Hochschule für Technik und Wirtschaft Dresden
Fakultät Maschinenbau

HTWD

Aufgabenblatt für die Diplomarbeit

im Studiengang / Studienrichtung: **Fahrzeugtechnik / Kraftfahrzeugtechnik**

Name des Diplomanden/ Matrikel-Nr.: **Max Lenke / 49388**

Thema **Implementierung und Test von Bewertungsverfahren für die Objekterkennung**

Kurzbeschreibung:

Aufgrund der zunehmenden Verbreitung von Lidar-Sensoren in Bereichen wie autonomes Fahren, Robotik und Überwachungssystemen ist es von Bedeutung, die Leistungsfähigkeit und Zuverlässigkeit dieser Sensoren bei der Erkennung mehrerer Objekte zu analysieren. Die GOSPA-Metrik standardisiert die Bewertung des Multi-Object Tracking von Lidar-Sensoren und liefert Kostenwerte, die den Vergleich erleichtern. Diese Diplomarbeit zielt darauf ab, die Eignung von Lidar-Sensoren für die Multi-Objektdetektion zu bewerten, indem die GOSPA-Metrik in MATLAB angewendet wird. Durch die Durchführung von Evaluierungsversuchen und der Analyse der Ergebnisse soll die Leistungsfähigkeit verschiedener Lidar-Sensoren quantifiziert werden, um Rückschlüsse auf ihre optimale Nutzung zu ziehen.

Arbeitspunkte:

- Einarbeitung in Bewertungsmethoden für Objektdetektion (GOSPA_Metrik in Matlab)
- Implementierung einer Test-APP in Matlab zur effektiven Bewertung verschiedener Lidar-Sensoren
- Durchführung und Auswertung von Referenzversuchen
- Konzeption einer Implementierung in die Software ecu.test der Firma tracetronic

Die Ergebnisse der Arbeit sind in einem Poster zu dokumentieren.

Betreuer HTW: Prof. Dr. rer. nat. Toralf Trautmann
Dipl.-Ing. (FH) Franziskus Mendt

Ausgehändigt am: 15.05.2024 Einzureichen bis: 16.10.2024

.....
Prof. Dr. rer. nat. Toralf Trautmann
Verantwortlicher Hochschullehrer

.....
Prof. Dr.-Ing. Thomas Himmer
Prüfungsausschussvorsitzender

Hinweise zum Erstellen der Diplomarbeit unter
https://www.htw-dresden.de/hochschule/fakultaeten/maschinenbau/studium/diplomsemester
Die Aufgabenstellung kann nach Absprache und dem Vorliegen von Teilergebnissen erweitert bzw. eingeengt werden.

Hochschule für Technik und Wirtschaft Dresden
Fakultät Maschinenbau
Studiengang Fahrzeugtechnik
Studienrichtung Kraftfahrzeugtechnik

Diplomarbeit

Implementierung und Test von Bewertungsverfahren für die Objekterkennung

vorgelegt von:
Max Lenke

Betreuer HTW	Prof. Dr. rer. nat. Toralf Trautmann
	Dipl. Ing. (FH) Franziskus Mendt
Abgabetermin	13.11.2024

Inhaltsverzeichnis

Abkürzungsverzeichnis

APD	Avalanche Photodiode
CARLA	Car Log Analyser
CV	Constant Velocity
CW	Continious Wave
CT	Constant Turn
ECE	Euclidean Cluster Extraction
EKF	Extended Kalman Filter
FN	False Negatives
FP	False Positives
GNN	Global Nearest Neighbor
GPS	Global Positioning System
GOSPA	Generalized Optimal Subpattern Assignment
GUI	Graphical User Interface
HOTA	Higher Order Tracking Accuracy
IDSW	ID Switches
IMM	Interacting Multiple Model
JPDA	Joint Probabilistic Data Association
KF	Kalman Filter
KI	Künstliche Intelligenz
LiDAR	Light Detection and Ranging
MATLAB	Matrix Laboratory
MEMS	Micro-Electro-Mechanical Systems
MOT	Multiple Object Tracking
MOTA	Multi Object Tracking Accuracy
MOTP	Multi Object Tracking Precision
NN	Nearest Neighbor
OSPA	Optimal Subpattern Assignment
PIN	Positive-Intrinsic-Negative
RADAR	Radio Detection And Ranging
RANSAC	Random Sample Consensus
ROI	Region of Interest
RTK	Real-Time Kinematic
SLAM	Simultaneous Localization and Mapping
ToF	Time of Fligth
TP	True Positives
UKF	Unscented Kalman Filter

Verzeichnis der Formelzeichen und Symbole

Symbol	Einheit	Beschreibung
Φ	rad	Winkel der Objektreflexion
d	m	Entfernung zwischen Objekt und Sensor
c	m/s	Ausbreitungsgeschwindigkeit des Lichtes
ToF	s	Laufzeit des Laserimpulses
v, $\vec{v}_{\text{rel}}$	m/s	Relativgeschwindigkeit
f_d	Hz	Dopplerfrequenz
λ	m	Wellenlänge des ausgesendeten Signals
$\vec{R}$	m	Entfernung zwischen Sensor und Objekt
t	s	Zeitpunkt der Messung
$d(p_i, p_j)$	m	Euklidische Distanz zwischen Punkten p_i und p_j
p_i, p_j	m	Punkte im Raum
$x_i, x_j, y_i, y_j, z_i, z_j$	m	Koordinaten der Punkte p_i und p_j
d_{ij}	m	Euklidische Distanz zwischen z_i und $\hat{x}_j$
z_i	m	Position der Messung i
$\hat{x}_j$	m	Geschätzte Position des Ziels j
C_{ij}	-	Kosten für die Zuordnung von z_i zu $\hat{x}_j$
P_{ij}	-	Wahrscheinlichkeit, dass z_i zu $\hat{x}_j$ zugeordnet wird
S_j	-	Unsicherheit der Zielposition j
Var(x_i)	-	Varianz von x_i
Cov(x_i, x_j)	-	Kovarianz von x_i und x_j
x_i, x_j	-	Zufallsvariablen
Q	-	Prozessrausch-Kovarianzmatrix
q_{ij}	-	Element der Matrix Q
R	-	Messrausch-Kovarianzmatrix
r_{ij}	-	Element der Matrix R

Zeichen	Einheit	Bedeutung
$FP, IDFP$	-	False Positive
$FN, IDFN$	-	False Negative
$TP, IDTP$	-	True Positive
TN	-	True Negative
$IDSW$	-	Identity Switches
$gtDet$	-	Ground Truth
$d_{t,i}$	m	Distanz/Überlappung von Ziel i gtDet im Frame t
c_t	-	Anzahl korrekt zugeordneter Hypothesen im Frame t
$prDet$	-	Vom Tracker erkanntes Objekt
$prID$	-	Zugewiesene Identität für eine prDet
TPA	-	Korrekte Assoziationen für einen True Positive
FNA	-	Fehlende Assoziationen für einen True Positive
FPA	-	Falsche Assoziationen für einen True Positive
$A(c)$	-	Assoziationsscore für einen True Positive c
d_{OSPA}	-	OSPA-Gesamtfehler
d_{GOSPA}	-	GOSPA-Gesamtfehler
X, Y	-	Mengen geschätzter und tatsächlicher Positionen
π	-	Optimale Zuordnung zwischen X und Y
c	m	Maximale Distanz (Cutoff)
p	-	Exponent zur Gewichtung großer Fehler
$\|X\|, \|Y\|$	-	Anzahl der Objekte in X bzw. Y
α	-	Gewichtung für den Kardinalitätsfehler in GOSPA
θ	rad	Winkel zwischen $\vec{n}$ und $\vec{r}$
$\vec{n}, \vec{r}$	-	Normal- und Referenzvektor der Ebene
$x_{\text{mean}}, y_{\text{mean}}, z_{\text{mean}}$	m	Koordinaten des Schwerpunktes
L, W, H	m	Länge, Weite, Breite eines Objekts
$P_d(r)$	-	Detektionswahrscheinlichkeit abh. von Distanz r
s, sz	-	Schrumpfrate für L/W und H
$L_{\text{shrink}}, W_{\text{shrink}}, H_{\text{shrink}}$	m	L, W, H nach Schrumpf-Korrektur
$\dot{x}, \dot{y}, \dot{z}$	m/s	Geschwindigkeit entlang der x-, y- und z-Achse
x_cv, x_cv	-	Zustandsvektoren
ω	rad/s	Winkelgeschwindigkeit

Zeichen	Einheit	Bedeutung
$Q_{\text{CV}}, Q_{\text{CT}}$	-	Prozessrauschmatrizen
z_k	-	Messvektor
x_k	-	Geschätzter Zustand des Objekts
H	-	Beobachtungsmatrix, wählt Komponenten aus x_k
w_k	-	Rauschvektor für Messfehler
Π_{ij}	-	Matrix für Übergangswahrscheinlichkeit

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Die fortschreitende Digitalisierung und Automatisierung prägen zunehmend verschiedene Bereiche wie autonomes Fahren, Robotik und Überwachungssysteme. In diesem Kontext gewinnen Light Detection and Ranging (LiDAR)-Sensoren an Bedeutung, da sie präzise und zuverlässige Umgebungsdaten liefern. Diese sind für die Entscheidungsfindung in sicherheitsrelevanten Situationen autonomer Anwendungen essenziell. LiDAR-Systeme ermöglichen die Erstellung detaillierter 3D-Modelle der Umgebung, was die Erkennung und Verfolgung mehrerer Objekte erheblich verbessert. Diese hohe räumliche Auflösung und die Fähigkeit zur präzisen Entfernungsmessung machen LiDAR zu einem unverzichtbaren Bestandteil moderner Sensortechnologien.

