Echtzeitfähige Schätzung generischer Linienzüge für das kartenlose automatisierte Fahren mittels Deep Learning

Zur Erlangung des akademischen Grades einer

DOKTORIN DER INGENIEURWISSENSCHAFTEN (Dr.-Ing.)

von der KIT-Fakultät für Maschinenbau
des Karlsruher Instituts für Technologie (KIT)

genehmigte

DISSERTATION

von

M.Sc. Annika Meyer
geb. in Heidelberg

Tag der mündlichen Prüfung: 21.04.2023

Hauptreferent: Prof. Dr.-Ing. Christoph Stiller
Korreferent: Prof. Dr.-Ing. Klaus Dietmayer

Zusammenfassung

Im automatisierten Fahren werden üblicherweise hochgenaue Karten eingesetzt, um die Wahrnehmung zu unterstützen und zu ergänzen. Diese Karten sind aufwändig zu erstellen und veralten schnell, da sich die Verkehrswelt stets ändert. Daher wird die Karte immer mehr von Wahrnehmungsmodulen ergänzt oder ersetzt. Dies stellt neue Anforderungen an die Wahrnehmung: Um die Kartenschnittstelle eines automatisierten Systems durch Detektionen aus Sensordaten zu ersetzen, muss ein Wahrnehmungsmodul die Kartenmerkmale detektieren können. Bisher gibt es im Forschungsfeld keine geeigneten Ansätze, um insbesondere linienförmige Kartenelemente im urbanen Raum zu detektieren. Hier wird meist auf zu einfache Repräsentationen zurückgegriffen, die insbesondere Kreuzungen nicht abbilden können. Eine korrekte Fahrstreifenschätzung ist jedoch die essenzielle Grundvoraussetzung, um langfristig Karten zu ersetzen.

Diese Arbeit präsentiert einen Ansatz, der diese Lücke schließt. Dazu wird ein vorwärtsgerichtetes, neuronales Netz präsentiert, das beliebige Linienzüge durch eine Gitterstruktur diskretisiert in Echtzeit schätzen kann. Mit dieser Repräsentation lassen sich Fahrstreifenränder, Markierungen aber auch implizite Merkmale wie Mittellinien von Fahrstreifen detektieren. Dies war in einer vorwärtsgerichteten Netzarchitektur bisher nicht möglich.

Die breite Anwendbarkeit des Ansatzes wird auf Fahrstreifenmittellinien, Fahrstreifenrändern sowie den Markierungen sowohl auf Autobahnen als auch im innerstädtischen Bereich gezeigt. Mithilfe der Klassifikation jedes Liniensegments können verschiedenste Kartenmerkmale gleichzeitig identifiziert werden: Markierungen können nach ihrer Bedeutung (gestrichelt oder durchgezogen) klassifiziert oder Mittellinien von Fahrstreifenrändern unterschieden und gemeinsam prädiziert werden.

Der in dieser Arbeit vorgestellte Ansatz ermöglicht nicht nur aktuell relevante Anwendungen der Kartenwahrnehmung, Kartierung und Lokalisierung. Auch

in zukünftigen, möglicherweise vollständig gelernten Systemen, behält die diskretisierte Darstellung von Kartenelementen ihre Relevanz.

Vorwort

Zuallererst danke ich meinem Doktorvater Prof. Dr.-Ing. Christoph Stiller für die außerordentlich guten Bedingungen am Institut für Mess- und Regelungstechnik. Meine Promotionszeit hat bereits mit dem spannenden MAD-Projekt begonnen, in dem ich die Idee des kartenlosen Fahrens untersucht habe. Dank der einzigartigen Möglichkeit zur Arbeit direkt am Versuchsträger und viel Zeit für freie Forschung, konnte ich sehr unterschiedliche, spannende Ideen verfolgen. Genauso möchte ich Prof. Dr.-Ing. Klaus Dietmayer dafür danken, dass er mein Koreferat übernommen hat und mir somit den Abschluss meiner Promotion ermöglicht.

Die Arbeit wäre so nicht möglich gewesen, ohne die Unterstützung der Kollegen, des Sekretariats, Werners und der Werkstätten. Dabei möchte ich mich ganz besonders für die motivierenden Worte von Erna Nagler bedanken, die, auch wenn sie sich noch so lautstark ärgern kann, immer strahlt und ein positives Wort für jeden hat.

Ich möchte mich auch bei den diversen Konstellationen der Lane Group bedanken, die durch einen wunderbaren Zusammenhalt und gegenseitige Unterstützung viele Möglichkeiten eröffnet und mich über Durststrecken hinweg motiviert haben. Insbesondere gilt der Dank meinem Gruppenleiter, der mich zu vielen Ideen inspiriert hat und sich immer wieder in jedes Thema eindenkt und wertvolles Feedback gibt. Zu guter Letzt ist das Wichtigste an der Arbeit, jeden Tag positiv und motiviert zu starten. Dafür danke ich Kevin Rösch für ein tägliches Standup mit Motivation, Hinterfragen von Ideen und Unterstützung jeglicher Art.

Gerade die letzten Wochen einer Disseration sind stressig und anstrengend. Daher danke ich ganz besonders meinen Korrekturlesern, die sich immer wieder dazu bereiterklärt haben, Teile der Arbeit zu lesen und mir wertvolles Feedback zu geben. Gerade Kevin Rösch, Fabian Poggenhans und Meike Rietdorf danke ich dabei für ihre spontanen und kurzfristigen Einsätze.

Die tolle Arbeitsatmosphäre war aber natürlich nicht das einzige, was mich durch diese Arbeit gebracht hat. Vielmehr ist das auch ein Vermächtnis der Unterstützung meiner Familie und Freunde. Vielen Dank, dass ihr mich durch die vielen Auf und Abs dieser Disseration ge- und ertragen habt und mir jede Abwesenheit – gedanklich und physisch – verzeihen konntet! Namentlich nennen möchte ich hier Katharina Bleistein und Meike Rietdorf, die mich immer wieder aufgebaut, therapiert oder mit Kaffee versorgt haben.

Abschließend möchte ich meinen unendlichen Dank an Jan-Hendrik Pauls aussprechen. Er hat mich bis ans Ziel motiviert und in schwierigen Phasen immer wieder aufgebaut. Ohne dich wäre ich heute nicht an diesem Punkt.

Karlsruhe, Dezember 2022

Annika Meyer

Inhaltsverzeichnis

Zusammenfassung . . . . . **i**

Vorwort . . . . . **iii**

Abkürzungen und Symbole . . . . . **ix**

1 Einleitung **1**
1.1 Motivation 2
1.2 Beiträge der Arbeit 4
1.3 Aufbau der Arbeit 4

2 Grundlagen **5**
2.1 Eigenschaften und Maße im euklidischen Raum 5
2.1.1 Fehlermaße 5
2.1.2 Klassifikationsmaße 6
2.2 Graphentheorie 7
2.3 Maschinelles Lernen 7
2.3.1 k-Means 8
2.4 Neuronale Netze 9
2.4.1 Faltende neuronale Netze 10
2.4.2 Encoder-Decoder-Architektur 12
2.4.3 Ausgabeschicht und Anwendungen 14
2.4.4 Aktivierungsfunktionen der Ausgabeschicht 15
2.4.5 Rekurrente neuronale Netze 17
2.4.6 Transformer 18
2.4.7 Graphenbasierte neuronale Netze 18
2.4.8 Optimierer 19
2.4.9 Kostenfunktionen 20

3 Stand der Forschung 21
3.1 Dichte Fahrstreifenschätzung 22
3.1.1 Fahrstreifenfläche 22
3.1.2 Fahrstreifenränder 23
3.1.3 Distanzbeschreibung 25
3.1.4 Diskussion 25
3.2 Dünnbesetzte, Parametrische Fahrstreifenschätzung 26
3.2.1 Anker 26
3.2.2 Rekurrente neuronale Netze 27
3.2.3 Graphenbasierte neuronale Netze 28
3.2.4 Transformer 28
3.3 Kreuzungen 29

4 Räumlich diskretisierte Linienschätzung 31
4.1 Problemdefinition 31
4.2 Echtzeitfähige Detektion von Linien 33
4.3 Architektur 36
4.3.1 Encoder 37
4.3.2 Decoder und Skalierung 38
4.3.3 Rezeptives Feld 39
4.4 Linienrepräsentation 40
4.4.1 Kartesische Start- und Endpunkte 41
4.4.2 Mittelpunkt-Richtung-Koordinaten 42
4.4.3 Randkoordinaten 43
4.4.4 Euler-Winkel 44
4.4.5 Diskussion der Linienrepräsentation 45
4.5 Prädiktoren 45
4.5.1 Dynamische Zuordnung 47
4.5.2 Anker 48
4.6 Kostenfunktion 51

5 Anwendungsspezifische Nachverarbeitung für das automatisierte Fahren 55
5.1 Graphkonstruktion 56
5.1.1 Zellstruktur 56

5.1.2 Euklidischer Abstand . . . 56
5.2 Non-Maximum-Suppression . . . 57
5.2.1 DBSCAN . . . 58
5.2.2 Non-Maximum-Suppression als Klassifikation im Graphen . . . 59
5.3 Verbindungsschätzung . . . 61
5.3.1 Baumsuche . . . 61
5.3.2 Kuhn-Munkres-Zuordnung . . . 62
5.3.3 MuSSP . . . 63
5.3.4 Gelernte Verbindung im Graphen . . . 64

6 Evaluation . . . 65
6.1 Datensätze . . . 66
6.1.1 Argoverse 2.0: Mittellinien . . . 67
6.1.2 KAI: Markierungen . . . 67
6.1.3 TuSimple: Fahrstreifenränder . . . 68
6.1.4 Datensatzaufbereitung . . . 68
6.2 Evaluationsmetrik . . . 70
6.3 Trainingssystem . . . 72
6.4 Experimente . . . 73
6.4.1 Rezeptives Feld . . . 73
6.4.2 Linienrepräsentationen und Aktivierungsfunktionen 74
6.4.3 Anker . . . 77
6.4.4 Dynamische Zuordnung . . . 81
6.4.5 Skalierung . . . 83
6.4.6 Diskretisierung . . . 85
6.4.7 Gewichtung der Kostenfunktion . . . 86
6.5 Qualitative Ergebnisse . . . 88
6.6 Qualitative Ergebnisse der Nachverarbeitung . . . 92

7 Schlussfolgerungen und Ausblick . . . 95

A Anhang . . . 101
A.1 Architektur . . . 102
A.2 Konfigurationen der Experimente . . . 105
A.3 Datensätze . . . 111

A.4 Parameter der Nachverarbeitung 115
A.5 Beispielbilder der Nachverarbeitung auf TuSimple 116
A.6 Verteilung der Liniensegmente je Zelle im Bild 118

Literaturverzeichnis **119**

Abkürzungen und Symbole

Abkürzungen

1D	Randkoordinaten
Acc	Accuracy
CNN	faltendes neuronales Netz
DBSCAN	Density Based Spatial Clustering of Applications with Noise
E2E	Ende-zu-Ende
Eu	Euler-Winkel
F1	F1-Maß
ffNN	vorwärtsgerichtetes neuronales Netz
FCN	vollständig faltendes neuronales Netz
FN	Falsch-Negativ-Schätzung
FP	Falsch-Positiv-Schätzung
GNN	graphenbasiertes neuronales Netz
GT	Ground Truth
IoU	Intersection over Union
Kf	Konfidenzabweichung
KfTP	Konfidenzabweichung der TPs
KAI	Kartenänderungsdatensatz von Pauls u.a.
Li	lineare Aktivierungsfunktion
LeakyReLU	Leaky Rectified Linear Unit
MAE	durchschnittliche, absolute Abweichung
MCMC	Markow-Ketten-Monte-Carlo-Verfahren
MLP	Multi-Layer Perzeptron
M	Mittelpunkt-Koordinaten

MPNN	Message Passing Neural Network
MR	Mittelpunkt-Richtung-Koordinaten
MSE	durchschnittliche, quadratische Abweichung
MuSSP	Minimum-update Successive Shortest Path
NN	neuronales Netz
NMS	Non-Maximum-Suppression
Pr	Precision
R	Richtungskoordinaten
Re	Recall
ReLU	Rectified Linear Unit
RNN	rekurrentes Netz
RMSE	Wurzel der durchschnittlichen, quadratischen Abweichungen
SE	kartesische Start- und Endpunkte
Si	Sigmoid-Funktion
SSE	Summe der quadratischen Abweichungen
TN	Wahr-Negativ-Schätzung
TP	Wahr-Positiv-Schätzung

Notationen

Skalare	Kleinbuchstaben	x, α
Vektoren	fettgedruckter Kleinbuchstabe	$\mathbf{v}$
Position im Vektor	hochgestellte Indizes	x^0, x^n
Tensoren	Großbuchstabe	V
2. Norm / Länge		$\|\mathbf{v}\|_2$
Mengen	Großbuchstabe	S
Kardinalität		$\|S\|$
Bildkoordinaten	uv-Indizes	s_u
Zellkoordinaten	xy-Indizes	s_x
Funktionen	griechische Kleinbuchstaben mit (...)	$\delta(\ldots)$
Winkel	griechische Kleinbuchstaben	α

Symbole

$1D$	1D-Randkoordinaten.
α	Start-Winkel in Euler-Koordinaten.
b_{1D}	Startpunkt in Randkoordinaten.
β	End-Winkel in Euler-Koordinaten.
$\mathbf{b}$	Startpunkt in Zellkoordinaten.
$\tilde{c}_{ij}$	Konfidenz einer Kante im GNN.
$\tilde{c}_i$	Konfidenz eines Knoten im GNN.
c	Konfidenz in YOLinO.
δ	Distanzfunktion.
$\mathbf{d}_{uv}$	Richtung in uv-Koordinaten.
$\mathbf{d}$	Differenz zwischen Start-/Endpunkt in Zellkoordinaten.
$\mathcal{E}$	Menge der Kanten im Graph.
e	Kante im Graph.
eu	Euler-Koordinaten.
e_{1D}	Enpunkt in Randkoordinaten.
$\mathbf{e}$	Endpunkt in Zellkoordinaten.
g	Geometrieindikator.
g	Ground Truth Linienzug.

$\mathbf{g}$	Geometrie.
G	Menge der Ground Truth Linienzüge.
h	Höhe des Eingangsbildes.
k	Klassenindikator.
$\mathbf{k}$	Klasse als Wahrscheinlichkeitsverteilung.
K	Menge der möglichen Klassen.
$\hat{\ell}$	GT-Liniensegment im Zellgitter Z.
$\hat{L}$	Menge der GT-Liniensegmente im Zellgitter Z.
l_{uv}	Länge in uv-Koordinaten.
ℓ	geschätztes Liniensegment im Zellgitter Z.
L	Menge der Liniensegmente im Zellgitter Z.
$\mathcal{L}_c$	Konfidenzkosten.
$\mathcal{L}_g$	Lokalisierungskosten.
$\mathcal{L}_k$	Klassifikationskosten.
md	Mittelpunkt-Richtung-Koordinaten.
$\mathbf{m}_{uv}$	Mittelpunkt in uv-Koordinaten.
$\mathbf{m}$	Mittelpunkt in Zellkoordinaten.
o	Spalte im Zellgitter.
$\lvert P \rvert$	Anzahl an Prädiktoren.
p	Prädiktor.
P	Menge der Prädiktoren je Zelle.

r	Zeile im Zellgitter.
se	Kartesische Koordinaten.
$\mathcal{V}$	Menge der Knoten im Graph.
v	Ein Knoten im Graph.
$\|V\|$	Anzahl der Trainingsvariablen e.g. $\|P\| \cdot (4 + \|K + 1)\|$.
w	Breite des Eingangsbildes.
ω_c	Gewicht der Konfidenzkosten.
ω_0	Gewicht der Konfidenzkosten für nicht-verantwortliche Prädiktoren.
ω_1	Gewicht der Konfidenzkosten für verantwortliche Prädiktoren.
ω_g	Gewicht der Geometriekosten.
ω_k	Gewicht der Klassifikationskosten.
z	Zelle.
Z	Menge der Zellen.

1 Einleitung

Mit den fortschreitenden Automatisierungsmöglichkeiten ist ein assistiertes Fahren Teil jedes neuen Fahrzeugs und ein voll-automatisiertes Fahren eine greifbare Zukunftsvision. In Systemen zum automatisierten Fahren sowie den aktuellen Forschungsarchitekturen werden Fahrentscheidungen gelöst, indem mehrere Module zum Tragen kommen. Eine typische, jedoch stark vereinfachte Architektur besteht dabei aus Komponenten wie in Abbildung 1.1 dargestellt ist.

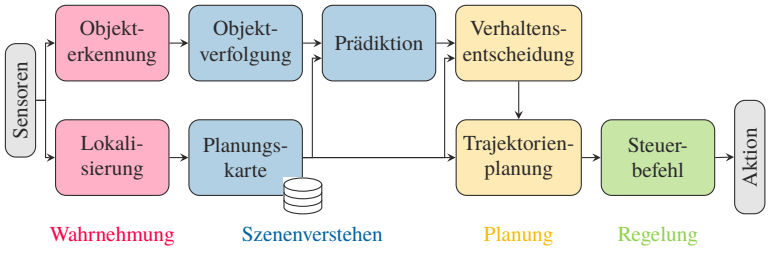


Abbildung 1.1: Vereinfachte Systemarchitektur für automatisierte Fahrzeuge

Zunächst werden dynamische Objekte in der Umgebung detektiert und deren zukünftige Positionen prädiziert. Gleichzeitig wird das Fahrzeug in einer bereits existierenden Planungskarte lokalisiert, die die statische Umgebung sowie Verkehrsregeln beinhaltet. Mit diesen Informationen kann eine Verhaltensentscheidung getroffen und eine Zieltrajektorie berechnet werden. Das letzte Modul regelt die Steuerbefehle, die direkt an die Aktorik des Fahrzeugs gegeben werden.

Da aus historischer Sicht das notwendige Szenenverstehen im Fahrzeug rechentechnisch einfach nicht möglich war, werden die Wahrnehmung und große Teile des Szenenverstehens bisher durch *hochgenaue Planungskarten (HD-Karten)* unterstützt.

1.1 Motivation

Diese hochgenauen Karten können vorab mithilfe deutlich größerer Rechenkapazitäten berechnet und optimiert werden und bieten somit auch Wissen über voraus liegende Straßenabschnitte, die noch nicht sichtbar sind. Ansätze zur automatischen Generierung derartigen Kartenmaterials gibt es bereits [Pog19]. Problematisch ist jedoch, dass diese Karten ein statisches Abbild sind, die (Verkehrs-)Welt sich jedoch dynamisch ändert und damit die Karteninformationen veralten. Diese Karten immer aktuell zu halten, birgt somit einen großen Aufwand. Neben diesen dauerhaften Änderungen wird ein automatisiertes Fahrzeug zudem mit temporären Veränderungen durch bspw. Baustellen oder Verkehrsunfälle konfrontiert. Zusätzlich verursacht die Lokalisierung in einer hochgenauen Karte bereits Ungenauigkeiten, die zu den Ungenauigkeiten der Karte hinzukommen.

Mit Zunahme der Rechenkapazitäten, die in einem automatisierten Fahrzeug zur Verfügung stehen, und der erhöhten Reichweite und Präzision moderner Sensoren werden Karten daher immer mehr auch durch Wahrnehmungsmodule ergänzt oder sogar ersetzt. Dies stellt neue Anforderungen an die Wahrnehmung: Um die Kartenschnittstelle eines automatisierten Systems durch Detektionen aus Sensordaten zu ersetzen, muss ein Wahrnehmungsmodul die Kartenmerkmale detektieren können.

Nach Poggenhans [Pog19] ergeben sich aus der Anwendung des automatisierten Fahrens sieben Anforderungen an die Generierung hochgenauer Karten. Davon sind für eine Schätzung im Fahrzeug vor allem die *befahrbare Fläche jenseits der Fahrstreifen*, die genaue *Geometrie der Fahrstreifen*, Informationen über die Fahrstreifen und mögliche *Fahrstreifenwechsel*, die *Verkehrsregeln* für alle Verkehrsteilnehmer und die *Verknüpfung der Verkehrsregeln mit den Fahrstreifen* relevant. Die befahrbare Fläche kann wie von der Autorin in [MSOS18] vorgeschlagen mit aktuellen Forschungsansätzen wie bspw. einer semantischen Segmentierung geschätzt werden. Diese kann jedoch nicht auf die Fahrstreifen angewendet werden, da die Komplexität von Kreuzungen nicht abgebildet werden kann.

Alle übrigen Anforderungen können erst mit einer genauen Schätzung der Geometrie der Fahrstreifen gelöst werden, da die Verkehrsregeln bspw. nur in Abhängigkeit zu einem Fahrstreifen existieren können. Somit ist eine Fahr-

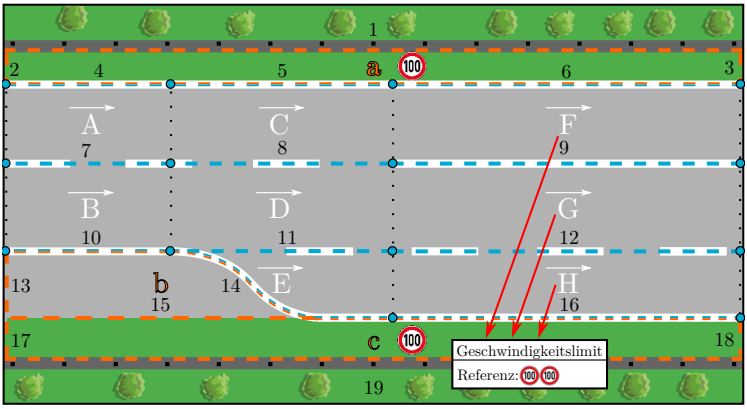


Abbildung 1.2: Aufbau einer HD-Karte mit Fahrstreifen, die durch je zwei Linienzüge als rechter und linker Rand begrenzt werden. Die Linienzüge verlaufen an der Position der Fahrspurmarkierungen (hellblau). Zusätzlich sind Verkehrsregeln und deren Bezüge zu Fahrstreifen visualisiert. Die befahrbare Fläche außerhalb der Fahrstreifen ist in Orange umrandet dargestellt. Bildquelle: [PPJ+18]

streifenschätzung die essenzielle Grundvoraussetzung, um hochgenaue Karten zu ersetzen. Da eine korrekte Fahrstreifengeometrie bisher nicht in Echtzeit im urbanen Raum detektiert werden kann, präsentiert diese Arbeit einen Ansatz zur Fahrstreifenschätzung, der auch im urbanen Raum einsetzbar ist.

Fahrstreifen werden in der aktuellen Forschung innerhalb einer Karte, wie bspw. Lanelet2 [PPJ+18], über zwei Linienzüge begrenzt (siehe Abbildung 1.2). Die Herausforderung für eine Fahrstreifenschätzung im Fahrzeug stellen hierbei, insbesondere im urbanen Bereich, Kreuzungen dar. Hier verlaufen Fahrstreifen per Definition übereinander oder werden zusammengeführt. Genauso kann ein Fahrstreifen sich in zwei separate Fahrstreifen aufspalten. Zudem sind für fahrzeugfeste Sensorik Teile der Kreuzungen verdeckt und müssen dennoch geschätzt werden.

1.2 Beiträge der Arbeit

Diese Arbeit stellt ein Verfahren zur Fahrstreifenschätzung vor, das die Generalisierfähigkeit von neuronalen Netzen mit einer geeigneten Linienrepräsentation für den urbanen Raum kombiniert. Es können nicht nur Linienzüge wie Fahrstreifenränder, sondern sogar jegliche linienförmigen Elemente in Echtzeit detektiert und klassifiziert werden. Dadurch lassen sich zum Beispiel gleichzeitig Fahrstreifenränder und Mittellinien detektieren ohne einen zweiten Algorithmus entwerfen zu müssen. Dies bildet die Grundlage, um selbst komplexe Kreuzungsmodelle in Echtzeit wahrnehmen zu können.

Die vorliegende Arbeit liefert dabei die zuvor nicht gelösten Beiträge im Rahmen einer echtzeitfähigen Deep-Learning-Architektur, genannt *YOLinO*:

1. eine echtzeitfähige Linienschätzung, die Sensordaten ohne Zwischenrepräsentation verarbeitet
2. eine für komplexe Topologien, wie Kreuzungen, geeignete Linienrepräsentation
3. gleichzeitige Abstraktion von Mittellinien, Fahrbahnmarkierungen und Fahrstreifenrändern
4. eine auf Linienzüge spezialisierte Non-Maximum-Suppression und Nachverarbeitung

1.3 Aufbau der Arbeit

Zunächst legt Kapitel 2 die Grundlagen für diese Arbeit, die insbesondere auf das Thema der neuronalen Netze eingehen. Mit Kapitel 3 werden die Grundlagen um den Stand der Forschung erweitert. Kapitel 4 umfasst die Konzeption und Implementierung des Ansatzes, der in Kapitel 6 evaluiert wird. Für eine anwendungsspezifische Sicht auf mögliche Nachverarbeitungen diskutiert Kapitel 5 verschiedene Ansätze aus dem Bereich der Non-Maximum-Suppression und Verbindungsschätzung. Die Erkenntnisse aus den vorherigen Kapiteln werden in Kapitel 7 zusammengefasst und mit einem Ausblick abgeschlossen.

2 Grundlagen

In diesem Kapitel werden zunächst die verwendeten Metriken und Maße in Abschnitt 2.1 eingeführt. Im Anschluss stellt Abschnitt 2.2 die Grundbegriffe der Graphentheorie auf, gefolgt von einem Einblick in das maschinelle Lernen inklusive des Clusteringalgorithmus k-Means in Abschnitt 2.3. Zum Abschluss des Kapitels beschreibt Abschnitt 2.4 verschiedene Varianten der neuronalen Netze und klärt Begrifflichkeiten. Die in diesem Kapitel eingeführten Grundlagen entsprechen dem Referenzwerk von Goodfellow u.a. [GBC16] soweit nicht anders angegeben.

2.1 Eigenschaften und Maße im euklidischen Raum

2.1.1 Fehlermaße

Die L2-Norm gehört zur Klasse der Lp-Normen, die einem Vektor einen nicht-negativen Skalar zuweisen. Im Falle $p = 2$ beschreibt diese Norm intuitiv die euklidische Distanz des Vektors $\mathbf{x}$ zum Ursprung mit

$$\|\mathbf{x}\|_2 = \sqrt{\sum_i x_i^2} \tag{2.1}$$

Die durchschnittliche, absolute Abweichung (MAE) befindet sich im gleichen Raum wie die Eingangsvektoren und ist definiert als

$$MAE(\mathbf{x}, \mathbf{y}) = \frac{1}{n} \sum_{i=1}^{n} |y_i - x_i|. \tag{2.2}$$

Die durchschnittliche, quadratische Abweichung (MSE) wird häufig als Kostenfunktion für neuronale Netze eingesetzt. Sie ist definiert als

$$MSE(\mathbf{x}, \mathbf{y}) = \frac{1}{n} \sum_{i=1}^{n} (y_i - x_i)^2. \tag{2.3}$$

Die Wurzel der durchschnittlichen, quadratischen Abweichungen (RMSE) bewertet Ausreißer in einem Datensatz stärker als bspw. der MAE. Beide Maße existieren im gleichen Raum wie die Eingangsvektoren. Sie ist definiert als

$$RMSE(\mathbf{x}, \mathbf{y}) = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (y_i - x_i)^2}. \tag{2.4}$$

2.1.2 Klassifikationsmaße

Der Recall / die Trefferquote beschreibt die Wahrscheinlichkeit, mit der ein positives Objekt korrekt positiv klassifiziert wird. Für eine Schätzung berechnet sich der Recall aus den wahr-positiven Objekten (TP) und den falsch-negativen Objekten (FN) mit

$$Re = \frac{TP}{TP + FN} \tag{2.5}$$

Die Precision / die Genauigkeit beschreibt die Wahrscheinlichkeit, mit der ein positiv klassifiziertes Objekt tatsächlich positiv ist. Für eine Schätzung berechnet sich die Precision aus den wahr-positiven Objekten (TP) und den falsch-positiven Objekten (FP) als

$$Pr = \frac{TP}{TP + FP} \tag{2.6}$$

Das F1-Maß setzt den Recall und die Precision ins Verhältnis, indem das harmonische Mittel der beiden berechnet wird:

$$F_1 = \frac{2 \cdot Re \cdot Pr}{Re + Pr} \tag{2.7}$$

Die Accuracy betrachtet zusätzlich die Wahr-Negativ-Schätzung (TNs) und gibt ein Maß über die Qualität der Gesamtschätzung mit

$$Acc = \frac{TP + TN}{TP + TN + FP + FN}. \tag{2.8}$$

2.2 Graphentheorie

Die folgenden Grundlagen der Graphentheorie wurden aus dem Basiswerk *Graphentheorie* von P. Tittmann [Tit22] entnommen.

Ein Graph $\mathcal{G}$ ist definiert als Menge von Knoten $\mathcal{V}$ verbunden über Kanten $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. Ein *gerichteter Graph* weist dabei jeder Kante zusätzlich eine Richtung zu und kann somit zwischen zwei Knoten bis zu zwei Kanten beinhalten (jeweils in entgegengesetzter Richtung). Ein *ungerichteter Graph* hat diese Eigenschaft nicht und somit nur eine Kante zwischen zwei Knoten. Im Verlauf dieser Arbeit wird zudem von *azyklischen* Graphen gesprochen. Diese weisen keine Ringschlüsse in den Kanten auf. Die Anzahl der Kanten, die von einem Knoten abgehen, wird als *Grad* des Knotens bezeichnet, das Maximum über alle Knoten als *Maximalgrad*.

Ein für diese Arbeit relevanter spezieller Graph ist ein gerichteter *Baum*. Dieser erfüllt die Bedingung eines gerichteten, azyklischen Graphen und ist zusätzlich zusammenhängend. Der Knoten ohne Vorgänger wird dabei als *Wurzelknoten* und die Knoten ohne Nachfolger als *Blätter* bezeichnet. Alle übrigen, *inneren* Knoten, besitzen genau eine eingehende Kante, sodass die Gesamtanzahl der Kanten $|\mathcal{E}| = |\mathcal{V}| - 1$ entspricht.

2.3 Maschinelles Lernen

Mit maschinellem Lernen werden verschiedene Aufgaben wie Klassifikation oder Regression gelöst. Dabei wird zu Eingabedaten eine Ausgabe generiert und die Parameter des Verfahrens mit einem Datensatz gelernt. Dieser besteht aus einem *Trainings-, Validierungs- und Testdatensatz*. Mit dem Trainingsdatensatz werden die Modelle trainiert und anschließend Hyperparameter auf

dem Validierungsdatensatz ermittelt. Zur Evaluation kann auf einem Testdatensatz berechnet werden, wie gut die Modelle auf neue Daten generalisieren.

Wenn die Modellstruktur mehr Parameter bietet als die Daten benötigen und somit das Modell überangepasst werden kann, spricht man von *Overfitting*. Hierbei kann ein Modell nicht mehr auf neue Daten generalisieren und lernt die Trainingsdaten auswendig. Das Gegenstück dazu ist das sogenannte *Underfitting*, bei dem ein Modell das Problem mit zu wenig Parametern abbildet.

Während des Trainings eines Algorithmus wird das Modell mit dem Validierungsdatensatz überprüft. Sobald sich die Ergebnisse dort nicht mehr verbessern, wird die Optimierung abgebrochen (engl. *early stopping*). Ab diesem Zeitpunkt liegt häufig bei dem Modell ein Overfitting vor.

Um direkt im Training stärker zu generalisieren, wird ein Trainingsschritt nicht individuell für ein Trainingsbeispiel, sondern für mehrere gleichzeitig – einen sogenannten Batch – durchgeführt. Dabei lassen sich die Optimierungsschritte über mehrere Beispiele gleichzeitig berechnen und generalisieren daher besser. Kandel und Castelli [KC20] empfehlen eine *Batchgröße* von 32 oder 64.

Überwachtes Lernen ist die am häufigsten eingesetzte Trainingsstrategie. Hierzu muss ein Datensatz zu jeder Eingabe die korrekte Ausgabe (Ground Truth (GT)) beinhalten, sodass aus dem Distanzmaß zwischen der GT und der Prädiktion eine Kostenfunktion formuliert werden kann. Für diesen Zusammenhang kann ein Gradient berechnet und somit eine Optimierung durchgeführt werden.

Im Gegensatz zu überwachtem Lernen benötigt *unüberwachtes Lernen* keine vorgegebene GT. Stattdessen werden inhärent in den Daten vorhandene Informationen genutzt, um einen Algorithmus zu optimieren. Zu diesen Verfahren gehören Gruppierungsalgorthmen – auch Clustering oder Cluster-Verfahren – wie k-Means. Ein Cluster-Verfahren versucht eine gegebene Datenmenge in Gruppen zu unterteilen und so Cluster zu bilden.

2.3.1 k-Means

Der k-Means-Algorithmus wird auf Datenmengen in euklidischen Räumen angewendet und liefert für eine gegebene Datenmenge der Größe N genau k Cluster. Das Optimierungsziel dieses Algorithmus ist die Minimierung der

quadratischen Summe (SSE) der Distanzen aller Datenpunkte N zum Mittelpunkt ihres zugewiesenen Clusters c mit Mittelpunkt μ_c, formalisiert als

$$J = \sum_{n=1}^{N} \sum_{c=1}^{C} r_{nc} |\mathbf{x}_n - \mu_c|^2. \tag{2.9}$$

r_{nc} beschreibt die Indikatorfunktion, die 1 annimmt, wenn Datenpunkt n Teil des Clusters c ist. Zur Optimierung dieser Kostenformulierung werden initial die Datenpunkte zufällig gewählten Clustern zugewiesen und anschließend iterativ die Mittelpunkte μ_c der aus dieser Zuweisung resultierenden Cluster berechnet. Wenn sich die Zuordnung für keinen Datenpunkt mehr durch eine Iteration ändert, so ist das Optimierungsziel erreicht. [Bis06]

2.4 Neuronale Netze

Der folgende Abschnitt beschreibt die für diese Arbeit wichtigen Grundlagen der neuronalen Netze. Neuronale Netze (NN) beschreiben eine Untergruppe von Algorithmen aus dem maschinellen Lernen, die durch Kombination einzelner Schichten beliebige Funktionen abbilden können. Es wird dabei von einer *Eingangsschicht*, *verdeckten Schichten* und der *Ausgangsschicht* gesprochen (siehe Abbildung 2.1).

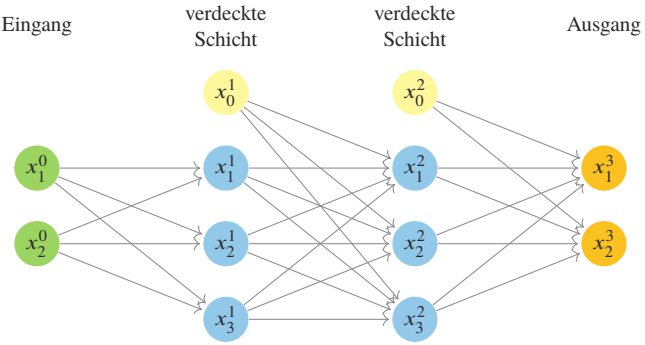


Abbildung 2.1: Ein vollständig verknüpftes neuronales Netz mit Neuronen in der Eingangsschicht, verdeckten Schichten und der Ausgangsschicht. Die zweite und dritte Schicht nehmen zusätzlich zu den Eingängen je einen Schwellwert auf.

Jede Schicht besteht aus mehreren Neuronen und nimmt Daten der Neuronen der vorhergehenden Schicht entgegen. Je Neuron werden diese Eingänge gewichtet aufsummiert und eine Aktivierungsfunktion darauf angewendet. Abhängig vom Modell wird zusätzlich je Schicht ein ungewichteter Schwellwert addiert. Befinden sich keine Zyklen innerhalb des Netzes, so handelt es sich um ein vorwärtsgerichtetes neuronales Netz (ffNN).

Die hier beschriebene und in Abbildung 2.1 gezeigte Variante ist zudem ein *vollständig verknüpftes Netz*: Jedes Neuron der nachfolgenden Schicht ist mit allen Neuronen der Vorgängerschicht verknüpft. Wenn zudem jedes Neuron über einen Schwellwert verfügt, so spricht man auch von einem *Multi-Layer Perzeptron (MLP)*.

2.4.1 Faltende neuronale Netze

Für diese Arbeit von größerer Bedeutung als vollständig verknüpfte Netze sind faltende neuronale Netze (CNNs). Sie werden in der Verarbeitung von räumlich strukturierten Informationen wie bspw. 2D-Bilddaten eingesetzt. Die Ausgabe jeder Schicht ist kein Vektor, sondern ein mehrdimensionaler Tensor, der auch *Merkmalskarte* (engl. Feature Map) genannt wird. Anstatt in jeder Schicht eine gewichtete Summe über alle Vorgängerneuronen vorzunehmen, wird ein Kernel mit einem Ausschnitt der Vorgängerschicht gefaltet. Die Struktur einer 2D-Merkmalskarte mit einem 3×3 Kernel zeigt Abbildung 2.2.

