbasierte Ansätze ihre Genauigkeitsvorteile durch einen deutlich höheren Rechenaufwand. Dadurch überschreiten sie die verfügbaren Echtzeit- und Speicherbudgets ressourcenbeschränkter On-Board-Systeme und sind ohne zusätzliche Optimierungsmaßnahmen wie Quantisierung oder TensorRT-Beschleunigung praktisch nicht einsetzbar.

ZUSAMMENFASSUNG UND AUSBLICK

Die vorliegende Arbeit untersuchte systematisch sechs 3D-Deep-Learning-Architekturen zur echtzeitfähigen Detektion kleiner, punktarmer Objekte in LiDAR-Punktwolken auf ressourcenbeschränkten On-Board-Systemen. Die zentrale Herausforderung, die mit zunehmender Distanz stark abnehmende Punktdichte, wurde sowohl modellübergreifend analysiert als auch durch einen gezielten Optimierungsansatz adressiert.

Die Ergebnisse zeigen, dass die Detektionsleistung primär von der architekturbedingten Erhaltung geometrischer Information abhängt: vollständig sparsame und sequenzielle Ansätze (VoxelNeXt, LION) erzielen die höchste Detektionsqualität, da sie Quantisierungsverluste klassischer Voxel- oder Projektionsverfahren vermeiden. Beide Modelle erreichen im Nahbereich bis 10 m Recall-Werte über 92 %, im Bereich ab 10 m bricht die Leistung jedoch deutlich ein. Der Super-Resolution-Ansatz konnte diese Limitation im FOKUS-Bereich (10–15 m) teilweise kompensieren, während die Integration von Shape Priors aufgrund fehlender Sequenzdaten und begrenzter Datenbasis verworfen wurde. Eine konkrete Empfehlung für den Einsatz im FASTDA-Projekt wurde in Abschnitt 5.2 formuliert.

Über die unmittelbare Anwendung hinaus eröffnen die gewonnenen Erkenntnisse mehrere grundlegende Forschungsrichtungen, die zu einer allgemeinen Verbesserung der Detektion kleiner, punktarmer Objekte in LiDAR-Daten beitragen könnten.

Lernbasierte Punktwolken-Vervollständigung: Der in dieser Arbeit verfolgte Super-Resolution-Ansatz basiert auf einer geometrischen Interpolation und ist durch die verfügbare Mindestpunktdichte limitiert. Generative Modelle etwa Diffusionsmodelle oder Generative Adversarial Network (GAN)-basierte Punktwolken-Vervollständigung könnten direkt auf Objektebene trainiert werden, um aus wenigen Punkten plausible vollständige Objektgeometrien zu rekonstruieren [46]. Im Gegensatz zur reinen Interpolation könnten solche Modelle gelerntes Formwissen nutzen, um auch jenseits von 15 m noch sinnvolle Repräsentationen zu erzeugen.

Architektonische Weiterentwicklung sparsamer Modelle. Die Ergebnisse dieser Arbeit sowie aktuelle Forschungsentwicklungen deuten darauf hin, dass vollständig sparsame Detektionsarchitekturen einen vielversprechenden Ansatz zur effizienten Verarbeitung von LiDAR-Punktwolken darstellen [16]. Gleichzeitig eröffnen Mamba-basierte State-Space-Modelle neue Möglichkeiten zur Modellierung globaler Kontextinformationen bei linearer Skalierung der Rechenkomplexität [47]. Zukünftige Arbeiten könnten daher untersuchen, inwieweit sich beide Konzepte innerhalb einer gemeinsamen Architektur kombinieren lassen. Denkbar wäre beispielsweise ein vollständig sparsames Backbone, das durch ein leichtgewichtiges Mamba-basiertes Mo-

dul ergänzt wird und ausschließlich in Bereichen geringer Punktdichte eingesetzt wird. Eine solche adaptive Architektur könnte die hohe Effizienz sparsamer Detektoren bewahren und gleichzeitig die Merkmalsextraktion für weit entfernte oder nur schwach abgetastete Objekte verbessern.

LiDAR-Kamera-Fusion. Eine alternative Strategie zur Verbesserung der Detektion weit entfernter Verkehrshütchen besteht in der Fusion von LiDAR- und Kameradaten. Während die Anzahl der LiDAR-Messpunkte pro Objekt mit zunehmender Entfernung stark abnimmt und dadurch die Extraktion diskriminativer Merkmale erschwert wird, liefern Kameras auch in größeren Distanzen weiterhin dichte Bildinformationen mit zusätzlichen Farb-, Textur- und Konturmerkmalen [48].