Die Leistungsfähigkeit und Zuverlässigkeit von LiDAR-Sensoren bei der Verfolgung von Objkekten sind entscheidende Faktoren, die ihre Eignung für sicherheitskritische Anwendungen bestimmen. Um diese Eigenschaften objektiv zu bewerten, ist die Entwicklung und Anwendung standardisierter Bewertungsverfahren unerlässlich. Die Generalized Optimal Subpattern Assignment (GOSPA)-Metrik stellt hierbei ein fortschrittliches Verfahren dar, das die Leistung von LiDAR-Sensoren in der Objektverfolgung quantitativ analysiert und Kostenwerte liefert, die einen direkten Vergleich verschiedener Systeme ermöglichen.

Ziel dieser Diplomarbeit ist es, die Eignung von LiDAR-Sensoren für die Multi-Objektdetektion zu bewerten, indem die GOSPA-Metrik in der Software Matrix Laboratory (MATLAB) implementiert und angewendet wird. Durch die Entwicklung eines Evaluierungsframeworks sollen verschiedene LiDAR-Sensoren systematisch getestet und deren Leistung hinsichtlich Präzision und Zuverlässigkeit analysiert werden. Ein zentraler Aspekt dieser Arbeit ist die Implementierung einer MATLAB Test-App, die eine effektive Bewertung unterschiedlicher LiDAR-Sensoren ermöglicht und die Anwendung der GOSPA-Metrik vereinfacht. Darüber hinaus wird ein Konzept entwickelt, das die Integration der MATLAB-App in die Software `ecu.test` der Firma tracetronic vorsieht.

Die vorliegende Arbeit verfolgt folgende Forschungsfragen:

- Wie kann die GOSPA-Metrik effektiv in MATLAB implementiert werden, um die Leistungsfähigkeit von LiDAR-Sensoren bei der Multi-Objektdetektion zu bewerten?
- Inwieweit ermöglicht die entwickelte MATLAB-App eine vergleichende Analyse verschiedener LiDAR-Technologien und welche Unterschiede in deren Leistungsbewertung sind dabei ersichtlich?
- Wie können Konzepte zur effizienten Erstellung von Referenzdaten für die GOSPA-

Metrik die schnellere Testung von LiDAR-Technologien vorrantreiben?

Durch die Beantwortung dieser Fragen soll ein umfassendes Verständnis für die Leistungsfähigkeit von LiDAR-Sensoren im Kontext der Objektverfolgung geschaffen werden. Die Ergebnisse dieser Arbeit sollen dazu beitragen, Empfehlungen für die Validierung von LiDAR-Sensoren hinsichtlicher der Leistungsfähigkeit von Objektverfolgung zu geben und somit die Effizienz und Zuverlässigkeit solcher Systeme zu garantieren. Diese Diplomarbeit trägt zur Weiterentwicklung von Testverfahren für LiDAR-Sensoren hinsichtlich der Objekterkennung bei.

2 Technische Grundlagen Light Detection and Ranging

In diesem Kapitel werden die grundlegenden Konzepte vermittelt, die für das Verständnis der nachfolgenden Arbeit von zentraler Bedeutung sind. Zunächst erfolgt eine detaillierte Darstellung des Funktionsprinzips von LiDAR-Sensoren, gefolgt von einer eingehenden Erläuterung ihres Aufbaus. Hierbei werden die verschiedenen Messmethoden sowie die einzelnen Komponenten der Sensoren behandelt. Im Anschluss daran wird das Konzept der Mehrfachobjektverfolgung (Multiple Object Tracking (MOT)) eingeführt, wobei zentrale Begriffe, wesentliche Schritte und grundlegende Konzepte erläutert werden. Ein besonderes Augenmerk liegt hierbei auf der Verarbeitung von Punktwolken, der Datenassoziation sowie der Zustandsschätzung. Abschließend werden Metriken vorgestellt, die zur Bewertung der Leistungsfähigkeit von MOT-Algorithmen herangezogen werden können.

2.1 Funktionsweise

Grundlegend basiert das Mess-Prinzip von LiDAR auf dem Aussenden von Lichtstrahlen und dem Empfangen der Reflexionen. Während Radio Detection And Ranging (RADAR)-Sensoren elektromagnetische Wellen im Radiofrequenzbereich nutzen, basiert das LiDAR-Prinzip auf einem optischen Messverfahren, bei dem fokussierte Lichtimpulse von Objekten in der Umgebung reflektiert werden. Die zurückkehrenden Lichtstrahlen werden von der Empfangseinheit des LiDAR-Sensors detektiert. Die erfolgreiche Durchführung der Messung setzt voraus, dass das beleuchtete Objekt reflektierende Eigenschaften aufweist und der Lichtweg frei von Störungen ist. [1]

Das System misst die Distanz zum Objekt. Jeder erfasste Punkt repräsentiert die reflektierte Position eines Objekts im Raum. Die Gesamtheit dieser Punkte bildet eine Punktwolke (engl.: Pointcloud), die das Umfeld in drei Dimensionen (x, y und z) darstellt. Abbildung 2.1 veranschaulicht beispielhaft eine Punktwolke welche mit einem LiDAR-Sensor (Ouster OS1) aufgenommen wurde. Die Punktwolke bildet einen Volkswagen Passat ab.

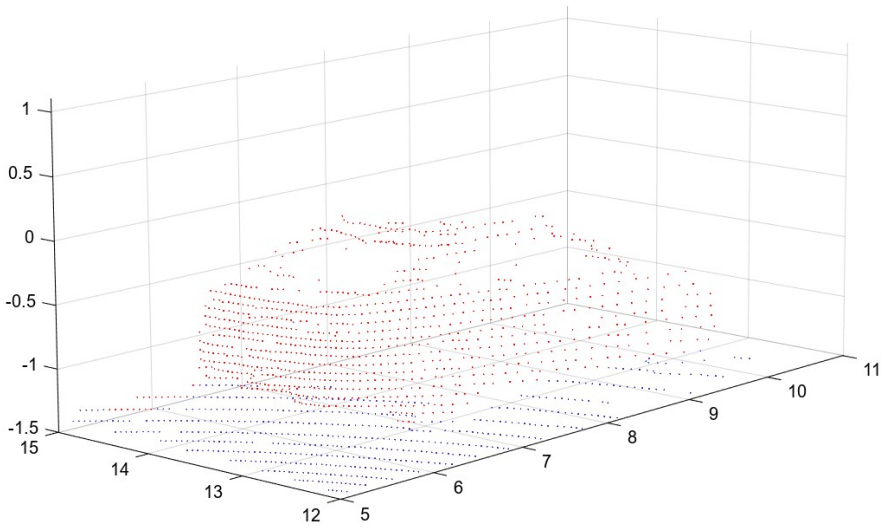


Abbildung 2.1. Punktwolke Volkswagen Passat

2.2 Abstands- und Gechwindigkeitsmessung

Die Abstands- bzw. Geschwindigkeitsmessung bei LiDAR-Sensoren erfolgt durch zwei grundlegende Prinzipien: Time of Fligth (ToF) und Dopplereffekt.

Bei der **ToF**-Messung werden ein oder mehrere Lichtimpulse ausgesendet, die an einem eventuell vorhandenen Objekt reflektiert werden. Die erwartete Puls-Antwort eines festen und einzelnen Objektes, wie beispielsweise die eines Fahrzeuges, hat die Form einer Gaußkurve. Da der ausgesandte Lichtimpuls die Entfernung zwischen Sender und Empfänger zweimal durchlaufen muss, entspricht die Zeit t der doppelten Entfernung zum Objekt.