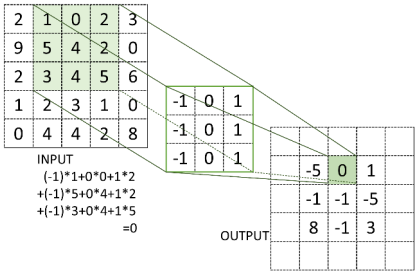


Abbildung 2.2: 2D-Faltung mit einem 3×3 Kernel. Bildquelle: [PWZ+22]

Die Kernel jeder Schicht werden im Training gelernt und fungieren wie die Gewichte der vollständig verknüpften Netze. Da derselbe Kernel nach und

nach auf das gesamte Bild angewendet wird, können faltende Schichten Merkmale unabhängig von ihrer Position in den Eingangsdaten (bspw. Bilddaten) erkennen und sind gut parallelisierbar.

Im 2D-Fall berechnet sich für die Position (r, c) in einer Merkmalskarte X^j die Faltung des Kernels W der Größe $n \times m$ mit der Eingangsmatrix X^{j-1} als

$$X^j = \sigma(\sum_{k=1}^{n} \sum_{l=1}^{m} W_{kl}^j X_{kl}^{j-1}). \tag{2.10}$$

Auch für CNNs wird die *Aktivierungsfunktion* σ auf die gewichtete Summe angewendet. Für verdeckte Schichten in CNNs ist die Rectified Linear Unit (ReLU) die Standardwahl. Sie bildet die Eingabe z mit

$$\sigma(z) = \begin{cases} z, & \text{wenn } z > 0 \\ 0, & \text{sonst} \end{cases} \tag{2.11}$$

ab.

Die Leaky Rectified Linear Unit (LeakyReLU) ist eine abgeschwächte Form der ReLU, bei der der negative Teil der Funktion mit $0 < a \ll 1$ skaliert wird, sodass

$$\sigma(z) = \begin{cases} z, & \text{wenn } z > 0 \\ az, & \text{sonst.} \end{cases} \tag{2.12}$$

Dadurch können verschwindende Gradienten (engl. vanishing gradients) in der Optimierung vermieden werden. Sie findet Anwendung in den in dieser Arbeit eingesetzten Architekturen der YOLO-Familie [RDGF16].

Durch die Beschränkung der faltenden Schichten auf nur einen Teil der Vorgängerschicht reduziert sich der Einfluss der Eingangsdaten auf die Ausgangsdaten. Der Bereich der Eingangsdaten, der für ein Neuron im Netz sichtbar ist, nennt sich *rezeptives Feld*. Abbildung 2.2 visualisiert diese Beschränkung in Grün.

Um das rezeptive Feld zu vergrößern, können die Faltungskernel durch eine *Aufweitung* (engl. dilation) auf eine größere Fläche der Vorgängerschicht angewendet werden. Dabei wird bspw. nur jedes zweite Vorgängerpixel in die Faltung aufgenommen.

2.4.2 Encoder-Decoder-Architektur

In der Forschung hat sich mit Verwendung von CNNs das Prinzip einer Encoder-Decoder-Struktur etabliert. Hier werden die Eingangsdaten mit einem sogenannten *Encoder* (auch Backbone) auf einen kleineren Tensor – das *Embedding* – komprimiert und anschließend mit einem *Decoder* (auch Frontend) wieder hochskaliert. Dies hat den Effekt, dass das Netz lernt, im Embedding wichtige Strukturen zu extrahieren, aus denen die Prädiktion hinreichend gut geschätzt werden kann. Die U-Form dieser Anwendung ist in Abbildung 2.3 visualisiert: mit dem Encoder im linken Teil, dem Embedding an der unteren Spitze und dem Decoder im rechten Teil.

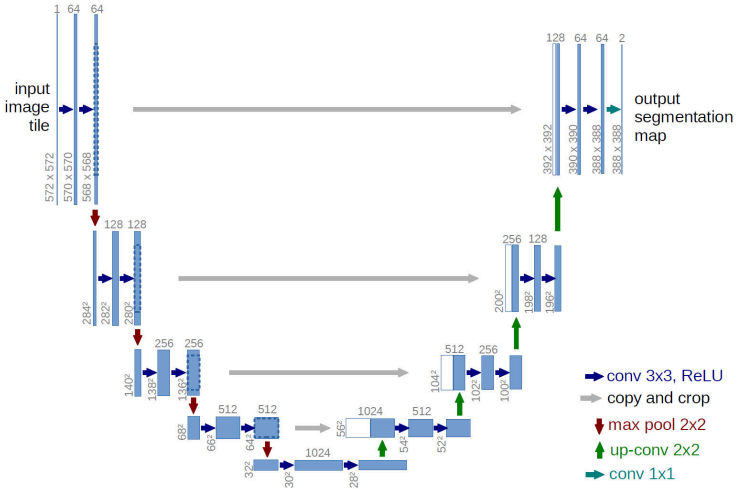


Abbildung 2.3: Encoder-Decoder Struktur mit Sprungverbindungen für eine semantische Segmentierung. Der Encoder komprimiert Eingangsdaten im linken Teil zu einem Embedding (untere Spitze). Der Decoder dekomprimiert die Daten im rechten Teil hin zu einer Prädiktion. Bildquelle: [RFB15]

Die Komprimierung der Eingangsdaten erfolgt in einem CNN durch sogenannte *Pooling-Schichten*. Diese fassen die vorherige Schicht blockweise mit einer Aggregationsfunktion zusammen. Typische Aggregationen sind das Maximum oder der Durchschnitt der Eingangsdaten (siehe Abbildung 2.4).

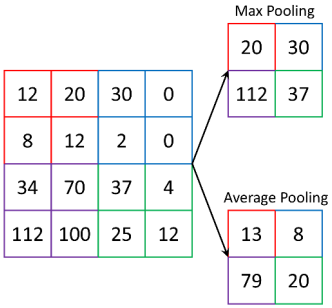


Abbildung 2.4: Prinzip einer Pooling-Schicht: Je 2×2 Block wird das Maximum bzw. der Mittelwert gebildet. Bildquelle: [PWZ+22]

Da bei der Komprimierung der Daten die Positionsinformation leicht verloren gehen kann und Gradienten bei zu langen Wegen durch eine Architektur immer kleiner werden, wurden *Sprungverbindungen* (engl. skip connection, [LSD15]) etabliert. Die Informationen der Merkmalskarten aus dem Encoder werden direkt an den Decoder weitergeleitet und dort mit den dekomprimierten Daten fusioniert.

Die Decoder-Schichten invertieren die Komprimierung der Encoder-Schichten mit *transponierten, faltenden Schichten*. In [LDG+17] werden zusätzlich *Merkmalspyramiden* angewendet. Diese werten verschiedene dekodierende Schichten der Netzarchitektur aus und schätzen unterschiedlich aufgelöste Prädiktionen, die anschließend zu einer Prädiktion fusioniert werden (siehe Abbildung 2.5).

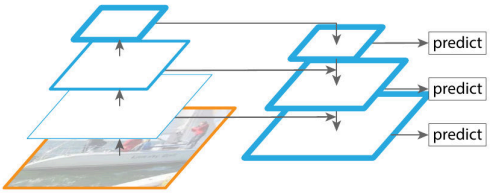


Abbildung 2.5: Prinzip einer Merkmalspyramide. Aus mehreren Schichten werden Prädiktionen extrahiert, die unterschiedlich aufgelöst sind. Bildquelle: [LDG+17]

2.4.3 Ausgabeschicht und Anwendungen

Dieser Abschnitt soll veranschaulichen, wie die bisher genannten Strukturen eingesetzt werden. Ein vollständig verknüpftes Netz definiert eine eindimensionale Ausgabestruktur. Passend zur Anwendung werden diese Neuronen auf bestimmte Ausgaben trainiert, um bspw. Regression oder Klassifikation durchzuführen. Bspw. können die Merkmale eines Datenpunktes klassifiziert werden. Dabei wird die Klassifikation als Wahrscheinlichkeitsverteilung über alle möglichen Klassen K formuliert, sodass der Ausgabevektor die Länge $|K|$ hat.

Abbildung 2.6 zeigt unterschiedliche Anwendungen und dazugehörige faltende Architekturen. Mit einer Mischung aus vollständig verknüpften und faltenden Schichten können Eingangsdaten als mehrdimensionale Struktur (z.B. Bilder) verarbeitet und gleichzeitig eine eindimensionale Ausgabestruktur geschätzt werden (siehe Abbildung 2.6 oben). Hier lässt sich bspw. eine Bildklassifikation mit einem Ausgabevektor der Länge $|K|$ durchführen.

Ein FCN definiert hingegen eine mehrdimensionale Ausgabestruktur, da nur faltende Schichten verwendet werden. Hier kann bspw. eine semantische Segmentierung wie in Abbildung 2.6 unten vorgenommen werden. Dabei wird bspw. ein Eingangsbild der Größe $w \times h$ mit drei Farbkanälen verarbeitet und pixelweise klassifiziert, sodass die Ausgabestruktur des Netzes einem Tensor der Größe $w \times h \times |K|$ entspricht. Es wird also für jedes Pixel des Eingangsbildes ebenfalls eine Wahrscheinlichkeitsverteilung der Länge $|K|$ geschätzt.

Alternativ kann an Stelle einer semantischen Segmentierung bspw. eine pixelweise Regression durchgeführt werden. Dazu wird statt der Wahrscheinlichkeitsverteilung ein Vektor der Länge v je Pixel geschätzt, sodass der Ausgabetensor die Form $w \times h \times v$ hat.

Die Dimension der Ausgabe wird durch das Training festgelegt und kann nicht variieren. Dies ist für variable Prädiktionen eine Herausforderung, da zusätzliche Variablen geschätzt werden müssen, die die Größe der Schätzung bestimmen.

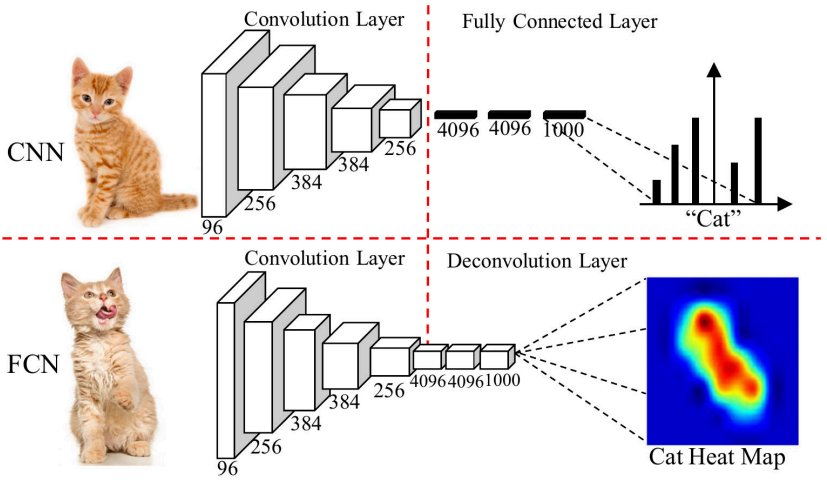


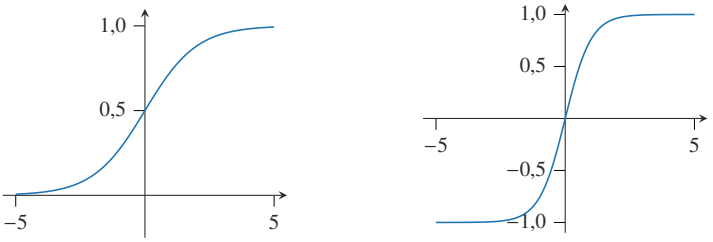
Abbildung 2.6: Anwendung verschiedener vorwärtsgerichteter Netzarchitekturen. *oben*: faltendes neuronales Netz als Kombination aus faltenden Schichten (engl. Convolutional Layer) und vollständig verknüpften Schichten (engl. Fully Connected Layer). Eine 1D-Ausgabestruktur wird als Bildklassifikation mit sechs Klassen trainiert. *unten*: Ein vollständig faltendes neuronales Netz (FCN) wird als semantische Segmentierung eines Bildes mit einer binären Klassifikation (engl. Heat Map) trainiert. Bildquelle: [PWZ+22]

2.4.4 Aktivierungsfunktionen der Ausgabeschicht

Um die Ausgabe eines neuronalen Netzes ggf. in die richtigen Wertebereiche zu legen, werden Informationen der letzten Schicht mit anderen Aktivierungsfunktion als in den verdeckten Schichten transformiert. Für Probleme, deren Wertebereich nicht beschränkt ist, wird die *lineare Aktivierungsfunktion* eingesetzt. Sie wendet keine Transformation auf die Eingangswerte an, sodass

$$\sigma_l(z) = z. \tag{2.13}$$

Für binäre bzw. multivariate Klassifikation eignen sich die Sigmoid- (siehe Abbildung 2.7a) bzw. die Softmax-Funktion. Häufig werden auch diskrete Schätzprobleme als solche umgesetzt.



(a) Sigmoid (b) Tangens Hyperbolikus

Abbildung 2.7: Aktivierungsfunktionen für Prädiktionen im Intervall. In a) wird die Sigmoid-Funktion für Ausagebn im Intervall $[0, 1]$ dargestellt. In b) wird der Tangens Hyperbolikus visualisiert, der Eingangswerte in das Intervall $[-1, 1]$ transformiert.

Die *Sigmoid-Funktion* wird für Bernoulli-Verteilungen, also binäre Klassifikationen, verwendet. Sie bildet die Eingabewerte auf das Intervall $[0, 1]$ ab, mit

$$\sigma_{sg}(z) = \frac{1}{1 + e^{-z}}. \tag{2.14}$$

Die *Softmax-Funktion* wird für Multi-Bernoulli-Verteilungen verwendet, bei der eine Klassifikation mit $|K|$ möglichen Klassen als Wahrscheinlichkeitsverteilung geschätzt wird. Das Netz berechnet dann $|K|$ Ausgabewerte mit jeweils der Wahrscheinlichkeit $P(z_i|I)$ abhängig von den Eingangsdaten I. Für Klasse i berechnet sich die Softmax-Funktion als

$$\sigma_{sf}(z_i) = \frac{e^{z_i}}{\sum_j^{|K|} e^{z_j}}. \tag{2.15}$$

Äquivalent dazu werden die GT-Elemente als *One-Hot-Encoding* formuliert, bei dem die Klasse k als Vektor z der Länge $|K|$ beschrieben wird, für dessen Einträge z_i gilt

$$z_i = \begin{cases} 1, & \text{wenn } i = k \\ 0, & \text{sonst.} \end{cases} \tag{2.16}$$

Die Ausgabeschicht für ein Modell zur Lösung eines Regressionsproblems, bei dem der Lösungsbereich auf ein Intervall begrenzt ist, kann ebenfalls

eine Sigmoid-Funktion (siehe Abbildung 2.7a) oder ein *Tangens Hyperbolikus* (siehe Abbildung 2.7b) mit

$$\sigma_t(z) = \tanh z = \frac{\sinh z}{\cosh z} = \frac{e^z - e^{-z}}{e^z + e^{-z}} \tag{2.17}$$

für periodische Variablen eingesetzt werden. Typischerweise ist jedoch bei einer Regression die lineare Aktivierungsfunktion effektiver.

2.4.5 Rekurrente neuronale Netze

Um variable Ausgabe- und Eingabestrukturen verwenden zu können, wurden rekurrente Netze (RNNs) [RHW86] vorgeschlagen. Ein rekurrentes Netz ist allgemein formuliert eine iterativ angewendete Netzarchitektur, die – teilweise unter Verwendung eines Zustands – Sequenzen aus Eingabedaten einliest und/oder Sequenzen an Ausgaben produziert. Die feste Ausgabestruktur muss hier nur je Iteration beibehalten werden. Dies ist in Abbildung 2.8 visualisiert. Die grün visualisierten Blöcke entsprechen der iterativen Anwendung des gleichen neuronalen Netzes.

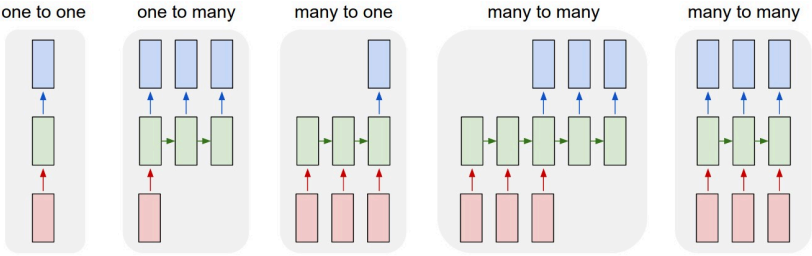


Abbildung 2.8: Varianten rekurrenter Netze. Es lassen sich sowohl sequenzielle Eingangsdaten verwenden (rot) als auch sequenzielle Ausgabedaten produzieren (blau). Dies wird durch iterative Ausführung einer Netzarchitektur (grün) erreicht, die zwischen den Iterationen einen Zustand speichert. Die Variante 'one-to-one' entspricht einem ffNN. Bildquelle: `http://karpathy.github.io/2015/05/21/rnn-effectiveness/`

RNNs sind für viele Anwendungen sehr erfolgreich, jedoch für die Perzeption von echtzeitfähigen Anwendungen nicht gut geeignet. Zum einen weisen sie keine feste Komplexität auf, da die Laufzeit von der Länge der Sequenz

abhängt und sie nicht parallelisiert werden können. Zum anderen werden die Gradienten durch die ggf. zahlreichen Iterationen des Netzes beim Training sehr klein, sodass kein Gradientenabstieg möglich ist. Dies wird als Problem der *verschwindenden Gradienten* (engl. vanishing gradients) bezeichnet.

2.4.6 Transformer

In der aktuellen Forschung werden rekurrente Netze von sogenannten *Transformern* [VSP+17, DBK+21] abgelöst. Diese kombinieren die Parallelisierbarkeit der CNNs mit der sequenziellen Verarbeitung der RNNs. Hier wird die Sequenz nicht schrittweise verarbeitet und kein Zustand über die Schritte hinweg gespeichert. Stattdessen wird für jede Schätzung die gesamte Eingabe bereitgestellt. Ein *Attention-Modul* [VSP+17] berechnet zunächst ein Embedding fester Länge für die gesamte Eingabesequenz. Die Gewichtung der einzelnen Teile wird dabei durch die Attention bestimmt, die jeden Teil in ihrer Relevanz bewertet. Ein ffNN berechnet anschließend daraus ein Embedding für die Sequenz.

Zur Schätzung einer Ausgabe werden Abfragen (engl. Queries) auf Basis der bisher prädizierten Ausgabesequenz verwendet, um aus der kodierten Eingabesequenz Informationen auszuwählen und auszuwerten. Dabei ist die Generierung der Ausgabe auto-regressiv: Alle bereits ausgegebenen Prädiktionen dienen als Eingabe für die nächste Prädiktion. So entsteht eine zusammenhängende Sequenz. 2022 wurde diese Idee als Vision Transformer [DBK+21] auf Bilder angewendet.

Durch diese Architektur sind Transformer im Gegensatz zu RNNs robust gegenüber verschwindenden Gradienten, da nur kurze Pfade zwischen der Ausgabe und der Eingabe liegen. Zudem muss der Encoder für die Eingabesequenz nur einmal inferiert werden und weist damit eine feste Komplexität auf. Der Decoder hingegen folgt weiterhin einem iterativen Verfahren.

2.4.7 Graphenbasierte neuronale Netze

Graphenbasierte neuronale Netze (GNN) verarbeiten Eingangsdaten, die als Graph repräsentiert sind (siehe Abschnitt 2.2). Nachrichtenverschickende neu-

ronale Netze (MPNN) [GSR+17] sind eine Spezialform der graphenbasierten neuronalen Netze (GNNs). Sie fassen Ansätze zusammen, die sogenannte Nachrichten zwischen den Knoten (und Kanten) des Graphen austauschen und diese Repräsentation anschließend zur Prädiktion nutzen. Dazu werden je nach Ansatz bspw. ein MLPs auf die Merkmalsvektoren zweier Knoten und ihrer Kante oder aller Kanten eines Knotens angewendet. Die variable Anzahl der Kanten eines Knotens wird mit einer Aggregationsfunktion (Maximum oder Mittelwert) auf einen festen Merkmalsvektor zusammengefasst.

Diese Schritte werden mehrfach wiederholt, sodass Informationen immer weiter durch den Graphen gereicht werden können. Die so entstandene Repräsentation des Graphen kann dann verwendet werden, um bspw. eine Klassifikation auf den Merkmalsvektoren jedes Knotens oder einer Kante durchzuführen. Dazu wird wiederum eine beliebige Netzarchitektur verwendet wie bspw. ein MLP.

2.4.8 Optimierer

Wenn die Architektur eines neuronalen Netzes bestimmt ist, muss für das Training eine Zielfunktion (auch Kostenfunktion genannt) formuliert werden. Für überwachte Lernverfahren ist dies typischerweise ein Distanzmaß zwischen der Schätzung und der GT. Die Kostenfunktion hat per Definition die optimale Lösung als globales Minimum. Durch die Nichtlinearität der neuronalen Netze ist diese Optimierung jedoch nicht geschlossen lösbar und muss iterativ erfolgen. Dazu wird der Gradient berechnet und schrittweise die Gewichte des Netzes optimiert – dies wird als *Gradientenabstieg* bezeichnet. Die Schrittweite wird dabei als *Lernrate* bezeichnet.

Mit einer differenzierbaren Kostenfunktion kann für jedes Gewicht des Netzes zu jedem Trainingsbeispiel ein Gradient berechnet werden, der durch die Schichten der Netzarchitektur propagiert wird und so die Gewichte jeder Schicht aktualisiert. Dieses Verfahren nennt sich *Backpropagation*.

Um den Gradientenabstieg durchzuführen, gibt es verschiedene Optimierer zur Auswahl. Sie unterscheiden sich durch ihre Parametrisierung und Fähigkeit bspw. lokale Minima zu umgehen. Ein Vergleich verschiedener Ansätze findet sich in [GBC16, S. 302]. Der *Adam*-Optimierer [KB15] liefert für Deep

Learning Anwendungen die besten Trainingsergebnisse und ist gleichzeitig robust gegenüber der Wahl der Hyperparameter.

2.4.9 Kostenfunktionen

Aktivierungsfunktionen mit Exponentialanteil (Sigmoid, Softmax) können mit einer logarithmischen Kostenfunktion kombiniert werden. Beide Aktivierungsfunktionen skalieren die Werte auf das Intervall $[0, 1]$, sodass die Netzausgaben als Wahrscheinlichkeiten interpretiert und für Klassifikationsaufgaben eingesetzt werden können. Die hier üblicherweise verwendete Kostenfunktion ist die Kreuzentropie. Sie berechnet für jede Klasse k und die geschätzte Pseudo-Wahrscheinlichkeit $P(k|I)$ die Kreuzentropie über alle Klassen in K und das Eingangsbild I als

$$CE(k) = -\sum_{i}^{K} \mathbb{1}_i^k \log P(k|I) \tag{2.18}$$

wobei $\mathbb{1}_i^k$ die Indikatorfunktion beschreibt, die 1 annimmt, wenn $i = k$. Für Regressionsprobleme wird der quadratische Fehler (siehe Unterabschnitt 2.1.1) eingesetzt.

3 Stand der Forschung

Im Bereich des automatisierten Fahrens existieren bereits Ansätze, die linienförmige Merkmale – insbesondere Kartenmerkmale – mit gelernten Methoden detektieren können. Zu den relevantesten zählen Arbeiten aus der Fahrstreifenerkennung, die sich in dichte Schätzverfahren in Abschnitt 3.1 und dünnbesetzte, parametrische Schätzverfahren in Abschnitt 3.2 aufteilen. Eine gesonderte Betrachtung der Kreuzungsanwendungen bietet Abschnitt 3.3.

Unabhängig von der Algorithmik werden die Informationen aus Sensordaten gewonnen. Hierfür werden Bilder aus einer Frontkamera [OBB16, BMP17, PSL+18, SAN18, CZT19, GNB+19, HMLL19, Phi19, GCP+19, QWL20, TBP+21, LCZT21] eingesetzt. Rundum-Bilddaten fusioniert aus mehreren Kamerabildern [LWWZ22a, LWWZ22b], Bildsequenzen über mehrere Zeitschritte [BMH+18, LHM+19, ZVB21, OWP+22] und Luftbildaufnahmen [AFKR19] kommen ebenfalls zum Einsatz. Da Kartenmaterialien in einer Draufsicht repräsentiert werden, setzen [BMH+18, LHM+19, ZVB21, OWP+22] eine Projektion der Bilddaten in eine Draufsicht ein.

In einigen Arbeiten werden zudem Lidardaten verwendet. Diese werden in einer Draufsichtzellstruktur mit Intensität [HML+19, LHM+19, ZVB21, OWP+22] und Höhe [LHM+19, ZVB21, OWP+22] je Zelle beschrieben. Sowohl das HD-MapNet [LWWZ22a] als auch VectorMapNet [LWWZ22b] verarbeiten die Punktwolken des Lidars direkt mit einer geeigneten Architektur und transformieren die Daten mit einer Netzarchitektur ebenfalls in eine Draufsicht. Fusionierte Daten kommen daher bei [LHM+19, ZVB21, OWP+22, LWWZ22a, LWWZ22b] zum Einsatz.

Alle Ideen zum Einsatz fusionierter oder projizierter Daten lassen sich auf beliebige Algorithmik übertragen, da insbesondere neuronale Netze durch die Encoder-Decoder-Struktur modular handhabbar sind. Somit kann der Encoder bei jedem Ansatz um die nötige Fusion und Transformation erweitert werden,

sodass der Encoder eine geringe Relevanz bzgl. der Betrachtung geeigneter Repräsentationskonzepte aufweist.

3.1 Dichte Fahrstreifenschätzung

In diesem Abschnitt werden Ansätze zusammengefasst, deren Schätzung eine dichte Fläche beschreibt, aber über Punkte repräsentiert wird. Dies bedeutet, dass die Schätzung (typischerweise ein neuronales Netz) einen Linienzug oder eine Fläche als einzelne Punktschätzungen (z.B. Menge aus Pixeln) abtastet. Die Ausdehnung und der Zusammenhang dieser Punkte wird nicht explizit geschätzt, was zu Problemen führen kann, die später in dieser Arbeit genauer betrachtet werden.

3.1.1 Fahrstreifenfläche

Viele Ansätze schätzen die zu befahrende Fläche des eigenen Fahrstreifens [OBB16] oder die gesamte Straßenfläche [OBB16, CZT19, FWCL20]. Eine im Rahmen dieser Doktorarbeit entstandene Publikation [MSOS18] schätzt zusätzlich zum eigenen Fahrstreifen weitere Klassen wie die Nachbarfahrstreifen in gleiche Richtung sowie die der Gegenrichtung.

Alternativ gibt es Ansätze, die nicht die physischen Grenzen des Fahrstreifens klassifizieren, sondern einen möglichen Fahrkorridorbereich [BMP17]. Diese Korridore werden in einem schwach überwachten Lernverfahren aus den zukünftigen Positionen des Fahrzeugs gewonnen und repräsentieren dabei den lateralen Bereich zwischen den Rädern des Fahrzeugs. Diese Korridore sind dadurch schmaler als der tatsächliche Fahrstreifen. Für Verhaltensentscheidungen und Planungsalgorithmen ist diese Art der Darstellung nicht ausreichend, da keinerlei Information über Fahrstreifenwechsel oder die tatsächliche Fahrbahnbreite gegeben sind.

Typischerweise werden Ansätze mit CNNs als semantische Segmentierung (siehe Unterabschnitt 2.4.3) umgesetzt, bei der jeder Pixel im Bild klassifiziert wird. Die erfolgreichsten Ansätze unterscheiden sich dabei durch die Verwendung von fusionierten Eingabedaten [CZT19] und ausgefeilteren Netzstrukturen. Allen ist gemein, dass eine Encoder-Decoder-Struktur (siehe Un-

terabschnitt 2.4.2) eingesetzt wird, bei der ein beliebiger Backbone mit einem dekodierenden Frontend verknüpft wird, das eine pixelweise Klassifikation der Straßen- oder Fahrstreifenfläche vornimmt.

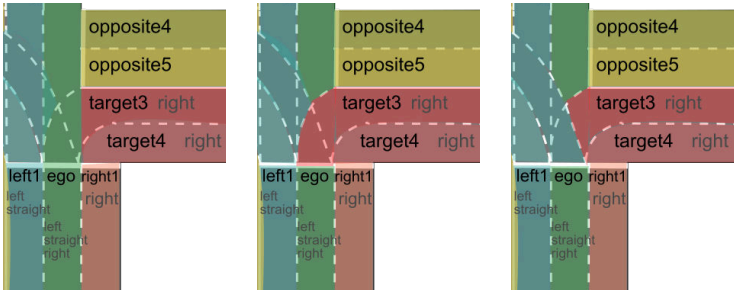
Mit dieser punkt- bzw. pixelweisen Schätzung lassen sich zwar Bereiche gleicher Klassifikation bestimmen, die Netze selbst inferieren dabei jedoch keine Flächen- oder Instanzinformationen, sondern betrachten jedes Pixel separat. Sie lernen die Flächen also auch nicht explizit [AFKR19, GNB+19]. Durch diesen Umstand können Prädiktionen entstehen, die den einfachen geometrischen Annahmen eines Fahrstreifens widersprechen, wie bspw. Abbildung 3.1 zeigt.

Abbildung 3.1: Fehlerhafte semantische Segmentierung der Fahrstreifen [MSOS18] auf dem Argoverse-Datensatz (Training auf Cityscapes).

Darüber hinaus bieten sämtliche Ansätze, die ein eindimensionales Klassifikationssystem vorweisen, keine geeignete Repräsentation für Kreuzungen an. Hier bedarf es einer hierarchischen Klassenstruktur (siehe Abbildung 3.2), die kreuzende Fahrstreifen unterscheiden kann. Zudem sollte eine Information über die Fahrtrichtung jedes Fahrstreifens kodiert werden. Dies hat jedoch bisher keine erfolgreiche Anwendung gefunden.

3.1.2 Fahrstreifenränder

Semantische Segmentierung wird alternativ auf die Pixel angewendet, die auf den Fahrstreifenrand (im Gegensatz zur Fläche) fallen. Zwei Ansätze [AF-



(a) Visualisierung des Ego-Fahrstreifens: grüne Fläche verläuft durch die Kreuzung

(b) Visualisierung des Rechtsabbiegers: rote Fläche startet bereits an der Haltelinie

(c) Visualisierung des Linksabbiegers: blaue Fläche startet bereits an der Haltelinie

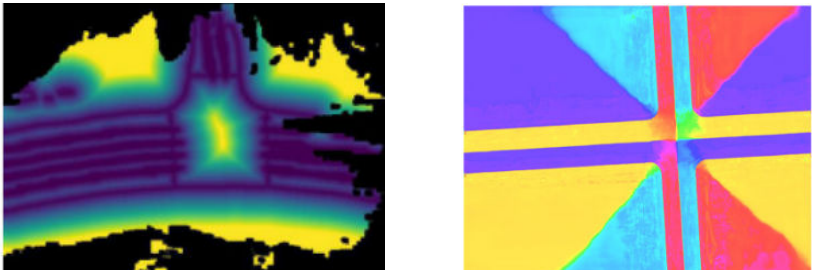
Abbildung 3.2: Drei Varianten zur Klassifikation der Fläche einer Kreuzung. Von der Haltelinie des Ego-Fahrstreifens führen zu allen drei Richtungen Fahrstreifen aus der Kreuzung hinaus. Alle Fahrtrichtungen teilen sich denselben Fahrstreifen, der sich ab der Haltelinie aufspaltet und somit überlagert. Dies ist nur mit einem hierarchischen Klassensystem als semantische Segmentierung abbildbar.

KR19, LWWZ22a] beschränken sich dabei auf die Erkennung der tatsächlichen Markierungen und ggf. deren Klasse. Diese alleine bilden jedoch noch keine ausreichend nützlichen Informationen, insbesondere für nicht-markierte Straßen. Andere Ansätze schätzen daher sämtliche sichtbaren sowie nicht explizit sichtbaren Begrenzungen eines Fahrstreifens [PSL+18, SAN18, BMH+18, GNB+19, HMLL19, Phi19, LCZT21, LWWZ22a].

Ein Problem der semantischen, pixelweisen Segmentierung als Fahrstreifenrand sind Kreuzungen, bei denen sich die Fahrstreifenränder überschneiden, zu einem fusionieren oder trennen. Hier müsste erneut für ein einzelnes Pixel eine Mehrdeutigkeit geschätzt werden, die nicht vorgesehen ist. Zudem erfordert die Angabe eines Fahrstreifens als unstrukturierte Pixelmenge eine aufwändige Nachverarbeitung, um daraus verwendbare Fahrstreifen für die Planung zu berechnen, insbesondere, wenn sich mehrere Fahrstreifen kreuzen oder berühren. Diese Nachverarbeitung lässt sich vereinfachen durch eine Schätzung einer weiteren Klassifikation: der Richtung jedes Pixels [LWWZ22a] oder der Schätzung des Nachfolgepixels [Phi19].

3.1.3 Distanzbeschreibung

Eine weitere dichte Beschreibung einer Fahrstreifengeometrie ist die pixelweise Distanzbeschreibung (siehe Abbildung 3.3). Hier wird pixelweise eine Regression vorgenommen, die den Abstand [BMH+18, OWP+22] oder den Abstandsvektor [LHM+19, HML+19] zum nächsten Fahrstreifenrand beschreiben soll. Diese Form der Darstellung erleichtert eine Nachverarbeitung, ist jedoch in kreuzenden Szenarien nicht eindeutig, sodass bspw. Ort u.a. [OWP+22] den Kreuzungsbereich explizit ausschließen.



(a) [OWP+22] (b) [LHM+19]

Abbildung 3.3: Distanzkarten in verwandten Arbeiten

3.1.4 Diskussion

Für die Weiterverwendung im automatisierten Fahren, bieten dichte Ansätze einen guten Einstieg in die Thematik, lösen die eigentliche Aufgabe jedoch nicht vollständig. Eine Verdichtung der Flächen- bzw. Randinformation hin zu einem planbaren Linienzug muss in jedem Fall erfolgen, sodass die im folgenden Abschnitt vorgestellten Ansätze diesen klar vorzuziehen sind.

Zusätzlich wird mit einer pixelweisen Beschreibung eine dichte Prädiktion geschätzt, die sehr großer Netzarchitekturen bedarf und damit verbunden große Inferenzzeiten aufweist.

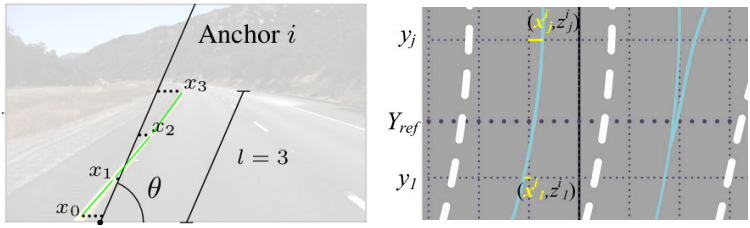
3.2 Dünnbesetzte, Parametrische Fahrstreifenschätzung

3.2.1 Anker

Mit der Veröffentlichung des TuSimple Datensatzes und Benchmark[1] wurde die Fahrstreifenrandschätzung auf Autobahnszenarien deutlich vorangetrieben. Um die Fahrstreifen im Frontbild zu bestimmen, wird für bestimmte Bildzeilen die Position der Pixel, die auf einem Fahrstreifenrand liegen, geschätzt [QWL20, LCZT21, ZHL+22].

Mit der Annahme, dass Fahrstreifen vom unteren Bildbereich zum Horizont verlaufen, werden aus diesen Punkten zusammenhängende Linienzüge konstruiert. Diese Annahme ist insbesondere im urbanen Raum unzutreffend, da Kreuzungsarme durch horizontal verlaufende Linien beschrieben werden müssen.

Eine Weiterentwicklung des Bildzeilenprinzips sind Ankerpunkte wie in Abbildung 3.4. Hier werden feste entlang der Fahrtrichtung definierte Ankerlinien im 2D- [SAN18, TBP+21, LLHY20] bzw. 3D-Raum [GCP+19] definiert und die orthogonalen Abweichungen von diesen geschätzt.



(a) [TBP+21]: Abweichung des Fahrstreifenrandes (grün) von einem Anker definiert über die Orientierung θ.

(b) [GCP+19]: Abweichung x_j^i der Mittellinie (blau) von einem Anker i (gepunktet, vertikal) an definierten Stützpunkten y_j.

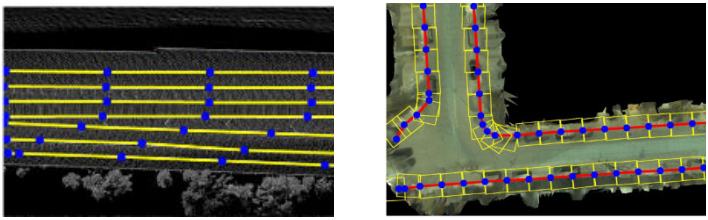
Abbildung 3.4: Ankerdarstellungen in verwandten Arbeiten.