Benchmark-Erweiterung für punktarmer Objekte. Schließlich zeigt diese Arbeit, dass etablierte Benchmarks wie nuScenes oder KITTI die Herausforderungen extrem punktarmer Objekte (< 5 Punkte) nur unzureichend abbilden. Die Schaffung eines öffentlich verfügbaren Benchmark-Datensatzes, der gezielt den Distanzbereich kritischer Punktarmut adressiert, könnte der Forschungsgemeinschaft eine einheitliche Grundlage bieten, um Fortschritte in diesem spezifischen, sicherheitsrelevanten Teilproblem der 3D- Objekterkennung vergleichbar zu machen.

Diese Richtungen verdeutlichen, dass die Verbesserung der Detektion kleiner, punktarmer Objekte nicht allein durch inkrementelle Modelloptimierung, sondern durch ein Zusammenspiel aus Datenstrategie, Architekturdisein und Trainingsmethodik erreicht werden kann.

LITERATUR

- [1] Jiapei Wei, Azizan As'array, Khairil Anas Md Rezali, Mohd Zuhri Mohamed Yusoff, Haohao Ma und Kunlun Zhang. "A Review of YOLO Algorithm and Its Applications in Autonomous Driving Object Detection". In: *IEEE Access* 13 (2025), S. 93688–93711. DOI: [10.1109/ACCESS.2025.3573376](https://doi.org/10.1109/ACCESS.2025.3573376).
- [2] Manikanta Kotthapalli, Deepika Ravipati und Reshma Bhatia. "YOLOv1 to YOLOv11: A Comprehensive Survey of Real-Time Object Detection Innovations and Challenges". In: *arXiv preprint arXiv:2508.02067* (2025). DOI: [10.48550/arXiv.2508.02067](https://doi.org/10.48550/arXiv.2508.02067).
- [3] Ke Wang, Tianqiang Zhou, Xingcan Li und Fan Ren. "Performance and Challenges of 3D Object Detection Methods in Complex Scenes for Autonomous Driving". In: *IEEE Transactions on Intelligent Vehicles* 8.2 (2023), S. 1699–1716. DOI: [10.1109/TIV.2022.3213796](https://doi.org/10.1109/TIV.2022.3213796).
- [4] Ying Li, Lingfei Ma, Zilong Zhong, Fei Liu, Michael A. Chapman, Dongpu Cao und Jonathan Li. "Deep Learning for LiDAR Point Clouds in Autonomous Driving: A Review". In: *IEEE Transactions on Neural Networks and Learning Systems* 32.8 (2021), S. 3412–3432. DOI: [10.1109/TNNLS.2020.3015992](https://doi.org/10.1109/TNNLS.2020.3015992).
- [5] Nang Htet Htet Aung, Paramin Sangwongngam, Rungroj Jintamethasawat, Shashi Shah und Lunchakorn Wuttisittikulkij. "A Review of LiDAR-Based 3D Object Detection via Deep Learning Approaches Towards Robust Connected and Autonomous Vehicles". In: *IEEE Transactions on Intelligent Vehicles* 10.1 (2025), S. 526–547. DOI: [10.1109/TIV.2024.3415771](https://doi.org/10.1109/TIV.2024.3415771).
- [6] Noor Ul Ain Tahir, Zuping Zhang, Muhammad Asim, Junhong Chen und Mohammed ELAffendi. "Object Detection in Autonomous Vehicles under Adverse Weather: A Review of Traditional and Deep Learning Approaches". In: *Algorithms* 17.3 (2024). ISSN: 1999-4893. DOI: [10.3390/a17030103](https://doi.org/10.3390/a17030103). URL: <https://www.mdpi.com/1999-4893/17/3/103>.
- [7] You Li und Javier Ibanez-Guzman. "Lidar for Autonomous Driving: The Principles, Challenges, and Trends for Automotive Lidar and Perception Systems". In: *IEEE Signal Processing Magazine* 37.4 (2020), S. 50–61. DOI: [10.1109/MSP.2020.2973615](https://doi.org/10.1109/MSP.2020.2973615).
- [8] Peide Wang. "Research on Comparison of LiDAR and Camera in Autonomous Driving". In: *Journal of Physics: Conference Series* 2093.1 (Nov. 2021), S. 012032. DOI: [10.1088/1742-6596/2093/1/012032](https://doi.org/10.1088/1742-6596/2093/1/012032). URL: <https://doi.org/10.1088/1742-6596/2093/1/012032>.
- [9] IBM. *How LiDAR Works*. <https://www.ibm.com/topics/lidar>. 2023.