Die Entfernung d kann durch die Gleichung (2.1) beschrieben werden.

$$d = \frac{c \cdot ToF}{2} \tag{2.1}$$

Dabei kennzeichnet die Größe c die Ausbreitungsgeschwindigkeit des Lichtes im vorliegenden Medium Luft von etwa 3×10^8 Meter pro Sekunde während ToF die gemessene Zeit darstellt. Abbildung 2.2 veranschaulicht das Prinzip der Zeitmessung sowie den Zusammenhang der Zeitpunkte t_0 (Pulserzeugung) und t_1 (Detektion der Reflexion).

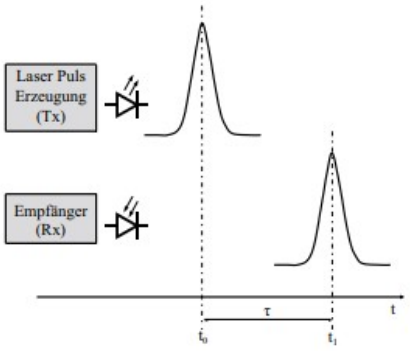


Abbildung 2.2. ToF Messungsprinzip [1]

Befinden sich mehrere Objekte innerhalb eines Messkanals, ermöglicht ein geeignetes Auswerteverfahren bei ausreichendem Abstand zwischen den Objekten die Erfassung mehrerer Zielobjekte. Diese Eigenschaft wird als Mehrzielfähigkeit des Systems bezeichnet. In Abbildung 2.3 sind zwei Pulsantworten zu erkennen. Die Gaußkurven zeigen unterschiedliche Maxima bei Φ , dem Winkel der Objektreflexion in Radiant. Dies weist darauf hin, dass es sich um Reflexionen von zwei verschiedenen Objekten handelt. [1, 2]

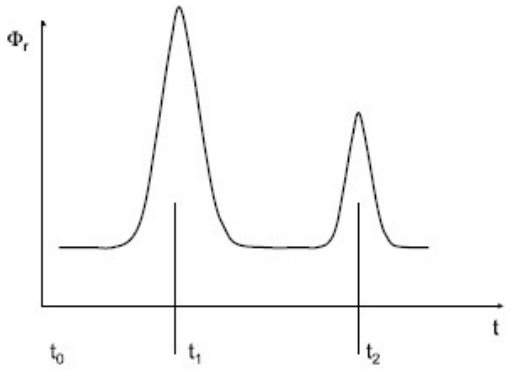


Abbildung 2.3. Pulsantwort zweier Objekte [1]

Der **Dopplereffekt** bietet eine Methode zur Messung der Geschwindigkeit von Objekten. Durch die Messung der Frequenzänderung des zurückgestreuten Lichts lässt sich die Relativgeschwindigkeit des Objekts in Bezug auf andere Objekte bestimmen. Verwendet wird dieser Effekt z.B. in der Verkehrskontrolle und Meteorologie. Die Relativgeschwindigkeit v kann durch die Gleichung 2.2 berechnet werden. Die Variable f_d beschreibt die Doppler-Verschiebung (Frequenzunterschied) und λ die Wellenlänge des ausgesendeten Signals.

$$v = \frac{2f_d}{\lambda} \tag{2.2}$$

Relativgeschwindigkeit durch Ableitung

Die höheren Anforderungen und die damit verbundenen Kosten bei der Messung der Dopplerfrequenz im Lichtspektrum erschweren deren Umsetzung. Deshalb wird stattdessen

die Differenzenbildung von zwei, idealerweise mehreren, aufeinander folgenden Abstandsmessungen verwendet. Bei dieser Methode wird die Änderungsrate der Entfernung über die Zeit betrachtet, um die Relativgeschwindigkeit $\vec{v}_{\text{rel}}$ zu berechnen.

Dabei bezeichnet $\vec{R}$ die zu einem bestimmten Zeitpunkt gemessene Entfernung zwischen dem Sensor und dem Objekt, während t den Zeitpunkt der Messung angibt. Mit der Gleichung 2.3 kann die Differentiation zur Berechnung der Relativgeschwindigkeit $\vec{v}_{\text{rel}}$ durchgeführt werden.

$$\vec{v}_{\text{rel}} = \frac{d\vec{R}}{dt} = \lim_{\Delta t \to 0} \frac{\Delta \vec{R}}{\Delta t} \tag{2.3}$$

2.3 Komponenten

Ein LiDAR-Sensor setzt sich aus verschiedenen Komponenten zusammen, die dazu dienen, Lichtimpulse auszusenden, deren Reflexionen zu empfangen und die empfangenen Signale zu verarbeiten. Obwohl es unterschiedliche Methodiken und Aufbauvarianten von LiDAR-Sensoren gibt, basieren sie grundsätzlich auf den selben wesentlichen Komponenten. Im Folgenden werden die einzelnen Komponenten sowie ihre jeweiligen Funktionen erläutert.

Sendezweig/Laserquelle

Eine Laserdiode erzeugt Lichtimpulse, die auf das Ziel gerichtet werden. Diese Impulse können im sichtbaren oder unsichtbaren Bereich des Spektrums liegen, wobei die Wellenlängen und die Art des Lasers die Leistung und Reichweite des LiDAR-Systems beeinflussen. Die Wellenlänge der Lichtpulse bei LiDAR-Sensoren in der Automobilanwendung betragen 850 nm bis 1 µm. Die Strahlungsspitzenleistung der eingesetzten Hochleistungsdioden kann 75 W oder mehr betragen. Aufgrund der Anforderungen an die Laserschutzklasse, muss die durchschnittliche Leistung der gepulsten Lasersignale bei 5 mW liegen, was eine Pulswiederholungsrate von 20 kHz bis 100 kHz erfordert. [1][2]

Empfangszweig/Detektor

Der Detektor empfängt das zurückgestreute Licht, welches von den Objekten reflektiert wird. Typischerweise werden Positive-Intrinsic-Negative (PIN)-Dioden oder Avalanche Photodiode (APD) verwendet. Diese Detektoren wandeln das Lichtsignal in elektrische Signale um, die dann zur weiteren Verarbeitung verwendet werden. Spezielle Filtersubstrate filtern das einfallende Licht, bevor es auf die Detektordioden trifft, um bestimmte Wellenlängen auszublenden und die Signalqualität zu erhöhen. Wenn Licht auf die Fotodiode trifft, wird ein Strom erzeugt, der proportional zur empfangenen Strahlungsleistung ist. Dies kann zu sehr geringen Strömen führen. Die elektronische Auswertung des Sensorstroms umfasst die Verstärkung und Umwandlung dieses Stroms in eine elektrische Spannung. Zusätzlich entstehen Störströme infolge von Umgebungsbeleuchtung sowie durch den Dunkelstrom der Diode selbst. Der Dunkelstrom bezeichnet den elektrischen Strom, der in der Diode

auch ohne Lichteinwirkung fließt. Diese langsam variierenden Ströme können jedoch weitgehend durch die Elektronik und Signalverarbeitung eliminiert werden. [2]

Optik

Die optischen Komponenten sind entscheidend für die Fokussierung des Laserstrahls und die Erfassung des zurückkommenden Lichtes. Dazu gehören Linsen, Spiegel und optische Prismen, die für die präzise Fokussierung des Laserstrahls verantwortlich sind. Die Optik beeinflusst die Größe und Form des Lichtbündels sowie die Richtung des Strahls. Linsen und Parabolspiegel werden oft verwendet, um das Licht zu bündeln und die Strahlrichtung zu steuern. Bei der Erfassung des zurückkommenden Lichtes spielen ebenfalls Linsen und andere Fokussier-Optiken eine wichtige Rolle, da sie das Licht auf den Detektor fokussieren. [2]

Elektronik und Signalverarbeitung

Diese Komponenten verarbeiten die vom Detektor empfangenen elektrischen Signale, um die Zeit zwischen der Aussendung des Lichtimpulses und dem Empfang des zurückgestreuten Signals zu messen. Die Signalverarbeitungseinheit berechnet die Entfernungen und kann zusätzliche Daten wie die Intensität des reflektierten Lichts extrahieren. Dies ermöglicht die Erstellung von hochpräzisen dreidimensionalen Karten und Modellen der Umgebung. Um Pulse im Bereich von wenigen Nanosekunden präzise auszuwerten, ist eine Abtastrate im Bereich von mehreren 100 MHz erforderlich. Dies setzt eine hohe Auflösung der gemessenen Spannungswerte voraus, was wiederum einen geringen Eigenrauschpegel der Schaltungstechnik impliziert. Zudem muss der Leistungsverbrauch des gesamten Systems in Relation zur Anzahl der benötigten Laserpulse für ein 3D-Bild betrachtet werden. Daher sollte eine hohe Abtastrate und eine Vielzahl parallel auszuwertender Fotodioden angestrebt werden. Desweiteren werden bei der Signalverarbeitung Filtertechniken eingesetzt, die niederfrequente Störkomponenten herausfiltern. Spezielle Schaltungskonzepte, wie die Differenzverstärkung, werden verwendet um die Auswirkungen von Störströmen minimieren. Ein Beispiel für einen Störstrom ist der vorher erwähnte Dunkelstrom des Sendezweigs. [2]

2.4 Kategorisierung von LiDAR-Sensoren

In diesem Kapitel werden die unterschiedlichen LiDAR-Sensoren systematisch kategorisiert, um einen klaren Überblick über die verfügbaren Technologien zu bieten. Trotz der Verwendung ähnlicher Komponenten lassen sich LiDAR-Sensoren basierend auf ihrer Messmethode und Technik in verschiedene Kategorien unterteilen. LiDAR-Sensoren werden nach zwei grundlegenden Messprinzipien klassifiziert: abtastend (engl.: scanning) und nicht abtastend (engl.: non-scanning). Abbildung 2.4 veranschaulicht diese Unterteilung der Sensoren und zeigt die vielfältigen Ansätze zur Erfassung räumlicher Daten. Im Folgenden werden die technischen Unterschiede der Sensoren präsentiert, um deren Vor- und Nachteile sowie geeignete Anwendungsbereiche besser nachvollziehbar zu machen.