[1] Tusimple `https://github.com/TuSimple/tusimple-benchmark`

Durch die Verwendung von Ankern, die entlang der Fahrtrichtung bzw. vertikal im Bild definiert sind, lassen sich ebenso keine Querverbindungen an Kreuzungen darstellen. Dies ist darauf zurückzuführen, dass für jede Position entlang der Anker nur ein Stützpunkt beschrieben wird. Durch die Einführung einer Mehrfachbelegung der Anker [GCP+19] wird die Aufspaltung und Zusammenführung von Fahrstreifen ermöglicht. Horizontale Linien sind jedoch nicht praktikabel realisierbar.

3.2.2 Rekurrente neuronale Netze

Die rigide Struktur eines ffNN lässt sich mit einer rekurrenten Architektur [HML+19, LHM+19], umgehen (siehe Unterabschnitt 2.4.5), indem bspw. die Stützpunkte eines Linienzugs iterativ prädiziert werden. Abbildung 3.5 visualisiert zwei dieser Beispiele.



(a) [HML+19] (b) [LHM+19]

Abbildung 3.5: Beispielergebnisse iterativer Verfahren

Um ein RNN einzusetzen, muss zunächst eine geeignete Repräsentation der Eingangsdaten generiert werden. Hierfür lernen die Modelle zunächst eine Zwischenrepräsentation, die Distanzkarten [LHM+19, HML+19] und/oder Startpunkte [LHM+19] als binäre Klassifikation darstellen. Auf Basis dieser Informationen werden iterativ die Stützpunkte geschätzt. [HML+19] nutzt eine Regression auf dem Vektor zum nächsten Stützpunkt und eine Klassifikation in Split-/Mergepunkte. [LHM+19] wendet eine semantische Segmentierung auf einzelne Bildausschnitte an und schätzt so die Position des nächsten Stützpunktes.

Die graphenähnliche Repräsentation dieser Ansätze ist vielversprechend, da teilweise zusätzlich teilende und fusionierende Fahrstreifen geschätzt werden

können [HML+19]. Die rekurrenten Netze führen jedoch zu einer nicht konstanten Ausführungszeit, verschwindenden Gradienten im Trainingsprozess und hohen Inferrenzzeiten. Der Ansatz von Liang u.a. [LHM+19] konnte zudem in einer Masterarbeit [Zan21] mit öffentlichen Datensätzen nicht ansatzweise reproduziert werden. Der Ansatz von [HML+19] wird rein in Autobahnszenarien angewandt. Eine Fahrtrichtung der geschätzten Linienzüge wird von keinem der bisher genannten Ansätze umgesetzt.

3.2.3 Graphenbasierte neuronale Netze

Aufgrund der Struktur der Linienzüge kann zur Schätzung des Zusammenhangs zwischen Liniensegmenten ein graphenbasiertes neuronales Netz (GNN) [ZVB21] angewendet werden. Dafür wird ein initialer Graph benötigt, der die Szene bestehend aus Linienstücken beschreibt. Zürn u.a. [ZVB21] verwenden hierfür einen Region-Proposal-Ansatz, der rechteckige Objekthypothesen aufstellt. Die Zentrumspunkte der Boxhypothesen werden als Stützpunkte verwendet, zwischen denen das GNN eine Verbindung klassifiziert. Parallel wird eine dichte Richtungskarte geschätzt, um die Stützpunkte in einer Nachverarbeitung mit einem Richtungsattribut versehen zu können.

Die Verarbeitung der Einzelhypothesen mit einem Graphen hin zu verbundenen Linienzügen ist für die Anwendung zielführend. Ein Region-Proposal-Ansatz als Detektor der Hypothesen ist hingegen sehr zeitaufwändig und durch den iterativen Charakter nicht in konstanter Komplexität lösbar. Die Verwendung von rechteckigen Hypothesen ist mit Hinblick auf die linienförmige Struktur der Detektion nicht sinnvoll. Die Ergebnisse zeigen Schwachstellen insbesondere in kreuzenden Szenarien.

3.2.4 Transformer

Das VectorMapNet [LWWZ22b] ist ein Ansatz, der die Sequenz eines Linienzugs mit einem Transformer (siehe Unterabschnitt 2.4.6) umsetzt. In einem zweischrittigen Verfahren (siehe Abbildung 3.6) wird zunächst eine grobe Beschreibung des Linienzugs über zwei bis vier Schlüsselpunkte bestimmt und anschließend zu jedem dieser Linienzugobjekte ein Linienzug geschätzt. Die Koordinaten des Linienzugs werden auto-regressiv generiert, sodass die

Schätzung nicht parallel stattfinden kann. Lediglich unterschiedliche Instanzen können unabhängig voneinander und damit parallel geschätzt werden.

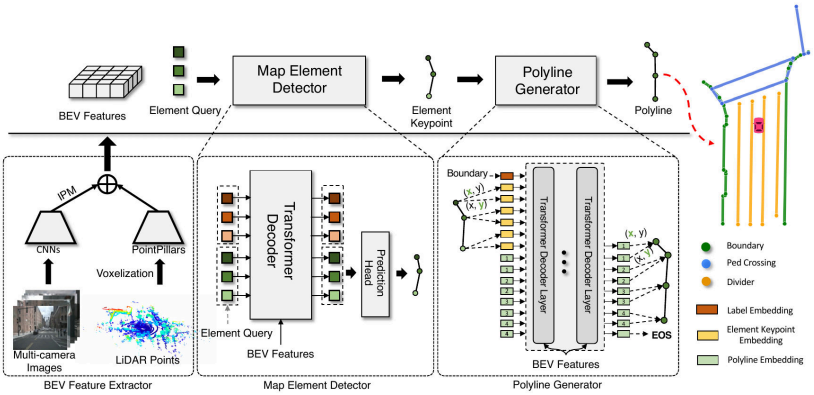


Abbildung 3.6: Transformer-Ansatz für Linienzüge [LWWZ22b]

3.3 Kreuzungen

Die Forschung im Bereich der Fahrstreifenerkennung betrachtet häufig ausschließlich nicht-kreuzende Bereiche der Straße und lässt die Komplexität von Kreuzungen unbeachtet. Zur Bestimmung von Kreuzungsgeometrien ist es wichtig, dass ein Algorithmus zusammenführende Fahrstreifen repräsentieren und trotz Verdeckungen eine valide Schätzung abgeben kann. Viele Ansätze scheitern jedoch bereits daran, dass quer zur Fahrtrichtung verlaufende Linien nicht repräsentierbar sind.

Von den bisher genannten Arbeiten, die Sensordaten direkt mit einem neuronalen Netz verarbeiten, zeigen [LHM+19, LWWZ22a, LWWZ22b] die Anwendung auf einer vollständigen Kreuzungsschätzung. Es wird dabei eine aufwändige Verarbeitung mit einer iterativen Netzarchitektur [LHM+19, HML+19, ZVB21] oder eine einfache Nachverarbeitung mit aufwändigen Zwischenrepräsentationen [LWWZ22a] eingesetzt. Liang u.a. [LHM+19] schätzen zudem keine Fahrstreifen, sondern nur Straßenränder.

In der Literatur wird für Kreuzungen eine modellbasierte Schätzung vorgenommen [GLW+14, RZB17, BB19, Pog19]. In [MWLS19] hat die Autorin ebenfalls ein Topologie- und Fahrstreifenmodell vorgestellt.

Sowohl die eigenen Arbeiten [MWLS19, MWL20] als auch [GLW+14, RZB17, BB19] verwenden aufgrund der gemischt-ganzzahligen Problemstellung einen sampling-basierten Schätzer, das Markow-Ketten-Monte-Carlo-Verfahren. Zur Schätzung des Topologiemodells werden eine Vielzahl von Modellvarianten M mit Messungen aus Sensordaten S verglichen und das Modell mit der maximalen A-Posteriori-Wahrscheinlichkeit $P(M|S,V)$ unter Annahme von A-Priori-Wissen V gewählt. Eine direkte Optimierung ist hier aufgrund der Mischung aus ganzzahligen und kontinuierlichen Variablen nicht immer möglich. Bisher liefern diese Verfahren keine zufriedenstellenden Ergebnisse, die in Echtzeit eine Schätzung abgeben können. Auch in eigenen Arbeiten im Rahmen dieser Doktorarbeit konnte das Problem nicht vollständig mit modellbasierten Ansätzen gelöst werden [Wal18, Wil18, MWLS19, Oes19, MWL20, Wal20], da es primär an geeigneten Detektoren mangelt.

Typisch für diese Verfahren ist die Verwendung von vorverarbeiteten Detektionen (bspw. Markierungen und Bordsteine), die für die Modellschätzung aufbereitet und repräsentiert werden. Diese Zwischenrepräsentation stellt eine Einschränkung des möglichen Lösungsraums dar. Idealerweise zieht ein Ansatz das Wissen direkt aus den Sensordaten, um eine Modellschätzung abgeben zu können.

Bisherige Ansätze verwenden dazu überwiegend punktbasierte Detektionen, aufgrund des Mangels an linienförmigen Detektoren. Diese Repräsentation ist nicht ideal, wenn die zu detektierenden Objekte Linienzüge darstellen. Bspw. die (Aus-)Richtung einer Linie lässt sich mit Punktdetektionen nicht repräsentieren.

4 Räumlich diskretisierte Linienschätzung

Um eine geeignete Lösung für Linienschätzung im Kontext automatisierter Fahrzeuge vorstellen zu können, wird in Abschnitt 4.1 zunächst das Problem konkretisiert und Anforderungen an die Lösung gestellt. In Abschnitt 4.2 wird die Basisidee der YOLO-Architekturen diskutiert und auf Linienzüge übertragen. Die Details des Konzepts werden in Abschnitt 4.3 bis Abschnitt 4.6 vorgestellt. Teile dieses Konzepts wurden bereits in [MSPS21] vorgestellt.

Zunächst wird die Architektur des neuronalen Netzes in Abschnitt 4.3 konstruiert. Dabei sind insbesondere die Skalierung der Zellstruktur und das rezeptive Feld wichtige Aspekte.

Nachdem die grobe Struktur des Ansatzes diskutiert wurde, beschreibt Abschnitt 4.4 vier mögliche Liniendarstellungen, die als Netzausgabe verwendet werden können.

In Abschnitt 4.5 wird ein wichtiges Konzept für das Verfahren vorgestellt: die Zuordnung der GT-Linien zu Prädiktoren. Es wird geklärt, wie ein GT-Element für das Training einer Prädiktion zugeordnet werden kann und welche Konsequenzen diese Konzepte haben.

In Abhängigkeit von der Zuordnung formuliert Abschnitt 4.6 die Trainingskosten und betrachtet zudem die Gewichtung der einzelnen Teilkosten nach Kendall u.a. [KGC18].

4.1 Problemdefinition

Gegeben sind Sensorinformationen eines automatisierten Fahrzeugs. Es soll ein Ansatz entworfen werden, der Kartenmerkmale, insbesondere Fahrstreifen,

auf einer kausalen Datengrundlage detektieren kann. Da der Fokus dieser Arbeit zunächst auf der Repräsentation von Linienzügen mit neuronalen Netzen liegt, sollen nur Einzelbilder verarbeitet werden. Eine Lösung für Bildsequenzen unterscheidet sich lediglich in der Aggregation im Encoder, nicht jedoch im eigentlichen Algorithmus selbst.

Die Anwendung im urbanen Raum führt zu den folgenden Anforderungen an die Repräsentation des Modells:

- kreuzende Linienstücke sind zwei eindeutig separierte Instanzen,
- die Fahrtrichtung ist kodiert,
- Verkehrsregelattribute (bspw. durchgezogene und gestrichelte Linien) sind kodiert,
- es wird keine strikte Modellannahme über den Verlauf der Linienzüge getroffen.

Das Modell soll nicht manuell vorgegeben, sondern als ein neuronales Netz gelernt werden, um inhärente Informationen direkt aus den Sensordaten extrahieren zu können. Gelernte Modelle haben sich insbesondere im Wahrnehmungsbereich überlegen gegenüber menschlich gestalteten Algorithmen und damit beschränkten Modellen gezeigt. Zudem soll ein vorwärtsgerichtetes Netz zum Einsatz kommen, um die Rechenkomplexität des Verfahrens konstant zu halten. Bei rekurrenten Netzen und anderen iterativen Verfahren variiert die Komplexität hingegen mit steigender Anzahl der Objekte.

Für das automatisierte Fahren sind insbesondere Kamerainformationen relevant, da verkehrsregelnde Informationen – wie Fahrstreifen – menschenles- und -interpretierbar dargestellt werden. Andere Sensoren, wie Lidar und Radar, sind für Fahrstreifen von zweitrangiger Bedeutung, da sie primär den physisch befahrbaren Bereich (Wände, Bordsteine, ...) jedoch weniger die verkehrsregelnden Elemente (Markierungen, Schilder, ...) und ihre Bedeutung aufnehmen können. Für die Verarbeitung der 2D-Darstellung von Kamerabildern bieten sich insbesondere faltende neuronale Netze (CNNs) an. Dies kann auf 2D-Projektionen von Lidar und Radar sowie auf eine Draufsichtsprojektion der Kameradaten übertragen werden.

Schließlich stellt ein automatisiertes System die Anforderung einer weichen Echtzeit mit üblicherweise mindestens 10 Bildern pro Sekunde.

4.2 Echtzeitfähige Detektion von Linien

Die Struktur von komplexen Kreuzungsmodellen ist für den Einsatz von vorwärtsgerichteten neuronalen Netzen nicht ideal. Insbesondere die Tatsache, dass Kreuzungen eine unterschiedliche Anzahl an Armen oder Fahrstreifen aufweisen und damit nicht einfach über eine feste Anzahl an Variablen beschrieben werden kann, sind Kreuzungen auch mit den fixen Prädiktionstensoren eines vorwärtsgerichteten Netzen nicht direkt abbildbar (siehe Abschnitt 2.4).

Die YOLO-Familie [RDGF16, RF17, RF18, BWL20] stellt eine Reihe von Publikationen dar, die eine dynamische Anzahl Objekte in Bildern detektieren – ohne ein aufwändiges mehrstufiges oder iteratives Verfahren. Stattdessen wird ein vorwärtsgerichtetes neuronales Netz präsentiert, bei dem das Eingangsbild mit einer Gitterstruktur überlagert wird. Jede dieser Gitterzellen schätzt Hypothesen für eine Objektdetektion (siehe Abbildung 4.1), die über die Position, die Höhe und Breite der Umrandungsbox des Objekts sowie die Konfidenz beschrieben wird. Zusätzlich wird ab [RF17] eine Klassifikation je Objekt geschätzt.

Somit ergibt sich eine einfache Netzarchitektur, die mit vorwärts verknüpften Schichten in Echtzeit Ergebnisse inferiert, leicht konvergiert und keine besonderen Trainingsstrategien benötigt. Zudem bieten diese Verfahren eine konstante Rechenkomplexität unabhängig von der tatsächlichen Anzahl der Objekte in der Szene.

Mit der originalen YOLO-Repräsentation sind beschränkt ausgedehnte Objekte gut abbildbar. Ein Linienzug beschreibt hingegen keine Fläche. Eine Umrandungsbox eines Linienzuges, der bspw. einen abbiegenden Fahrstreifen beschreibt, umfasst leicht das gesamte Bild und ist somit wenig informativ für den eigentlichen Verlauf.

Als Alternative können an Stelle der Umrandungsboxen komplette Linienzüge geschätzt werden. Dazu müsste das Netz eine Beschreibung des gesamten Zuges prädizieren. Das Grundproblem eines Linienzuges für die Schätzung

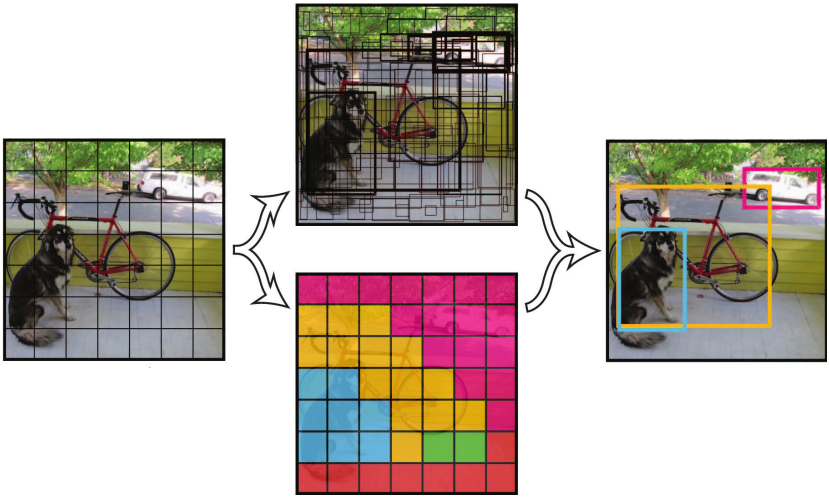


Abbildung 4.1: Schematische Darstellung der Objektdetektion mit YOLO. Im Eingangsbild wird eine Zellstruktur definiert, in der dann Umrandungsboxen, eine Konfidenz und eine Klasse geschätzt wird. Aus diesen Angaben wird die finale Prädiktion bestimmt. Ursprüngliche Bildquelle: [RDGF16]

ist jedoch, dass er je nach Komplexität eine variable Anzahl an Stützpunkten benötigt.

Stattdessen wird in dieser Arbeit eine Diskretisierung vorgeschlagen, die die Vorteile der YOLO-Architektur einsetzt: Die Linienzüge werden durch das Zellgitter unterteilt und somit nicht mehr als Gesamtobjekt, sondern als Linienstücke diskretisiert prädiziert.

Durch die Abtastung der Linienzüge durch die Zellkanten begrenzt sich jedes Linienstück automatisch auf die Zelle. Aus diesen Teilstücken bildet sich dann der gesamte Linienzug, der flexibel auch Kreuzungen und komplexe Topologien abbilden kann. Im Gegensatz zu verwandten Arbeiten wird damit eine Architektur entworfen, die eine 2D-Diskretisierung der Szene ermöglicht und gleichzeitig mit einer sehr einfachen Architektur auskommt.

Wie in Abschnitt 4.1 beschrieben muss die Repräsentation in der Lage sein, mehrere ggf. kreuzende Linienstücke in einer Zelle darzustellen. Dazu kann das Konzept der Hypothesenschätzung ausgenutzt werden. Jede Zelle schätzt

nicht nur ein Linienstück, sondern stellt mehrere sich ggf. kreuzende oder überlagernde Linienhypothesen auf. Im Folgenden wird jede Komponente der Prädiktion einer Zelle als *Prädiktor* $p \in P$ bezeichnet. Im Netz entspricht dies je einer Position im Ausgabetensor, an der das Netz eine Liniensegmenthypothese ℓ prädiziert. Hier gilt es daher eine geeignete geometrische Beschreibung der Hypothesen zu ermitteln (Abschnitt 4.4).

Die A-Priori-Verteilung der Liniensegmente wird hierbei durch sogenannte *Anker* modelliert (Abschnitt 4.5), relativ zu denen die Prädiktoren eine Abweichung bzw. eine Korrektur ausgeben. Die Anker ermöglichen darüber hinaus eine Spezialisierung der Prädiktoren, bspw. auf horizontale oder vertikale Liniensegmente. In verwandten Arbeiten zeigt sich, dass die Verwendung von Ankern einen großen Vorteil bringt [RF17, SAN18, GCP+19, TBP+21, LLHY20]. Durch das Einbringen von A-Priori-Wissen muss dabei nicht mehr die absolute Position einer Prädiktion geschätzt werden, sondern nur die Abweichung zu einem Anker.

Die zu lernenden Liniensegmente werden als *GT-Liniensegment* oder *Zielbeschreibung* (engl. target) bezeichnet. In einem Trainingsbeispiel existieren jedoch in einer Zelle durchaus mehr als ein tatsächlich richtiges GT-Liniensegment, sodass eine Kostenfunktion (Abschnitt 4.6) diese n-zu-m-Zuordnung zwischen Prädiktoren und GT-Linien explizit beachten muss. Im Sinne der Verantwortlichkeit, die Redmon u.a. bereits für Objektdetektion formuliert haben [RDGF16], sollte sich jeder Prädiktor für einen speziellen Linientyp während des Trainings ausbilden. Das bedeutet konkret, dass bspw. ein auf horizontale Segmente spezialisierter Prädiktor nur dann durch eine hohe Konfidenz aktiviert wird, wenn in der Szene horizontale Linien vorhanden sind. Dazu wird vorgeschlagen, die GT-Liniensegmente den Prädiktoren auf Basis einer euklidischen Distanz zuzuordnen.

Um aus der Vielzahl an prädizierten Hypothesen abschließend die richtigen auszuwählen, kann eine Non-Maximum-Suppression (NMS) angehängt werden. Diese filtert die Hypothesen, sodass lediglich valide übrig bleiben. Für einige Anwendungen ist es zudem wichtig, die Linien als verbundene Linienzüge darzustellen. Sowohl anwendungsspezifische NMS als auch Verbindungsmöglichkeiten werden in Kapitel 5 vorgestellt.

Zur Lösung der gegebenen Problemstellung kann somit der YOLO-Ansatz auf Linienzüge übertragen werden. Damit können Kartenbestandteile wie Markie-

rungen, Bordsteine, Fahrstreifenränder oder Fahrstreifenmittellinien repräsentiert werden. Da die YOLO-Architektur zudem eine Klassifikation je Objekt bereitstellt, können mit der vorgeschlagenen Architektur alle Kartenbestandteile gleichzeitig geschätzt und klassifiziert werden. Da Fahrstreifen und insbesondere deren Mittellinien die größten Herausforderungen bieten, wird die Konzeption in dieser Arbeit mit Fokus auf Fahrstreifenmittellinien wie in Abbildung 4.2 beschrieben. Sie ist jedoch auf jede Art von Linienzug übertragbar.

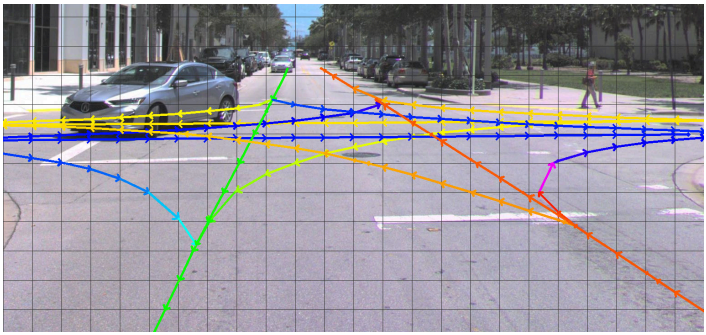


Abbildung 4.2: Eine Beispielkreuzung aus dem Argoverse-Datensatz mit den Fahrstreifenmittellinien. Die Farben visualisieren die Orientierung der Linienstücke und geben somit einen Eindruck der Fahrtrichtung.

4.3 Architektur

Die erste Version der YOLO-Architektur [RDGF16] ist ein CNN und besteht aus 24 faltenden Schichten gefolgt von zwei vollständig verknüpften Schichten. Dieses Netz wird in [RF17] durch ein sogenanntes Darknet-19 mit 19 faltenden Schichten ersetzt. In [RF18] wird die Architektur auf 53 faltende Schichten (Darknet-53) erhöht und um das Konzept einer Merkmalspyramide (siehe Unterabschnitt 2.4.2) ergänzt. Hier wird nicht mehr nur eine Gitterstruktur prädiziert, sondern drei aufeinander aufbauende, skalierte Strukturen, die jeweils Objekt-Hypothesen aufstellen. Im Gegensatz zu Darknet-19 weist das Darknet-53 Sprungverbindungen (siehe Unterabschnitt 2.4.2) zwischen Eingangsschichten und prädizierenden Schichten auf. In [Sku20] wurde das Darknet-53 für Liniendetektion untersucht. Es konnte jedoch keine Verbesse-

rung gegenüber dem Darknet-19 erzielt werden, sodass in dieser Arbeit das Darknet-19 verwendet wird.

4.3.1 Encoder

Der Encoder des Darknet-19 (siehe Abbildung 4.3) komprimiert die eingehenden Bildinformationen, mit 19 faltenden und fünf Pooling-Schichten, um den Faktor 32. Für ein Bild von bspw. 320×640 Pixel werden die Informationen auf eine Zellstruktur 10×20 Zellen verdichtet, die jeweils über ein Embedding mit 1024 Variablen beschrieben werden.

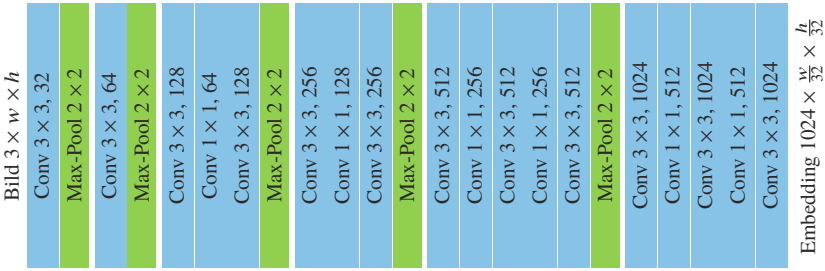


Abbildung 4.3: Der Encoder des Darknet19 mit 18 faltenden Schichten (blau) und fünf Pooling-Schichten (grün). Weitere Details finden sich in Abschnitt A.1.

Der Encoder des Netzes ist generisch austauschbar, solange es sich um einen komprimierenden Encoder handelt. Genauso können Architekturen verwendet werden, die bspw. Lidardaten verarbeiten oder Bild- und Lidardaten fusionieren oder eine Transformation in eine Draufsicht lernen (siehe Kapitel 3). Der wichtige Aspekt ist lediglich die Komprimierung auf ein repräsentatives Embedding.

Merkmalspyramiden sind für Liniendetektionen ebenfalls nicht sinnvoll, da bei mehreren Skalen nur die Mehrdeutigkeiten erhöht werden, nicht jedoch die Detektionen verbessert. Da Objekte über die Zelle hinaus reichen, liefern unterschiedliche Skalierungen unterschiedliche Objektgrößen, sodass sich die Merkmalsextraktion im Netz darauf spezialisieren kann. Liniensegmente sollten hingegen wie oben beschrieben auf die Ausmaße einer Zelle begrenzt sein.

Durch Merkmalspyramiden und mehrere Skalen würden somit nur genau die gleichen Liniensegmente mit unterschiedlichen Abtastraten abgebildet werden. Somit reicht die letzte Schicht einer Merkmalspyramide (mit der feinsten Auflösung des Gitters) aus, um alle Linienzüge zu prädizieren.

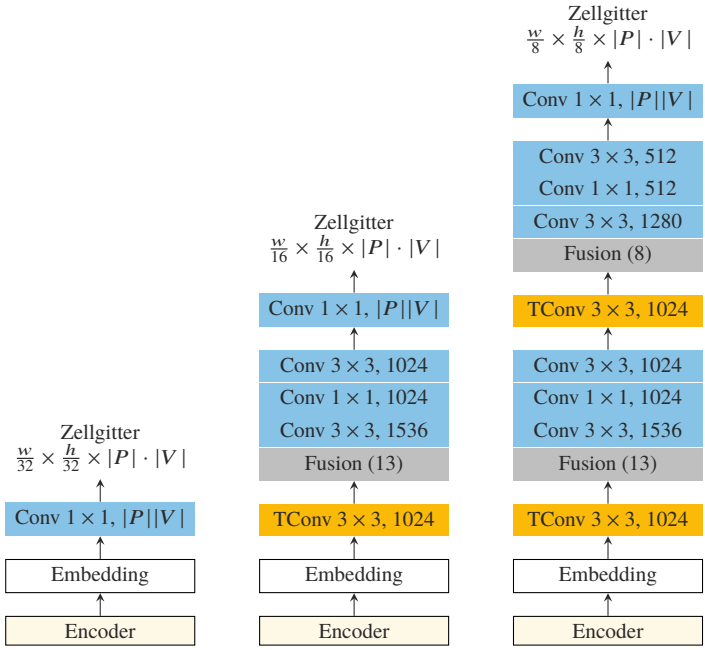
4.3.2 Decoder und Skalierung

Mit dem Darknet-19-Backend und dem YOLO-Frontend produziert die Ausgabeschicht ein $n \times m$ Zellgitter mit jeweils $|P|$ Prädiktoren, die jeweils eine Hypothese beschreiben. Dazu besteht der Decoder aus einer faltenden Schicht, die das Embedding auf das zu prädizierende Zellgitter mit jeweils Zellen der Größe $|P| \cdot |V|$ komprimiert (siehe Abbildung 4.4). $|V|$ beschreibt die Anzahl der Variablen, die nötig ist, eine Hypothese zu beschreiben. Die Anzahl der Prädiktoren pro Zelle $|P|$ bestimmt somit die maximale Anzahl an Liniensegmenten pro räumlichen Zellbereich, die das Netz schätzen kann.

Da die Prädiktion durch die Diskretisierung in räumliche Zellen ungenauer wird, ist die Auflösung der finalen Zellstruktur elementar. In dieser Arbeit werden daher verschiedene Skalierungsstufen untersucht, die zu einer Auflösung von 8×8 Pixel, 16×16 Pixel, 32×32 Pixel pro Zelle führen.

Um die Liniensegmente in höher aufgelösten Zellgittern aufbauend auf dem Darknet-19-Embedding schätzen zu können, wird die Architektur um weitere (transponierte) faltende Schichten erweitert (siehe Abbildung 4.4), sodass Zellstrukturen mit 8×8 Pixel und 16×16 Pixel Zellen prädiziert werden können. Das Zellgitter besteht dadurch automatisch aus $\frac{w}{8} \times \frac{h}{8}$ bzw. $\frac{w}{16} \times \frac{h}{16}$ Zellen abhängig von der Bildgröße mit $w \times h$ Pixel.

Um die Präzision durch die Komprimierung auf das Embedding nicht zu verlieren, werden Sprungverbindungen (siehe Unterabschnitt 2.4.2) zwischen den gleich großen Schichten im Encoder und Decoder verwendet. Hier werden die Ausgaben der Encoder-Schicht mit den Ausgaben der Decoder-Schicht zusammengeführt. Die vollständigen Architekturen mit allen Sprungverbindungen sind in Abschnitt A.1 dargestellt. Als Aktivierungsfunktion der verdeckten Schichten wird wie bereits bei den YOLO-Architekturen die LeakyReLu (siehe Unterabschnitt 2.4.4) eingesetzt.

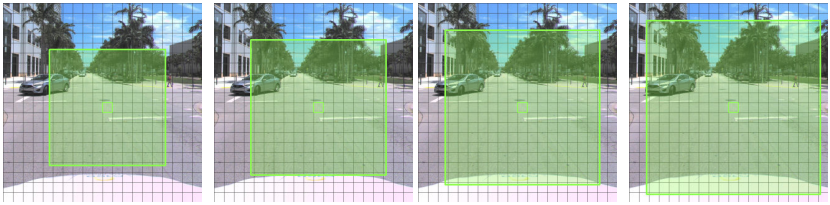


(a) 32×32 Pixel Zellgitter (b) 16×16 Pixel Zellgitter (c) 8×8 Pixel Zellgitter

Abbildung 4.4: Drei Dekoder-Varianten mit unterschiedlicher Auflösung pro Zelle. Faltende Schichten werden in Blau, transponierte faltende Schichten in Orange und Fusionsschichten in Grau visualisiert. Die Fusionsschichten bekommen zusätzlich Informationen aus den Sprungverbindungen, die die Merkmalskarten der achten bzw. 13. faltenden Schicht aus dem Encoder (siehe Abbildung 4.3) weitergeben. Weitere Details finden sich in Abschnitt A.1.

4.3.3 Rezeptives Feld

Für fahrstreifenbezogene Detektionen sollte das rezeptive Feld (siehe Unterabschnitt 2.4.1) möglichst groß gewählt werden, um die gesamte Szene bewerten zu können. Diese ist für die Interpretation von Fahrstreifen und insbesondere der Erkennung impliziter Merkmale wie der Mittellinie deutlich relevanter als bei einer Objekterkennung, da Verkehrsschilder, Verkehrsteilnehmer und andere Fahrstreifen bspw. Auskunft über die Fahrtrichtung geben. Genauso ist der Verlauf paralleler Markierungen ein wichtiger Hinweis für eine Detektion.



(a) $d = 1$
374 × 374 Pixel

(b) $d = 2$
436 × 436 Pixel

(c) $d = 3$
498 × 498 Pixel

(d) $d = 4$
560 × 560 Pixel

Abbildung 4.5: Rezeptive Felder für verschiedene Aufweitungen d visualisiert für eine Zelle von 32 × 32 Pixel in einem Bild von 640 × 640 Pixel aus dem Argoverse-Datensatz [WQA+21]

Für die Darknet-19 Architektur ist das rezeptive Feld 374×374 Pixel und damit deutlich kleiner als der sichtbare Bildbereich (siehe Abbildung 4.5). Durch eine Aufweitung der einzelnen faltenden Schichten (engl. dilation) lässt sich das rezeptive Feld um jeweils 62 Pixel vergrößern. Hier ist zu prüfen, ob diese Aufweitung tatsächlich im Training einen Vorteil bietet (siehe Experimente in Unterabschnitt 6.4.1).

4.4 Linienrepräsentation

Jede geschätzte Hypothese beschreibt ein Liniensegment als $\ell = (\mathbf{g}, \mathbf{k}, c)$. Es besteht aus einer geometrischen Beschreibung $\mathbf{g}$, einer optionalen Klassifikation $\mathbf{k}$ und einer Konfidenz c. Mit der Konfidenz werden nach erfolgreichem Training valide Prädiktionen schwellwertbasiert gefiltert.

Zur geometrischen Repräsentation der Liniensegmente bestehen mehrere Optionen. Eine geeignete Repräsentation definiert die Linien in den Koordinaten einer Zelle und ermöglicht die Berechnung einer differenzierbaren Distanz zwischen zwei Liniensegmenten, um eine geeignete Fehlerfunktion formulieren zu können.

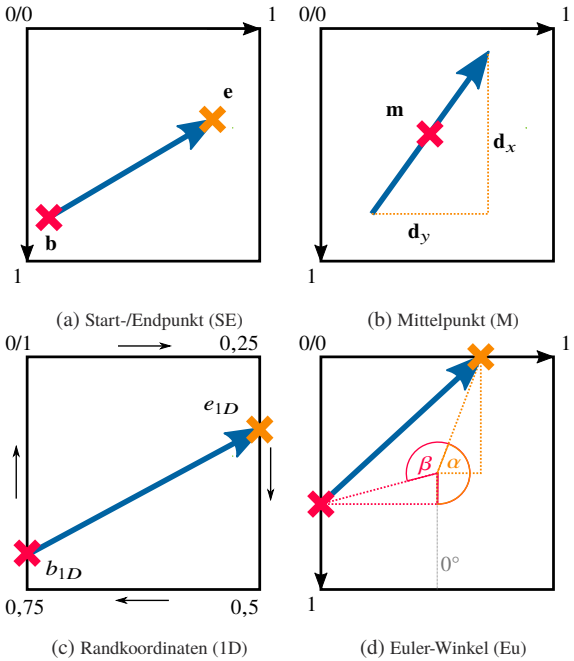


Abbildung 4.6: Vier verschiedene Repräsentationen zur Beschreibung eines Liniensegments innerhalb einer Zelle.

4.4.1 Kartesische Start- und Endpunkte

Bei Betrachtung von Linienzügen als Kombination von Stützpunkten kommt zunächst eine Repräsentation über kartesische Start- und Endpunkte (SE) infrage. Sie definiert ein Liniensegment ℓ_{se} über je einen Startpunkt $\mathbf{b}$ und einen Endpunkt $\mathbf{e}$ als

$$\ell_{se} := \begin{pmatrix} \mathbf{b} \\ \mathbf{e} \end{pmatrix} \quad \text{mit} \quad \mathbf{b}, \mathbf{e} \in [0,1]^2. \tag{4.1}$$

Der Ursprung des Koordinatensystems liegt dabei auf der oberen linken Ecke jeder Zelle. Zur Vereinheitlichung wird das Koordinatensystem so auf die Zelle normiert, dass die Zellbreite (und -höhe) genau 1 entsprechen. Abbildung 4.6a

zeigt die Repräsentation grafisch. Diese Darstellung bietet eine hohe Flexibilität, da Liniensegmente frei in einer Zelle definiert werden können.

In Anlehnung an [RDGF16] wird die euklidische Distanz zwischen den Start- und Endpunkten der Liniensegmente in der Prädiktion und der GT als Distanzmaß verwendet.

4.4.2 Mittelpunkt-Richtung-Koordinaten

Da die SE Abweichungen entlang der Liniendefinition genauso bestraft wie parallele Verschiebungen, bietet sich als Alternative eine Repräsentation an, die die Position und Rotation entkoppelt betrachtet. In Mittelpunkt-Richtung-Koordinaten (MR) wird jedes Liniensegment über

$$\ell_{md} := \begin{pmatrix} \mathbf{m} \\ \mathbf{d} \end{pmatrix} \quad \text{mit} \quad \mathbf{m} \in [0,1]^2, \mathbf{d} \in [-1,1]^2 \tag{4.2}$$

beschrieben, wobei $\mathbf{m}$ den Mittelpunkt und $\mathbf{d}$ den x- bzw. y-Anteil des Vektors darstellt. Die SE $\ell_{se} = (\mathbf{b}, \mathbf{e})^T$ werden mit

$$\mathbf{m} = \frac{\mathbf{b} + \mathbf{e}}{2} \tag{4.3}$$

$$\mathbf{d} = \mathbf{e} - \mathbf{b} \tag{4.4}$$

in den entsprechenden Raum konvertiert. Abbildung 4.6b gibt eine grafische Visualisierung der Linienrepräsentation.