- [10] Blickfeld GmbH. *LiDAR Technologie – Funktionsweise und Anwendungen*. <https://www.blickfeld.com/de/lidar-technologie>. Abgerufen am 20. März 2026. 2023.
- [11] Georgios Zamanakos, Lazaros Tsochatzidis, Angelos Amanatiadis und Ioannis Pratikakis. “A comprehensive survey of LIDAR-based 3D object detection methods with deep learning for autonomous driving”. In: *Computers Graphics* 99 (2021), S. 153–181. ISSN: 0097-8493. DOI: <https://doi.org/10.1016/j.cag.2021.07.003>. URL: <https://www.sciencedirect.com/science/article/pii/S0097849321001321>.
- [12] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan und Oscar Beijbom. “nuScenes: A Multimodal Dataset for Autonomous Driving”. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, S. 11618–11628. DOI: [10.1109/CVPR42600.2020.01164](https://doi.org/10.1109/CVPR42600.2020.01164).
- [13] Andreas Geiger, Philip Lenz, Christoph Stiller und Raquel Urtasun. “Vision meets robotics: The kitti dataset”. In: *The international journal of robotics research* 32.11 (2013), S. 1231–1237.
- [14] Alex H. Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang und Oscar Beijbom. “PointPillars: Fast Encoders for Object Detection From Point Clouds”. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, S. 12689–12697. DOI: [10.1109/CVPR.2019.01298](https://doi.org/10.1109/CVPR.2019.01298).
- [15] Zetong Yang, Yanan Sun, Shu Liu und Jiaya Jia. “3DSSD: Point-Based 3D Single Stage Object Detector”. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, S. 11037–11045. DOI: [10.1109/CVPR42600.2020.01105](https://doi.org/10.1109/CVPR42600.2020.01105).
- [16] Yukang Chen, Jianhui Liu, Xiangyu Zhang, Xiaojuan Qi und Jiaya Jia. “VoxelNeXt: Fully Sparse VoxelNet for 3D Object Detection and Tracking”. In: *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2023, S. 21674–21683. DOI: [10.1109/CVPR52729.2023.02076](https://doi.org/10.1109/CVPR52729.2023.02076).
- [17] Zhe Liu, Jinghua Hou, Xinyu Wang, Xiaoqing Ye, Jingdong Wang, Hengshuang Zhao und Xiang Bai. “LION: Linear Group RNN for 3D Object Detection in Point Clouds”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2025. URL: <https://happinesslz.github.io/projects/LION>.
- [18] MMDetection3D Contributors. *MMDetection3D: OpenMMLab next-generation platform for general 3D object detection*. <https://github.com/open-mmlab/mmdetection3d>. 2020.
- [19] OpenPCDet Development Team. *OpenPCDet: An Open-source Toolbox for 3D Object Detection from Point Clouds*. <https://github.com/open-mmlab/OpenPCDet>. 2020.