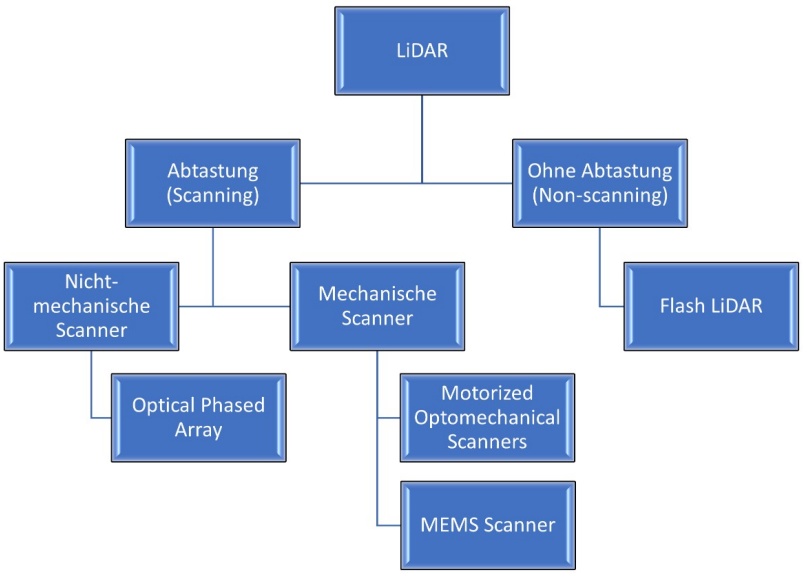


Abbildung 2.4. Kategorisierung von LiDAR-Sensoren [3]

2.4.1 Scanning LiDAR

Bei einem Scanning-LiDAR wird der Laserstrahl entweder mechanisch oder elektronisch durch eine Strahlablenkungseinheit gelenkt, um nacheinander verschiedene Punkte im Sichtfeld abzutasten. Jeder Punkt wird sequenziell beleuchtet und die reflektierten Signale werden gemessen sowie verarbeitet. Dies ermöglicht die Erstellung detaillierter und präziser 3D-Daten, was für Anwendungen wie Kartierung und Geländevermessung von entscheidender Bedeutung ist. Das Funktionsprinzip von Scanning-LiDAR ist in Abbildung 2.5 dargestellt.

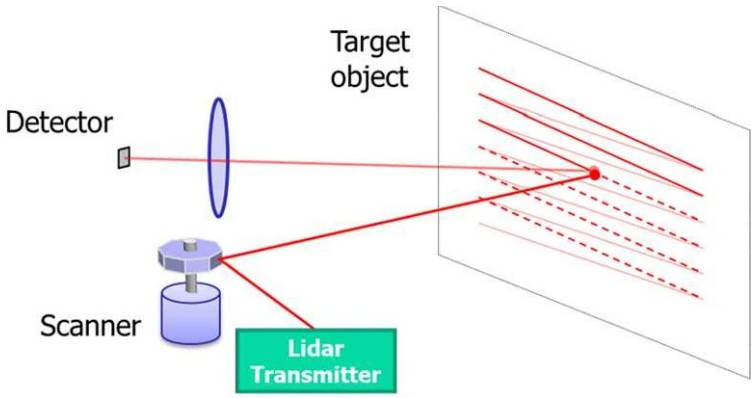


Abbildung 2.5. Schematische Darstellung der Scanning-Messmethodik [4]

Phased Array LiDAR

Phased Array LiDAR nutzt phasengesteuerte Antennen, um den Wellenstrahl ohne mechanische Bewegung zu lenken. Dieses Konzept ist für alle Wellenlängen im elektromagnetischen Spektrum einsetzbar, einschließlich Mikrowellen (RADAR) und optischen Wellenlängen (LiDAR). Durch die elektronische Steuerung der Phasenverschiebung kann der Strahl

präzise und schnell gerichtet werden. Vorteile von Phased Array LiDAR sind das Fehlen beweglicher Teile, was die Zuverlässigkeit und Lebensdauer erhöht, sowie die Möglichkeit schneller Richtungsänderungen und hoher Aktualisierungsraten.

Trotz dieser Vorteile stehen Phased Array LiDAR-Systeme vor mehreren Herausforderungen. Die komplexe und kostspielige Steuerungselektronik erschwert Entwicklung und Produktion. Zudem ist eine hohe Signalqualität für präzise Richtungssteuerungen trotz Störungen erforderlich. Die dichte Antennenanordnung erzeugt erhebliche Wärme, die effizient abgeführt werden muss, um die Systemstabilität zu sichern. [5]

Micro-Electro-Mechanical Systems (MEMS)-LiDAR

MEMS-LiDAR verwendet mikroelektromechanische Systeme zur Steuerung des Lasers oder der Lichtempfänger. Typischerweise wird ein MEMS-Spiegel eingesetzt, um den Laserstrahl schnell und präzise zu lenken. Durch die Miniaturisierung der mechanischen Komponenten ermöglicht MEMS-LiDAR kompakte und leichte Systeme, die sich besonders für den Einsatz in Fahrzeugen mit begrenztem Raumangebot eignen. Zudem bieten MEMS-LiDAR eine hohe Schaltfrequenz, was zu schnellen Aktualisierungsraten und einer verbesserten Erfassungsgenauigkeit führt. Allerdings stehen MEMS-LiDAR auch vor Herausforderungen wie der Anfälligkeit für mechanische Ermüdung und begrenzten Lebensdauern der beweglichen Teile. In Abbildung 2.6 ist der Aufbau eines typischen MEMS-LiDAR-Sensors dargestellt. [4]

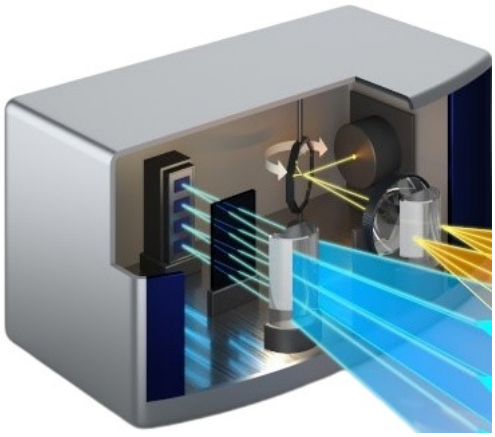


Abbildung 2.6. Aufbau eines MEMS-LiDAR-Sensors [6]

Rotierende LiDAR

Rotierende LiDAR-Sensoren nutzen mechanische Rotationseinheiten, um den Laserstrahl über ein breites Sichtfeld zu verteilen. Diese Systeme bestehen aus einem oder mehreren Lasern, die auf rotierenden Plattformen montiert sind, wodurch eine 360-Grad-Abdeckung ermöglicht wird. Durch die kontinuierliche Rotation können diese LiDAR-Systeme eine umfassende und detaillierte Erfassung der Umgebung gewährleisten. Die rotierenden Mechanismen ermöglichen eine hohe Reichweite und Präzision bei der Abstandsmessung, sind jedoch auch mit Nachteilen wie höherem Energieverbrauch, größerem Gewicht und erhöhter Anfälligkeit für mechanische Verschleißerscheinungen verbunden. Trotz dieser

Herausforderungen bleiben rotierende LiDAR-Sensoren aufgrund ihrer etablierten Technologie und zuverlässigen Leistung eine weit verbreitete Wahl in vielen industriellen Anwendungen. Abbildung 2.7 zeigt den Aufbau eines einfachen LiDAR-Sensors mit Rotationsplattform.