Bei dieser Darstellung gilt es zu beachten, dass die Orientierung über den unskalierten x- bzw. y-Anteil beschrieben wird. Hier könnte alternativ der Kosinus-/Sinusanteil der Orientierung und eine Länge verwendet werden. Dies führt jedoch zum einen dazu, dass fünf statt vier Parameter optimiert werden müssen. Zudem ist die Inferenz der exakten Kosinus-/Sinusanteile voneinander abhängig, da nur ausgewählte Kombinationen $\cos^2 \alpha + \sin^2 \alpha = 1$ erfüllen. Idealerweise sollteb jedoch die Zielformulierungen unabhängig lernbar sein. Diese theoretische Vermutung konnte in Experimenten bestätigt werden: Mit einer Kosinus-/Sinus-/Längendarstellung konvergieren die Netze nur schwer bis gar nicht.

Falls für ein Training die Sigmoid-Funktion eingesetzt werden soll, so muss für die Orientierung $\mathbf{d}_x, \mathbf{d}_y$ die Sigmoid-Aktivierungsfunktion so skaliert werden, sodass ein Bereich von $[-1, 1]$ abgedeckt ist. Wird die Abweichung zu einem Anker gelernt, so ist sogar ein Bereich von $[-2, 2]$ notwendig. Ähnliche Ansätze verwenden jedoch die lineare Aktivierungsfunktion [SMAG19] für Richtungsangaben.

Hier wird ebenfalls die euklidische Distanz zwischen den geometrischen Beschreibungen der Liniensegmente in der Prädiktion und der GT verwendet.

4.4.3 Randkoordinaten

Da für verbundene Linienzüge die Segmente an den Zellkanten starten und enden, untersucht diese Arbeit zudem eine beschränkte Repräsentation aus Randkoordinaten (1D). Hier wird der Startpunkt b_{1D} und Endpunkt e_{1D} eines Liniensegments auf der Zellkante als im Uhrzeigersinn gemessenen Abstand entlang der Kante zur oberen linken Ecke definiert (siehe Abbildung 4.6c). Der Umfang jeder Zelle wird dabei ebenfalls auf 1 normiert, sodass

$$\ell_{1D} := (b_{1D}, e_{1D}) \quad \text{mit} \quad b_{1D}, e_{1D} \in [0, 1]. \tag{4.5}$$

Distanzfunktion

Die Distanzfunktion lässt sich als Distanz zwischen Start- und Endpunkt formulieren, mit

$$\delta_{1D}(\ell, \hat{\ell}) := (b_{1D} - \hat{b}_{1D})^2 + (e_{1D} - \hat{e}_{1D})^2. \tag{4.6}$$

Diese Distanzfunktion ist jedoch am Ursprung (linke obere Ecke) nicht stetig definiert, sodass Punkte nah an 0 und Punkte nah an 1 eine hohe Distanz zueinander aufweisen, obwohl sie im euklidischen Raum eine kurze Distanz aufweisen.

Durch eine geschickte Formulierung kann dieser Sprung vermieden werden. Dazu wird das Minimum aus der Distanz im Uhrzeigersinn $d_{1D}(\ell, \hat{\ell})$ und gegen den Uhrzeigersinn verwendet, sodass

$$\begin{aligned}\delta_{1D}(\ell, \hat{\ell}) := \min((b_{1D} - \hat{b}_{1D})^2, (1 - (b_{1D} - \hat{b}_{1D}))^2) \\ + \min((e_{1D} - \hat{e}_{1D})^2, (1 - (e_{1D} - \hat{e}_{1D}))^2).\end{aligned} \tag{4.7}$$

Diese ist zwar nicht global differenzierbar, jedoch lokal, sodass ein Gradientenabstieg praktisch umsetzbar ist.

4.4.4 Euler-Winkel

Zur Vermeidung von Sprüngen schlagen [SMAG19] Euler-Winkel (Eu) vor. Übertragen auf die hier vorliegenden Linien kann ebenso der Sprung vermieden werden. Ein Segment ist über einen Start- und Endpunkt auf dem Rand definiert. Jeder Punkt auf dem Rand einer Zelle wird beschrieben als Schnittpunkt zwischen dem Zellrand und einem durch den Winkel α bzw. β definierten Strahl (siehe Abbildung 4.6d). Das Koordinatensystem ist dabei auf der Zellmitte aufgestellt, sodass das Netz ein Liniensegment als

$$\ell_{eu} := \begin{pmatrix} \cos(\alpha) \\ \sin(\alpha) \\ \cos(\beta) \\ \sin(\beta) \end{pmatrix} \quad \text{mit} \quad \ell_{eu} \in [0, 1]^4 \tag{4.8}$$

lernt. Hier ist es wichtig hervorzuheben, dass nicht der Winkel direkt gelernt wird, sondern der Kosinus- bzw. Sinusanteil des Winkels. Zur Verbesserung der Abhängigkeit zwischen dem Kosinus- und Sinusanteil, kann mit $\cos^2 \alpha + \sin^2 \alpha = 1$ eine zusätzlich Regularisierung im Training eingeführt werden.

Distanzfunktion

Durch die gegebene Periodizität der Darstellung ist die Distanz kontinuierlich definierbar zwischen den einzelnen Komponenten als

$$\delta_{eu}(\ell, \hat{\ell}) = \left(\begin{pmatrix} \cos\alpha \\ \sin\alpha \\ \cos\beta \\ \sin\beta \end{pmatrix} - \begin{pmatrix} \cos\hat{\alpha} \\ \sin\hat{\alpha} \\ \cos\hat{\beta} \\ \sin\hat{\beta} \end{pmatrix} \right)^2 . \tag{4.9}$$

Aktivierungsfunktion

Für diese periodische Darstellung wird der Tangens Hyperbolikus als Aktivierungsfunktion, wie in [Sku20] vorgeschlagen, eingesetzt.

4.4.5 Diskussion der Linienrepräsentation

In einer Publikation zu dieser Arbeit [MSPS21] wurden die Euler-Winkel, Randkoordinaten und kartesische Start- und Endpunkte gegeneinander verglichen. Tabelle 4.1 zeigt die Resultate auf dem TuSimple-Datensatz. Die Mittelpunkt-Richtung-Koordinaten wurden für die vorliegende Arbeit erst neu entwickelt und sind daher noch nicht Teil dieses ersten Vergleichs. Hier zeigte sich kein Vorteil der randgebundenen Darstellungen gegenüber den kartesischen Start- und Endpunkten, sodass für die Experimente in dieser Arbeit der Fokus auf die freien Repräsentationen kartesische Start- und Endpunkte (SE) und Mittelpunkt-Richtung-Koordinaten (MR) gelegt wird. Diese bieten die größtmögliche Flexibilität in der Beschreibung der zu detektierenden Liniensegmente.

4.5 Prädiktoren

Idealerweise wird ein Liniensegment immer vom selben Prädiktor prädiziert, da sich so spezialisierte Merkmalsextraktoren ausbilden können sowie ein

Typ	tAcc	FPR	FNR	pF1	pRe	pPr
1D	.887	.157	.160	.616	.955	.455
Eu	.877	**.137**	.168	.685	**.956**	.533
SE	**.914**	.159	**.112**	**.740**	.950	**.607**

Tabelle 4.1: Vergleich zwischen den Linienrepräsentationen Randkoordinaten (1D), Euler-Winkel (Eu) und kartesische Start- und Endpunkte (SE) auf dem TuSimple-Validierungsdatensatz aus [MSPS21]. Hier wurden die Instanzmetriken des TuSimple-Benchmarks (Accuracy (tAcc), Falsch-Positiv-Rate (FPR), Falsch-Negativ-Rate (FNR)) sowie ein punktbasiertes F1-Maß (pF1), Recall (pRe) und Precision (pPr) verglichen.

stabiles Trainingsergebnis gewährleistet werden kann. Dazu muss zunächst eine klare Zuordnung der Liniensegmente der GT zu einem Prädiktor erfolgen. Insbesondere in Kreuzungsszenarien (siehe Abbildung 4.7) müssen dabei pro Zelle mehrere Liniensegmente geschätzt werden, die sich bspw. kreuzen oder parallel verlaufen. Genauso bieten perspektivische Bilder mit geringem Nickwinkel nahe am Horizont die Herausforderung, dass sehr ähnliche Liniensegmente in dieselbe Zelle fallen.

In dieser Arbeit wird zum einen eine dynamische Zuordnung untersucht, bei der in jedem Trainingsschritt die Zuordnung neu auf Basis der Prädiktion berechnet wird. Zum anderen werden aus dem Datensatz Anker berechnet und den Prädiktoren fest zugewiesen, sodass die GT-Liniensegmente bereits vor dem Training zugeordnet werden können.

Mit steigender Anzahl der Prädiktoren pro Zelle ist ein Prädiktor für immer weniger Liniensegmente im Datensatz zuständig, sodass die Anzahl und deren Modellierung direkt bestimmen wie stark das Netz für einen Prädiktor generalisieren muss bzw. wie spezialisiert es sein kann. Je mehr Prädiktoren die Architektur bereitstellt, desto feingranularer können Linientypen unterschieden werden. Gleichzeitig reduziert sich der Einfluss jedes Trainingsbeispiels auf einen Prädiktor. Es gilt also die Anzahl der Prädiktoren abzuwägen und einen guten Mittelweg zu finden.

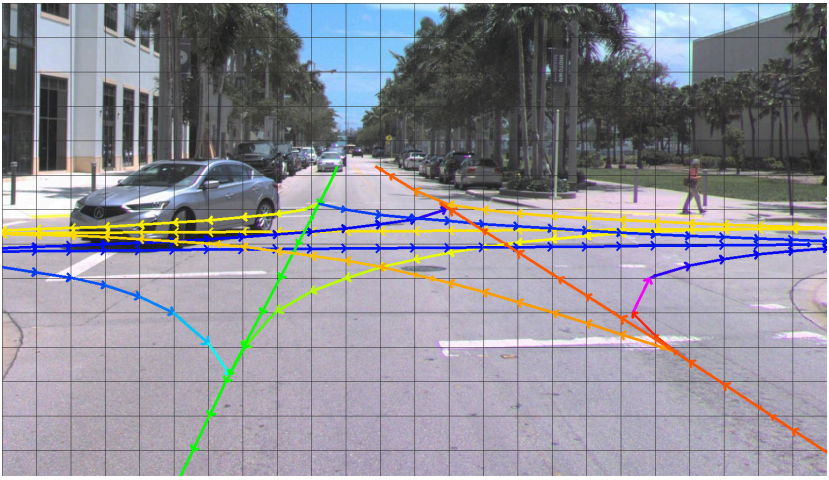


Abbildung 4.7: Nahe am Horizont und in Kreuzungsarmen konvergieren Linienzüge durch die perspektivische Verzerrung zu einer Fluchtlinie. Dies birgt für die Linienschätzung Herausforderungen. Beispielbild aus dem Argoverse-Datensatz [WQA+21].

4.5.1 Dynamische Zuordnung

Für die dynamische Zuordnung werden die GT-Linien nach jeder Inferenz im Training mit der Prädiktion verglichen. Innerhalb jeder Zelle wird für jedes Paar die euklidische Distanz im jeweiligen Assoziationsraum berechnet und mithilfe eines Kuhn-Munkres-Algorithmus (engl. Hungarian method) [Mun57] die beste 1-zu-1-Zuordnung bestimmt.

Für den Assoziationsraum bestehen dabei zwei Möglichkeiten: Entweder wird jeder Prädiktor einer GT über die geometrischen Attribute $\mathbf{g}$ und die Klassifikation $\mathbf{k}$ zugewiesen. Dabei wird die Konfidenz c der Prädiktion nicht beachtet. Alternativ werden für die Zuweisung zur GT alle Attribute $(\mathbf{g}, \mathbf{k}, c)$ inkl. der Konfidenz eines Prädiktors verwendet. Welche Variante sich besser eignet, wird in Experimenten untersucht.

Der Vorteil dieses Ansatzes ist die automatische Ausbildung der Linientypen, die durch einen Prädiktor prädiziert werden. Da alle Prädiktoren gleich initialisiert werden, liegen sie zunächst im selben Punkt der Zelle. Die Prädiktoren

werden dann im Training mit verschiedenen GT-Linien optimiert, sodass sich nach und nach verschiedene Prädiktortypen herausbilden, ohne diese explizit vorzugeben.

4.5.2 Anker

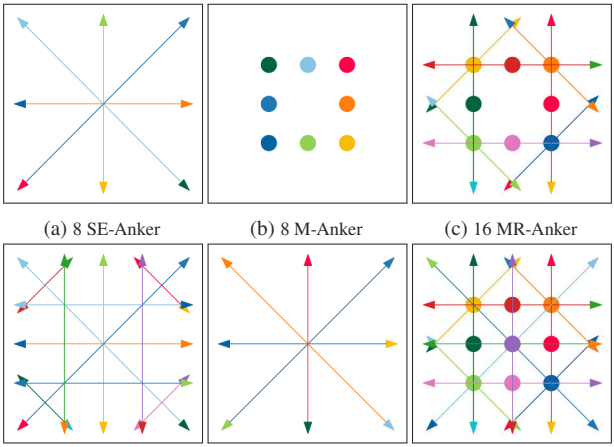
Als Alternative lässt sich A-Priori-Wissen nutzen, um eine feste Zuordnung zu bestimmen. Die zweite Version der YOLO-Ansätze [RF17] führt die Anker für YOLO ein. Sie werden mit einem k-Means-Clustering (siehe Unterabschnitt 2.3.1) aus dem Datensatz berechnet und entsprechen somit typischen Vertretern der Objektboxen. Diese Vertreter bestimmen zum einen die geometrische Beschreibung eines Prädiktors und dienen zum anderen als geometrische Anker, von dem lediglich eine Abweichung gelernt werden muss.

Der Vorteil von Ankern besteht zum einen in einer deutlich schnelleren Trainingszeit. Da während des Trainings keine Zuordnung berechnet werden muss, fällt im Vergleich zur dynamischen Zuordnung ein Großteil der Berechnungen weg. Zum anderen wird die Prädiktion zu Beginn des Trainings bereits bei einer guten Schätzung initialisiert, wenn die Abweichung von Ankern und keine absolute Position gelernt wird.

Für Liniensegmente lassen sich Anker gleichverteilt festlegen oder datengetrieben bestimmen. Zunächst gilt zu beachten, dass alle Hypothesen gleichzeitig geschätzt werden, und damit jeder Anker in der Schätzung nur einmal belegt werden kann. In der Konsequenz bedeutet dies, dass jede GT-Linie nur dann im Training auftaucht, wenn keine andere Linie dem entsprechenden Anker zugeordnet wurde. Die Wahl und Verteilung der Anker ist somit von elementarer Bedeutung für den Trainingserfolg.

Für MR-Linienrepräsentation stellt sich zunächst die Frage, ob sowohl der Mittelpunkt als auch die Richtung für einen Anker relevant sind. Bei genauer Betrachtung der Anwendungsfälle wird klar, dass ähnliche Linienstücke am Horizont im Winkelraum nahe beieinander liegen, im Mittelpunktraum jedoch gut unterscheidbar sind (siehe Abbildung 4.7). Es wird auch klar, dass sich semantisch unterschiedliche Linientypen nicht durch ihre Position in einer Zelle unterscheiden, sondern vielmehr durch ihre Richtung.

Gleichverteilte Anker



(a) 8 SE-Anker (b) 8 M-Anker (c) 16 MR-Anker

(d) 24 SE-Anker (e) 8 R-Anker (f) 24 MR-Anker

Abbildung 4.8: Gleichverteilte Anker für die MR-Repräsentation. Die Farben dienen der Unterscheidung der Linien.

Anker für Mittellinien, Fahrstreifenränder oder Markierungen in Innenstadtbereichen sollten idealerweise den gesamten Merkmalsraum gleichmäßig abdecken. Für Autobahnszenarien, bei denen die Gegenfahrbahn nicht von Relevanz ist, ist nur die Hälfte der Anker nötig, da Liniensegmente nur vom unteren zum oberen Bildrand verlaufen, niemals jedoch in die entgegengesetzte Richtung.

Dabei ist Folgendes insbesondere für Mittellinien in Innenstadtbereichen zu beobachten: Liniensegmente verlaufen durch die gesamte Zelle von Rand zu Rand. Die Position innerhalb einer Zelle folgt einer Gleichverteilung, da – abhängig von der Perspektive und den im resultierenden Bild gelegten Zellrändern – ein Liniensegment an jeder Position des Zellrandes starten und enden kann. Für die in Abschnitt 4.4 präsentierten Linienrepräsentationen bieten sich daher die Anker in Abbildung 4.8 an.

SE-Anker Für acht Anker im kartesischen Raum (SE) sind die Liniensegmente so definiert, dass sie am Zellrand starten und enden und den gesamten

Rotationsbereich in 45°-Schritten abtasten (siehe Abbildung 4.8a). Für 24 Anker werden zusätzlich parallel verschobene Liniensegmente hinzugefügt, die jeweils die Ecken bzw. Seitenbereiche abdecken (siehe Abbildung 4.8d).

M-Anker Für die Mittelpunkt-Koordinaten (M) bieten sich acht Positionen im gleichen Abstand um den Mittelpunkt der Zelle angeordnet an. Diese sind genau auf jeweils ein Viertel der Diagonale bzw. Zellbreite positioniert (siehe Abbildung 4.8b). Der Mittelpunkt der Zelle selbst bietet sich hier nicht an, da dieser für einen Großteil der Liniensegmente, die durch die Zelle laufen, der nächstliegende Punkt wäre, was zu Mehrfachzuweisungen führen würde.

R-Anker Acht Anker, die über die Richtungskoordinaten (R) definiert werden, tasten ebenfalls den Raum in Schritten von 45° ab (siehe Abbildung 4.8e).

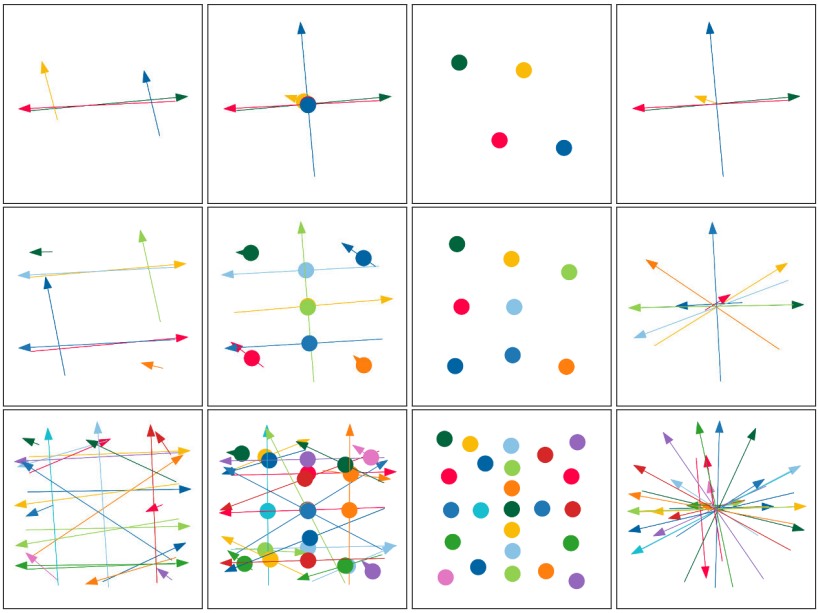
MR-Anker Die Anker aus Mittelpunkt und Richtung (MR) kombinieren die Ideen aus den vorherigen Definitionen und ergeben damit 16 Varianten (siehe Abbildung 4.8c). Für 24 Anker wird der Mittelpunkt der Zelle aufgenommen (siehe Abbildung 4.8f), da dieser in Kombination mit der Richtung wieder eindeutig zugeordnet werden kann.

Datengetriebene Anker

Als Alternative können analog zu Redmon und Farhadi [RF17] Vertreter aus dem Datensatz mithilfe eines k-Means-Clusterings (siehe Unterabschnitt 2.3.1) bestimmt werden. Hier werden k Cluster im Datensatz berechnet und deren Mittelwerte als Anker verwendet. Bei der Vorgabe der Anker muss untersucht werden, ob die vorgeschlagene Strategie, diese mittels k-Means-Clustering vorzugeben, auch für Liniensegmente geeignet ist, und in welchem Raum dieses Clustering ablaufen sollte.

Um diese Abwägung zwischen Modellkomplexität und Spezialisierung zu vereinfachen, wird das Clustering für die Mittelpunkt-Richtung-Koordinaten nur auf Mittelpunkt M und Richtung R als jeweils alleinstehendes Clustermerkmal, genauso wie auf der Kombination MR der beiden ausgeführt. Für die Start-

und Endpunkte (SE) lassen sich Anker nur auf dem vollen Merkmalsraum definieren. Die aus dem Argoverse-Datensatz extrahierten Clusterrepräsentanten werden in Abbildung 4.9 dargestellt.



(a) SE (b) MR (c) M (d) R

Abbildung 4.9: Mit dem k-Means-Algorithmus bestimmte Repräsentanten auf dem Trainingsdatensatz. Hier visualisiert für vier, acht und 24 Prädiktoren. Alle Dimensionen, die nicht zum Clusterraum gehören, wurden als Mittelwert des Clusters visualisiert.

4.6 Kostenfunktion

Die Liniensegmente der GT lassen sich demnach sowohl mit Ankern als auch dynamisch im Training den Prädiktoren zuordnen. Um die Gewichte des Netzes und somit die Prädiktion zu optimieren, müssen für jedes Trainingsbeispiel Kosten formuliert werden. Diese Kosten werden zwischen einem GT-

Liniensegment und dem zugeordneten Prädiktor unabhängig von der Zuordnungsvariante berechnet.

Dazu wird die gewichtete Summe aus geometrischen Kosten $\mathcal{L}_g$, Klassifikationskosten $\mathcal{L}_k$ und Konfidenzkosten $\mathcal{L}_c$ für jede Prädiktion $\ell \in L$ berechnet als

$$\mathcal{L} = \omega_g \mathcal{L}_g + \omega_c \mathcal{L}_c + \omega_k \mathcal{L}_k . \tag{4.10}$$

Da es nicht für jeden Prädiktor des Netzes eine passende GT gibt, sondern überhaupt nur in einem Bruchteil der Zellen Linienzüge verlaufen, werden die geometrischen und Klassifikationskosten nur für zugeordnete Prädiktoren bestimmt. Hier spricht man auch von *verantwortlichen* Prädiktoren. Die Konfidenzkosten werden hingegen für jeden Prädiktor berechnet. Gleichung 4.11 bis 4.13 zeigen die Umsetzung der einzelnen Teilkosten für ein prädiziertes Liniensegment $\ell = (\mathbf{g}, \mathbf{k}, c)$, das ggf. einem GT-Segment $\hat{\ell} \in \hat{L}$ zugewiesen wurde.

$$\mathcal{L}_g(\ell, \hat{\ell}) = \begin{cases} \delta(\mathbf{g}, \hat{\mathbf{g}}), & \text{wenn } \ell \text{ verantwortlich für } \hat{\ell} \\ 0, & \text{sonst} \end{cases} \tag{4.11}$$

$$\mathcal{L}_c(\ell, \hat{\ell}) = \begin{cases} \omega_1 (c-1)^2, & \text{wenn } \ell \text{ verantwortlich für } \hat{\ell} \\ \omega_0 (c-0)^2, & \text{sonst} \end{cases} \tag{4.12}$$

$$\mathcal{L}_k(\ell, \hat{\ell}) = \begin{cases} (\mathbf{k} - \hat{\mathbf{k}})^2, & \text{wenn } \ell \text{ verantwortlich für } \hat{\ell} \\ 0, & \text{sonst} \end{cases} \tag{4.13}$$

Die *geometrischen Kosten* $\mathcal{L}_g$ werden über die Distanzfunktion δ abhängig von der Linienrepräsentation bestimmt (siehe Abschnitt 4.4). Mit den *Konfidenzkosten* $\mathcal{L}_c$ wird eine hohe Konfidenz entweder belohnt oder bestraft, je nachdem ob ein Prädiktor verantwortlich ist.

Für die *Klassifikationskosten* $\mathcal{L}_k$ des Prädiktors schlagen Redmon und Farhadi in der zweiten Version von YOLO [RF17] den quadratischen Abstand zwischen den prädizierten Pseudo-Wahrscheinlichkeiten $\mathbf{k}(c)$ für jede Klasse $c \in K$ und dem One-Hot-Encoding der GT vor.

Die Bestimmung der Gewichte ω der einzelnen Kostenterme ist nicht trivial und gleichzeitig sehr wichtig [RDGF16]. Die Kostenterme können unterschiedlich relevant für das Problem sein und unterschiedliche Wertebereiche aufweisen. In Experimenten können diese empirisch bestimmt werden.

Alternativ haben Kendall u.a. [KGC18] einen Ansatz vorgeschlagen, bei dem die Teilkosten jeweils mit ihrer Varianz σ^2 skaliert werden. Die Varianz wird während des Trainings als zusätzlicher Parameter $\tilde{\sigma}$ optimiert. Für Klassifikationsaufgaben mit einer Sigmoid- oder Softmax-Aktivierung ($\mathcal{L}_k$ und $\mathcal{L}_c$) gilt dabei eine andere Gewichtung als Regressionsaufgaben ($\mathcal{L}_g$):

$$\mathcal{L} = \frac{1}{2\sigma_g^2}\mathcal{L}_g + \log \sigma_g + \frac{1}{\sigma_c^2}\mathcal{L}_c + \log \sigma_c + \frac{1}{\sigma_k^2}\mathcal{L}_k + \log \sigma_k \tag{4.14}$$

Die Netzausgabe für das Gewicht wird dabei auf $\tilde{\sigma} = \log(\sigma^2)$ trainiert, da so ein stabiles Training möglich ist. [KGC18]

5 Anwendungsspezifische Nachverarbeitung für das automatisierte Fahren

In vielen YOLO-basierten Netzen wird eine Vielzahl von Hypothesen inklusive einer Konfidenz für jede Hypothese geschätzt. Da durch die Architektur keine Limitierung der validen Hypothesen existiert, sondern diese nur über eine Kostenfunktion bestraft wird, kann eine Prädiktion mehrere valide Hypothesen für ein Objekt beinhalten. Um nur eine Hypothese je Objekt zu erhalten, wird bei derartigen Ansätzen eine Non-Maximum-Suppression (NMS) angehängt, die aus den Prädiktionen auswählt. Zudem erwarten einige Anwendungen verbundene Linienzüge, sodass in diesem Kapitel ebenfalls Empfehlungen zur Verknüpfung der Liniensegmente gegeben werden. Da die diskutierten Ansätze teilweise in Masterarbeiten im Rahmen dieser Arbeit entstanden sind, wurden Ideen aus eigens betreuten Masterarbeiten [Sku20, MSPS21, Buc21, Sch22] für das folgende Kapitel entnommen.

Das Kapitel ist wie folgt aufgebaut. Zunächst wird die Repräsentation als Graph dargestellt (Abschnitt 5.1), da diese die Verarbeitung vereinfacht. In Abschnitt 5.2 werden zwei NMS-Ansätze für Liniensegmente vorgestellt, die ein Clustering bzw. ein GNN einsetzen. Die Schätzung der Verbindung zwischen den Liniensegmenten kann als Breitensuche (Unterabschnitt 5.3.1), Zuordnungsproblem (Unterabschnitt 5.3.2), graphenbasierte Pfadoptimierung (Unterabschnitt 5.3.3) oder als gelernte Pfadprädiktion (Unterabschnitt 5.3.4) gelöst werden.

Alle Varianten, die sich differenzierbar implementieren lassen, können mit dem bestehenden YOLinO-Ansatz kombiniert werden und ermöglichen somit eine vollständige Optimierung in einem Ende-zu-Ende-Verfahren. Qualitative Ergebnisse werden in Abschnitt 6.6 präsentiert und ausgewertet.

5.1 Graphkonstruktion

Die Prädiktion von YOLinO kann nicht nur als Menge von Liniensegmenten bzw. als Zellgitter interpretiert werden, sondern auch als Graphstruktur (siehe Abschnitt 2.2). Dabei stellt der Graph die lokale Nachbarschaft der Liniensegmente dar.

Die Liniensegmente $\ell = (\mathbf{g}, \mathbf{k}, c)$ werden im Graph als Knoten $v \in \mathcal{V}$ repräsentiert. Da für die in dieser Arbeit eingesetzten Ansätze ein geringer Maximalgrad im Graphen von Vorteil ist, gilt es, die Kanten $\mathcal{E}$ zwischen den Knoten geschickt zu wählen. Um den Maximalgrad des Graphen gering zu halten, sollen nur benachbarte Liniensegmente verbunden werden. Dazu bieten sich verschiedene Möglichkeiten an, die Nachbarschaft zu bestimmen.

5.1.1 Zellstruktur

Die Zellstruktur und damit die Position der Liniensegmente im Zellgitter (r, o) kann zur Erstellung des Graphen verwendet werden. Es werden alle Knoten v verbunden, deren Liniensegmente maximal ρ_z Zellen voneinander entfernt sind, sodass

$$\mathcal{E} := \{e = (v_i, v_j) \mid \min(|r_i - r_j|, |o_i - o_j|) \leq \rho_z\}. \tag{5.1}$$

Mit dieser Methodik kann der Graph sehr schnell konstruiert werden. Der Grad der Knoten steigt jedoch mit Anzahl der Prädiktoren mit $((2\rho_z + 1)^2 - 1) \cdot |P|$, sodass der Graph bereits bei kleinen Werten für ρ_z einen hohen Maximalgrad aufweist. Für $\rho_z = 2$ hat jeder Knoten bspw. bereits $24 \cdot |P|$ Nachbarn.

5.1.2 Euklidischer Abstand

Die Verwendung des Zellgitters ermöglicht eine einfache Konstruktion, nimmt jedoch diagonal zur Zellstruktur mehr Segmente auf als orthogonal dazu. Um eine gleichverteilte Nachbarschaft zu erstellen, kann alternativ ein Radius ρ_{uv} als maximaler euklidischer Abstand zwischen dem Startpunkt $\mathbf{b}$ und dem End-

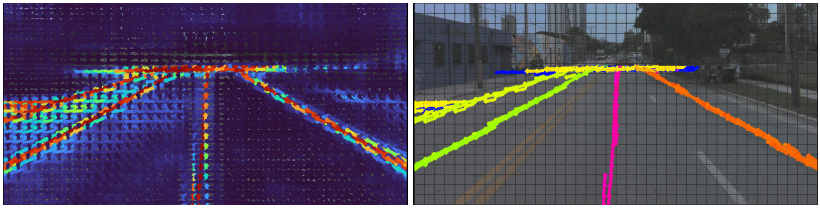
punkt **e** zweier Liniensegmente in Bildkoordinaten verwendet werden. Dazu werden alle Knoten v verbunden, für deren Kanten e gilt

$$\mathcal{E} := \{e = (v_i, v_j) \mid \min(\|\mathbf{b}_i - \mathbf{b}_j\|_2, \|\mathbf{e}_i - \mathbf{e}_j\|_2) \leq \rho_{uv}\}. \tag{5.2}$$

Diese Variante ist zwar langsamer, weist jedoch einen geringeren Maximalgrad auf.

5.2 Non-Maximum-Suppression

Die Architektur prädiziert zur Laufzeit zu einem gegebenen Bild die Prädiktion als Zellstruktur. Da für jede Zelle $|P|$ Hypothesen aufgestellt werden, sieht die vollständige Prädiktion wie in Abbildung 5.1a aus. Über die Konfidenz lassen sich zunächst die validen Prädiktionen mit $c > 0{,}5$ filtern. Teilweise sind jedoch noch Duplikate im Vergleich zur GT vorhanden. Eine gefilterte Prädiktion mit einem hohen Recall zeigt Abbildung 5.1b.



(a) Prädiktionen (b) Prädiktionen mit $c > 0{,}5$

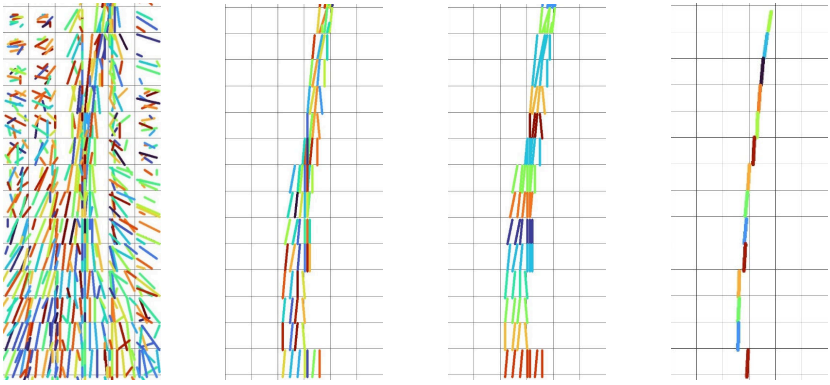
Abbildung 5.1: Beispielhafte Prädiktion mit einer Zellauflösung von 16×16 Pixel. a) Die Konfidenz jedes Liniensegments wird hier farblich kodiert. In Dunkelblau werden Liniensegmente mit $c = 0$ und in Dunkelrot mit $c = 1$ dargestellt. b) Jedes Liniensegment mit $c > 0{,}5$ wird farblich kodiert nach der Orientierung eingefärbt.

In den YOLO-Ansätzen wird der Intersection over Union (IoU) als Maß für die Überlappung von Objekten verwendet, um eine NMS umzusetzen [RDGF16]. Der IoU ist nicht auf Liniensegmente übertragbar, da diese keine Ausdehnung aufweisen, sodass eine auf Liniensegmente angepasste NMS nötig ist. Eine geeignete Linien-NMS muss aus der durch die Konfidenz gefilterten Prädiktion eine möglichst eindeutige Menge an Liniensegmenten bestimmen.

5.2.1 DBSCAN

Dazu bietet sich ein Clusteringalgorithmus an, der Duplikate zu einem Cluster gruppiert und einen Repräsentanten je Cluster bestimmt [MSPS21]. Alle übrigen Elemente des Clusters werden unterdrückt.

Der DBSCAN-Algorithmus [EKSX96] gruppiert Prädiktionen dichtebasiert und kann so parametrisiert werden, dass parallel verlaufende Linien einen Cluster bilden. Aufeinander folgende Liniensegmente sollen nicht Teil desselben Clusters werden, sondern beibehalten werden. Abbildung 5.2 zeigt das Prinzip grafisch.



(a) Rohe Prädiktion (b) Gefiltert (c) Cluster (d) Repräsentanten

Abbildung 5.2: Mit dem DBSCAN-Algorithmus werden die prädizierten Liniensegmente zunächst mit $c > 0{,}5$ gefiltert und zu einem Cluster mit einem Repräsentanten zusammengefasst. a) Alle Prädiktionen der Zellen. b) Alle Liniensegmente mit einer hohen Konfidenz. c) Mit DBSCAN ermittelte Cluster farblich kodiert. d) Die Repräsentanten der Cluster.

Die prädizierten Liniensegmente werden zunächst nach ihrer Konfidenz gefiltert (siehe Abbildung 5.2b) und anschließend in einen geeigneten Raum transformiert, der den Mittelpunkt-Richtung-Koordinaten (MR) ähnelt, jedoch die Länge explizit betrachtet (siehe Abbildung 5.3).

Jedes Liniensegment wird über $\hat{\mathbf{g}} = (m_u, m_v, l_{uv}, d_u, d_v)^\intercal$ beschrieben, wobei $\mathbf{m}_{uv}$ das Zentrum des Liniensegments in Bildkoordinaten, l_{uv} die Länge des Segments und $\mathbf{d}_{uv}$ die normierte Richtung darstellt. Jede Dimension wird

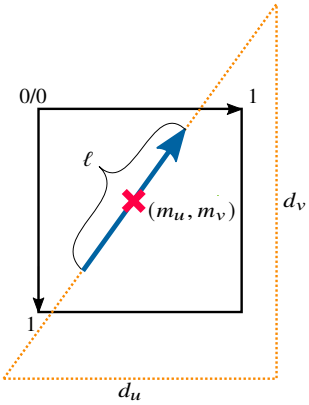


Abbildung 5.3: Für das DBSCAN-Clustering werden Liniensegmente über das Zentrum des Liniensegments in Bildkoordinaten $\mathbf{m}_{uv}$, die Länge des Segments l_{uv} und die normierte Richtung $\mathbf{d}_{uv}$ dargestellt.

zusätzlich über je einen Vorfaktor $\lambda_m, \lambda_l, \lambda_d$ skaliert, der empirisch für jeden Datensatz und jede Skale bestimmt wird.

Auf dem so aufgespannten Clusterraum werden mit DBSCAN die Cluster bestimmt (siehe Abbildung 5.2c). Dazu wird jedes Liniensegment mit $w_i = c_i^{10}$ gewichtet, sodass unpassende Prädiktoren weniger in die Cluster einfließen.

Die resultierenden Cluster werden zu einem Repräsentanten zusammengefasst (siehe Abbildung 5.2d). Die geometrische Beschreibung des Repräsentanten wird als über die Konfidenzen gewichteter Mittelwert bestimmt und das Maximum der Konfidenzen des Clusters als neue Konfidenz dem Repräsentanten zugewiesen.