- [20] Yongxin Shao, Aihong Tan, Zhetao Sun, Enhui Zheng, Tianhong Yan und Peng Liao. "PV-SSD: A Multi-Modal Point Cloud 3D Object Detector Based on Projection Features and Voxel Features". In: *IEEE Transactions on Emerging Topics in Computational Intelligence* 8.5 (2024), S. 3436–3449. DOI: [10.1109/TETCI.2024.3389710](https://doi.org/10.1109/TETCI.2024.3389710).
- [21] Pooja Goswami, Ram Vaishnav, Tanishq Anand und Prateek Dayal. "A Comprehensive Review on LiDAR Based 3D Deep Learning Object Detection Algorithms". In: *2025 International Conference on Computer, Electrical Communication Engineering (ICCECE)*. 2025, S. 1–6. DOI: [10.1109/ICCECE61355.2025.10941027](https://doi.org/10.1109/ICCECE61355.2025.10941027).
- [22] Siheng Chen, Baoan Liu, Chen Feng, Carlos Vallespi-Gonzalez und Carl Wellington. "3D Point Cloud Processing and Learning for Autonomous Driving: Impacting Map Creation, Localization, and Perception". In: *IEEE Signal Processing Magazine* 38.1 (2021), S. 68–86. DOI: [10.1109/MSP.2020.2984780](https://doi.org/10.1109/MSP.2020.2984780).
- [23] L. Fan, J. Cao, X. Liu, X. Li, L. Deng, H. Sun und Y. Peng. "Voxel self-attention and center-point for 3D object detector". In: *iScience* 27.9 (2024), S. 110759. DOI: [10.1016/j.isci.2024.110759](https://doi.org/10.1016/j.isci.2024.110759).
- [24] R. Qi Charles, Hao Su, Mo Kaichun und Leonidas J. Guibas. "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, S. 77–85. DOI: [10.1109/CVPR.2017.16](https://doi.org/10.1109/CVPR.2017.16). URL: <https://doi.org/10.1109/CVPR.2017.16>.
- [25] Y. Zhou und O. Tuzel. "VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, S. 4464–4473. DOI: [10.48550/arXiv.1711.06396](https://doi.org/10.48550/arXiv.1711.06396).
- [26] Maosheng Ye, Shuangjie Xu und Tongyi Cao. "HVNet: Hybrid Voxel Network for LiDAR Based 3D Object Detection". In: *arXiv preprint arXiv:2003.00186*. Accepted to CVPR 2020. 2020. DOI: [10.48550/arXiv.2003.00186](https://doi.org/10.48550/arXiv.2003.00186).
- [27] Ultralytics. *Anchor-Based Detectors*. <https://www.ultralytics.com/de/glossary/anchor-based-detectors>.
- [28] Shaoqing Ren, Kaiming He, Ross Girshick und Jian Sun. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.6 (2017), S. 1137–1149. DOI: [10.1109/TPAMI.2016.2577031](https://doi.org/10.1109/TPAMI.2016.2577031).
- [29] Peigeng Li. "A Comparative Analysis of Single-Stage Detectors from the Perspectives of Anchor-Free and Anchor-Based Approaches". In: *Highlights in Science, Engineering and Technology* 72 (Dez. 2023), S. 774–782. DOI: [10.54097/s37sgq58](https://doi.org/10.54097/s37sgq58).

- [30] Yu Wang, Shaohua Wang, Yicheng Li und Mingchun Liu. "Developments in 3-D Object Detection for Autonomous Driving: A Review". In: *IEEE Sensors Journal* 25.12 (2025), S. 21033–21053. DOI: [10.1109/JSEN.2025.3562284](https://doi.org/10.1109/JSEN.2025.3562284).
- [31] Andreas Geiger, Philip Lenz und Raquel Urtasun. "Are we ready for autonomous driving? The KITTI vision benchmark suite". In: *2012 IEEE Conference on Computer Vision and Pattern Recognition*. 2012, S. 3354–3361. DOI: [10.1109/CVPR.2012.6248074](https://doi.org/10.1109/CVPR.2012.6248074).
- [32] Rafael Padilla, Sergio L. Netto und Eduardo A. B. da Silva. "A Survey on Performance Metrics for Object-Detection Algorithms". In: *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*. 2020, S. 237–242. DOI: [10.1109/IWSSIP48289.2020.9145130](https://doi.org/10.1109/IWSSIP48289.2020.9145130).
- [33] Sazan Ali Kamal Mohammed, Mohd Zulhakimi Ab Razak, Abdul Hadi Abd Rahman und Maria Abu Bakar. "An Efficient Intersection Over Union Algorithm for 3D Object Detection". In: *IEEE Access* 12 (2024), S. 169768–169786. DOI: [10.1109/ACCESS.2024.3495761](https://doi.org/10.1109/ACCESS.2024.3495761).
- [34] J. A. Hanley und B. J. McNeil. "The meaning and use of the area under a receiver operating characteristic (ROC) curve". In: *Radiology* 143.1 (1982), S. 29–36. DOI: [10.1148/radiology.143.1.7063747](https://doi.org/10.1148/radiology.143.1.7063747).
- [35] et al. Suo. "Characterizing and Optimizing the Tail Latency for Safe Level-4 Autonomous Vehicle Systems". In: *arXiv preprint* (2023). DOI: [10.48550/arXiv.2305.07147](https://doi.org/10.48550/arXiv.2305.07147).
- [36] Ouster. *OS2 Rev 7 Specifications*. <https://store.clearpathrobotics.com/products/os2-rev-7>. Rotation Rate: 10 or 20 Hz. 2023.
- [37] Leonardo Christino und Fernando Osório. "GPU-Services: GPU Based Real-Time Processing of 3D Point Clouds Applied to Robotic Systems and Intelligent Vehicles". In: *Robotics*. Hrsg. von Fernando Santos Osório und Rogério Sales Gonçalves. Cham: Springer International Publishing, 2016, S. 152–171. ISBN: 978-3-319-47247-8. DOI: [10.1007/978-3-319-47247-8_10](https://doi.org/10.1007/978-3-319-47247-8_10).
- [38] Gopi Krishna Erabati und Helder Araujo. "DeLiVoTr: Deep and Lightweight Voxel Transformer for 3D Object Detection". In: *Intelligent Systems with Applications* 22 (2024), S. 200361. DOI: [10.1016/j.iswa.2024.200361](https://doi.org/10.1016/j.iswa.2024.200361).
- [39] Shaoshuai Shi, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang und Hongsheng Li. "PV-RCNN: Point-Voxel Feature Set Abstraction for 3D Object Detection". In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, S. 10526–10535. DOI: [10.1109/CVPR42600.2020.01054](https://doi.org/10.1109/CVPR42600.2020.01054).
- [40] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu und Alexander C. Berg. "SSD: Single Shot Multi-Box Detector". In: *European Conference on Computer Vision*. Springer. 2016, S. 21–37. DOI: [10.1007/978-3-319-46448-0_2](https://doi.org/10.1007/978-3-319-46448-0_2). arXiv: [1512.02325 \[cs.CV\]](https://arxiv.org/abs/1512.02325).