Abbildung 2.7. Aufbau eines rotierenden LiDAR-Sensors [6]

2.4.2 Non-Scanning LiDAR

Bei einem Flash-LiDAR wird ein breiter Laserimpuls verwendet, der das Licht in einem einzigen Impuls über das gesamte Sichtfeld verteilt. Die Reflexionen werden durch eine Detektor-Matrix aufgenommen. Dies ermöglicht die gleichzeitige Erfassung aller Punkte im Sichtfeld und ist besonders nützlich für schnelle 3D-Erfassungen und Echtzeitanwendungen, da es eine schnelle und umfassende Abdeckung bietet. Allerdings begrenzt die Streuung der Lichtintensität über einen relativ großen Winkel die Reichweite der Messungen. Das Funktionsprinzip von Flash-LiDAR ist in Abbildung 2.8 dargestellt. [4]

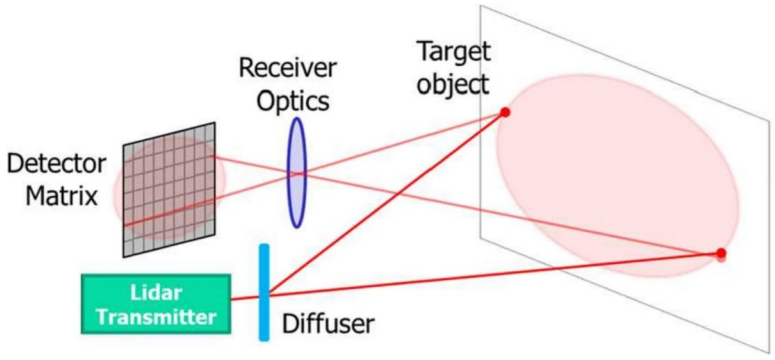


Abbildung 2.8. Schematische Darstellung der Flash-Messmethodik [4]

Flash-LiDAR und Continious Wave (CW)-LiDAR

Flash LiDAR nutzt einen einzigen intensiven Laserimpuls, um die gesamte Szene gleichzeitig zu beleuchten. Diese Methode ermöglicht eine schnelle 3D-Erfassung der Umgebung ohne mechanische Bewegung, was die Zuverlässigkeit erhöht und die Lebensdauer des Systems verlängert.

CW-Laser-LiDAR verwendet ebennfalls die Flah-Messmethodik. Dabei wird jedoch kontinuierliches Licht anstatt Impulse verwendet. Die Distanzmessung erfolgt durch die Phasenverschiebung zwischen ausgesendetem und empfangenem Licht (vgl. Kapitel 2.2). Damit ist eine sehr hohe Präzision und Auflösung bei kürzeren Distanzen ermöglicht. CW-LiDAR-Sensoren bieten eine höhere Messgenauigkeit und bessere räumliche Auflösung, was besonders in der Kartierung und Vermessung von Vorteil ist. Zudem sind sie weniger anfällig für Störungen durch andere Lichtquellen oder LiDAR-Systeme und können aufgrund eines besseren Signal-Rausch-Verhältnisses längere Messreichweiten erreichen.

3 Grundlagen der Mehrfach-Objekt-Verfolgung

Die Mehrfach-Objekt-Verfolgung (MOT) ist ein zentraler Bereich des maschinellen Sehens und spielt eine wesentliche Rolle in Anwendungen wie dem autonomen Fahren, der Videoüberwachung und der Robotik und der Sportanalyse. Das Ziel von MOT besteht darin, mehrere sich bewegende Objekte über eine Sequenz von Bildern hinweg zu verfolgen. Dies erfordert die Identifikation und Verfolgung jedes einzelnen Objekts, selbst wenn diese sich kreuzen oder nahe beieinander liegen.

Der Prozess des MOT beginnt mit der Datenerfassung, bei der Sensoren wie LiDAR, RADAR oder Kameras eingesetzt werden, um die Umgebung in Form von Punktwolken oder Bilddaten zu erfassen. Da der Schwerpunkt dieser Arbeit auf der Datenaufnahme mittels LiDAR liegt, werden im Folgenden ausschließlich Methoden der Objektverfolgung behandelt, die speziell für den Einsatz mit LiDAR-Technologien relevant sind. LiDAR-Daten sind besonders wertvoll für das Verfolgen mehrerer Objekte, da sie die Entfernung und Form von Objekten in der Umgebung genau abbilden. Die dreidimensionalen Punktwolken dienen als Basis für die Objekterkennung, bei der spezifische Algorithmen die Objekte identifizieren und ihre Positionen bestimmen.

3.1 Definition wichtiger Begriffe

In diesem Kapitel werden grundlegende Begriffe erläutert, die für die Objekterkennung und -verfolgung relevant sind. Diese Definitionen schaffen die Basis für ein besseres Verständnis der folgenden Kapitel.

Detektion/Messung bezeichnet die Identifikation eines Objekts oder Punktes innerhalb eines gegebenen Datenraums, etwa einer Punktwolke oder eines Bildes. Eine Detektion ist eine Momentaufnahme, die die Position und Eigenschaften eines Objekts zum Zeitpunkt der Erfassung beschreibt. Messungen enthalten oft Rauschanteile und können durch Umwelteinflüsse beeinflusst sein.

Track/Verfolgung bezeichnet die kontinuierliche Schätzung der Position und anderer Zustände eines Objekts über eine Serie von Detektionen oder Messungen. Die Zuverlässigkeit des Tracks wird dabei maßgeblich von der Genauigkeit der zugrunde liegenden Detektionen beeinflusst. Ein Track umfasst typischerweise Informationen zur Position, Geschwindigkeit und Bewegungsrichtung des Objekts.

Ground Truth/Grundwahrheiten sind als Referenzdaten zu verstehen, die den tatsächlichen Zustand eines Objekts oder Systems beschreiben. In der Regel werden Ground Truth Daten manuell oder durch hochpräzise Sensoren erfasst und stellen den „wahren“ Verlauf oder die exakte Position eines Objekts dar. Diese Daten dienen als Vergleichsgrundlage, um die Genauigkeit von Detektionen und Tracks zu bewerten und Algorithmen zu validieren.

3.2 Grundproblematiken von MOT mit LiDAR

Schrumpfen der Detektionen

Eine der zentralen Problematiken ist das sogenannte Schrumpfproblem, welches auftritt, wenn sich ein Objekt zunehmend vom LiDAR-Sensor entfernt. In solchen Fällen verringert sich die Punktdichte der erfassten Detektionen, was zu einer Verkleinerung und Vereinfachung der Begrenzungsrahmen der Objekte führt. Durch die reduzierte Detailgenauigkeit wird die präzise Verfolgung und Klassifikation der Objekte erschwert, da weniger Informationen für die Bestimmung ihrer genauen Position, Größe und Bewegung verfügbar sind. Zudem steigt die Unsicherheit bei der Positionsbestimmung, was die Stabilität des Tracking-Systems beeinträchtigt und die Wahrscheinlichkeit von Fehlzuweisungen oder vorzeitigen Löschungen von Tracks erhöht. Um diese Herausforderungen zu bewältigen, sind adaptive Algorithmen erforderlich, die die veränderten Bedingungen berücksichtigen und die Detektionswahrscheinlichkeit dynamisch anpassen. Dadurch kann die Genauigkeit und Robustheit des MOT-Systems auch in Szenarien mit variierenden Entfernungen und Punktdichten verbessert werden.

Rauschen

Eine weitere bedeutende Problematik bei der Verwendung von LiDAR-Daten im MOT ist das Rauschen. LiDAR-Sensoren sind anfällig für Rauschen, das durch verschiedene Umweltbedingungen wie Regen, Nebel oder Staub sowie durch technische Unvollkommenheiten des Sensors selbst verursacht werden. Dieses Rauschen führt zu Fehlmessungen und verrauschten Punktwolken, die die Genauigkeit der Objektidentifikation und Verfolgung erheblich beeinträchtigen können. Rauschen kann zu fehlerhaften Detektionen führen, bei denen irrelevante Punkte als Objekte erkannt werden oder echte Objekte nicht zuverlässig erfasst werden. Dies erschwert die Zuordnung von Detektionen zu bestehenden Tracks und erhöht das Risiko von Zuordnungsfehlern sowie von verfrühten oder fehlerhaften Track-Löschungen. Um dem Rauschen entgegenzuwirken, werden verschiedene Filter- und Verarbeitungsmethoden eingesetzt, die das Rauschen reduzieren und die Qualität der Punktwolken verbessern. Zudem können probabilistische Modelle und Sensorfusionstechniken verwendet werden, um die Auswirkungen von Rauschen zu minimieren und die Zuverlässigkeit des MOT-Systems zu erhöhen.