5.2.2 Non-Maximum-Suppression als Klassifikation im Graphen

Um die Graphenstruktur für eine gelernte NMS zu verwenden, bieten sich graphenbasierte neuronale Netze (GNNs) an (siehe Unterabschnitt 2.4.7). Dazu müssen durch das Netz Duplikate unterdrückt und valide Hypothesen hervorgehoben werden. Daraus ergibt sich eine binäre Klassifikation für jeden Knoten

des Graphen in valide und nicht valide Hypothesen, die als Schätzung einer Konfidenz $\tilde{c}_i$ je Knoten umgesetzt wird. Die GT zu dieser Klassifikation lässt sich über die Distanz der Prädiktion zur GT bestimmen, sodass immer die der GT am nächsten liegende Prädiktion als valide Hypothese klassifiziert werden soll [Buc21].

Die Repräsentation der Knoten besteht dabei idealerweise aus der Prädiktion mit geometrischer Beschreibung, Klassifikation und Konfidenz und wird um ein gelerntes Embedding erweitert, das detaillierte Informationen aus der YOLinO-Architektur extrahiert [Sch22].

Da die meisten Datensätze ein großes Rauschen aufweisen, ist es jedoch nicht immer zielführend, nur die Liniensegmente als valide Prädiktion zu akzeptieren, die der GT am nächsten sind. Vielmehr ist die Konsistenz des Gesamtergebnisses wichtig.

Genauso bietet die Zellstruktur eine Herausforderung für die Bestimmung der nächsten Prädiktion, was Abbildung 5.4a veranschaulicht. Bei einer 1-zu-1-Zuweisung der GT zu Prädiktionen, würden alle Prädiktionen (rot) den GT (blau) zugewiesen, obwohl bereits die Kombination aus Prädiktion 0 und 1 die gesamte GT abdeckt.

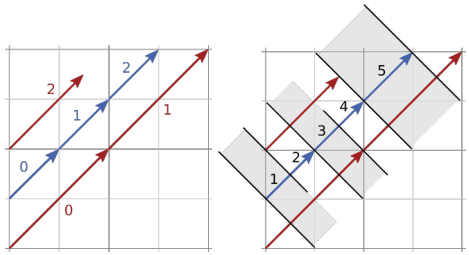


Abbildung 5.4: Projektion der Prädiktion auf die GT. a) zeigt die einzelnen Segmente der Prädiktion (rot) und GT (blau). b) visualisiert die Abschnitte gleicher Projektion, für die jeweils Kosten formuliert werden. Bildquelle: [Sch22]

Zur Lösung der beiden Herausforderungen wird ein weicher Kostenterm präsentiert, der die Projektion der Prädiktion auf die GT verwendet [Sch22]. Dabei wird die GT in Abschnitte gleicher Projektion aufgeteilt. In Abbildung 5.4b werden die Prädiktionen auf die GT projiziert. Die Abschnitte ergeben sich

dann aus dem Start und Ende der Projektionen. In Abschnitt 1 ist bspw. nur die Prädiktion 0 zuständig. In Abschnitt 2 sind es sowohl Prädiktion 0 als auch Prädiktion 2.

Daraus lassen sich für jeden Abschnitt a des GT-Liniensegments und den Prädiktionen $\Omega = \{\ell_0, \ell_1, ...\}$, die auf diesen Abschnitt fallen, die Kosten formulieren als

$$\mathcal{L}_s := |a| \cdot \left(\left(\sum_{i}^{|\Omega|} \tilde{c}_i \right) - 1 \right)^2 . \tag{5.3}$$

Somit wird das GNN darauf trainiert, nur eine der Prädiktionen, die auf die gleiche GT fallen über die Konfidenz $\tilde{c}_i$ des Knotens, als valide zu klassifizieren.

5.3 Verbindungsschätzung

Neben der Unterdrückung der Duplikate durch NMS kann es für einige Anwendungen notwendig sein, die vollständigen Linienzüge und somit die Verbindung der Liniensegmente zu schätzen.

5.3.1 Baumsuche

Für Szenarien auf Autobahnen und baulich getrennten Kraftfahrstraßen kann eine hilfreiche Annahme getroffen werden: Da die Gegenfahrbahn nicht relevant ist und zudem keine Kreuzungen existieren, verlaufen die Fahrstreifen entlang der Fahrtrichtung und starten am Bildrand bzw. lateral verschoben zur Fahrzeugpose. Mit dieser Annahme lässt sich ein spezieller Graph, ein Suchbaum (siehe Abschnitt 2.2), aufspannen, der die Verarbeitung deutlich vereinfacht [Sku20]. Wenn der Graph azyklisch und zusammenhängend ist sowie nur jeweils eine Kante in einen Knoten führt, so erfüllt dieser die Anforderungen eines Suchbaums.

Für jeden zusammenhängenden Graphen in der Prädiktion soll ein Suchbaum entstehen, da es sich hierbei um einzelne Instanzen handelt. Um den Graphen azyklisch zu machen, werden die prädizierten Liniensegmente zusätzlich nach

ihrer Orientierung gefiltert. Zeigt ein Liniensegment von oben nach unten im Bild (±45°), so wird dieses entfernt. Damit nur jeweils eine Eingangskante am Knoten existiert, wird der Baum von den Blättern her aufgespannt und nur eine Verbindung zum jeweils nächsten Liniensegment eingefügt.

Für jeden Suchbaum kann eine Breitensuche ausgehend vom Wurzelknoten erfolgen (siehe Abbildung 5.5), die jeweils einen Linienzug ermittelt [MSPS21].

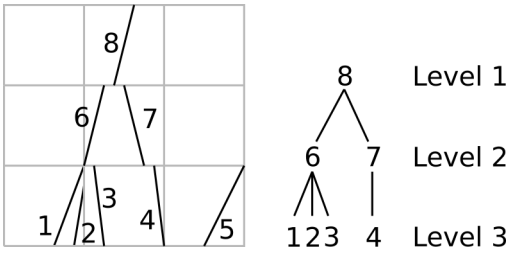


Abbildung 5.5: Breitensuche zur Verbindung der Liniensegmente. Segmente 1, 2, und 3 sowie Segmente 6 und 7 teilen sich je einen Nachfolger. Segment 8 ist ein Wurzelknoten, der die Breitensuche startet. Für jede Ebene wird der Mittelwert bestimmt, sodass Segmente 6 und 7 sowie Segmente 1 bis 4 zusammengefasst werden. Segment 5 hat keinen Nachfolger und wird verworfen, da der Linienzug zu kurz ist.

Unter der Annahme, dass es keine Aufspaltungen der Fahrstreifen gibt, wird jede Ebene des Baums zu einem Liniensegment zusammengefasst und der gewichtete Mittelwert der Ebene als Repräsentant verwendet. Die Konfidenzen der Liniensegmente dienen als Gewichte. Aus dieser Beschreibung entsteht somit je Wurzelknoten eine Instanz eines Linienzugs ohne doppelte Liniensegmente.

5.3.2 Kuhn-Munkres-Zuordnung

Wenn die Prädiktion bereits durch ein NMS gefiltert ist und somit keine Duplikate mehr existieren, kann die Schätzung der Verbindung als Assoziationsproblem aufgefasst werden [Sch22]. Dazu werden für jedes Paar aus Liniensegmenten Kosten über die euklidische Distanz der Start- und Endpunkte formuliert, sodass

$$\mathcal{L}_H(v_i, v_j) = \|\mathbf{e}_i - \mathbf{b}_j\|_2. \tag{5.4}$$

Diese Kosten können mit dem Kuhn-Munkres-Algorithmus [Mun57] optimiert werden, sodass eine 1-zu-1-Zuordnung entsteht. Eine Verbindung zweier Liniensegmente ergibt sich somit dann, wenn der Startpunkt des einen Liniensegments nah am Endpunkt des anderen Liniensegments liegt und sie optimalerweise einander zugeordnet werden.

5.3.3 MuSSP

Auch die Verbindungsschätzung kann die Graphstruktur ausnutzen, indem jeder Linienzug als Pfad im Graph beschrieben wird. In der Objektverfolgung ist dies bereits eine bekannte Problemformulierung: Objekte werden zu verschiedenen Zeitpunkten detektiert und müssen über die Zeit hinweg verfolgt und somit im Objektgraphen verbunden werden. Dieses Problem kann als Mehrpfadsuche im Graphen mit Kanten- und Knotenkosten formuliert werden [Sch22]. Ein Lösungsansatz dafür ist MuSSP (Minimum-update Successive Shortest Path) [WWW+19].

Da MuSSP die Annahme stellt, dass der Graph gerichtet ist und kaum Zyklen enthält, muss die Graphenkonstruktion weiter verfeinert werden. Dazu werden nur Liniensegmente mit einem Nachfolger verbunden, deren Endpunkt innerhalb eines Radius ρ_d zum Startpunkt des Nachfolgers liegt und die Abweichung in der Orientierung unter einem Schwellwert ρ_α liegt. Zusätzlich wird ein Liniensegment nur mit nachfolgenden Liniensegmenten verbunden, wenn das nachfolgende Liniensegment im Öffnungswinkel ρ_β des Vorgängers liegt.

Dadurch ergibt sich ein Graph, bei dem ein Liniensegment nur mit möglichen Nachfolgern verbunden ist. Zusätzlich werden im Graphen eine Senke und eine Quelle als Knoten für den Start- bzw. das Ende eines Linienzugs eingefügt und diese mit allen Liniensegmenten verbunden.

Im resultierenden Graphen werden die kostenminimalen Pfade gesucht. Die Kosten für eine Kante zwischen Knoten sind dabei positiv und die Knotenkosten negativ. Die Kosten der Kanten zur Senke bzw. von der Quelle sind ebenfalls positiv. Mit diesem Prinzip wird ein gewichteter Graph erzeugt, der nur hinreichend lange Pfade zueinander passender Liniensegmente als Linienzüge erlaubt. Die konkrete Parametrisierung des Ansatzes findet sich im Anhang Tabelle A.11.

5.3.4 Gelernte Verbindung im Graphen

Die in Unterabschnitt 5.2.2 vorgestellte NMS auf Basis von GNNs lässt sich auch zur Schätzung eines Verbindungsgraphen einsetzen [Sch22]. Dazu werden die NMS-Kosten mit einem weiteren Kostenterm kombiniert. Die Konfidenz der Kanten $\tilde{c}_{ij}$ und Knoten $\tilde{c}_i$ kann dann wie bei einer NMS als Prädiktion interpretiert werden, die die Wahrscheinlichkeit beschreibt, dass eine Kante bzw. ein Knoten existiert. Daraus lässt sich dann ein Verbindungsgraph erstellen.

Die Kanten des Graphen sollen zusätzlich mit Kosten versehen werden, die möglichst nur eine Kante als Verbindung zwischen zwei Liniensegmenten des gleichen Linienzugs zulässt. Sind mehrere Liniensegmente einer GT zugeordnet, so darf die Summe der geschätzten Konfidenzen maximal 1 sein.

Dazu werden alle Prädiktionen i einer GT j als Assoziation $\Theta = \{(i,j)^n\}$ zugeordnet und anschließend für jede GT die Kosten aufgestellt, dass je GT nur eine valide Kante existieren darf mit

$$\mathcal{L}_+ := \left(\left(\sum_{i,j}^{\Theta} \tilde{c}_{ij} \right) - 1 \right)^2 . \tag{5.5}$$

6 Evaluation

In diesem Kapitel wird das in Kapitel 4 vorgestellte Konzept hinsichtlich der wesentlichen Aspekte evaluiert und die Parametrisierung diskutiert. Im Hinblick auf die Schätzung von Linienzügen als Kombination aus Liniensegmenten werden Empfehlungen abgegeben, wie das Konzept konkret umgesetzt werden kann. Es wird zum einen untersucht, welchen Einfluss die Aufweitung der Netzarchitektur (Unterabschnitt 6.4.1) hat und wie sich verschiedene geometrische Repräsentationen der Liniensegmente (Unterabschnitt 6.4.2) und die in Unterabschnitt 4.5.2 vorgestellten Ankervarianten (Unterabschnitt 6.4.3) auswirken. Die dynamische Zuordnung aus Unterabschnitt 4.5.1 wird anschließend in Unterabschnitt 6.4.4 untersucht.

Zum anderen betrachtet dieses Kapitel den Einfluss unterschiedlich großer Zellgitter zur Diskretisierung der Linienzüge auf das Trainingsergebnis (Unterabschnitt 6.4.5) und den Diskretisierungsfehler (Unterabschnitt 6.4.6). In Unterabschnitt 6.4.7 wird abschließend untersucht, ob sich die Gewichtung der Kostenfunktionen lernen lässt oder konstant sein soll.

Um diese Experimente durchführen zu können, stellt Abschnitt 6.1 zunächst eine Reihe von Datensätzen vor, die für die Problemstellung relevante Daten enthalten und wählt aus, welche in dieser Arbeit zur Evaluation verwendet werden. Da die Datensätze die GT nicht direkt bereitstellen, wird eine nötige Vorverarbeitung in Unterabschnitt 6.1.4 diskutiert.

In Abschnitt 6.2 werden Metriken eingeführt, anhand derer die Ergebnisse und damit die Experimente vergleichbar gemacht werden. Abschnitt 6.3 ergänzt dazu die nötigen Informationen zum System und den Trainingsparametern. Auf Basis dieser Information werden die Konzeptentscheidungen in Abschnitt 6.4 untersucht. Einen qualitativen Einblick in anwendungsrelevante Ergebnisse der Prädiktion gibt Abschnitt 6.5. Die Ergebnisse der in Kapitel 5 vorgestellten Möglichkeiten zur Nachverarbeitung präsentiert Abschnitt 6.6.

6.1 Datensätze

Für diese Arbeit wird in Experimenten gezeigt, dass unterschiedliche Kartenmerkmale präzise detektiert werden können. Dafür werden Datensätze benötigt, die Markierungen, Fahrstreifenränder oder Mittellinien beinhalten und dazu eine pixelgenaue Transformation in Kamera- oder Lidardaten bereitstellen. Für die Untersuchung des Konzepts wird ausschließlich ein Datensatz benötigt, der möglichst komplexe Szenen beinhaltet. In Abschnitt 6.5 wird die Konzeption dann auf die übrigen Datensätze übertragen, um die generischen Einsatzmöglichkeiten aufzuzeigen.

Datensätze, die lediglich die Fahrstreifenfläche annotieren [BSFC08, FKG13, MSOS18, RKS+18, YCW+20], sind für diesen Ansatz ohne unverhältnismäßigen Aufwand nicht nutzbar, da aus einem Umrandungspolygon der Fläche nicht eindeutig eine Linienzugbeschreibung wie bspw. ein linker und rechter Fahrstreifenrand berechnet werden kann. Das Gleiche gilt für Datensätze mit Markierungsannotationen in Form von semantischen Segmentierungen oder Umrandungspolygonen [LKY+17].

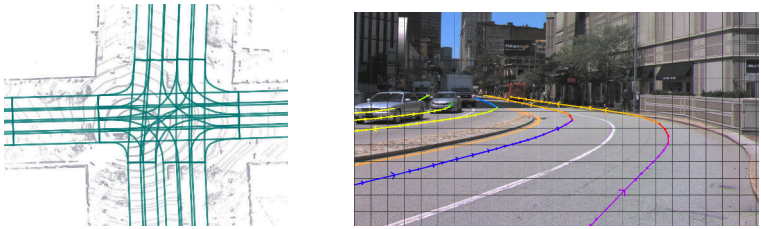
Da der Ansatz langfristig für den Einsatz im Fahrzeug gedacht ist, wird nur mit Realdatensätzen gearbeitet und keine synthetischen Daten wie bspw. [GCZ+20] verwendet. Der Caltech Lane [Aly08] Datensatz wird nicht betrachtet, da das Bildmaterial nicht mehr den heutigen Standards entspricht.

Datensätze mit wenigen Bildern werden ebenfalls ausgeschlossen, da diese für ein erfolgreiches Training nicht ausreichen. Das betrifft den Oxford Road Boundaries Datensatz [SGDMN21] mit 175 Bildern.

Für diese Arbeit werden Ergebnisse auf Argoverse 2.0 (Unterabschnitt 6.1.1), KAI (Unterabschnitt 6.1.2) und TuSimple (Unterabschnitt 6.1.3) vorgestellt. Die Datensätze Nuscenes [CBL+20], Lyft, BDD100k [YCW+20], CULanes [PSL+18], LLamas [BS19], VPGNet [LKY+17], VIL100 [ZZF+21] und ApolloScape [MZZ+19] wurden ebenfalls im Vergleich betrachtet und in Abschnitt A.3 zusammengefasst, bieten jedoch keinen weiteren Vorteil gegenüber den ausgewählten Datensätzen.

6.1.1 Argoverse 2.0: Mittellinien

Der Argoverse Datensatz (Version 2.0) [WQA+21] umfasst Kamera- und Lidardaten sowie eine hochgenaue Karte (siehe Abbildung 6.1). Die bereitgestellte Karte basiert auf Fahrstreifenrändern, aus denen die mitgelieferte Software Mittellinien der Fahrstreifen berechnet. Zusätzlich zu den Positionen beinhaltet die Karte den vollständigen Fahrstreifengraphen, sodass abgeleitet werden kann, welche Positionen und Fahrstreifen von der eigenen Position erreichbar sind. Mit den enthaltenen Lidartiefeninformationen kann das vorhandene Kartenmaterial in die Bilder transformiert und somit als GT-Daten verwendet werden. Die bereitgestellte Transformation ist jedoch nicht immer perfekt und kann dadurch verrauschte GT erzeugen (siehe Abbildung 6.1b).



(a) Karte einer Kreuzung mit Lidar-Daten (b) Projektion der Mittellinien in das Kamerabild

Abbildung 6.1: Zwei Beispielszenen aus dem Argoverse-Datensatz. In b) wurde eine Szene ausgewählt, in der die von Argoverse bereitgestellte Lokalisierung und Projektion in das Kamerabild fehlerbehaftet ist. Die dargestellten Mittellinien verlaufen nahe am Horizont nicht immer auf der Straßenfläche.

6.1.2 KAI: Markierungen

Der Kartenänderungsdatensatz von Pauls u.a. (KAI) [PSH+18] beinhaltet Annotationen für Luftbildaufnahmen von deutschen Autobahnen (siehe Abbildung 6.2). Dabei wurden im Luftbild die Positionen und der Verlauf von durchgezogenen und gestrichelten Markierungen als Linienzüge angegeben, die auf die Luftbilder projiziert werden können.

Abbildung 6.2: Luftbildaufnahme mit Annotation durchgezogener (gelb) und gestrichelter Markierungen (rot). Bildquelle der Luftbilder: ©Stadt Karlsruhe | Liegenschaftsamt

6.1.3 TuSimple: Fahrstreifenränder

TuSimple[1] ist ein Datensatz, der Bilder einer Frontkamera auf Autobahnszenarien in den Vereinigten Staaten von Amerika beinhaltet (siehe Abbildung 6.3). Es wurden in diesen Bildern die Position der Fahrstreifenränder der Fahrtrichtung des Ego-Fahrzeugs annotiert. Diese liegen als Position in definierten Bildzeilen (in rot dargestellt) vor. Die gestellte Aufgabe ist durch die Beschränkung auf Autobahnen sehr einfach formuliert, jedoch durch den mitgelieferten Benchmark gut zum Vergleich mit anderen Ansätzen nutzbar.

6.1.4 Datensatzaufbereitung

Für die weiteren Experimente wird der Argoverse-Datensatz mit dem impliziten Merkmal von Mittellinien in Innenstädten verwendet, da dieser unter allen Datensätzen die schwierigste Aufgabe bereitstellt und somit das interessanteste Evaluationspotential bietet. Da Innenstädte und Autobahnen unterschiedliche Anforderungen aufweisen, wird für einige Experimente zusätzlich der Vergleich zum TuSimple-Datensatz gezogen.

Der Argoverse-Datensatz wird zunächst vorverarbeitet, da die Daten nicht in geeigneter Form vorliegen. Es wird eine reduzierte Menge der Bilder verwendet, da es sich um Sequenzen handelt und die aufeinanderfolgenden Bilder

[1] TuSimple `https://github.com/TuSimple/tusimple-benchmark`

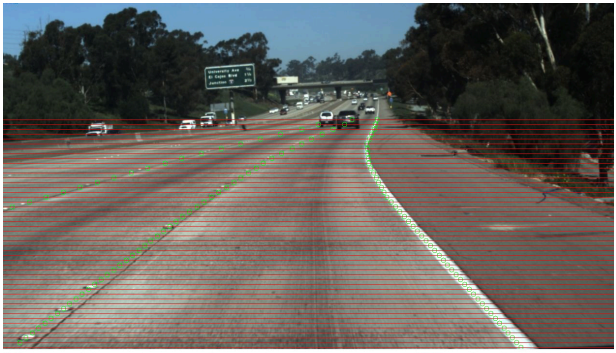


Abbildung 6.3: Beispielszene aus dem TuSimple-Datensatz mit Fahrstreifenannotationen (grün) auf Autobahnen. In Rot wird hier die Abtastung der Linienzüge in Bildzeilen visualisiert. Bildquelle: `https://github.com/TuSimple/tusimple-benchmark`

kaum Unterschiede aufweisen, wodurch sie für das Training wenig relevant sind. Es wird jedes 20. Bild jeder Sequenz verwendet, sodass pro Sequenz mit 16 Bildern trainiert wird. Da das Frontbild im Format 1550×2048 Pixel (Breite $\times$ Höhe) vorliegt, die Fahrstreifen jedoch nur im unteren Teil des Bildes existieren und für die Prädiktion idealerweise eine Bildgröße teilbar durch Zellgröße verwendet wird, wird nur ein quadratischer Ausschnitt 1536×1536 Pixel am unteren Rand des Bildes für das Training eingesetzt, der aufgrund der Rechenzeit auf 640×640 Pixel skaliert wird.

Die Kartendaten in einem Umkreis von 80 m um die Fahrzeugpose werden mit der bereitgestellten Software in das Bild der Frontkamera projiziert und in Bildkoordinaten umgerechnet. Um die Menge an Stützpunkten gering zu halten, wird auf alle projizierten Linienzüge der Ramer-Douglas-Peucker-Algorithmus (RDP) [Ram72] angewendet, der die Anzahl der Stützpunkte eines Linienzugs unter Erhaltung der Struktur reduziert[2]. Um die projizierten Linienzüge in die entsprechende Zielformulierung zu übertragen, werden die Linienzüge mit dem Zellgitter abgetastet. Algorithmus 1 zeigt die dazu nötigen Schritte.

[2] Für die Umsetzung des RDP wird die `simplify`-Methode von shapely mit einer Toleranz von 0,8 verwendet.

Für die Berechnung des Schnitts zwischen Zelle und Liniensegment wird die `intersection`-Methode von shapely[3] verwendet.

Algorithmus 1 Berechnung der Liniensegmente L im Zellgitter Z durch Abtastung der GT-Linienzüge G

$L \leftarrow \emptyset$
for each $g \in G$ **do**
  $\tilde{g} \leftarrow RDP(g)$ ▷ Ramer-Douglas-Peucker
  for each $z \in Z$ **do**
    $\hat{L}_z \leftarrow z \cap \tilde{g}$ ▷ Linienzüge innerhalb der Zelle z
    for each $\ell \in \hat{L}_z$ **do**
      $n \leftarrow |\ell|$
      $L \leftarrow L \cup (\ell^1, \ell^n)$ ▷ Erster und letzter Punkt je Zug ℓ
    end for
  end for
end for

6.2 Evaluationsmetrik

Die durch ihre Parameter verschiedenen Modelle werden zunächst auf ihre *Detektionsfähigkeit* hin evaluiert. Während des Trainings wird dazu eine zellbasierte Evaluation verwendet.

Die zellbasierte Evaluation vergleicht die Prädiktionen und GT innerhalb einer Zelle und entspricht somit der bereits vorgestellten Assoziation der Prädiktoren (siehe Abschnitt 4.5). Bei Verwendung von Ankern ist die Assoziation bereits durch die Position im Training gegeben (siehe Unterabschnitt 4.5.2). Bei der dynamischen Zuordnung wird die Assoziation nach der Inferenz durch Minimierung der Assoziationskosten ermittelt (siehe Unterabschnitt 4.5.1).

Eine Prädiktion gilt als positiv, wenn die Konfidenz $c > 0{,}5$ ist. Wahrpositiv (TP) ist eine Prädiktion dann, wenn sie einem GT-Liniensegment zugeordnet ist. Als negativ werden Liniensegmente bezeichnet, deren Konfidenz mit

[3] Shapely `https://github.com/shapely/shapely`

$c \leq 0{,}5$ geschätzt wird, sodass die wahr-negativen Schätzungen (TN) alle negativen Prädiktionen sind, die keinem GT-Liniensegment zugeordnet sind. Ist die Kombination aus Konfidenz und Zuordnung nicht passend so spricht man von einer Falsch-Positiv-Schätzung (FP) bzw. Falsch-Negativ-Schätzung (FN). Eine Übersicht gibt Tabelle 6.1. Für die gesamte Zuordnung wird der Recall (Re), die Precision (Pr), das F1-Maß (F1) und die Accuracy (Acc) berechnet (siehe Unterabschnitt 2.1.2).

		assoziiert	nicht assoziiert
Konfidenz	$c > 0{,}5$	wahr-positiv (TP)	falsch-positiv (FP)
	$c \leq 0{,}5$	falsch-negativ (FN)	wahr-negativ (TN)

Tabelle 6.1: Bestimmung der Wahr-Positiv-Schätzung (TP), Falsch-Positiv-Schätzung (FP), Wahr-Negativ-Schätzung (TN) und Falsch-Negativ-Schätzung (FN) einer Liniensegmentprädiktion mit Konfidenz c.

Für alle wahr-positiven Liniensegmente soll zusätzlich die *Abweichung* vom GT-Partner in Bild- oder UV-Koordinaten bestimmt werden. Hierzu wird die durchschnittliche, absolute Abweichung (MAE) von den folgenden Fehlermaßen berechnet:

1. euklidische Distanz der Start- und Endpunkte ($\|\cdot\|_2$),
2. Differenz der Mittelpunkte (MP),
3. Differenz der Längen (L) und
4. Differenz der Konfidenzen (KfTP).

Zusätzlich betrachten die Experimente die Differenz der Konfidenzen unter allen Prädiktionen (Kf).

Für die in diesem Kapitel präsentierten Ergebnisse werden die Modelle zum Zeitpunkt der Konvergenz ausgewertet. Die Epoche der Konvergenz wird in den Tabellen mit Kv bezeichnet. Je nach Relevanz für das Experiment wird entweder nur die Inferenzzeit als $t_{\bowtie}$ oder die Summe aus Inferenzzeit und Kostenberechnung als $t_{\mathcal{L}}$ je Bild im Batch angegeben. Um die Lesbarkeit der Tabellen zu verbessern, wird je Block das **beste**, zweitbeste und *drittbeste* Ergebnis hervorgehoben.

6.3 Trainingssystem

Die Trainings in diesem Kapitel werden auf einem Basisexperiment aufgebaut, das sich bereits in Vorexperimenten als geeignete Konfiguration gezeigt hat. Die Konfiguration des Basisexperiments ist in Abschnitt A.2 aufgelistet. Für jedes Experiment wird eine Batchgröße von 64 oder 32 Bildern gewählt.

Mit einem Early Stopping nach 15 Epochen ohne Kostensenkung[4] wird ein Overfitting vermieden. Wenn ein Training nicht konvergiert, so wird es nach 80 Epochen abgebrochen. Die Netzarchitektur wird mit vortrainierten Gewichten initialisiert.

Als Optimierer (siehe Unterabschnitt 2.4.8) wird Adam [KB15] verwendet, da dieser am robustesten gegenüber der Wahl der Hyperparameter ist und gleichzeitig gute Ergebnisse liefert.

Um eine bessere Generalisierung zu erreichen, werden die Bilder in jedem Trainingsschritt leicht augmentiert. Analog zu YOLOv2 [RF17] wird die Helligkeit, der Kontrast und der Farbton angepasst. Zusätzlich wird nach YOLOv3 [RF18] zufällig zugeschnitten und rotiert, sowie zufällige Bereiche ausgeschnitten und zufällig befüllt. Eine Spiegelung der Bilder ist für eine Fahrstreifenerkennung nicht zu empfehlen, da hierdurch die Fahrtrichtung einer Straße umgekehrt wird und sich das Ego-Fahrzeug auf der linken Fahrbahn befindet. Die Parameter der Augmentierung befinden sich in Abschnitt A.2.

Es werden jeweils vier Experimente gleichzeitig auf einem Knoten mit zwei Intel Xeon Platinum 8368 CPUs (je 38 Kerne) mit vier NVIDIA A100-40 GPUs (40 GB) ausgeführt[5]. Die Implementierung wurde mit PyTorch 1.11.0, CUDA 11.3 und Python 3.6.8 auf einem Red Hat 8.4 Linux System umgesetzt.

[4] Da nur alle 3 Epochen auf dem Validierungsdatensatz evaluiert wird, werden bei einem Early Stopping nach 15 Epochen 5 Berechnungen zur Validierung der Konvergenz herangezogen.

[5] Die Trainings in dieser Arbeit wurden auf dem Supercomputer HoreKa durchgeführt, der vom Ministerium für Wissenschaft, Forschung und Kunst Baden-Württemberg und vom Bundesministerium für Bildung und Forschung gefördert wird.

6.4 Experimente

Die Kernelemente des vorgestellten Konzepts werden in diesem Abschnitt untersucht und einzelne Gestaltungsoptionen einander gegenüber gestellt. Beginnend bei einer initialen Konfiguration werden einzelne Parameterbelegungen ausgetauscht und deren Auswirkungen betrachtet. Zwischen den Experimenten werden die Ergebnisse diskutiert und das weitere Vorgehen bestimmt, sodass im Laufe des Kapitels Entscheidungen für eine geeignete Parametrisierung getroffen werden. Die Visualisierung der Prädiktion im Bild erfolgt typischerweise wie in Abbildung 6.4.

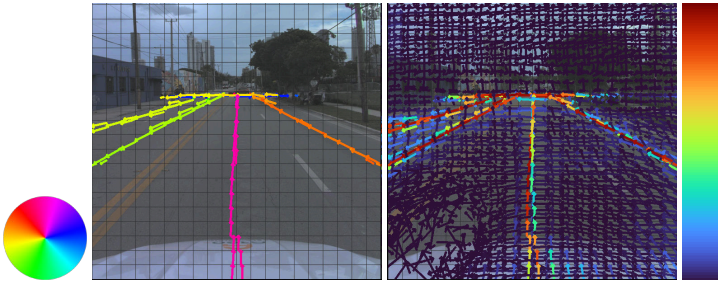


Abbildung 6.4: Prädiktion mit der Basiskonfiguration auf dem Argoverse-Datensatz. Eine Prädiktion wird sowohl vollständig (rechts) als auch gefiltert (links) visualisiert. Die Farbschemata der beiden Varianten sind daneben dargestellt. Links wird mit der Farbe die Orientierung der Prädiktion kodiert und somit ein Eindruck der Fahrtrichtung vermittelt. Rechts kodiert die Farbe die Konfidenz eines Liniensegments von Dunkelblau ($c = 0$) bis rot ($c = 1$).

6.4.1 Rezeptives Feld

Die Vergrößerung des rezeptiven Feldes durch Aufweitung der Kernel in den faltenden Schichten (Unterabschnitt 4.3.3) wird in diesem Abschnitt untersucht. Tabelle 6.2 zeigt Experimente bei dem der Kernel auf jedes, jedes zweite, dritte und vierte Pixel angewendet wird.

Die Tabelle zeigt, dass die Aufweitung im Training keine Verbesserung bringt. Mit jedem Aufweitungsschritt verringert sich das F1-Maß. Für weitere Experimente wird daher **keine Aufweitung eingesetzt**.

d	F1	Re	Pr	Acc	Kf	KfTP	$\|\cdot\|_2$	MP	L
1	**0,44**	**0,47**	0,41	**0,96**	**0,06**	**0,18**	**3,58**	**3,06**	**4,21**
2	0,43	**0,47**	0,40	**0,96**	**0,06**	**0,18**	3,62	3,13	4,27
3	0,43	0,43	**0,43**	**0,96**	**0,06**	0,19	3,60	3,12	4,22
4	0,42	**0,47**	0,39	**0,96**	**0,06**	0,19	3,72	3,23	4,40

Tabelle 6.2: Ergebnisse des Experiments mit Aufweitungen (d) von ein bis vier Pixeln.

6.4.2 Linienrepräsentationen und Aktivierungsfunktionen

Im Konzeptteil werden vier Linienrepräsentationen vorgestellt (siehe Abschnitt 4.4). Da die Randkoordinaten und Euler-Winkel bereits in [MSPS21] keinen Vorteil gegenüber den kartesischen Start- und Endpunkten gezeigt haben, beschränkt sich dieser Abschnitt auf die kartesischen Start- und Endpunkte (SE) sowie die erst im Laufe dieser Arbeit entwickelten Mittelpunkt-Richtung-Koordinaten (MR).

Linie	A_g,A_c	F1	Re	Pr	Acc	Kf	KfTP	$\|\cdot\|_2$	MP	L	Kv
SE	Li,Li	*0,42*	0,45	*0,40*	**0,96**	*0,08*	0,23	3,55	3,69	**2,53**	48
SE	Li,Si	*0,42*	0,45	0,39	**0,96**	**0,06**	0,22	3,58	3,70	2,62	27
SE	Si,Li	0,43	*0,46*	*0,40*	**0,96**	0,07	0,19	4,72	4,57	5,09	69
SE	Si,Si	*0,42*	0,47	0,39	**0,96**	**0,06**	0,17	4,67	4,49	5,08	30
MR	Li,Li	**0,44**	0,47	**0,42**	**0,96**	0,07	0,21	**3,53**	**3,05**	*4,19*	51
MR	Li,Si	0,43	**0,48**	*0,40*	**0,96**	**0,06**	*0,18*	*3,56*	3,07	*4,19*	39
MR	Si,Li	**0,44**	0,47	**0,42**	**0,96**	**0,06**	0,19	3,74	3,32	4,72	66
MR	Si,Si	**0,44**	**0,48**	0,41	**0,96**	**0,06**	**0,16**	3,77	*3,30*	4,74	42

Tabelle 6.3: Experiment zum Vergleich verschiedener Linienrepräsentationen (Linie) und Aktivierungsfunktionen für Geometrie A_g und Konfidenz A_c.

In Tabelle 6.3 werden die Ergebnisse der Trainings dargestellt. Es wird je ein Netz mit der jeweiligen Linienrepräsentation und einer Auswahl an Aktivierungsfunktionen trainiert. Für die Geometrie und die Konfidenz wird jeweils

entweder die Sigmoid-Funktion (Si) oder die lineare Aktivierungsfunktion (Li) gewählt.

Die MR-Darstellung erreicht unabhängig von der Aktivierungsfunktion ein besseres F1-Maß. Die Konfidenzen werden unabhängig von der Linienrepräsentation mit einer Sigmoid-Aktivierung (*,Si) präziser geschätzt. Dieser Unterschied zeigt sich insbesondere bei der Differenz der Konfidenz der TPs (KfTP).

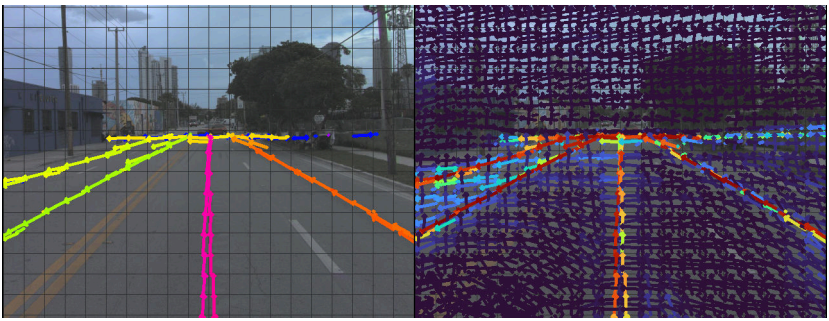


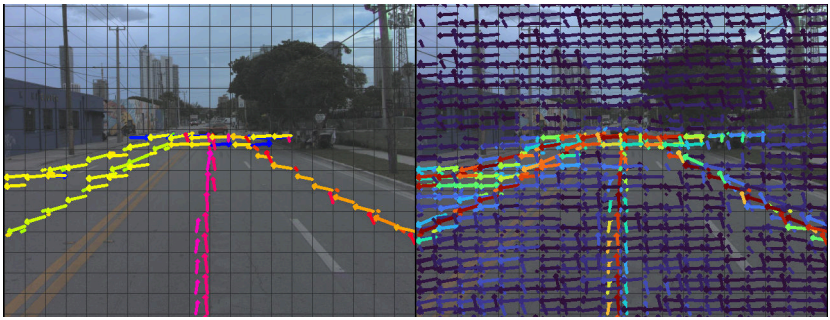
Abbildung 6.5: Beispielbilder für das beste Ergebnis (MR Li,Si) der Untersuchung der Linienrepräsentation und Aktivierungsfunktionen.