- [41] Albert Gu und Tri Dao. *Mamba: Linear-Time Sequence Modeling with Selective State Spaces*. 2024. arXiv: 2312.00752 [cs.LG]. URL: <https://arxiv.org/abs/2312.00752>.
- [42] Xu Yan, Jiantao Gao, Jie Li, Ruimao Zhang, Zhen Li, Rui Huang und Shuguang Cui. "Sparse Single Sweep LiDAR Point Cloud Segmentation via Learning Contextual Shape Priors from Scene Completion". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. 2021. DOI: 10.48550/arXiv.2012.03762.
- [43] Jinwoo Kim u. a. "Up-Sampling Method for Low-Resolution LiDAR Point Cloud to Enhance 3D Object Detection in an Autonomous Driving Environment". In: *Sensors* 23.1 (2022), S. 322. DOI: 10.3390/s23010322.
- [44] Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W. Mahoney und Kurt Keutzer. "A Survey of Quantization Methods for Efficient Neural Network Inference". In: *arXiv preprint arXiv:2103.13630* (2021). URL: <https://arxiv.org/abs/2103.13630>.
- [45] Yuxiao Zhou und Kecheng Yang. "Exploring TensorRT to Improve Real-Time Inference for Deep Learning". In: *2022 IEEE 24th Int Conf on High Performance Computing Communications; 8th Int Conf on Data Science Systems; 20th Int Conf on Smart City; 8th Int Conf on Dependability in Sensor, Cloud Big Data Systems Application (HPCC/DSS/SmartCity/DependSys)*. 2022, S. 2011–2018. DOI: 10.1109/HPCC-DSS-SmartCity-DependSys57074.2022.00299.
- [46] Keneni W. Tesema, Lyndon Hill, Mark W. Jones, Muneeb I. Ahmad und Gary K.L. Tam. "Point Cloud Completion: A Survey". In: *IEEE Transactions on Visualization and Computer Graphics* 30.10 (2024), S. 6880–6899. DOI: 10.1109/TVCG.2023.3344935.
- [47] Dingkang Liang, Xin Zhou, Wei Xu, Xingkui Zhu, Zhikang Zou, Xiaoqing Ye, Xiao Tan und Xiang Bai. "PointMamba: A Simple State Space Model for Point Cloud Analysis". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2024. DOI: 10.48550/arXiv.2402.10739.
- [48] Hai Wang, Junhao Liu, Haoran Dong und Zheng Shao. "A Survey of the Multi-Sensor Fusion Object Detection Task in Autonomous Driving". In: *Sensors* 25.9 (2025). ISSN: 1424-8220. DOI: 10.3390/s25092794. URL: <https://www.mdpi.com/1424-8220/25/9/2794>.