3.3 Methoden zur Objektdetektion

Nach der Datenerfassung ist die Objekterkennung oder -detektion der erste entscheidende Schritt im Verfolgungs-Prozess. Dabei werden Objekte in den erfassten Daten identifiziert und ihre Positionen bestimmt. Die Algorithmen analysieren geometrische Eigenschaften von Punktwolken, um Objekte von ihrer Umgebung zu trennen. Verschiedene Punktdichten weisen auf verschiedene Objekte hin. Dies ist besonders herausfordernd in verrauschten Daten oder komplexen Szenarien mit vielen Objekten die sich überlappen. Um dies zu bewältigen, kommen Techniken wie die Segmentierung zum Einsatz. Auch maschinelles Lernen kann verwendet werden, um Objekte zu identifizieren und klassifizieren. Vertraute Verfahren, darunter die „Euclidean Cluster Extraction“, werden oft verwendet, um eine genaue und zügige Erkennung sicherzustellen. Abbildung 3.1 visualisiert eine segmentierte Punktwolke. Die verschiedenen Segmente wurden mit Begrenzungsrahmen markiert, um Personen und Fahrzeuge als erkannte Objekte darzustellen. Im Folgenden werden gängige Methoden zur Segmentierung von Punktwolken vorgestellt.

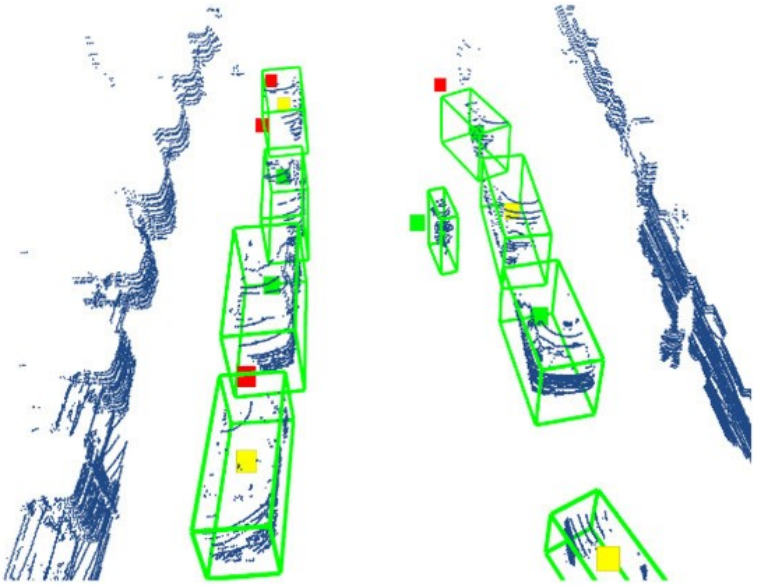


Abbildung 3.1. Beispiel einer Segmentierung innerhalb einer Punktwolke [7]

3.3.1 Erkennung von Ebenen und geometrischen Strukturen

Random Sample Consensus

Der Random Sample Consensus (RANSAC)-Algorithmus stellt ein robustes Verfahren zur Schätzung von Modellparametern in Datenmengen mit hohem Anteil an Ausreißern dar. In der Punktwolkenverarbeitung findet RANSAC häufig Anwendung bei der Erkennung geometrischer Strukturen wie Ebenen oder Zylinder. Der Algorithmus verwendet wiederholt kleine Zufallsstichproben zur Schätzung von Modellparametern. Datenpunkte, die dem Modell ausreichend entsprechen, werden als Übereinstimmung klassifiziert, während abweichende Datenpunkte als Ausreißer verworfen werden. [8]

Diese Methode eignet sich besonders zur Segmentierung von Umweltstrukturen wie Gebäuden, Fahrspuren und Kanten in urbanen Bereichen. Die RANSAC basierte Linien- und Kurvenerkennung ist dabei eine wichtige Technik für die Bestimmung von Fahrbahnen und Hindernissen. Aufgrund der hohen Rechenintensität und der empfindlichen Parameterwahl wird sie jedoch seltener zur Erkennung beweglicher Objekte, wie beispielsweise Fahrzeugen, eingesetzt. [7][9]

Supervoxel Clustering

Supervoxel Clustering ermöglicht eine effiziente Segmentierung von Punktwolken durch die Unterteilung in kleinere, zusammenhängende Regionen, die als Supervoxels bezeichnet werden. Diese Regionen basieren auf räumlicher Nähe und anderen geometrischen Eigenschaften, wodurch die Punktwolke strukturiert und vereinfacht wird. Zu Beginn wird die Punktwolke verdichtet und von Rauschen befreit, um unerwünschte Störungen zu entfernen. Anschließend werden die Supervoxels als grundlegende Einheiten für die Klassifikation gebildet, wobei verschiedene Merkmale wie Form, Dichte und Höhe berücksichtigt werden. [10]

Die Ausrichtung der Normalenvektoren der berechneten Ebenen in den Supervoxeln ermöglicht dabei eine Klassifikation der Bereiche. Da viele künstliche Strukturen, insbesondere in der modernen Architektur, weitgehend rechteckig und plan gestaltet sind, orientieren sich Wandflächen oft parallel zur XY-Ebene und weisen minimale Abweichungen in der Z-Richtung auf. Diese geometrische Eigenschaft erlaubt es, die Supervoxel basierend auf der Ausrichtung ihrer Normalenvektoren in die Klassen „Boden“, „Wand“ und „Sonstiges“ zu unterteilen. Durch diese vereinfachte Strukturierung lassen sich besonders großflächige Umweltstrukturen wie Straßen und Gebäude effizient analysieren, da übersichtliche Einheiten entstehen, die eine Grundlage für weiterführende Segmentierung und Objekterkennung bilden. [11]

3.3.2 Erkennung beweglicher Objekte in dynamischen Szenen

PointNet und PointNet++

PointNet stellt ein tiefes neuronales Netzwerk dar, das speziell für die direkte Verarbeitung von Punktwolken entwickelt wurde, ohne dass diese in Raster- oder Voxelstrukturen überführt werden müssen. Die Methode nutzt die Reihenfolgeunabhängigkeit der Daten und ermöglicht die effiziente Extraktion globaler Merkmale für die Klassifikation und Segmentierung von 3D-Objekten. PointNet++ erweitert die Methode durch eine hierarchische Strukturierung, welche die Analyse lokaler Details innerhalb der Punktwolke ermöglicht. Diese Erweiterung verbessert die Erkennung komplexer geometrischer Strukturen und bietet daher hohe Genauigkeit bei der Segmentierung und Klassifikation anspruchsvoller Szenen, beispielsweise zur Unterscheidung von Fahrzeugen und Fußgängern. [12]

Euclidean Cluster Extraction (ECE)

Der Algorithmus zur euklidischen Cluster-Extraktion gruppiert benachbarte Punkte in einer

Punktwolke basierend auf ihrem euklidischen Abstand. Punkte, deren Abstand unterhalb eines definierten Schwellenwerts liegt, werden zu einem Cluster zusammengefasst. Diese Methode ist besonders effektiv für die Segmentierung zusammenhängender Objekte wie Fahrzeuge oder Personen und eignet sich für Anwendungen, in denen zusammenhängende Strukturen innerhalb von Punktwolken identifiziert werden müssen. Allerdings wird das ECE-Verfahren exponentiell langsamer mit zunehmender Punktanzahl, was seine Berechnungseffizienz bei größeren Datenmengen einschränkt. Für kleinere Punktwolken bleibt es jedoch eine geeignete Methode, auch für Echtzeitanwendungen. [13]

Die euklidische Distanz zweier Punkte p_i, p_j kann durch Gleichung 3.1 berechnet werden.

$$d(p_i, p_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2} \tag{3.1}$$

Die Methode unterscheidet sich von anderen Clustering-Algorithmen durch die strikte Anwendung des Abstandskriteriums und konzentriert sich ausschließlich auf die geometrische Nähe von Punkten.

3.4 Methoden zur Datenassoziation/ Verfolgung

Nach der Objekterkennung ist die Datenassoziation der nächste entscheidende Schritt bei der Verfolgung von Objekten innerhalb von Punktwolken. Dabei stellen die vorher detektierten Objekte die Messung dar. Folglich wird ermittelt, welche Messungen aufeinanderfolgender Zeitpunkte denselben realen Objekten zugeordnet werden können. Um dies zu bewältigen, kommen Methoden wie der Nearest Neighbor (NN)-Algorithmus, der Joint Probabilistic Data Association (JPDA)-Algorithmus und der Hungarian Algorithmus zum Einsatz. Diese Methoden werden in den nächsten Kapiteln näher beleuchtet.

3.4.1 Nearest Neighbor Algorithmus

Prinzip: Der NN-Algorithmus weist jeder Messung das nächstgelegene Ziel basierend auf der vorhergesagten Position zu. Die Zuordnung basiert auf der minimalen euklidischen Distanz zwischen der Messung und der Zielvorhersage. Dies ist eine einfache und effiziente Methode zur Datenassoziation, die besonders bei wenigen Zielen und Messungen gut funktioniert.