Die Schätzung der Geometrie ist in diesem Experiment mit einer linearen Aktivierung (Li,*) am besten. Dabei zeigt die (SE Li,*)-Variante eine bessere Schätzung der Länge der Liniensegmente (L), während die (MR Li,*)-Variante die Start- und Endpunkte ($\|\cdot\|_2$) sowie den Mittelpunkt (MP) präziser schätzt. Die Sigmoid-Aktivierung ist für die Geometrie nicht erfolgversprechend, da große geometrische Fehler entstehen, die sich insbesondere in Abbildung 6.6 im Vergleich zu Abbildung 6.5 zeigen.

Zusammenfassend besteht zwischen den Linienrepräsentationen nur ein geringfügiger Unterschied, sodass beide für die vorliegende Aufgabe geeignet sind. Für weitere Experimente werden die **MR-Koordinaten verwendet, da das F1-Maß höher** ist.

Zudem sollten für eine höhere Präzision der Ergebnisse beide Linienrepräsentationen mit einer **linearen Aktivierungsfunktion für die Geometrie** trainiert werden. Für die Konfidenz ist der Unterschied zwischen Sigmoid und linearer Aktivierung nicht ausschlaggebend. Da aus theoretischer Sicht die **Sigmoid-**

(a) kartesische Start- und Endpunkte (SE) mit Sigmoid-Funktion (Si) für die Geometrie und lineare Aktivierungsfunktion (Li) für die Konfidenz



(b) kartesische Start- und Endpunkte (SE) mit Sigmoid-Funktion (Si) für die Geometrie und Sigmoid-Funktion (Si) für die Konfidenz

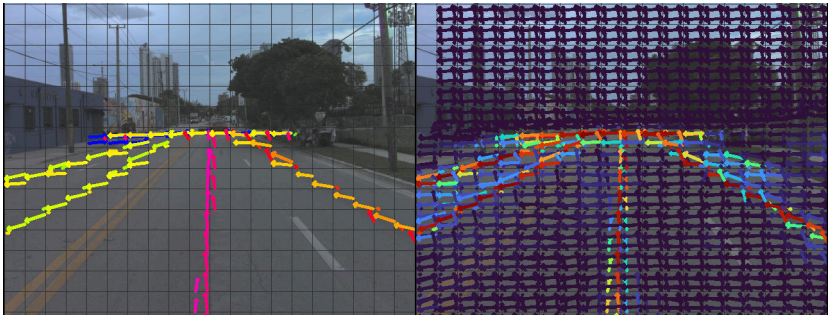


Abbildung 6.6: Beispielbilder für Ergebnisse mit der Sigmoid-Aktivierungsfunktion auf der SE-Darstellung.

Aktivierungsfunktion für die Konfidenz geeigneter ist, wird diese empfohlen und für die weiteren Experimente verwendet. Ein Beispielbild der besten Variante im Experiment findet sich in Abbildung 6.5.

Für dieses Experiment wird das Training mit einer quadratischen Kostenfunktion durchgeführt. Um den Exponentialanteil der Sigmoid-Funktion auszugleichen und somit verschwindende Gradienten zu vermeiden, wird in der Literatur empfohlen, eine logarithmische Kostenfunktion einzusetzen (siehe Abschnitt 2.4). Tabelle 6.4 zeigt die Ergebnisse mit einer binären Kreuzentropie als Kostenfunktion für die Sigmoid-Aktivierungsfunktion der Konfidenz.

L	A_g,A_c	Kfkt.	F1	Re	Pr	Acc	Kf	KfTP	$\|\cdot\|_2$	MP	L	Kv
SE	Li,Si	quad.	0,42	0,45	0,39	0,96	0,06	0,22	3,58	3,70	2,62	27
MR	Li,Si	quad.	0,43	0,48	0,40	0,96	0,06	0,18	3,56	3,07	4,19	39
SE	Li,Si	BCE	0,42	0,45	0,39	0,96	0,06	0,20	3,64	3,74	2,66	24
MR	Li,Si	BCE	0,43	0,46	0,41	0,96	0,06	0,19	3,67	3,13	4,32	24

Tabelle 6.4: Erweiterung der Experimente zum Vergleich der Linienrepräsentationen mit der binären Kreuzentropie (BCE) als Kostenfunktion (Kfkt.) im Vergleich zur quadratischen Kostenfunktion (quad.) für die Konfidenz.

Diese zeigt, dass die binäre Kreuzentropie keine Verbesserung der Ergebnisse bringt. Zudem zeigt sich mit dieser Kostenfunktion bereits nach 24 Epochen ein Overfitting auf den Trainingsdaten, sodass das gelernte Modell schlecht generalisiert. Somit wird weiterhin die **Sigmoid-Aktivierungsfunktion für die Konfidenz mit einem quadratischen Fehlermaß trainiert**.

6.4.3 Anker

Für die Anker der Prädiktoren (siehe Abschnitt 4.5) wird in diesem Abschnitt geprüft, welcher Clusterraum am geeignetsten ist. Da die Mittelpunkt-Richtungskoordinaten (MR) aus Mittelpunkt-Koordinaten (M) und Richtungskoordinaten (R) bestehen, werden in diesem Abschnitt sowohl die Kombination als auch die einzelnen Dimensionen untersucht. Zudem werden in Unterabschnitt 4.5.2 zwei Strategien zur Bestimmung der Anker vorgestellt (Clustering und Gleichverteilung), die in diesem Abschnitt ebenfalls betrachtet werden.

Anteil der Mehrfachzuweisungen

Zum Vergleich der verschiedenen Ankerstrategien und Clusterräume wird untersucht, wie häufig im Validierungsdatensatz mehr als ein GT-Liniensegment demselben Anker zugewiesen wird. Diese Metrik gibt Auskunft darüber, wie gut die GT mit den Ankern repräsentiert werden kann. Jede Mehrfachzuweisung kann nicht im Training auftauchen, da es keinen Prädiktor gibt, der das entsprechende GT-Liniensegment als Trainingszielvorgabe erhält.

		Anteil Mehrfachzuweisungen			Abstand zum oberen Bildrand (in Zellen)	
	Cluster	Schnitt	Median	Max.	Mittelw.	Varianz
R	6-Means	21,41	20,96	**55,78**	9,12	*18,06*
R	8 gleichverteilt	27,88	27,41	67,66	9,51	**17,14**
R	8-Means	*21,05*	*19,54*	58,33	*9,03*	18,10
R	16-Means	18,85	16,17	60,06	8,72	17,79
R	24-Means	**16,88**	**14,29**	*59,03*	**8,50**	18,57
M	6-Means	20,05	19,05	**61,54**	8,82	**19,53**
M	8 gleichverteilt	21,14	19,16	65,97	*8,61*	20,08
M	8-Means	*19,62*	*17,49*	*63,41*	8,62	*20,35*
M	16-Means	15,02	12,07	64,29	8,11	20,38
M	24-Means	**13,40**	**10,53**	61,90	**7,75**	20,56
MR	6-Means	20,06	19,23	*56,18*	8,99	18,84
MR	8-Means	20,10	18,59	59,38	8,86	19,17
MR	16 gleichverteilt	17,81	15,70	57,14	8,54	*19,14*
MR	16-Means	*17,13*	*14,88*	56,46	*8,37*	19,90
MR	24gleichverteilt	15,92	13,80	**53,36**	8,35	**18,78**
MR	24-Means	**15,05**	**12,50**	55,78	**8,14**	20,68

Tabelle 6.5: Anteil der Mehrfachzuweisungen der Anker je Bild als Durchschnitt, Median und Maximum des Datensatzes. In der rechten Spalte sind zudem die Positionen der Mehrfachzuweisungen im 20×20 Zellgitter als Mittelwert und Varianz über den Datensatz aufgetragen.

In Tabelle 6.5 wird der Anteil der Anker, bei denen eine Mehrfachzuweisung auftritt, für die Clustermerkmale Mittelpunkt-Richtung-Koordinaten (MR), Mittelpunkt-Koordinaten (M) und Richtungskoordinaten (R) verglichen. Hier zeigt sich, wie zu erwarten, dass eine höhere Anzahl der Anker zu weniger Mehrfachzuweisungen führt. Im Training führt eine Steigerung der Anzahl der Anker jedoch typischerweise zu weniger Trainingserfolg, da immer weniger Trainingsbeispiele für den einzelnen Prädiktor zur Verfügung stehen. Zusätzlich tritt **im Mittelpunktraum eine Doppelung im Schnitt weniger häufig** auf als in den anderen Räumen, jedoch gibt es hier mehr Ausreißer.

Beim Vergleich zwischen den k-Means-Prädiktoren und den gleichverteilten Ankervarianten sind die **k-Means-Prädiktoren die bessere Wahl** in Bezug auf die Mehrfachzuweisungen.

Position der Mehrfachzuweisungen

Neben der Betrachtung der Anzahl der Mehrfachzuweisungen ist die Position der Mehrfachzuweisungen im Bild relevant. Wenn eine Ankervariante bspw. primär am Horizont Doppelungen aufweist, ist diese einer über das ganze Bild verteilten Ankervariante deutlich vorzuziehen. Tabelle 6.5 zeigt die Abstände (in Anzahl an Zellen) vom oberen Bildrand für ein Zellgitter von 20×20 Zellen. Erneut zeigen hier die **Mittelpunkt-Koordinaten (M) die besseren Ergebnisse**, gefolgt von der Mittelpunkt-Richtung-Koordinaten (MR). Die Unterschiede zwischen den einzelnen Varianten sind hier jedoch deutlich geringer.

Trainingsergebnisse

Neben den Mehrfachzuweisungen werden bei der Bewertung der unterschiedlichen Clusterräume insbesondere die Trainingseigenschaften betrachtet. Die Ergebnisse eines solchen Vergleichs zeigt Tabelle 6.6.

Die Ergebnisse zeigen, dass die Mittelpunkt-Koordinaten (M) zwar weniger Doppelzuweisungen im Datensatz erzeugen, das Training mit diesen Ankern dennoch ein schlechteres F1-Maß erzielt als die Vergleichsexperimente. Zudem sind die geometrischen Abweichungen fast doppelt so hoch wie bei den übrigen Ankervarianten. Die Ergebnisse der Richtungskoordinaten (R) und Mittelpunkt-Richtung-Koordinaten (MR) sind ähnlich, wobei R einen höheren Recall (Re) und MR eine höhere Precision (Pr) aufweisen.

Generell **steigt das F1-Maß, je weniger Prädiktoren** das Netz zur Verfügung hat. Dies ist darin begründet, dass mit wenigen Prädiktoren ein einzelner Prädiktor deutlich häufiger verantwortlich ist und daher häufiger Kosten zugewiesen bekommt, von denen er lernen kann. Gleichzeitig **steigt die geometrische Präzision mit Anzahl der Prädiktoren**, da sich ein Prädiktor immer stärker auf bestimmte geometrische Eigenschaften spezialisieren kann. Die für Konvergenz nötige Anzahl an Trainingsepochen nimmt mit der Zahl der Prädiktoren zu.

A	$\|P\|$	C	F1	Re	Pr	Acc	Kf	KfTP	$\|\cdot\|_2$	MP	L	Kv	$t_{\mathcal{L}}$
R	6	kM	0,45	**0,50**	*0,41*	0,95	0,07	0,15	4,22	3,77	4,60	48	27
R	8	gl	**0,47**	0,47	**0,48**	*0,97*	0,04	**0,13**	5,69	4,76	6,87	42	26
R	8	kM	*0,44*	0,47	0,43	0,96	*0,05*	*0,16*	*4,06*	*3,66*	*4,36*	45	27
R	16	kM	0,40	*0,42*	0,39	0,98	0,04	0,21	3,48	3,25	4,11	36	29
R	24	kM	0,38	0,37	0,39	**0,99**	**0,02**	0,22	**3,09**	**2,98**	**3,80**	54	29
M	6	kM	**0,41**	**0,43**	**0,39**	*0,95*	0,08	*0,22*	7,58	7,19	6,51	33	28
M	8	gl	0,40	0,41	0,38	0,96	*0,06*	**0,20**	6,87	6,65	7,89	72	28
M	8	kM	*0,39*	0,41	0,38	0,96	*0,06*	0,21	*6,86*	**6,62**	5,43	78	27
M	16	kM	0,32	*0,34*	*0,30*	**0,98**	0,05	0,29	6,84	*6,76*	*5,70*	51	30
M	24	kM	0,29	0,29	*0,30*	**0,98**	**0,03**	0,30	**6,78**	6,80	**5,19**	69	30
MR	6	kM	**0,44**	**0,48**	0,41	0,95	0,07	0,19	3,83	3,27	4,28	30	27
MR	8	kM	**0,44**	0,47	0,41	*0,96*	*0,06*	**0,16**	3,51	3,05	4,14	39	27
MR	16	gl	0,41	*0,41*	**0,42**	0,98	0,03	0,19	3,17	2,67	3,76	66	29
MR	16	kM	*0,40*	0,40	0,41	0,98	0,03	0,21	2,79	*2,40*	3,06	42	29
MR	24	gl	0,39	*0,41*	*0,38*	**0,99**	**0,02**	*0,20*	*2,83*	2,36	*3,34*	57	29
MR	24	kM	0,38	*0,41*	0,36	**0,99**	0,03	0,21	**2,32**	**2,00**	**2,62**	57	30

Tabelle 6.6: Ergebnisse der Trainings mit unterschiedlichen Clusterräumen (A), Anzahl der Anker ($|P|$) und Clustervariante (C) als k-Means (kM) oder gleichverteilte Anker (gl).

Im Vergleich zwischen k-Means und gleichverteilten Ankern weisen die **gleichverteilten Anker immer ein besseres F1-Maß** auf als die k-Means-Variante mit der gleichen Anzahl an Prädiktoren. Gleichzeitig sind jedoch die **geometrischen Distanzen, insbesondere bei gleichverteilten R-Ankern, deutlich schlechter** als bei den k-Means-Varianten.

Es muss demnach anhand der Anwendung zwischen geometrischer Präzision und hohem F1-Maß priorisiert werden. Im Folgenden wird zugunsten eines höheren F1-Maßes priorisiert und daher mit **den Mittelpunkt-Richtung-Koordinaten mit 8-Means-Ankern** weitergearbeitet.

6.4.4 Dynamische Zuordnung

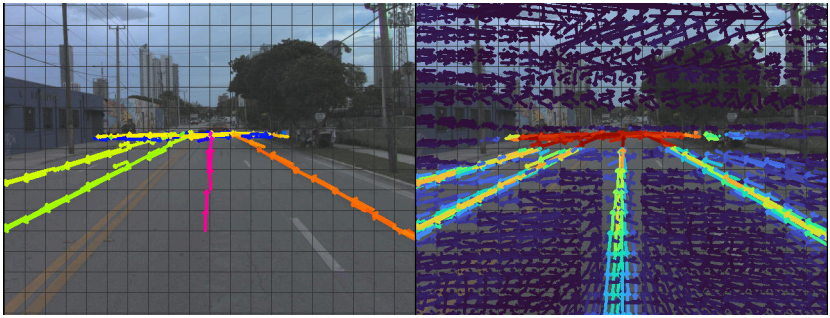
In Unterabschnitt 4.5.1 wird die dynamische Zuordnung der Prädiktoren zur GT vorgestellt. In diesem Abschnitt wird untersucht, ob dies einen Vorteil gegenüber den festen Ankern bietet. Es werden zudem zwei Strategien für die Assoziation vorgestellt: mit und ohne Konfidenz im Assoziationsraum (c). Tabelle 6.7 zeigt die Ergebnisse der Trainings zum Vergleich.

$\omega_1;\omega_0$	$\omega_g;\omega_c$	c	F1	Re	Pr	Acc	KfTP	$\|\cdot\|_2$	MP	L	Kv	$t_{\mathcal{L}}$
$\frac{1}{9};\frac{8}{9}$	$\frac{1}{2};\frac{1}{2}$	-	0,44	0,47	0,42	0,96	0,17	3,54	3,04	4,13	45	27
$\frac{1}{9};\frac{8}{9}$	$\frac{1}{2};\frac{1}{2}$	×	**0,46**	**0,53**	0,40	0,96	**0,24**	3,67	3,15	3,61	48	204
$\frac{1}{11};\frac{10}{11}$	$\frac{1}{2};\frac{1}{2}$	×	**0,46**	0,46	0,45	0,96	0,25	3,77	3,24	3,76	45	208
$\frac{1}{11};\frac{10}{11}$	$\frac{2}{3};\frac{1}{3}$	×	0,45	*0,44*	0,46	**0,97**	*0,26*	*3,63*	3,12	*3,56*	54	197
$\frac{1}{13};\frac{12}{13}$	$\frac{1}{2};\frac{1}{2}$	×	0,45	*0,44*	0,47	**0,97**	*0,26*	3,64	*3,13*	3,60	45	199
$\frac{1}{13};\frac{12}{13}$	$\frac{2}{3};\frac{1}{3}$	×	*0,43*	0,39	*0,50*	**0,97**	0,27	3,68	3,18	3,63	45	201
$\frac{1}{9};\frac{8}{9}$	$\frac{1}{2};\frac{1}{2}$	✓	*0,59*	**0,68**	0,52	**0,97**	**0,14**	4,79	4,05	4,76	48	196
$\frac{1}{11};\frac{10}{11}$	$\frac{1}{2};\frac{1}{2}$	✓	0,60	*0,64*	*0,58*	**0,97**	0,15	4,71	3,96	4,64	51	197
$\frac{1}{11};\frac{10}{11}$	$\frac{1}{3};\frac{2}{3}$	✓	**0,61**	*0,64*	0,59	**0,97**	**0,14**	4,75	4,01	**4,58**	51	197
$\frac{1}{11};\frac{10}{11}$	$\frac{1}{6};\frac{5}{6}$	✓	0,60	0,65	0,56	**0,97**	**0,14**	4,88	4,08	4,77	30	210
$\frac{1}{11};\frac{10}{11}$	$\frac{2}{3};\frac{1}{3}$	✓	*0,59*	0,62	0,56	**0,97**	0,19	4,76	*3,99*	4,66	30	194
$\frac{1}{13};\frac{12}{13}$	$\frac{1}{2};\frac{1}{2}$	✓	0,60	0,62	0,59	**0,97**	**0,14**	*4,72*	*3,99*	*4,63*	48	194
$\frac{1}{13};\frac{12}{13}$	$\frac{1}{3};\frac{2}{3}$	✓	0,60	0,61	*0,58*	**0,97**	*0,17*	4,77	4,03	4,61	24	201
$\frac{1}{13};\frac{12}{13}$	$\frac{1}{6};\frac{5}{6}$	✓	**0,61**	0,62	**0,60**	**0,97**	**0,14**	4,77	4,04	4,67	51	196
$\frac{1}{13};\frac{12}{13}$	$\frac{2}{3};\frac{1}{3}$	✓	*0,59*	0,61	*0,58*	**0,97**	*0,17*	**4,67**	**3,95**	*4,63*	48	200

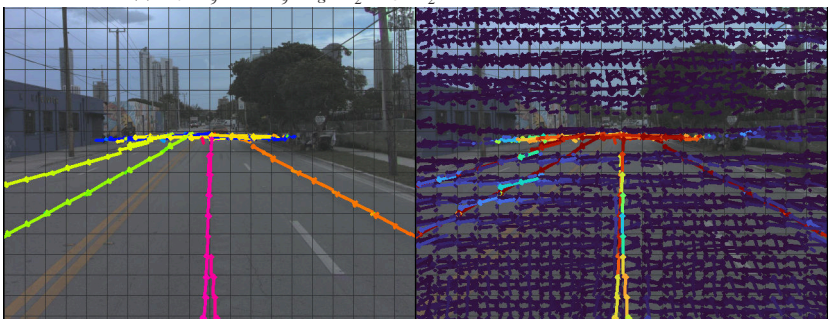
Tabelle 6.7: Vergleich der dynamischen Zuordnung mit (✓) und ohne (×) Einbeziehung der Konfidenz in die Assoziation (c) sowie mit unterschiedlicher Gewichtung der Konfidenzterme ($\omega_0;\omega_1$). Zum Vergleich zeigt die erste Zeile die Variante mit festen Ankern.

Die Unterschiede der beiden Varianten zeigen sich deutlich. Wenn die Konfidenz nicht in den Assoziationsraum aufgenommen wird, so ist das Ergebnis vergleichbar mit bisherigen Experimenten mit einer leichten Verbesserung des F1-Maßes und der Schätzung der Länge der Liniensegmente.

Wird die Konfidenz in den Assoziationsraum aufgenommen, so **steigt das F1-Maß um 15 Prozentpunkte**. Gleichzeitig sind die geometrischen Abweichungen mindestens ein halbes Pixel schlechter als ohne Konfidenz in der Assoziation.



(a) $\omega_0 = \frac{8}{9}$, $\omega_1 = \frac{1}{9}$, $\omega_g = \frac{1}{2}$, $\omega_c = \frac{1}{2}$ ohne Konfidenz in der Assoziation



(b) $\omega_0 = \frac{12}{13}$, $\omega_1 = \frac{1}{13}$, $\omega_g = \frac{1}{6}$, $\omega_c = \frac{5}{6}$ mit Konfidenz in der Assoziation

Abbildung 6.7: Beispielbilder für Trainings mit einer dynamischen Zuordnung. Die Schätzung der Konfidenz ist ohne Konfidenz im Assoziationsraum, insbesondere im vorderen Bildbereich, deutlich schlechter.

Bei Betrachtung der Ergebnisbilder in Abbildung 6.7 zeigt sich dieser Unterschied deutlich. Wenn die Konfidenz nicht im Assoziationsraum enthalten ist, wird die Konfidenz von Liniensegmenten am Horizont besser geschätzt als am unteren Bildrand. Dies ist darauf zurückzuführen, dass die Prädiktoren am Horizont sehr häufig in einem Trainingsbeispiel verantwortlich sind und somit Kosten zugewiesen bekommen. Am unteren Bildrand ist durch die Perspekti-

ve die Varianz der Mittellinien-Position sehr groß, sodass für viele Zellen im Training seltener verantwortliche Konfidenzkosten formuliert werden.

Durch die Verbesserung des F1-Maßes ist die dynamische Zuordnung den Ankern klar vorzuziehen. Wird ein schnelles, effizientes Training benötigt, gilt diese Einschätzung jedoch nicht, da ein Training mit **vordefinierten Ankern schneller konvergiert und kürzere Trainingszeiten** aufweist, sodass für die Einzelstudien in dieser Arbeit weiter mit den Ankern trainiert wird.

6.4.5 Skalierung

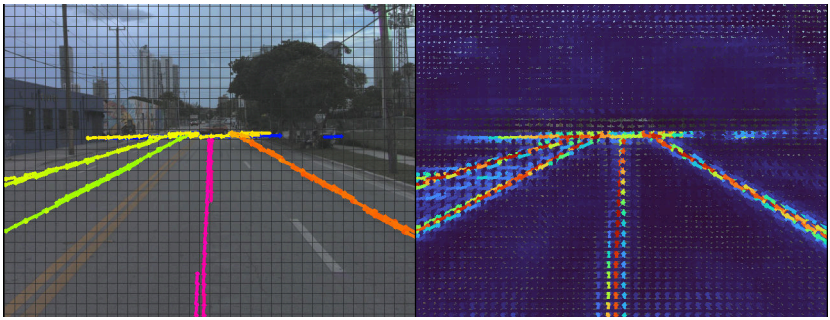
Die Anzahl der Zellen und damit deren Auflösung können mit einer Erweiterung des Decoders erhöht werden. In Unterabschnitt 4.3.2 werden die Architekturen für Zellen der Größe 32×32 Pixel, 16×16 Pixel und 8×8 Pixel vorgestellt. Tabelle 6.8 zeigt verschiedene Trainings auf den drei Skalen (□) mit unterschiedlicher Gewichtung der Konfidenzkostenterme (verantwortlich ω_1 und nicht verantwortlich ω_0). Für die Experimente muss eine Batchsize von 32 gewählt werden, da die Varianten mit der größten Skalierung einen deutlich größeren Speicher benötigen als die bisherigen Trainings.

□	$\omega_1;\omega_0$	F1	Re	Pr	Acc	Kf	KfTP	$\|\cdot\|_2$	MP	L	Kv	$t_{\bowtie}$
32	$\frac{1}{9};\frac{8}{9}$	0,44	0,47	0,42	0,96	0,06	0,17	3,54	3,04	4,13	45	10
16	$\frac{1}{9};\frac{8}{9}$	0,34	**0,53**	0,26	0,97	0,06	**0,19**	1,69	1,51	1,98	30	27
16	$\frac{1}{13};\frac{12}{13}$	0,37	0,46	0,32	**0,98**	**0,05**	0,21	1,66	1,47	1,94	45	27
16	$\frac{1}{17};\frac{16}{17}$	**0,38**	0,41	**0,35**	**0,98**	**0,05**	0,25	**1,62**	**1,44**	**1,92**	30	27
8	$\frac{1}{13};\frac{12}{13}$	0,27	**0,47**	0,19	0,98	0,05	**0,26**	0,80	0,72	0,99	27	42
8	$\frac{1}{17};\frac{16}{17}$	**0,28**	0,40	0,22	0,98	0,04	0,28	0,78	0,71	0,97	27	42
8	$\frac{1}{33};\frac{32}{33}$	0,24	0,20	**0,31**	**0,99**	**0,03**	0,39	**0,74**	**0,67**	**0,91**	12	43

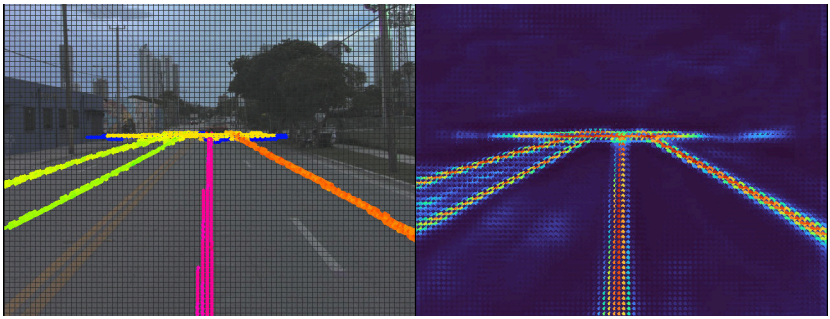
Tabelle 6.8: Ergebnisse der Trainings mit unterschiedlichen Skalierungen des Zellgitters (□).

Die Tabelle zeigt deutliche Unterschiede insbesondere in der Inferenzzeit des Netzes ($t_{\bowtie}$). Die Precision und damit das F1-Maß sinkt mit jeder Vergrößerung des Zellgitters ab, sodass der **initial gewählte Skalierungsfaktor von 32 das beste Ergebnis** liefert. Dies ist durch die verrauschten GT-Daten begründet,

durch die die Position der Mittellinie nicht präzise und konsistent bestimmt werden kann. Dadurch streuen die GT-Linien insbesondere am unteren Bildrand, was sich als Ungenauigkeit in der Prädiktion widerspiegelt. Dies zeigt sich insbesondere bei Betrachtung der Visualisierung in Abbildung 6.8: in mehreren Zellen am unteren Bildrand wird je ein Liniensegment für dieselbe GT-Linie prädiziert.



(a) 16×16 Pixel je Zelle



(b) 8×8 Pixel je Zelle

Abbildung 6.8: Prädiktion auf verschiedenen Skalierungsstufen. Je mehr Zellen das Zellgitter hat, desto mehr Zellen werden durch verrauschte GT-Daten aktiviert. Dies führt bei mehr Zellen zu einem höheren Recall. Dies kann durch eine NMS leicht behoben werden.

Der geometrische Fehler der Prädiktion verringert sich bei jedem Skalierungsschritt, sodass die mittlere Abweichung bei einer Auflösung von 8×8 Pixeln im Subpixelbereich liegt. Mit einer geeigneten Non-Maximum-Suppression (NMS), wie sie in Kapitel 5 eingeführt wurde, kann der Recall deutlich redu-

ziert und gleichzeitig die geometrische Präzision beibehalten werden, sodass die **Skalierung auf** 8×8 **Pixel Zellen in Kombination mit einer NMS eine sinnvolle Option** ist. Es gilt jedoch für jede Anwendung individuell abzuwägen, ob die **zwei- bis vierfache Inferenzzeit für höhere Skalen** gerechtfertigt ist.

6.4.6 Diskretisierung

Durch die Diskretisierung der Linienzüge in räumliche Zellen wird bei starken Krümmungen ein Fehler gegenüber der GT erzeugt. Dieser Fehler entsteht zum einen durch das Entfernen von Stützpunkten innerhalb einer Zelle und zum anderen bei den randgebundenen Linienrepräsentationen durch Extrapolation oder Kürzung von Linien, um am Rand der Zelle zu enden bzw. zu starten.

Um eine Einschätzung des Fehlers zu bekommen, wird zum einen die maximale Abweichung der entfernten Knotenpunkte zum Liniensegment je Zelle ($D_{\max}$) gemessen – dies entspricht der Idee der Hausdorff-Distanz zwischen dem Linienzug und dem neuen Liniensegment. Zum anderen wird die entfernte Fläche (A) zwischen dem Liniensegment und dem GT-Linienzug beschrieben (siehe Abbildung 6.9).

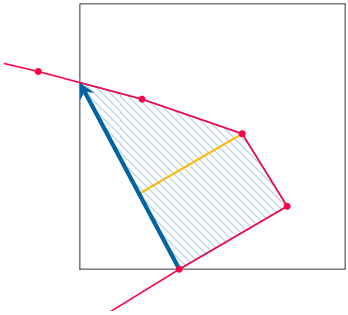


Abbildung 6.9: Bei starken Krümmungen der Linienzüge kann es zu Diskretisierungsfehlern kommen. Wenn der Linienzug (rot) durch das Liniensegment (blau) approximiert wird, wird die eingeschlossene Fläche (blau schraffiert) sowie der maximale Abstand zum Linienzug (gelb) evaluiert, um den Diskretisierungsfehler zu quantifizieren.

Tabelle 6.9 zeigt die Ergebnisse auf 700 zufällig gewählten Bildern aus dem Argoverse-Datensatz. Die Verkleinerung der Zellen reduziert den Diskretisie-

rungsfehler im gleichen Maß wie die Zellgröße. **Sämtliche Distanzmaße D halbieren sich bei jedem Skalenschritt**, bei dem jeweils die Zellbreite/-höhe ebenfalls halbiert wird. Die Fläche A zeigt diesen Trend ebenfalls. Gleichzeitig wird die Inferenzzeit, wie bereits im vorangehenden Abschnitt beschrieben, deutlich erhöht. Auch hier muss demnach eine Abwägung je nach Anwendung vorgenommen werden.

Skale	$\overline{D}_{\max}$	$\max(D_{\max})$	$\overline{A}_{\text{sum}}$	$\max(A_{\text{sum}})$	$t_{\bowtie}$
32	7,7	31,7	246	1335	10
16	4,0	15,2	71	345	27
8	2,1	7,8	20	112	42

Tabelle 6.9: Die maximale Distanz $D_{\max}$ der GT-Linie zu den Liniensegmenten durch die Diskretisierung je Bild, zusammengefasst als das Mittel $\overline{D}_{\max}$ und das Maximum im Datensatz $\max(D_{\max})$. Die Summe der Flächen A_{sum} zwischen den Liniensegmenten und den GT-Linien je Bild als Mittelwert $\overline{A}_{\text{sum}}$ und Maximum im Datensatz $\max(A_{\text{sum}})$. Zur Bestimmung der Daten wurden 700 zufällige Bilder aus dem Argoverse-Datensatz ausgewählt.

6.4.7 Gewichtung der Kostenfunktion

Die Gewichtung der Kostenterme kann, wie in den bisherigen Experimenten, empirisch oder nach [KGC18] über die Varianz σ^2 der Einzelterme erfolgen (siehe Abschnitt 4.6). Für die bisherigen Experimente wurden konstante Kostengewichte für verantwortliche Prädiktoren mit $\omega_1 = \frac{7}{8}$ und nicht-verantwortlichen Prädiktoren mit $\omega_0 = \frac{1}{8}$ eingesetzt. Die Geometriekosten wurden gegenüber den Konfidenzkosten bisher gleich gewichtet, sodass $\omega_g = \frac{1}{2}$ und $\omega_c = \frac{1}{2}$. Für die Experimente in diesem Abschnitt, bei denen die Gewichtung gelernt wird, wird ω_c konstant gesetzt, da dies nur ein Vorfaktor für die Einzelgewichte ω_1 und ω_0 ist, wie Gleichung 6.1 und Gleichung 6.2 zeigen. $\mathbb{1}_1$ und $\mathbb{1}_0$ beschreiben dabei die Indikatorfunktion aus Gleichung 4.12 für verantwortliche und nicht-verantwortliche Prädiktoren.

$$\mathcal{L} = \omega_g \mathcal{L}_g \quad + \quad \omega_c \mathcal{L}_c \tag{6.1}$$

$$\mathcal{L} = \omega_g \mathcal{L}_g \quad + \quad \omega_c (\mathbb{1}_1 \omega_1 (c-1)^2 + \mathbb{1}_0 \omega_0 (c-0)^2) \tag{6.2}$$

Hierbei sind die $\log \sigma_\cdot$ zusätzliche Parameter, die im Training angepasst werden. Die Gewichtung der Kostenterme ergibt sich daraus mit konstantem ω_c als

$$\omega_g = \frac{1}{2\sigma_g^2} \quad , \quad \omega_0 = \frac{1}{\sigma_0^2} \quad , \quad \omega_1 = \frac{1}{\sigma_1^2}. \tag{6.3}$$

Tabelle 6.10 und Tabelle 6.11 zeigen die Trainingsergebnisse der gelernten Variante mit unterschiedlicher Initialisierung der Gewichtung und Variation der Lernrate. Die Spalten ω^* beschreiben den Wert der Gewichte nach 80 Epochen, wobei die Gewichte zur Stabilität des Trainings auf das Intervall $[0{,}05; 20]$ begrenzt werden.

$\omega_1;\omega_0$	$\omega_g;\omega_c$	F1	Re	Pr	Acc	$\lVert\cdot\rVert_2$	MP	L	ω_1^*	ω_0^*	ω_g^*
$\frac{1}{9};\frac{8}{9}$	$\frac{1}{2};\frac{1}{2}$	0,44	0,47	0,42	0,96	3,54	3,04	4,13	-	-	-
$\frac{1}{2};\frac{1}{2}$	$\frac{1}{2};\frac{1}{2}$	0,13	**0,75**	0,07	0,66	4,00	3,45	4,83	20,00	2,45	2,25
$\frac{1}{9};\frac{8}{9}$	$\frac{1}{2};\frac{1}{2}$	**0,45**	0,45	**0,47**	**0,97**	**3,43**	**2,97**	**4,13**	2,32	20,00	2,41
1;1	1;1	0,13	**0,75**	0,07	0,66	4,00	3,44	4,85	20,00	2,28	2,23
1;8	1;1	**0,45**	0,48	0,43	0,96	3,47	3,00	4,16	2,49	20,00	2,43

Tabelle 6.10: Trainingsergebnisse mit Gewichtung der Kostenfunktion nach [KGC18] mit Lernrate 0,001. Zum Vergleich werden verschiedene Initialisierungen für die Konfidenzgewichte für verantwortliche Prädiktionen ω_1 und nicht verantwortliche Prädiktionen ω_0 sowie die Gewichte für Geometrie ω_g und Konfidenz ω_c ausprobiert. Mit ω^* werden die Gewichte bezeichnet, zu denen die Trainings konvergieren. Alle Gewichte wurden im Training auf das Intervall $[0{,}05; 20]$ begrenzt.

Die Tabelle zeigt, dass in diesem Experiment die Konfidenzgewichte (ω_0, ω_1) bereits im richtigen Verhältnis initialisiert werden müssen, um ein hohes F1-Maß zu erreichen. Diese Stellschraube zum Ausgleich zwischen Precision und Recall kann nicht durch das Netz gelernt werden.

Die **gelernte Gewichtung der Kostenfunktion bringt somit für diese Anwendung keinen Vorteil für die Gewichtung der Konfidenzen**, da die Initia-

$\omega_1;\omega_0$	$\omega_g;\omega_c$	F1	Re	Pr	Acc	$\|\cdot\|_2$	MP	L	ω_1^*	ω_0^*	ω_g^*
$\frac{1}{9};\frac{8}{9}$	$\frac{1}{2};\frac{1}{2}$	0,44	0,47	0,42	0,96	3,54	3,04	4,13	-	-	-
$\frac{1}{2};\frac{1}{2}$	$\frac{1}{2};\frac{1}{2}$	0,34	0,60	0,24	0,93	3,73	3,22	4,24	2,00	1,99	1,90
$\frac{1}{9};\frac{8}{9}$	$\frac{1}{2};\frac{1}{2}$	0,44	0,45	0,43	**0,96**	**3,49**	**3,05**	**4,02**	0,44	3,54	1,90
1;1	1;1	0,34	**0,61**	0,23	0,92	3,74	3,24	4,29	4,04	4,00	3,17
1;8	1;1	**0,45**	0,44	**0,45**	**0,96**	3,53	**3,05**	4,09	3,69	20,00	3,18

Tabelle 6.11: Trainingsergebnisse mit Gewichtung der Kostenfunktion nach [KGC18] mit Lernrate 0,0001. Zum Vergleich werden verschiedene Initialisierungen für die Konfidenzgewichte für verantwortliche Prädiktionen ω_1 und nicht verantwortliche Prädiktionen ω_0 sowie die Gewichte für Geometrie ω_g und Konfidenz ω_c ausprobiert. Mit ω^* werden die Gewichte bezeichnet, zu denen die Trainings konvergieren. Alle Gewichte wurden im Training auf das Intervall $[0{,}05; 20]$ begrenzt.

lisierung bereits im richtigen Verhältnis erfolgen muss, um ein erfolgreiches Training durchführen zu können. Die gelernten Gewichte können jedoch als Validierung und Feinabstimmung für die Gewichtung zwischen Geometrie und Konfidenz verwendet werden. Für zukünftige Experimente wäre es interessant, die Konfidenzgewichte konstant zu lassen und lediglich die Gewichtung der Geometriekosten zu lernen.

6.5 Qualitative Ergebnisse

Um einen qualitativen Eindruck der Ergebnisse zu bekommen, präsentiert dieser Abschnitt die Prädiktion auf dem TuSimple und dem KAI-Datensatz. Auf dem Argoverse-Datensatz werden anschließend vier Szenen diskutiert und Herausforderungen identifiziert. Die verwendeten Konfigurationen befinden sich im Anhang Abschnitt A.2.

In Abbildung 6.10 wird die Anwendung auf Markierungen in Luftbildern (KAI-Datensatz) aus der eigenen Veröffentlichung [MSPS21] gezeigt. Die Liniensegmente beschreiben dabei zwei Klassen von Markierungen: durchgezogen und gestrichelt. Über die Richtung des Liniensegments kann die Fahrtrichtung der Fahrstreifen geschätzt werden. Die Richtung wird dabei über die Sortierung als Start- und Endpunkt kontinuierlich mitgeschätzt. Nach [MSPS21]

erreicht die Schätzung ein punktbasiertes F1-Maß von 89 % und hat lediglich Schwierigkeiten mit ungewöhnlichen Objekten wie im rechten Bild.

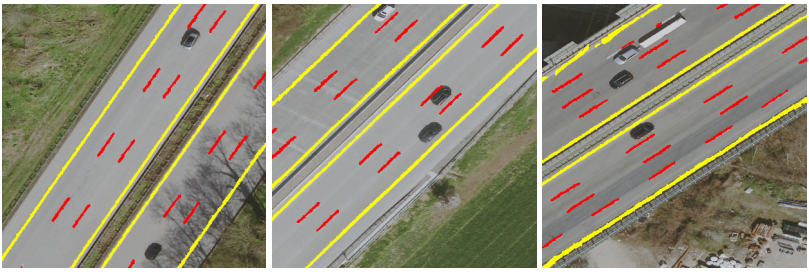


Abbildung 6.10: Prädiktionen auf dem KAI-Testdatensatz veröffentlicht in [MSPS21]. Die Klassifikation der Liniensegmente wird hier verwendet, um hetereogene Objekte zu unterscheiden. Alle Varianten schätzen ebenfalls eine Richtung der Markierung und beschreiben die Fahrtrichtung des Fahrstreifens damit korrekt. Bildquelle der Luftbilder: ©Stadt Karlsruhe | Liegenschaftsamt

Abbildung 6.11 zeigt Ergebnisse verschiedener Trainings auf dem TuSimple-Datensatz Hier werden die Fahrstreifen der Autobahnszenarien ebenfalls mit einer Fahrtrichtung geschätzt. Wie in [MSPS21] gezeigt erreichen die Ergebnisse den Stand verwandter Arbeiten, ohne jedoch die für Autobahnen spezialisierten Modellannahmen zu benutzen.

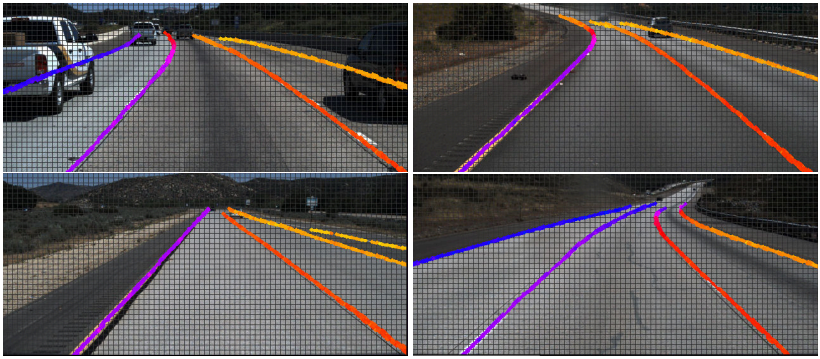


Abbildung 6.11: Prädiktionen auf dem TuSimple-Datensatz mit 8×8 Pixel Zellen. Die Liniensegmente sind farblich nach ihrer Orientierung im Bild kodiert, sodass die geschätzte Fahrtrichtung visualisiert wird.

Abbildung 6.12 zeigt Prädiktionen auf verschiedenen Szenen des Argoverse-Datensatzes mit festen Ankern bzw. der dynamischen Zuordnung im Training. Die ausgewählten Szenen zeigen die Eigenschaften der Detektion auf dem Argoverse-Datensatz. Die Liniensegmente sind hier farblich nach der Orientierung kodiert. Dadurch lässt sich unterscheiden, welcher Fahrstreifen in Fahrtrichtung (lila, rot, orange) und welcher in die Gegenrichtung (gelb) befahren wird.

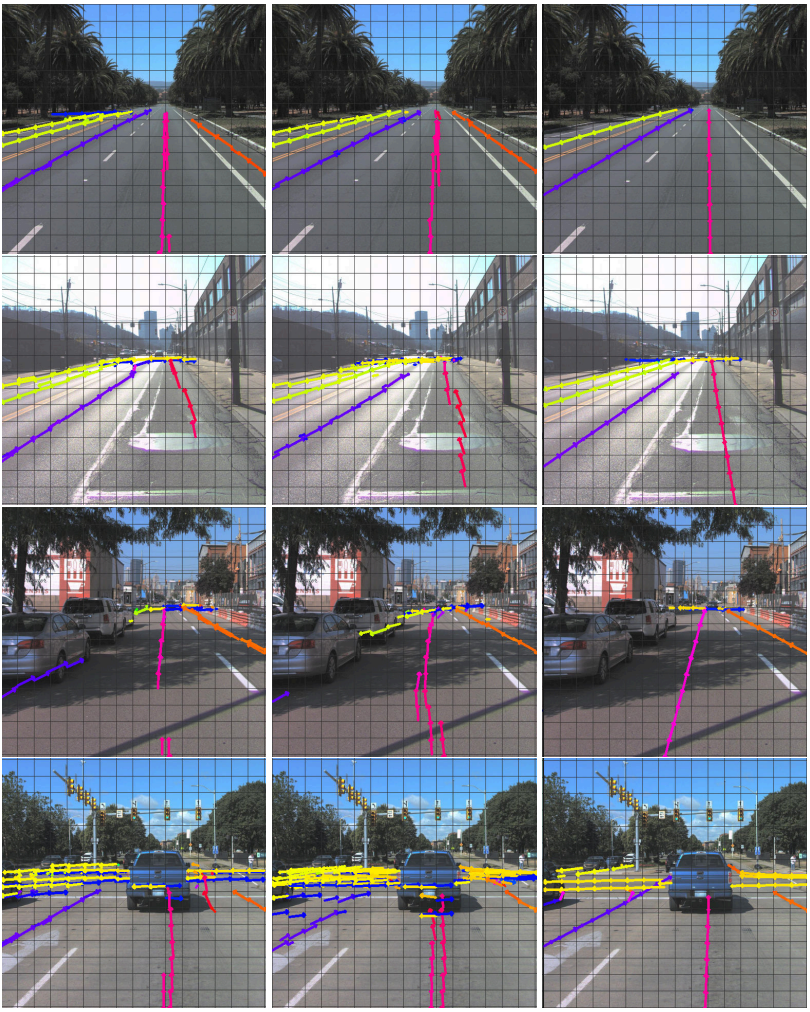
In der obersten Zeile ist eine vierspurige Straße mit einem Fahrradstreifen abgebildet. Dieser wird von dem hier präsentierten Ansatz einwandfrei detektiert, ist jedoch in der GT nicht vorhanden. Es gibt genauso Szenen, in denen der Fahrradstreifen in der Karte vorhanden und somit eine valide GT ist.

Die zweite Szene beinhaltet Bilder, die bei Gegenlicht aufgenommen wurden, sodass die Straße stark reflektiert. Trotz dieser Herausforderung kann noch eine weitestgehend korrekte Schätzung ermittelt werden.

Auch in der dritten Zeile stellen Schatten und parkende Fahrzeuge eine Herausforderung für die Prädiktion dar. Zudem handelt es sich hier um eine Straße mit nur einer Fahrtrichtung, sodass sich das Ego-Fahrzeug auf dem Fahrstreifen links außen befindet. Dies kommt nur selten im Trainingsdatensatz vor.

In der untersten Zeile wird eine Kreuzung gezeigt, die erfolgreich erkannt wurde. Hier tauchen jedoch deutlich mehr Fehler auf als in einfacheren Szenen. Insbesondere kann eine Detektion rein auf dem Kamerabild zwar bereits sehr viele Informationen ableiten, jedoch bedarf es für präzise Detektionen eines größeren Sichtfelds oder einer zeitlichen Sequenz von Bildern.

Die quantitativ bereits gezeigte Verbesserung der Ergebnisse durch die dynamische Zuordnung zeigt sich auch in den Beispielbildern. Die Linienzüge sind konsistenter und weisen weniger Fehlschätzungen auf.



(a) Anker (b) Dynamische Zuordnung (c) Ground Truth

Abbildung 6.12: Prädiktion auf dem Argoverse-Datensatz mit vordefinierten Ankern in Spalte a) und dynamischer Zuordnung in Spalte b). Die Farbe der Liniensegmente kodiert deren Ausrichtung im Bild und gibt somit einen Eindruck über die Fahrtrichtung.

6.6 Qualitative Ergebnisse der Nachverarbeitung

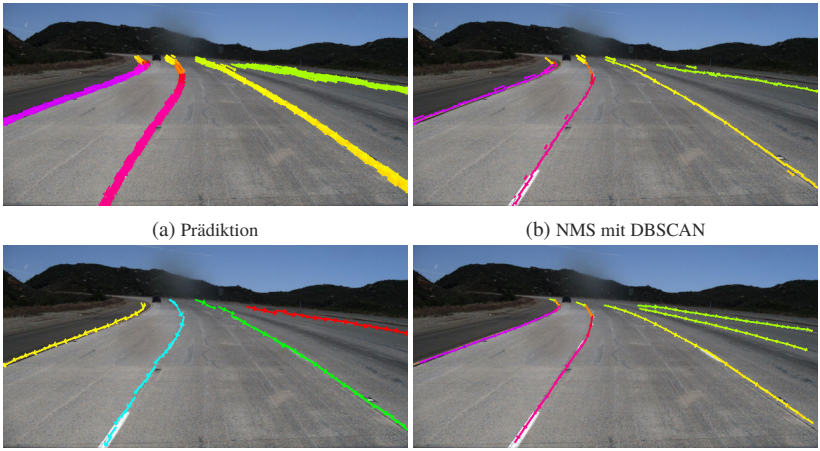
Die in Kapitel 5 vorgestellten Möglichkeiten für eine weitere Verarbeitung der Daten, sollen in diesem Abschnitt beleuchtet werden. Abbildung 6.13 und Abbildung 6.14 zeigen die Ergebnisse der Non-Maximum-Suppression, umgesetzt als DBSCAN-Clustering, sowie die Verbindungsschätzung durch Baumsuche, die in [MSPS21] gezeigt wurden.

(a) Prädiktion (b) NMS mit DBSCAN

Abbildung 6.13: Prädiktion und Non-Maximum-Suppression (NMS) mit DBSCAN auf dem KAI-Testdatensatz veröffentlicht in [MSPS21]. Die Farben visualisieren die Klasse der Markierungen: durchgezogen (gelb) und gestrichelt (rot). Bildquelle der Luftbilder: ©Stadt Karlsruhe | Liegenschaftsamt

Abbildung 6.15 zeigt die Ergebnisse der Nachverarbeitung auf dem TuSimple-Datensatz für einige Ansätze, die in [Sch22] untersucht wurden. Weitere Beispiele dazu finden sich in Abschnitt A.5.

DBSCAN, MuSSP und das GNN mit NMS-Kosten konnten für eine Non-Maximum-Suppression überzeugen. Die Instanzen der Fahrstreifen und eine korrekte Verbindung der Liniensegmente schätzt nur MuSSP korrekt. Außerdem kann die Kombination aus einem GNN und der Kuhn-Munkres-Zuordnung (GNN+KMZ) in einem zweischrittigen Verfahren aus NMS und Verbindungsschätzung ein valides Ergebnis liefern. Weder die harte Zuwei-



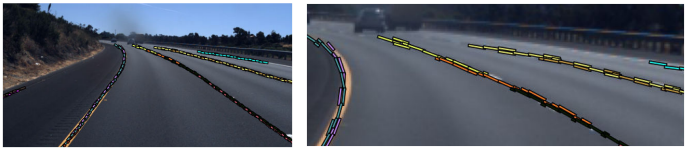
(a) Prädiktion (b) NMS mit DBSCAN

(c) Verbindungsschätzung als Baumsuche (d) Ground Truth

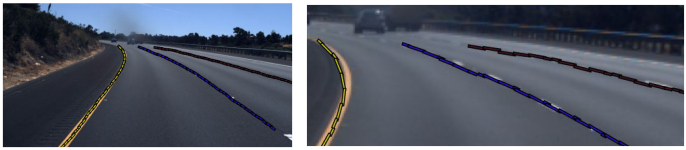
Abbildung 6.14: Prädiktion und Nachverarbeitung auf dem TuSimple-Datensatz veröffentlicht in [MSPS21]. a) zeigt die Prädiktion mit der Orientierung des Liniensegments farblich hervorgehoben. b) beinhaltet die Ergebnisse nach der NMS mit dem DBSCAN-Clustering. Die Farben visualisieren auch hier die Orientierung. In c) sind die Instanzen, die aus der Baumsuche resultieren, farblich getrennt visualisiert.

sung der NMS-Kosten noch die Verbindungskosten sind für die vorliegenden Anwendungen geeignet.

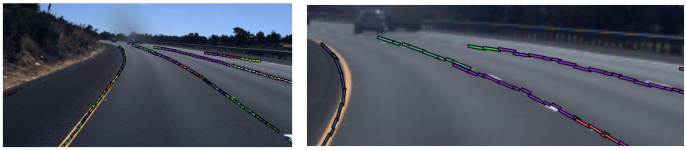
Auf innerstädtische Topologien lassen sich sowohl MuSSP als auch GNN+KMZ übertragen, wenn an Aufspaltungen und Zusammenführung von Fahrstreifen eine neue Instanz begonnen wird und diese Stellen bereits in der YOLinO-Architektur klassifiziert werden.



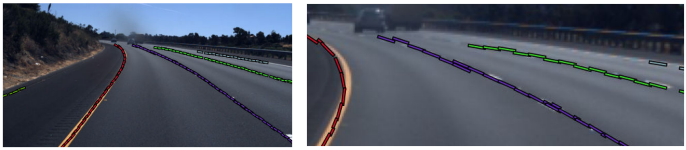
(a) Kuhn-Munkres-Zuordnung



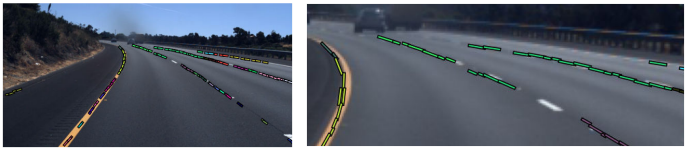
(b) MuSSP



(c) NMS-GNN



(d) NMS-GNN + Kuhn-Munkres-Zuordnung



(e) NMS-GNN + Verbindungskosten

Abbildung 6.15: Ergebnisbilder (links) und Ausschnitte daraus (rechts) evaluiert auf dem TuSimple-Datensatz für verschiedene Nachverarbeitungsschritte. Die besten Ergebnisse erzielt MuSSP. Die Farben der Liniensegmente identifizieren je eine Instanz. Segmente der gleichen Farbe werden als verbunden geschätzt. Bildquelle: [Sch22]

7 Schlussfolgerungen und Ausblick

In dieser Arbeit wurde ein Ansatz vorgestellt, der linienförmige Kartenmerkmale in Echtzeit detektieren kann. Sowohl explizit sichtbare Linien (Markierungen) als auch implizite Linien (Fahrstreifenmittellinien) können präzise geschätzt werden. Dazu setzt der Ansatz eine rein vorwärtsgerichtete Architektur ein und benötigt keine iterativen Verfahren, da die Linienzüge durch ein Zellgitter diskretisiert geschätzt werden. In den vorhergehenden Kapiteln und Veröffentlichungen im Rahmen dieser Arbeit [MSPS21] konnte gezeigt werden, dass der Ansatz echtzeitfähig ist, Linien in beliebigen Szenen erkennen kann und sogar für komplexe Kreuzungsgeometrien im urbanen Raum geeignet ist. Die vorgeschlagene Diskretisierung ermöglicht es, beliebige gerichtete, kreuzende und zusammenlaufende Liniengeometrien ohne Modellvorgaben in neuronalen Netzen zu repräsentieren. Dies war in einer vorwärtsgerichteten Netzarchitektur bisher nicht möglich.

Die vorgestellte Repräsentation gibt für jedes Liniensegment die Fahrtrichtung kontinuierlich an und kann Instanzen verschiedener Linienzüge eindeutig unterscheiden, ohne dazu auf eine Diskretisierung der Richtung oder eine Klassifikation zurückgreifen zu müssen. Die Anwendung wurde auf Fahrstreifenrändern, -mittellinien sowie den Markierungen sowohl auf Autobahnen als auch im innerstädtischen Bereich gezeigt (siehe Abschnitt 6.5). Der Stand der Forschung scheitert hier überwiegend bereits an der Repräsentation von Kreuzungen (siehe Kapitel 3).

Mithilfe der Klassifikation jedes Liniensegments können verschiedenste Kartenmerkmale gleichzeitig identifiziert werden: Markierungen können nach ihrer Semantik (gestrichelt und durchgezogen) klassifiziert oder Mittellinien von Fahrstreifenrändern unterschieden und gemeinsam prädiziert werden.

In Einzelstudien (siehe Kapitel 6) wurden die Aspekte der Konzeption, die für eine erfolgreiche Prädiktion relevant sind, gezeigt. Aus den Ergebnissen der Studien lassen sich folgende Empfehlungen ableiten:

1. Die Anzahl der Prädiktoren muss je nach Anwendung abgewägt werden. Je mehr Prädiktoren es gibt, desto besser wird die Geometrie der Liniensegmente geschätzt, jedoch sinkt die Präzision der Konfidenzschätzung.
2. Sowohl die Mittelpunkt-Richtung-Koordinaten (MR) als auch die kartesischen Start- und Endpunkte (SE) sind geeignet für eine Schätzung im Zellgitter. MR liefert dabei die besseren Ergebnisse.
3. Im Vergleich zwischen Ankerdefinitionen liefern Anker, die datengetrieben auf dem vollen MR-Clusterraum mittels Clustering bestimmt werden, die besten Trainingsergebnisse.
4. Eine dynamische Zuordnung im Training erreicht ein besseres F1-Maß als ein Training mit Ankern, weist jedoch deutlich längere Trainingszeiten und größere geometrische Abweichungen auf.
5. Die Skalierung des Zellgitters geht mit einem höheren Rauschen einher, ist jedoch in Verbindung mit einer NMS zu empfehlen. Die verbesserten Ergebnisse müssen für eine Anwendung gegen die höhere Inferrenzzeit abgewogen werden.

Ausblick

Der Ansatz ist so gestaltet, dass die Architektur einfach auf mehr oder andere Sensordaten erweitert werden kann. Alternativ kann auch der Encoder so erweitert werden, dass die Sensordaten in eine Draufsicht transformiert werden [LWWZ22b] und der Decoder entsprechend die Fahrstreifen in der Draufsicht schätzt. Die Repräsentation lässt sich ohne Änderung in andere Perspektiven, wie die Draufsicht, übertragen. Bei Verwendung von Ankern müssen diese an den Datensatz angepasst werden, weisen dann jedoch weniger Mehrfachzuweisungen auf, als es für perspektivische Bilder der Fall ist (siehe Unterabschnitt 6.4.3).

Die Anker werden in dieser Arbeit durch ein k-Means-Clustering bestimmt (siehe Unterabschnitt 4.5.2). Für zukünftige Arbeiten ist es interessant zu untersuchen, ob eine aufwändigere Optimierung Vorteile gegenüber dem k-Means hat. Damit könnte direkt bei Bestimmung der Anker die Anzahl der Mehrfachzuweisungen minimiert werden, sodass alle Liniensegmente im Trainingsdatensatz vollständig abgebildet werden könnten.

Alternativ können die Anker nicht für den gesamten Datensatz, sondern für jede Zelle im Bild individuell bestimmt werden, da sich die Verteilung der Liniensegmente zwischen den Zellen stark unterscheidet. Abbildung A.4 zeigt die Verteilung der Liniensegmente auf dem TuSimple-Datensatz.

Linienzüge und Kreuzungen

In Abschnitt 6.6 konnte gezeigt werden, dass die Liniensegmente mit verschiedenen Methoden zu Linienzügen kombiniert werden können. Für eine Non-Maximum-Suppression (NMS) eignen sich insbesondere ein graphenbasiertes neuronales Netz (GNN), ein Clustering mit DBSCAN sowie die Pfadsuche im Graphen mit MuSSP (siehe Abschnitt 5.2). Um aus den Liniensegmenten einen Linienzug zu schätzen bietet sich ebenfalls MuSSP sowie der Kuhn-Munkres-Algorithmus mit einer entsprechenden Kostenformulierung an (siehe Abschnitt 5.3). Da der Kuhn-Munkres-Algorithmus mittlerweile auch als differenzierbares neuronales Netz präsentiert wurde [APV21], sind alle Module des vorgestellten Systems lernbar. Mit der Kombination aus YOLinO, einem GNN und einer anschließenden gelernten Zuordnung (Kuhn-Munkres-Algorithmus) lässt sich ein einziges Ende-zu-Ende-Netz konstruieren, das die vollständigen, verbundenen Linienzüge im Bild direkt schätzen kann.

Alternativ kann Modellwissen einbezogen werden, um die Linienzüge in einer Szene zu schätzen. In den verwandten Arbeiten (Kapitel 3) wurde bereits der Mangel an geeigneten Liniendetektoren für eine modellgetriebene Kreuzungsschätzung hervorgehoben. Bisher verarbeiten diese Schätzer punktbasierte Messungen wie eine semantische Segmentierung oder Bilddaten sowie linienförmige Merkmale wie bspw. Trajektorien [MWLS19]. Ein Detektor, der Markierungen oder Mittellinien als Linien in Echtzeit präsentiert, existiert bisher nicht. Mit dem in dieser Arbeit vorgestellten Ansatz wurde diese Lücke geschlossen. Er liefert nicht nur Liniendetektionen, sondern gleichzeitig eine Hypothesenschätzung zur Verteilung der Liniensegmente im Bild. Mit modellbasierten Schätzungen, wie sie von der Autorin in [MWLS19, MWL20] vorgestellt wurden, kann diese Repräsentation verwendet werden, um eine vollständige Kreuzungsrepräsentation probabilistisch abzuschätzen. Eine zusätzliche NMS oder Verbindungsschätzung ist dafür gar nicht notwendig.

Zukünftige Anwendungen

Die Schätzung der Liniensegmente aus dieser Arbeit wurde für die Anwendung auf Fahrstreifen entworfen, ist jedoch generisch einsetzbar. Dadurch kann nicht nur die Anwendung im kartenlosen Fahren ermöglicht werden, sondern unterschiedliche andere Disziplinen.

Die in der Einleitung diskutierte Kartenwahrnehmung konnte mit dieser Arbeit um einen entscheidenden Aspekt erweitert werden. Es kann nun in Echtzeit die genaue Geometrie der Fahrstreifen geschätzt werden. Durch die in dieser Arbeit präsentierte Repräsentation lassen sich dann mit einem modellbasierten Ansatz Verkehrsregeln oder Nachbarschaftsbeziehungen ableiten. Ebenso kann die Schätzung mit vorhandenen Karten fusioniert werden.

Auch aktuelle Systeme, die auf eine hochgenaue Karte aufbauen, profitieren von dieser Arbeit. Eine semantische Lokalisierung in einer Karte benötigt einen Detektor, der in der Umgebung die Elemente der Karte erkennt, sodass die Detektionen mit dieser assoziiert werden können. Dies wird bisher über flächige Schätzungen wie eine semantische Segmentierung umgesetzt. In [MPHS22] konnte gezeigt werden, dass mit linienförmigen Merkmalen eine schnelle und hochgenaue Lokalisierung umgesetzt werden kann. Diese Arbeit liefert dazu den echtzeitfähigen Detektor.

Alternativ zu dem eingangs vorgestellten modularen Systemaufbau werden Ende-zu-Ende-Ansätze in der Forschung diskutiert [HSG+20], die aus Sensordaten direkt die Steuerbefehle für die Aktorik berechnen. Hier wird weder eine Karte benötigt noch eine Szenenrepräsentation berechnet. Diese Ansätze haben den Vorteil, dass die Informationen nie auf eine interpretierbare Zwischenrepräsentation verdichtet und dadurch reduziert werden müssen. Der klare Nachteil ist jedoch die mangelnde Nachvollziehbarkeit der Entscheidungen. Die Ergebnisse aus [HSG+20] zeigen, dass die Schätzung der Steuerbefehle verbessert wird, wenn neben den Steuerbefehlen interpretierbare Repräsentationen der Szene geschätzt werden. Die Autoren untersuchen dies mit einer semantischen Segmentierung. Da in modularen Systemen bisher eine Kartendarstellung elementar ist, eignet sich die in dieser Arbeit präsentierte Repräsentation ideal als Alternative für die Repräsentation in diesen Ende-zu-Ende-Ansätzen. Zudem ist die Berechnung der Linienrepräsentation effizienter als eine semantische Segmentierung und benötigt somit weniger Systemressourcen.

Somit ermöglicht der in dieser Arbeit vorgestellte Ansatz nicht nur aktuell relevante Anwendungen der Kartenwahrnehmung, Kartierung und Lokalisierung. Auch in zukünftigen, möglicherweise vollständig gelernten Systemen, behält die diskretisierte Darstellung von Kartenelementen ihre Relevanz.

A Anhang

A.1 Architektur

Name	Merkmalskarte	Filter	Kernel
Eingangsbild	3x640x640		
Conv		32	3x3
Max-Pool	32x320x320		2x2
Conv		64	3x3
Max-Pool	64x160x160		2x2
Conv		128	3x3
Conv		64	1x1
Conv		128	3x3
Max-Pool	128x80x80		2x2
Conv		256	3x3
Conv		128	1x1
Conv		256	3x3
Max-Pool	256x40x40		2x2
Conv		512	3x3
Conv		256	1x1
Conv		512	3x3
Conv		256	1x1
Conv		512	3x3
Max-Pool	512x20x20		2x2
Conv		1024	3x3
Conv		512	1x1
Conv		1024	3x3
Conv		512	1x1
Conv	1024x20x20	1024	3x3

Tabelle A.1: Darknet19-Encoder

Name	Merkmalskarte	Filter	Kernel
Embedding	1024x20x20		
Conv	$\|P\| \cdot \|V\|$x20x20	$\|P\| \cdot \|V\|$	1x1

Tabelle A.2: YOLinO-Decoder mit Skalierungsfaktor 32

Name	Merkmalskarte	Filter	Kernel	Stride
Embedding	1024x20x20			
Transp-Conv (up)	1024x40x40	1024	3x3	2
Zusammenführung	1024x40x40 + 512x40x40			
Conv		1536	3x3	1
Conv		1024	1x1	1
Conv	1024x40x40	1024	3x3	1
Conv	$\|P\| \cdot \|V\|$x40x40	$\|P\| \cdot \|V\|$	1x1	

Tabelle A.3: YOLinO-Decoder mit Skalierungsfaktor 16

Name	Merkmalskarte	Filter	Kernel	Stride
Embedding	1024x20x20			
Transp-Conv (up)	1024x40x40	1024	3x3	2
– Zusammenführung	1024x40x40 + 512x40x40			
Conv		1536	3x3	1
Conv		1024	1x1	1
Conv	1024x40x40	1024	3x3	1
Transp-Conv (up)	1024x80x80	1024	3x3	2
– Zusammenführung	1024x80x80 + 256x80x80			
Conv		1280	3x3	1
Conv		512	1x1	1
Conv	512x80x80	512	3x3	1
Conv	$\|P\| \cdot \|V\|$x80x80	$\|P\| \cdot \|V\|$	1x1	

Tabelle A.4: YOLinO-Decoder mit Skalierungsfaktor 8

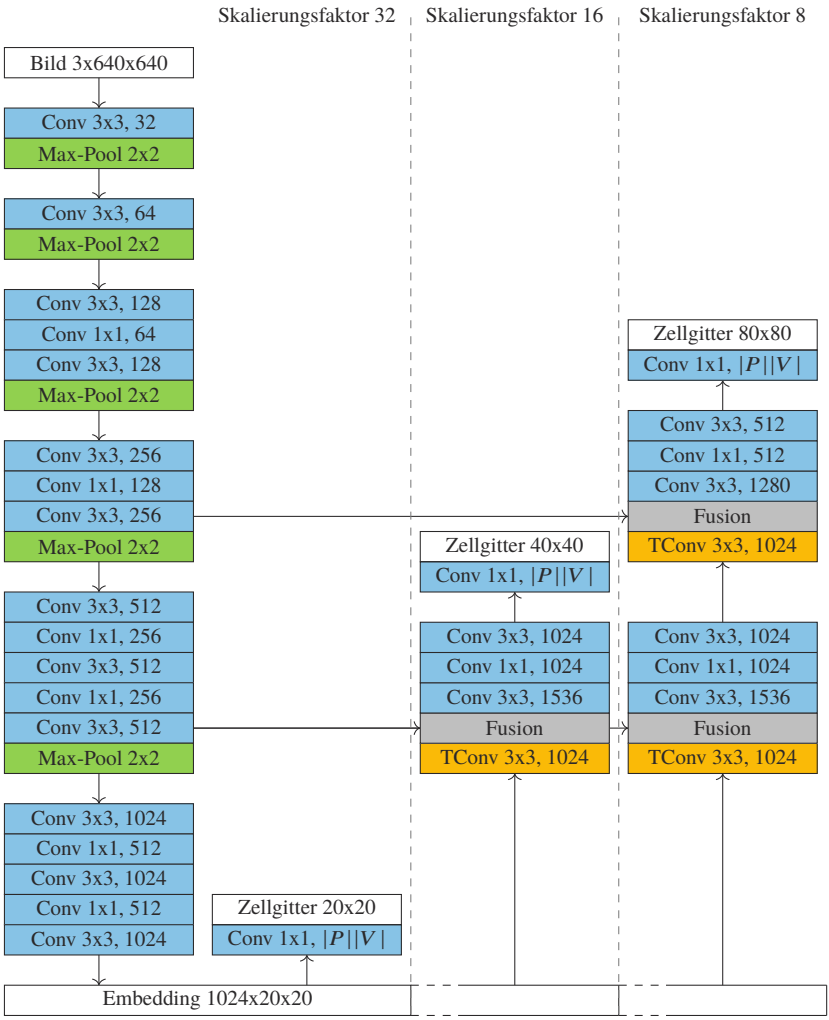


Abbildung A.1: Architektur mit drei Skalierungsvarianten

A.2 Konfigurationen der Experimente

Parameter	Wert
Aktivierung	Linear, Sigmoid
Anker-Offset	✓
Ankervariablen	SE
Ankerverteilung	k-Means
Anzahl Prädiktoren	8
Augmentierung	Zuschnitt, Rotation, Jitter, Auslassen
Augmentierung: Rotationsbereich	0,01
Augmentierung: Zuschnittsbereich	0,17
Dilation	1
Konfidenz	0.5
Kosten	Geometrie, Konfidenz
Kostenfunktion	Euklidsche Distanz, SSE
Kostenfunktion: Gewichte Konfidenz	5; 1
Kostenfunktion: Gewichte je Variable	10; 1
Kostenfunktion: Gewichtung	normiert
Lernrate	0,0001
Linienrepräsentation	SE
Optimierer	Adam
Skale	32

Tabelle A.5: Basisexperiment auf dem Argoverse-Datensatz

Parameter	Wert
Aktivierung	Linear,Sigmoid
Anker-Offset	✓
Ankervariablen	M,R
Ankerverteilung	dynamische Zuordnung
Anzahl Prädiktoren	8
Augmentierung	Zuschnitt,Rotation,Jitter,Auslassen
Augmentierung: Rotationsbereich	0,01
Augmentierung: Zuschnittsbereich	0,17
Dilation	1
Konfidenz	0,5
Kosten	Geometrie,Konfidenz
Kostenfunktion	Mittelwert der L2-Norm,MSE
Kostenfunktion: Gewichte Konfidenz	1;8
Kostenfunktion: Gewichte je Variable	1;1
Kostenfunktion: Gewichtung	Konstante Gewichte
Lernrate	0,0001
Linienrepräsentation	MR
Optimierer	Adam
Skale	32

Tabelle A.6: Konfiguration für Argoverse Prädiktion für Tabelle 6.7

Parameter	Wert
Aktivierung	Linear,Sigmoid
Anker-Offset	✓
Ankervariablen	M,R
Ankerverteilung	k-Means
Anzahl Prädiktoren	8
Augmentierung	Zuschnitt,Rotation,Jitter,Auslassen
Augmentierung: Rotationsbereich	0,01
Augmentierung: Zuschnittsbereich	0,17
Dilation	1
Konfidenz	0,5
Kosten	Geometrie,Konfidenz
Kostenfunktion	Mittelwert der L2-Norm,MSE
Kostenfunktion: Gewichte Konfidenz	1;8
Kostenfunktion: Gewichte je Variable	1;1
Kostenfunktion: Gewichtung	Gelernte Gewichte
Linienrepräsentation	MR
Optimierer	Adam
Skale	32

Tabelle A.7: Konfiguration für Argoverse Prädiktion für Tabelle 6.10 und Tabelle 6.11

Parameter	Wert
Aktivierung	Sigmoid,Sigmoid
Anker-Offset	×
Ankervariablen	SE
Ankerverteilung	k-Means
Anzahl Prädiktoren	8
Augmentierung	Zuschnitt,Rotation,Jitter,Auslassen
Augmentierung: Rotationsbereich	0,01
Augmentierung: Zuschnittsbereich	0,25
Dilation	8
Konfidenz	0,5
Kosten	Geometrie,Konfidenz
Kostenfunktion	norm,SSE
Kostenfunktion: Gewichte Konfidenz	5;1
Kostenfunktion: Gewichte je Variable	4;1
Kostenfunktion: Gewichtung	fixed
Lernrate	0,0001
Linienrepräsentation	SE
Optimierer	Adam
Skale	8

Tabelle A.8: Konfiguration für TuSimple Prädiktion Abbildung 6.11 und Abbildung 6.15

Parameter	Wert
Aktivierung	Linear,Sigmoid
Anker-Offset	✓
Ankervariablen	M,R
Ankerverteilung	k-Means
Anzahl Prädiktoren	8
Augmentierung	Zuschnitt,Rotation,Jitter,Auslassen
Augmentierung: Rotationsbereich	0,01
Augmentierung: Zuschnittsbereich	0,17
Dilation	1
Konfidenz	0,5
Kosten	Geometrie,Konfidenz
Kostenfunktion	normmean,msemean
Kostenfunktion: Gewichte Konfidenz	1;8
Kostenfunktion: Gewichte je Variable	1;1
Kostenfunktion: Gewichtung	Konstante Gewichte
Lernrate	0,0001
Linienrepräsentation	MR
Optimierer	Adam
Skale	32

Tabelle A.9: Konfiguration für Argoverse Prädiktion Abbildung 6.12a

Parameter	Wert
Aktivierung	Linear,Sigmoid
Anker-Offset	✓
Ankervariablen	M,R
Ankerverteilung	dynamische Zuordnung
Anzahl Prädiktoren	8
Augmentierung	Zuschnitt,Rotation,Jitter,Auslassen
Augmentierung: Rotationsbereich	0,01
Augmentierung: Zuschnittsbereich	0,17
Dilation	1
Konfidenz	0,5
Kosten	Geometrie,Konfidenz
Kostenfunktion	Mittelwert der L2-Norm,MSE
Kostenfunktion: Gewichte Konfidenz	1;10
Kostenfunktion: Gewichte je Variable	1;5
Kostenfunktion: Gewichtung	Konstante Gewichte
Lernrate	0,0001
Linienrepräsentation	MR
Optimierer	Adam
Skale	32

Tabelle A.10: Konfiguration für Argoverse Prädiktion Abbildung 6.12b

A.3 Datensätze

Cityscapes

Der ursprüngliche Cityscapes-Datensatz [COR+16] beinhaltet die semantische Segmentierung von Kamerabildern. Diese sind für die aktuelle Aufgabe alleine nicht nutzbar, da keinerlei Karteninformationen annotiert sind. Ein interner Datensatz wurde jedoch zu diesem Basisdatensatz erstellt, sodass zusätzlich Annotationen für Markierungen bestehen.