Die Zuordnung erfolgt durch Minimierung der Distanz d_{ij} zwischen einer Messung z_i und der Vorhersage des Zustands $\hat{x}_j$. Dieser mathematische Zusammenhang ist in Gleichung 3.2 beschrieben.

$$d_{ij} = \|z_i - \hat{x}_j\| \tag{3.2}$$

Vorteile: Der NN-Algorithmus ist aufgrund seiner Einfachheit und geringen Rechenkosten besonders gut für Szenarien mit einer kleinen Anzahl von Zielen und Messungen geeignet.

Einschränkungen: Der Algorithmus ist anfällig für Fehler in Szenarien mit mehreren Zielen, insbesondere wenn sich die Messungen von Zielen überschneiden oder die Objekte nahe beieinander liegen. In solchen Fällen kann es zu fehlerhaften Zuordnungen kommen, da der Algorithmus weder Unsicherheiten noch Kovarianzen der Messungen oder Zielvorhersagen berücksichtigt. Darüber hinaus beeinträchtigt Rauschen in den Messungen die Zuverlässigkeit der Zuordnung. [14]

3.4.2 Global Nearest Neighbor Algorithmus

Prinzip: Der Global Nearest Neighbor (GNN) Algorithmus basiert auf dem gleichen Grundkonzept wie der NN-Algorithmus, jedoch wird hier die Zuordnung für alle Ziele und Messungen gleichzeitig optimiert. Die Absicht besteht darin, eine Zuordnung zu bestimmen, die den Gesamtfehler über alle Ziel-Messungs-Paare verringert. Anstatt jede Messung einzeln zuzuordnen, sucht der GNN-Algorithmus nach der optimalen Kombination von Zuordnungen, die den kleinsten Gesamtabstand liefert. Dies wird häufig als Optimierungsproblem mit der Zielfunktion formuliert. Die Gleichung 3.3 liefert den kleinsten Gesamtabstand, wobei d_{ij} entsprechend der Gleichung 3.2 die Distanz zwischen Messung und vorhersage ist.

$$\min \sum_{i,j} d_{ij} \tag{3.3}$$

Vorteile: Der GNN Algorithmus liefert eine global optimierte Zuordnung, was die Genauigkeit bei Mehrziel-Szenarien deutlich verbessert. Er eignet sich besonders für Szenarien mit einer moderaten Anzahl von Zielen und Messungen, bei denen eine präzise Zuordnung entscheidend ist.

Einschränkungen: Obwohl der GNN-Algorithmus eine globale Optimierung für die Zuordnung liefert, ist er deutlich rechenintensiver als der einfachere NN-Algorithmus. Da er die Zuordnung für alle Ziele gleichzeitig berechnet, steigt die Rechenlast erheblich mit der Anzahl der Ziele und Messungen. Dies kann bei großen Datensätzen und einer hohen Zielanzahl zu Ineffizienzen führen. [14][15]

3.4.3 Hungarian Algorithmus

Prinzip: Der Hungarian Algorithmus ist ein Optimierungsalgorithmus, der verwendet wird, um Zuordnungsprobleme effizient zu lösen. Ziel ist es, die Gesamtkosten aller Zuordnungen zu minimieren, indem jede Messung einem Ziel zugeordnet wird, basierend auf einer Kostenmatrix. Die Kostenmatrix C beschreibt die Zuordnungskosten zwischen den Messungen z_i und den Zielvorhersagen $\hat{x}_j$. Die Gleichung 3.4 stellt diesen Zusammenhang

dar. Dabei stellt C_{ij} die Kosten für die Zuordnung der Messung z_i zum Ziel $\hat{x}_j$ dar.

$$\min \sum_{i,j} C_{ij} \tag{3.4}$$

Vorteile: Der Hungarian Algorithmus liefert eine optimale Lösung für Zuordnungsprobleme und kann in Kombination mit anderen Methoden verwendet werden, um die Effizienz zu steigern.

Einschränkungen: Er kann bei sehr großen Problemgrößen, d.h. bei vielen Zielen und Messungen, zeitaufwendig sein. Aus diesem Grund wird er oft in Kombination mit anderen Methoden verwendet, um die Effizienz zu steigern, insbesondere bei Echtzeitanwendungen oder Szenarien mit einer großen Anzahl von Zielen. [16]

3.4.4 Joint Probabilistic Data Association

Prinzip: Der JPDA-Tracker stellt einen statistischen Ansatz zur Lösung des Datenassoziationsproblems bei der Verfolgung mehrerer Ziele dar. Der JPDA-Algorithmus berechnet die Wahrscheinlichkeiten aller möglichen Zuordnungen von Messungen zu Zielen und kombiniert diese, um eine optimale Zuordnung zu bestimmen.

Der JPDA-Tracker setzt einen Ansatz ein, der die Zustände jedes Ziels anhand eines Bewegungsmodells vorhersagt. In Abbildung 3.2 wurde der systematische Ablauf der Mehrzielverfolgung mittels einem JPDA Tracker dargestellt.

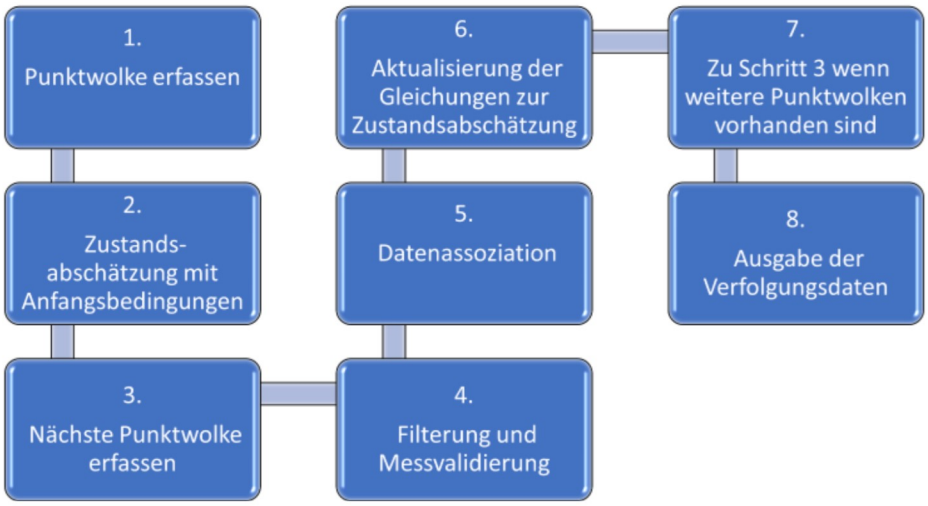


Abbildung 3.2. Flussdiagramm eines JPDA Trackers [17]

Die Wahrscheinlichkeit P_{ij}, dass eine Messung z_i einem Ziel $\hat{x}_j$ zugeordnet wird, kann vereinfacht durch die Gleichung 3.5 dargestellt werden.

$$P_{ij} \propto \exp\left(-\frac{d_{ij}}{S_j}\right) \tag{3.5}$$

Die Exponentialfunktion $\exp$ in Gleichung 3.5 spielt eine zentrale Rolle bei der Gewichtung der Wahrscheinlichkeit P_{ij}. Sie sorgt dafür, dass die Wahrscheinlichkeit exponentiell mit zunehmendem Abstand d_{ij} zwischen der Messung z_i und dem Ziel $\hat{x}_j$ abnimmt, während sie gleichzeitig die Unsicherheit S_j berücksichtigt. [18]

Dies bedeutet:

- **Kleiner Abstand + geringe Unsicherheit:** Eine Messung z_i, die nah an der vorhergesagten Position $\hat{x}_j$ liegt und geringe Unsicherheit S_j aufweist, führt zu einer hohen Wahrscheinlichkeit P_{ij}.
- **Großer Abstand + hohe Unsicherheit:** Eine Messung z_i, die weit von $\hat{x}_j$ entfernt ist und hohe Unsicherheit S_j aufweist, ergibt eine moderate Wahrscheinlichkeit P_{ij}.
- **Großer Abstand + geringe Unsicherheit:** Eine weit entfernte Messung z_i mit geringer Unsicherheit S_j führt zu einer niedrigen Wahrscheinlichkeit P_{ij}.

Vorteile: Der JPDA Algorithmus ermöglicht die gleichzeitige Nachverfolgung mehrerer Ziele, indem er Unsicherheiten bei der Datenassoziation berücksichtigt. Dies verleiht ihm eine hohe Effektivität in Umgebungen mit überlappenden Objekten und verrauschten Messdaten. Durch die Einbeziehung sämtlicher potenzieller Zuordnungen gewährleistet der JPDA Tracker eine robuste und zuverlässige Zielverfolgung, selbst in komplexen und dynamischen Szenarien.