Die Fahrbahnmarkierungen sind als Polygone in Bildkoordinaten annotiert und mit einer Klasse versehen, die die Verkehrsregelsemantik widerspiegelt: *Berandung von Fußgänger-Überwegen, Schmalstrich, Breitstrich, Schmalstrich durchgezogen, Breitstrich durchgezogen, schraffierter Bereichen, Haltelinie an Kreuzungen/Ampeln/Einmündungen, Zebrastreifen, Zick-Zack-Markierung als Hinweis auf Parken/Halten-Verboten, Begrenzung eines Fahrradstreifens* und alle *Pfeilkombinationen*. Zur Verwendung in dieser Arbeit müsste für die Polygone die Mittellinie der Geometrie geschätzt werden, da eine Schätzung der Randpolygone nicht zielführend ist.

CULanes

CULanes [PSL+18] beinhaltet Frontsicht-Kamerabilder aus Peking, die Annotationen für Fahrstreifenränder beinhalten. Die Szenen beschränken sich somit auf innerstädtische Szenen einer Großstadt. Die Annotationen sind als Pixelkoordinaten von B-Splines gegeben und daher für die Verwendung in dieser Arbeit geeignet. Jeder Fahrstreifenrand ist zudem bei Verdeckungen gegeben.

Hier gilt zu beachten, dass pro Bild maximal vier Fahrstreifenränder und in Kreuzungen alle Fahrstreifen nur bis zur Haltelinie annotiert wurde. Dies kann zur Verschlechterung eines Trainings beitragen.

Nuscenes

Nuscenes [CBL+20] stellt geeignete Sensordaten sowie eine Karte bereit. Diese Karte hat jedoch nur 2D Informationen, sodass die Projektion ins Kamerabild massive Fehler aufweist. Hier wäre nur eine Draufsichtschätzung möglich. Zudem ist die Kalibrierung der Kameras nicht ideal und produziert dadurch Fehler in der Distanz.

Lyft

Lyft[1] ist ähnlich zu Nuscenes stellt die Karte jedoch nur als eine pixelweise Ansicht ihrer eigentlichen Vektorkarte vor. Dies macht die Verarbeitung der Karten unpräzise, da keinerlei geometrische Informationen vorliegen und die Auflösung zu gering ist.

BDD100k

Der Datensatz BDD100K [YCW+20] ist ein rein kamerabasierter Datensatz. Zu 120 000 000 Bildern werden verschiedene Annotationen zu Verfügung gestellt. Neben einer Szenenklassifikation über die Gegend (*city street*, *residential*, *highway*), die Wetterverhältnisse (*clear, cloudy, ...*) und die Tageszeit (*daytime, night*) gibt es Annotationen der Fahrbahnmarkierungen.

Die sichtbaren Fahrbahnmarkierungen werden in diesem Datensatz als pixelweise Annotation und zusätzlich als vektorielle Annotationen in Bildkoordinaten bereitgestellt. Die Klassen der Markierungen wurden mit *road curb, crosswalk, double white, double yellow, double other color, single white, single yellow, single other color* bezeichnet und zusätzlich in *dashed* und *continuous* unterschieden. Als dritte Ebene wird angegeben, ob sich die Markierung parallel (*parallel*) oder orthogonal zur Fahrtrichtung (*perpendicular*) befindet.

[1] Lyft Datensatz `https://level-5.global/level5/data/`

LLamas

Der Llamas-Datensatz [BS19] beinhaltet Autobahnszenarien und stellt Markierungsdetektionen bereit, die für gestrichelte Markierungen mit beigefügtem Code zu Fahrstreifenrändern interpoliert werden können. Jede Markierung ist einer Fahrstreifenrandinstanz zugeordnet jedoch nicht klassifiziert. Die Autoren stellen zudem einen Benchmark bereit für die Segmentierung sowie die Fahrstreifenschätzung[2].

Apollo Synthetic

Der synthetische 3D-Fahrstreifen-Datensatz von Apollo [GCZ+20] stellt Bilder von simulierten Szenarien zur Verfügung, in denen die Fahrstreifenränder und -mittellinien in 3D-Kamerakoordinaten annotiert sind. Zusätzlich wurde für jeden Rand eine Vielzahl von Klassen bestimmt, die das Aussehen (*dashed,solid,single,double,parking,yellow,white*) und die Semantik im Straßenverkehr (*left,right,fork,merge*) beschreiben.

VPGNet

Der VPGNet-Datensatz [LKY+17] stellt für urbane Szenen Markierungsannotationen mit 17 verschiedenen Klassen bereit. Da die Annotationen jedoch als pixelweise Klassifikation angegeben sind und zudem in Kreuzungen nicht vollumfassend sind wird dieser Datensatz nicht verwendet.

VIL100

Der VIL100 [ZZF+21] stellt Fahrstreifenränder für Autobahn-Szenen als Linienannotation in Pixelkoordinaten bereit. Zudem werden die Informationen auch als semantische Segmentierung mit einer angenommenen Breite von 30px bereitgestellt.

[2] `https://unsupervised-llamas.com/llamas/benchmark_splines`

ApolloScape

ApolloScape [MZZ+19] beinhaltet, neben vielen für diese Arbeit nicht interessanten Aufgaben, Stereobilder mit Markierungsannotationen als pixelweise Klassifikation sowie einen Benchmark zu dieser Segmentierung.

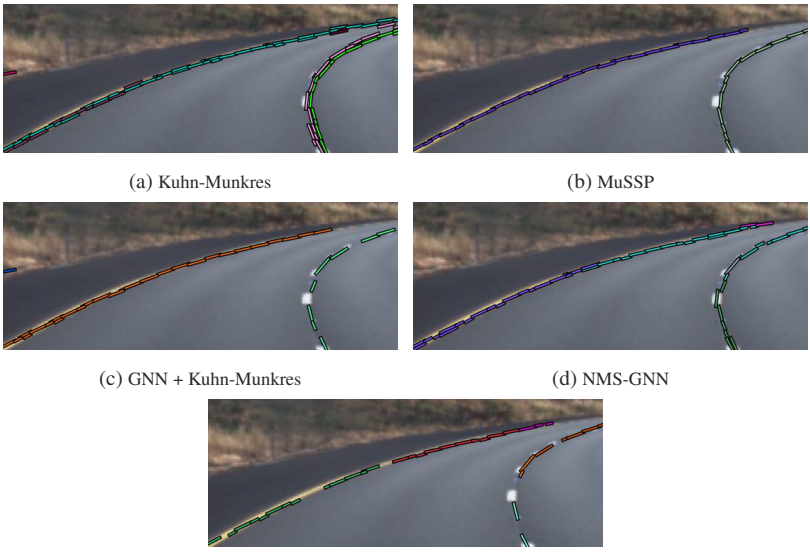
A.4 Parameter der Nachverarbeitung

Parameter	Wert
Distanzschwellwert ρ_d	2
Winkeldifferenz ρ_α	60°
Öffnungswinkel ρ_β	80°
Kosten der Quellenkante	100
Kosten der Senkenkante	100
Knotenkosten	-20
Kantenkosten	$100 \cdot \delta$

Tabelle A.11: Parameter für MuSSP abhängig von der Distanz δ der Knoten einer Kante [Sch22]

.

A.5 Beispielbilder der Nachverarbeitung auf TuSimple



(a) Kuhn-Munkres

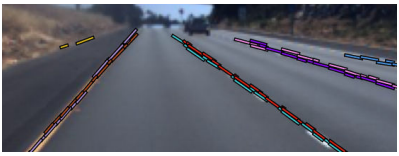
(b) MuSSP

(c) GNN + Kuhn-Munkres

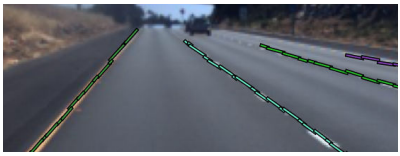
(d) NMS-GNN

(e) NMS-GNN + Verbindung

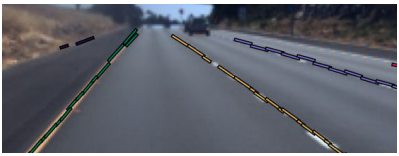
Abbildung A.2: Ergebnisse der Nachverarbeitung für ein Beispielbild aus dem TuSimple-Datensatz [Sch22]



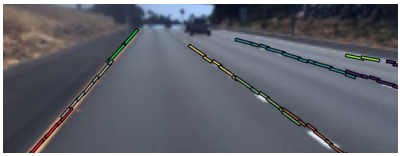
(a) Kuhn-Munkres



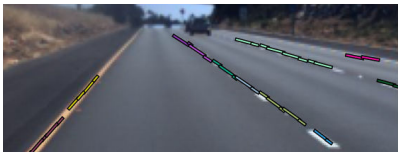
(b) MuSSP



(c) GNN + Kuhn-Munkres



(d) NMS-GNN



(e) NMS-GNN + Verbindung

Abbildung A.3: Ergebnisse der Nachverarbeitung für ein Beispielbild aus dem TuSimple-Datensatz [Sch22]

A.6 Verteilung der Liniensegmente je Zelle im Bild

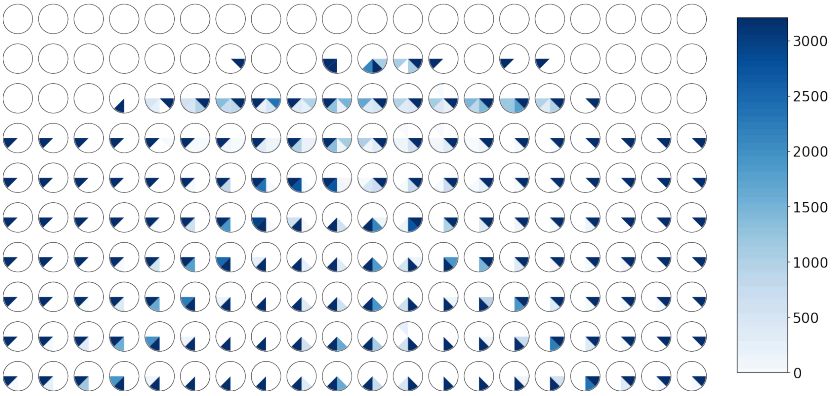


Abbildung A.4: Anzahl der Liniensegmente je Winkelbereich als zellbasierte Verteilung auf dem TuSimple-Trainingsdatensatz bei einer Bildgröße von 320×640 Pixeln und Zellen mit 32×32 Pixel.

Literaturverzeichnis

[AFKR19] Azimi, Seyed M. ; Fischer, Peter ; Körner, Marco ; Reinartz, Peter: Aerial LaneNet: Lane-Marking Semantic Segmentation in Aerial Imagery Using Wavelet-Enhanced Cost-Sensitive Symmetric Fully Convolutional Neural Networks. In: *IEEE Transactions on Geoscience and Remote Sensing* 57 (2019), Mai, Nr. 5, S. 2920–2938. – ISSN 1558–0644

[Aly08] Aly, Mohamed: Real Time Detection of Lane Markers in Urban Streets. In: *IEEE Intelligent Vehicles Symposium*. Eindhoven, Niederlande, Juni 2008. – ISSN 1931–0587, S. 7–12

[APV21] Adavanne, Sharath ; Politis, Archontis ; Virtanen, Tuomas: Differentiable Tracking-Based Training of Deep Learning Sound Source Localizers. In: *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*. New Paltz, NY, USA, Oktober 2021. – ISSN 1947–1629, S. 211–215

[BB19] Busch, Steffen ; Brenner, Claus: Discrete Reversible Jump Markov Chain Monte Carlo Trajectory Clustering. In: *IEEE Intelligent Transportation Systems Conference*. Auckland, Neuseeland, Oktober 2019. – ISBN 978–1–5386–7024–8, S. 1475–1481

[Bis06] Bishop, Christopher M.: *Pattern Recognition and Machine Learning*. 1. Auflage, 8. korrigierte Fassung (2019). New York, NY, USA : Springer Science+Business Media, LLC, 2006 (Information Science and Statistics). – ISBN 978–1–4939–3843–8

[BMH+18] Bai, M. ; Mattyus, G. ; Homayounfar, N. ; Wang, S. ; Lakshmikanth, S. K. ; Urtasun, R.: Deep Multi-Sensor Lane Detection. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. Madrid, Spanien, Oktober 2018, S. 3102–3109

[BMP17] Barnes, D. ; Maddern, W. ; Posner, I.: Find Your Own Way: Weakly-supervised Segmentation of Path Proposals for Urban

Autonomy. In: *IEEE International Conference on Robotics and Automation*. Singapur, Singapur, Mai 2017, S. 203–210

[BS19] Behrendt, Karsten ; Soussan, Ryan: Unsupervised Labeled Lane Markers Using Maps. In: *IEEE/CVF International Conference on Computer Vision Workshop*. Seoul, Südkorea, Oktober 2019. – ISSN 2473–9944, S. 832–839

[BSFC08] Brostow, Gabriel J. ; Shotton, Jamie ; Fauqueur, Julien ; Cipolla, Roberto: Segmentation and recognition using structure from motion point clouds. In: Forsyth, David (Hrsg.) ; Torr, Philip (Hrsg.) ; Zisserman, Andrew (Hrsg.) ; Springer (Veranst.): *Computer Vision – ECCV 2008*. Berlin, Heidelberg : Springer Berlin, Heidelberg, 2008. – ISBN 978–3–540–88682–2, S. 44–57

[Buc21] Buck, Jan: *Learning Non-maximum Suppression for Polyline Detection*. Karlsruhe, Deutschland, November 2021. – Masterarbeit am Institut für Mess- und Regelungstechnik, Karlsruher Institut für Technologie

[BWL20] Bochkovskiy, Alexey ; Wang, Chien-Yao ; Liao, Hong-Yuan M.: YOLOv4: Optimal Speed and Accuracy of Object Detection. In: *arXiv:2004.10934* (2020), April. – letzter Zugriff 25.06.2021 `https://arxiv.org/abs/2004.10934`

[CBL+20] Caesar, Holger ; Bankiti, Varun ; Lang, Alex H. ; Vora, Sourabh ; Liong, Venice E. ; Xu, Qiang ; Krishnan, Anush ; Pan, Yu ; Baldan, Giancarlo ; Beijbom, Oscar: nuScenes: A multimodal dataset for autonomous driving. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. virtuell, Jun 2020, S. 11621–11631

[COR+16] Cordts, Marius ; Omran, Mohamed ; Ramos, Sebastian ; Rehfeld, Timo ; Enzweiler, Markus ; Benenson, Rodrigo ; Franke, Uwe ; Roth, Stefan ; Schiele, Bernt: The Cityscapes Dataset for Semantic Urban Scene Understanding. In: *IEEE Conference on Computer Vision and Pattern Recognition*. Las Vegas, NV, USA, Juni 2016, S. 3213–3223

[CZT19] Chen, Zhe ; Zhang, Jing ; Tao, Dacheng: Progressive LiDAR Adaptation for Road Detection. In: *IEEE/CAA Journal of Auto-*

matica Sinica 6 (2019), Mai, Nr. 3, S. 693–702. – ISSN 2329–9274

[DBK+21] Dosovitskiy, Alexey ; Beyer, Lucas ; Kolesnikov, Alexander ; Weissenborn, Dirk ; Zhai, Xiaohua ; Unterthiner, Thomas ; Dehghani, Mostafa ; Minderer, Matthias ; Heigold, Georg ; Gelly, Sylvain ; Uszkoreit, Jakob ; Houlsby, Neil: An Image Is Worth 16x16 Words: Transformers for Image Recognition at Scale. In: *International Conference on Learning Representations*. virtuell, Mai 2021

[EKSX96] Ester, Martin ; Kriegel, Hans-Peter ; Sander, Jörg ; Xu, Xiaowei: A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In: *International Conference on Knowledge Discovery and Data Mining*. Portland, OR, USA, August 1996, S. 226–231

[FKG13] Fritsch, Jannik ; Kuehnl, Tobias ; Geiger, Andreas: A New Performance Measure and Evaluation Benchmark for Road Detection Algorithms. In: *International IEEE Conference on Intelligent Transportation Systems*. Den Haag, Niederlande, Oktober 2013, S. 1693–1700

[FWCL20] Fan, Rui ; Wang, Hengli ; Cai, Peide ; Liu, Ming: SNE-RoadSeg: Incorporating Surface Normal Information into Semantic Segmentation for Accurate Freespace Detection. In: Vedaldi, Andrea (Hrsg.) ; Bischof, Horst (Hrsg.) ; Brox, Thomas (Hrsg.) ; Frahm, Jan-Michael (Hrsg.): *Computer Vision – ECCV 2020*. Cham, Schweiz : Springer International Publishing, 2020. – ISBN 978–3–030–58577–8, S. 340–356

[GBC16] Goodfellow, Ian ; Bengio, Yoshua ; Courville, Aaron: *Deep Learning*. Erste Auflage. Cambridge, MA, USA : MIT Press, 2016. – ISBN 978–0–262–03561–3

[GCP+19] Garnett, Noa ; Cohen, Rafi ; Pe'er, Tomer ; Lahav, Roee ; Levi, Dan: 3D-LaneNet: End-to-End 3D Multiple Lane Detection. In: *IEEE International Conference on Computer Vision*. Seoul, Südkorea, Oktober 2019, S. 2921–2930

[GCZ+20] Guo, Yuliang ; Chen, Guang ; Zhao, Peitao ; Zhang, Weide ; Miao, Jinghao ; Wang, Jingao ; Choe, Tae E.: Gen-LaneNet: A Generalized and Scalable Approach for 3D Lane Detection.

In: Vedaldi, Andrea (Hrsg.) ; Bischof, Horst (Hrsg.) ; Brox, Thomas (Hrsg.) ; Frahm, Jan-Michael (Hrsg.): *Computer Vision – ECCV 2020*. Cham, Schweiz : Springer, 2020. – ISBN 978–3–030–58589–1, S. 666–681

[GLW+14] Geiger, A. ; Lauer, M. ; Wojek, C. ; Stiller, C. ; Urtasun, R.: 3D Traffic Scene Understanding From Movable Platforms. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36 (2014), Mai, Nr. 5, S. 1012–1025. – ISSN 0162–8828

[GNB+19] Ghafoorian, Mohsen ; Nugteren, Cedric ; Baka, Nóra ; Booij, Olaf ; Hofmann, Michael: EL-GAN: Embedding Loss Driven Generative Adversarial Networks for Lane Detection. In: Leal-Taixé, Laura (Hrsg.) ; Roth, Stefan (Hrsg.): *Computer Vision – ECCV 2018 Workshops*. Cham, Schweiz : Springer International Publishing, Januar 2019. – ISBN 978–3–030–11009–3, S. 256–272

[GSR+17] Gilmer, Justin ; Schoenholz, Samuel S. ; Riley, Patrick F. ; Vinyals, Oriol ; Dahl, George E.: Neural Message Passing for Quantum Chemistry. In: *International Conference on Machine Learning*. Sydney, Australien, August 2017, S. 1263–1272

[HML+19] Homayounfar, Namdar ; Ma, Wei-Chiu ; Liang, Justin ; Wu, Xinyu ; Fan, Jack ; Urtasun, Raquel: DAGMapper: Learning to Map by Discovering Lane Topology. In: *IEEE International Conference on Computer Vision*. Seoul, Südkorea, Oktober 2019, S. 2911–2920

[HMLL19] Hou, Yuenan ; Ma, Zheng ; Liu, Chunxiao ; Loy, Chen C.: Learning Lightweight Lane Detection CNNs by Self Attention Distillation. In: *IEEE/CVF International Conference on Computer Vision*. Seoul, Südkorea, Oktober 2019, S. 1013–1021

[HSG+20] Hawke, Jeffrey ; Shen, Richard ; Gurau, Corina ; Sharma, Siddharth ; Reda, Daniele ; Nikolov, Nikolay ; Mazur, Przemysław ; Micklethwaite, Sean ; Griffiths, Nicolas ; Shah, Amar ; Kndall, Alex: Urban Driving with Conditional Imitation Learning. In: *IEEE International Conference on Robotics and Automation*. virtuell, Mai 2020. – ISSN 2577–087X, S. 251–257

[KB15] Kingma, Diederik P. ; Ba, Jimmy: Adam: A Method for Stochastic Optimization. In: *International Conference on Learning Representations*. San Diego, CA, USA, Mai 2015

[KC20] Kandel, Ibrahem ; Castelli, Mauro: The Effect of Batch Size on the Generalizability of the Convolutional Neural Networks on a Histopathology Dataset. In: *ICT Express* 6 (2020), Dezember, Nr. 4, S. 312–315. – ISSN 2405–9595

[KGC18] Kendall, Alex ; Gal, Yarin ; Cipolla, Roberto: Multi-Task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics. In: *IEEE Conference on Computer Vision and Pattern Recognition*. Salt Lake City, UT, USA, Juni 2018, S. 7482–7491

[LCZT21] Liu, Lizhe ; Chen, Xiaohao ; Zhu, Siyu ; Tan, Ping: CondLaneNet: A Top-To-Down Lane Detection Framework Based on Conditional Convolution. In: *IEEE/CVF International Conference on Computer Vision*. virtuell, Oktober 2021, S. 3773–3782

[LDG+17] Lin, Tsung-Yi ; Dollár, Piotr ; Girshick, Ross ; He, Kaiming ; Hariharan, Bharath ; Belongie, Serge: Feature Pyramid Networks for Object Detection. In: *IEEE Conference on Computer Vision and Pattern Recognition*. Las Vegas, NV, USA, Juni 2017, S. 2117–2125

[LHM+19] Liang, Justin ; Homayounfar, Namdar ; Ma, Wei-Chiu ; Wang, Shenlong ; Urtasun, Raquel: Convolutional Recurrent Network for Road Boundary Extraction. In: *IEEE Conference on Computer Vision and Pattern Recognition*. Long Beach, CA, USA, Juni 2019, S. 9512–9521

[LKY+17] Lee, S. ; Kim, J. ; Yoon, J. S. ; Shin, S. ; Bailo, O. ; Kim, N. ; Lee, T. H. ; Hong, H. S. ; Han, S. H. ; Kweon, I. S.: VPGNet: Vanishing Point Guided Network for Lane and Road Marking Detection and Recognition. In: *IEEE International Conference on Computer Vision*. Venedig, Italien, Oktober 2017, S. 1965–1973

[LLHY20] Li, Xiang ; Li, Jun ; Hu, Xiaolin ; Yang, Jian: Line-CNN: End-to-End Traffic Line Detection With Line Proposal Unit. In: *IEEE Transactions on Intelligent Transportation Systems* 21 (2020), Januar, Nr. 1, S. 248–258. – ISSN 1558–0016

[LSD15] Long, Jonathan ; Shelhamer, Evan ; Darrell, Trevor: Fully Convolutional Networks for Semantic Segmentation. In: *IEEE Conference on Computer Vision and Pattern Recognition*. Boston, MA, USA, Juni 2015, S. 3431–3440

[LWWZ22a] Li, Qi ; Wang, Yue ; Wang, Yilun ; Zhao, Hang: HDMapNet: An Online HD Map Construction and Evaluation Framework. In: *International Conference on Robotics and Automation*. Philadelphia, PA, USA, Mai 2022, S. 4628–4634

[LWWZ22b] Liu, Yicheng ; Wang, Yue ; Wang, Yilun ; Zhao, Hang: VectorMapNet: End-to-end Vectorized HD Map Learning. In: *arXiv:2206.08920* (2022), Juni. – letzter Zugriff 17.11.2022 `https://arxiv.org/abs/2206.08920`

[MPHS22] Muñoz-Bañón, Miguel A. ; Pauls, Jan-Hendrik ; Hu, Haohao ; Stiller, Christoph: DA-LMR: A Robust Lane Marking Representation for Data Association. In: *International Conference on Robotics and Automation*. Philadelphia, PA, USA, Mai 2022, S. 2193–2199

[MSOS18] Meyer, Annika ; Salscheider, N. O. ; Orzechowski, Piotr F. ; Stiller, Christoph: Deep Semantic Lane Segmentation for Mapless Driving. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. Madrid, Spanien, Oktober 2018. – ISSN 2153–0866, S. 869–875

[MSPS21] Meyer, Annika ; Skudlik, Philipp ; Pauls, Jan-Hendrik ; Stiller, Christoph: YOLinO: Generic Single Shot Polyline Detection in Real Time. In: *IEEE/CVF International Conference on Computer Vision Workshops*. virtuell, Oktober 2021, S. 2916–2925

[Mun57] Munkres, James: Algorithms for the Assignment and Transportation Problems. In: *Journal of the society for industrial and applied mathematics* 5 (1957), Nr. 1, S. 32–38

[MWL20] Meyer, Annika ; Walter, Jonas ; Lauer, Martin: Fast Lane-Level Intersection Estimation using Markov Chain Monte Carlo Sampling and B-Spline Refinement. In: *IEEE Intelligent Vehicles Symposium*. virtuell, Oktober 2020, S. 71–76

[MWLS19] Meyer, Annika ; Walter, Jonas ; Lauer, Martin ; Stiller, Christoph: Anytime Lane-Level Intersection Estimation Based on Trajectories of Other Traffic Participants. In: *International*

Conference on Intelligent Transportation Systems. Auckland, Neuseeland, Oktober 2019, S. 3122–3129

[MZZ+19] Ma, Yuexin ; Zhu, Xinge ; Zhang, Sibo ; Yang, Ruigang ; Wang, Wenping ; Manocha, Dinesh: Trafficpredict: Trajectory prediction for heterogeneous traffic-agents. In: *AAAI Conference on Artificial Intelligence*. Honolulu, HI, USA, Januar 2019, S. 6120–6127

[OBB16] Oliveira, G. L. ; Burgard, W. ; Brox, T.: Efficient Deep Models for Monocular Road Segmentation. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. Daejeon, Südkorea, Oktober 2016, S. 4885–4891

[Oes19] Oesterle, Max: *Lane Topology and Geometry Estimation Based on Characteristic Road Features*. Karlsruhe, Deutschland, Juni 2019. – Bachelorarbeit am Institut für Mess- und Regelungstechnik, Karlsruher Institut für Technologie

[OWP+22] Ort, Teddy ; Walls, Jeffrey M. ; Parkison, Steven A. ; Gilitschenski, Igor ; Rus, Daniela: MapLite 2.0: Online HD Map Inference Using a Prior SD Map. In: *IEEE Robotics and Automation Letters* 7 (2022), Nr. 3, S. 8355–8362

[Phi19] Philion, Jonah: FastDraw: Addressing the Long Tail of Lane Detection by Adapting a Sequential Prediction Network. In: *IEEE Conference on Computer Vision and Pattern Recognition*. Long Beach, CA, USA, Juni 2019. – ISSN 1063–6919, S. 11574–11583

[Pog19] Poggenhans, Fabian: *Generierung hochdetaillierter Karten für das automatisierte Fahren*. – Doktorarbeit am Karlsruher Institut für Technologie, Karlsruhe, Deutschland (2019)

[PPJ+18] Poggenhans, Fabian ; Pauls, Jan-Hendrik ; Janosovits, Johannes ; Orf, Stefan ; Naumann, Maximilian ; Kuhnt, Florian ; Mayr, Matthias: Lanelet2: A High-Definition Map Framework for the Future of Automated Driving. In: *International Conference on Intelligent Transportation Systems*. Maui, HI, USA, November 2018. – ISSN 2153–0017, S. 1672–1679

[PSH+18] Pauls, Jan-Hendrik ; Strauss, Tobias ; Hasberg, Carsten ; Lauer, Martin ; Stiller, Christoph: Can We Trust Our Maps? An Evaluation of Road Changes and a Dataset for Map Validation.

In: *International Conference on Intelligent Transportation Systems*. Maui, HI, USA, November 2018. – ISSN 2153–0017, S. 2639–2644

[PSL+18] Pan, Xingang ; Shi, Jianping ; Luo, Ping ; Wang, Xiaogang ; Tang, Xiaoou: Spatial as Deep: Spatial CNN for Traffic Scene Understanding. In: *AAAI Conference on Artificial Intelligence*. New Orleans, LA, USA, Februar 2018, S. 7276–7283

[PWZ+22] Pan, Jingshan ; Wei, Zhiqiang ; Zhao, Yuhan ; Zhou, Yan ; Lin, Xunyu ; Zhang, Wei ; Tang, Chang: Enhanced FCN for Farmland Extraction from Remote Sensing Image. In: *Multimedia Tools and Applications* 81 (2022), November, Nr. 26, S. 38123–38150. – ISSN 1573–7721

[QWL20] Qin, Zequn ; Wang, Huanyu ; Li, Xi: Ultra Fast Structure-Aware Deep Lane Detection. In: Vedaldi, Andrea (Hrsg.) ; Bischof, Horst (Hrsg.) ; Brox, Thomas (Hrsg.) ; Frahm, Jan-Michael (Hrsg.): *Computer Vision – ECCV 2020*. Cham, Schweiz : Springer International Publishing, August 2020. – ISBN 978–3–030–58586–0, S. 276–291

[Ram72] Ramer, Urs: An Iterative Procedure for the Polygonal Approximation of Plane Curves. In: *Computer Graphics and Image Processing* 1 (1972), November, Nr. 3, S. 244–256. – ISSN 0146–664X

[RDGF16] Redmon, Joseph ; Divvala, Santosh ; Girshick, Ross ; Farhadi, Ali: You Only Look Once: Unified, Real-Time Object Detection. In: *IEEE Conference on Computer Vision and Pattern Recognition*. Las Vegas, NV, USA, Juni 2016, S. 779–788

[RF17] Redmon, Joseph ; Farhadi, Ali: YOLO9000: Better, Faster, Stronger. In: *IEEE Conference on Computer Vision and Pattern Recognition*. Honolulu, HI, USA, Juli 2017, S. 7263–7271

[RF18] Redmon, Joseph ; Farhadi, Ali: Yolov3: An Incremental Improvement. In: *arXiv:1804.02767* (2018), April. – letzter Zugriff 25.06.2021 `https://arxiv.org/abs/1804.02767`

[RFB15] Ronneberger, Olaf ; Fischer, Philipp ; Brox, Thomas: U-Net: Convolutional Networks for Biomedical Image Segmentation. In: *International Conference on Medical Image Computing and*

Computer-Assisted Intervention. München, Deutschland, Oktober 2015, S. 234–241

[RHW86] Rumelhart, David E. ; Hinton, Geoffrey E. ; Williams, Ronald J.: Learning Representations by Back-Propagating Errors. In: *nature* 323 (1986), Oktober, Nr. 6088, S. 533–536

[RKS+18] Roberts, Brook ; Kaltwang, Sebastian ; Samangooei, Sina ; Pender-Bare, Mark ; Tertikas, Konstantinos ; Redford, John: A Dataset for Lane Instance Segmentation in Urban Environments. In: Ferrari, Vittorio (Hrsg.) ; Hebert, Martial (Hrsg.) ; Sminchisescu, Cristian (Hrsg.) ; Weiss, Yair (Hrsg.): *Computer Vision – ECCV 2018*. Cham, Schweiz : Springer International Publishing, September 2018. – ISBN 978–3–030–01237–3, S. 543–559

[RZB17] Roeth, O. ; Zaum, D. ; Brenner, C.: Extracting Lane Geometry and Topology Information from Vehicle Fleet Trajectories in Complex Urban Scenarios Using a Reversible Jump MCMC Method. In: *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences* Bd. IV-1-W1, Copernicus GmbH, Mai 2017, S. 51–58

[SAN18] Suleymanov, T. ; Amayo, P. ; Newman, P.: Inferring Road Boundaries through and despite Traffic. In: *International Conference on Intelligent Transportation Systems*. Maui, HI, USA, November 2018, S. 409–416

[Sch22] Schmidt, Benjamin: *Classical and Deep Learning based Methods for Line Segment Connectivity and Non-Maximum Suppression in Automated Driving*. Karlsruhe, Deutschland, Dezember 2022. – Masterarbeit am Institut für Mess- und Regelungstechnik, Karlsruher Institut für Technologie

[SGDMN21] Suleymanov, T. ; Gadd, M. ; De Martini, D. ; Newman, P.: The Oxford Road Boundaries Dataset. In: *IEEE Intelligent Vehicles Symposium Workshops*. Nagoya, Japan, Juli 2021, S. 222–227

[Sku20] Skudlik, Philipp: *Parametrized Lane Centerline Estimation for Autonomous Driving using Deep Neural Networks*. Karlsruhe, Deutschland, Juli 2020. – Masterarbeit am Institut für Mess- und Regelungstechnik, Karlsruher Institut für Technologie

[SMAG19] Simony, Martin ; Milzy, Stefan ; Amendey, Karl ; Gross, Horst-Michael: Complex-YOLO: An Euler-Region-Proposal for Real-time 3D Object Detection on Point Clouds. In: Leal-Taixé, Laura (Hrsg.) ; Roth, Stefan (Hrsg.): *Computer Vision – ECCV 2018 Workshops*. Cham, Schweiz : Springer International Publishing, Januar 2019. – ISBN 978–3–030–11009–3, S. 197–209

[TBP+21] Tabelini, Lucas ; Berriel, Rodrigo ; Paixao, Thiago M. ; Badue, Claudine ; De Souza, Alberto F. ; Oliveira-Santos, Thiago: Keep Your Eyes on the Lane: Real-Time Attention-Guided Lane Detection. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. Nashville, TN, USA, Juni 2021, S. 294–302

[Tit22] Tittmann, Peter: *Graphentheorie: Eine Anwendungsorientierte Einführung*. Vierte Auflage. München, Deutschland : Carl Hanser Verlag, 2022. – ISBN 978–3–446–47247–1

[VSP+17] Vaswani, Ashish ; Shazeer, Noam ; Parmar, Niki ; Uszkoreit, Jakob ; Jones, Llion ; Gomez, Aidan N. ; Kaiser, Łukasz ; Polosukhin, Illia: Attention Is All You Need. In: *Advances in Neural Information Processing Systems*. Long Beach, CA, USA, Dezember 2017

[Wal18] Walter, Jonas: *Approximation der Kreuzungstopologie und der Fahrstreifen mittels Sampling-Methoden anhand der Trajektorien anderer Verkehrsteilnehmer*. Karlsruhe, Deutschland, Oktober 2018. – Bachelorarbeit am Institut für Mess- und Regelungstechnik, Karlsruher Institut für Technologie

[Wal20] Walter, Jonas: *MCMC-Schätzung der Kreuzungstopologie und der Fahrstreifen während des Zufahrens auf die Kreuzung*. Karlsruhe, Deutschland, Dezember 2020. – Masterarbeit am Institut für Mess- und Regelungstechnik, Karlsruher Institut für Technologie

[Wil18] Wilbrandt, Robert: *Clustering Methods for Lane Topology Estimation based on Trajectories of other Traffic Participants*. Karlsruhe, Deutschland, Oktober 2018. – Bachelorarbeit am Institut für Mess- und Regelungstechnik, Karlsruher Institut für Technologie

[WQA+21] Wilson, Benjamin ; Qi, William ; Agarwal, Tanmay ; Lambert, John ; Singh, Jagjeet ; Khandelwal, Siddhesh ; Pan, Bowen ; Kumar, Ratnesh ; Hartnett, Andrew ; Pontes, Jhony K. ; Ramanan, Deva ; Carr, Peter ; Hays, James: Argoverse 2: Next Generation Datasets for Self-Driving Perception and Forecasting. In: *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*. virtuell, August 2021

[WWW+19] Wang, Congchao ; Wang, Yizhi ; Wang, Yinxue ; Wu, Chiung-Ting ; Yu, Guoqiang: muSSP: Efficient Min-cost Flow Algorithm for Multi-object Tracking. In: *Advances in Neural Information Processing Systems*. Vancouver, Kanada, Dezember 2019

[YCW+20] Yu, Fisher ; Chen, Haofeng ; Wang, Xin ; Xian, Wenqi ; Chen, Yingying ; Liu, Fangchen ; Madhavan, Vashisht ; Darrell, Trevor: BDD100K: A Diverse Driving Dataset for Heterogeneous Multitask Learning. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. virtuell, Juni 2020, S. 2636–2645

[Zan21] Zanker, Dominik: *Deep Convolutional Feature Extraction for Road Boundaries*. Karlsruhe, Germany, Januar 2021. – Masterarbeit am Institut für Mess- und Regelungstechnik, Karlsruher Institut für Technologie

[ZHL+22] Zheng, Tu ; Huang, Yifei ; Liu, Yang ; Tang, Wenjian ; Yang, Zheng ; Cai, Deng ; He, Xiaofei: CLRNet: Cross Layer Refinement Network for Lane Detection. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. New Orleans, LA, USA, Juni 2022, S. 898–907

[ZVB21] Zürn, Jannik ; Vertens, Johan ; Burgard, Wolfram: Lane Graph Estimation for Scene Understanding in Urban Driving. In: *IEEE Robotics and Automation Letters* 6 (2021), Januar, Nr. 4, S. 8615–8622

[ZZF+21] Zhang, Yujun ; Zhu, Lei ; Feng, Wei ; Fu, Huazhu ; Wang, Mingqian ; Li, Qingxia ; Li, Cheng ; Wang, Song: VIL-100: A New Dataset and a Baseline Model for Video Instance Lane Detection. In: *IEEE/CVF International Conference on Computer Vision*. virtuell, Oktober 2021, S. 15681–15690