Einschränkungen: Trotz seiner Vorteile kann der JPDA Algorithmus bei einer großen Anzahl von Zielen und Messungen schnell komplex und rechenintensiv werden. Da die Wahrscheinlichkeiten für alle möglichen Kombinationen berechnet und kombiniert werden müssen, skaliert der Rechenaufwand exponentiell mit der Anzahl der Ziele und Messungen. [17][19]

3.5 Methoden der Zustandsschätzung

Die Zustandsschätzung ist der Prozess, bei dem dynamische Zustandsparameter eines Objekts – wie Position, Geschwindigkeit, Beschleunigung und Orientierung – basierend auf den verfügbaren Detektion geschätzt werden. Um präzise und robuste Schätzungen zu erzielen, werden mathematische Modelle und Filtertechniken verwendet, die Rauschen und Unsicherheiten sowohl in den Messungen als auch im Modell berücksichtigen. Bekannte Ansätze zur Zustandsschätzung sind der Kalman Filter (KF) und der Extended Kalman Filter (EKF). In komplexen Szenarien, in denen sich Objekte abrupt bewegen oder die Richtung ändern, kann der Ansatz des Interacting Multiple Model (IMM)-Filter eingesetzt werden, um die Schätzungen weiter zu verbessern. Der IMM-Filter verwendet mehrere Zustandsmodelle und wählt diese anhand der aktuellen Messdaten mithilfe von Übergangswahrscheinlichkeiten aus. Dadurch kann er sich flexibel an die unterschiedlichen Bewegungsmuster und dynamischen Zustände der zu verfolgenden Objekte anpassen. In den folgenden Kapiteln werden die verschiedenen Filtertechniken ausführlich erläutert, um

einen fundierten Einblick in die Methoden der Zustandsschätzung zu bieten.

3.5.1 Kalman Filter

Der KF-Filter verwendet einen rekursiven Algorithmus zur Schätzung des Zustands eines dynamischen Systems basierend auf einem mathematischen Modell und verrauschten Messdaten. Der Filter geht davon aus, dass sowohl die Systemdynamik, als auch die Messungen durch zufällige Fehler beeinflusst werden. Er verwendet die KF-Gleichungen, um eine optimale Schätzung des Zustands zu liefern, indem er die Vorhersagen mit den tatsächlichen Messungen kombiniert. Der KF-Filter schätzt sowohl den aktuellen Zustand $\hat{x}_k$, als auch die Unsicherheit P_k, die durch eine Kovarianzmatrix beschrieben wird.

Einschränkungen: Der KF-Filter funktioniert nur für lineare Systeme und setzt voraus, dass die Verteilungen des Zustands- und Messrauschens normalverteilt sind. Bei stark nichtlinearen Systemen liefert er keine genauen Schätzungen. [20][21][22][23]

3.5.2 Erweiterte Kalman-Filter

Der **Extended Kalman Filter (EKF)** erweitert den Kalman-Filter auf nichtlineare Systeme, indem er die System- und Messmodelle durch eine Taylor-Reihen-Approximation erster Ordnung linearisiert. Dies geschieht durch die Berechnung der Jacobi-Matrix der Systemdynamik und der Messgleichungen. Der EKF ist besonders nützlich, wenn die Dynamik und die Messungen eines Systems nichtlinear sind, wie es oft bei der Verfolgung von Flugzeugen, Fahrzeugen oder Robotern der Fall ist.

Einschränkungen: Die Linearisierung kann zu fehlerhaften Schätzungen führen, insbesondere wenn die Systemdynamik oder die Messgleichung stark nichtlinear ist.

Der **Unscented Kalman Filter (UKF)** wurde entwickelt, um die Einschränkungen des EKF zu überwinden. Anstatt die nichtlinearen System- und Messgleichungen zu linearisieren, nutzt der UKF eine spezielle Methode, welche die nichtlinearen Zusammenhänge direkt berücksichtigt und dadurch genauere Zustandsschätzungen ermöglicht. Statt eine Taylor-Approximation zu verwenden, nutzt die Methode eine Gruppe festgelegter Punkte, die die Verteilung des Zustands darstellen. Diese Punkte werden durch die nichtlineare Funktion geführt, um genauere Schätzungen zu ermöglichen. Dadurch kann der Zustand präziser bestimmt werden, ohne komplexe Approximationen anwenden zu müssen.

Einschränkungen: Der UKF ist zwar genauer als der EKF, er ist jedoch auch rechenintensiver. [20][21][22][23]

3.5.3 Interacting Multiple Model Filter

Der IMM Filter ist ein effizienter Algorithmus zur Zustandsschätzung in dynamischen Systemen, die zwischen mehreren Bewegungsmodellen wechseln. Im Gegensatz zu herkömmli-

chen Filtern berücksichtigt der IMM-Filter mehrere mögliche Modelle gleichzeitig und kombiniert deren Ergebnisse, um eine optimale Schätzung des Systemzustands zu liefern. Jeder Modellfilter verfolgt unabhängig die Systemdynamik und die Gewichtungen der Modelle werden basierend auf deren Übereinstimmung mit den eingehenden Messungen kontinuierlich angepasst. Der IMM-Filter ist besonders nützlich in Szenarien mit abrupten Zustandsänderungen, wie z.B. bei der Verfolgung von Fahrzeugen, die zwischen unterschiedlichen Bewegungsmodi (z.B. Beschleunigung, Kurvenfahrt) wechseln. Dies ermöglicht eine höhere Robustheit und Genauigkeit gegenüber traditionellen Filtern, die nur ein einzelnes Bewegungsmodell verwenden. [24][25]

3.6 Kovarianzmatrix und Rauschmodelle

Die Kovarianzmatrix ist ein zentrales mathematisches Werkzeug in der Statistik und Signalverarbeitung, das die Unsicherheiten und Abhängigkeiten zwischen mehreren Zufallsvariablen darstellt. Insbesondere in Anwendungen, wie der Mehrfach-Objekt-Verfolgung und der Sensordatenfusion, spielt die Kovarianzmatrix eine entscheidende Rolle bei der Modellierung von Prozessrauschen und Messrauschen. Durch die Verwendung der Prozessrausch-Kovarianzmatrix Q und der Messrausch-Kovarianzmatrix R wird eine präzise und zuverlässige Zustandsabschätzung ermöglicht. Diese Matrizen sind integraler Bestandteil moderner Filtermethoden. Dieses Kapitel erläutert die Struktur der Kovarianzmatrix und deren Anwendung in den Rauschmodellen. [20][24]

3.6.1 Struktur der Kovarianzmatrix

Die Kovarianzmatrix ist eine quadratische und symmetrische Matrix. Die Struktur einer solchen Kovarianz-Matrix ist in 3.6 dargestellt.

$$\Sigma = \begin{pmatrix} \text{Var}(x_1) & \text{Cov}(x_1, x_2) & \dots & \text{Cov}(x_1, x_n) \\ \text{Cov}(x_2, x_1) & \text{Var}(x_2) & \dots & \text{Cov}(x_2, x_n) \\ \vdots & \vdots & \ddots & \vdots \\ \text{Cov}(x_n, x_1) & \text{Cov}(x_n, x_2) & \dots & \text{Var}(x_n) \end{pmatrix} \tag{3.6}$$

Die Diagonalelemente $Var(x_i)$ einer Kovarianzmatrix repräsentieren die Varianzen der einzelnen Zufallsvariablen und messen deren Streuung um den Mittelwert. Im Gegensatz dazu stellen die Off-Diagonalelemente $Cov(x_i, x_j)$ die Kovarianzen zwischen Paaren von Zufallsvariablen dar und zeigen deren gemeinsame Variation sowie Korrelation auf.

Die Symmetrie der Kovarianzmatrix ($\text{Cov}(x_i, x_j) = \text{Cov}(x_j, x_i)$) spiegelt wieder, dass die Kovarianz zwischen zwei Variablen unabhängig von ihrer Reihenfolge ist. [20]

3.6.2 Prozessrauschen und Messrauschen

In der Zustandsabschätzung und im Tracking sind Prozessrauschen und Messrauschen wesentliche Komponenten, die durch Kovarianzmatrizen modelliert werden. Diese Rauschmodelle beschreiben die Unsicherheiten in den System- und Messmodellen und beeinflussen maßgeblich die Genauigkeit und Zuverlässigkeit von Filtern wie dem Kalman-Filter und dem IMM-Filter.

Prozessrauschen beschreibt die Ungewissheit in der Modellierung der Bewegungsdynamik von Objekten. Physikalische Modelle sind oft vereinfacht und erfassen nicht alle Einflüsse wie externe Kräfte oder Modellierungsfehler. Dieses Rauschen wird durch die Prozessrausch-Kovarianzmatrix Q in 3.7 dargestellt.

$$Q = \begin{pmatrix} q_{11} & q_{12} & \cdots & q_{1n} \\ q_{21} & q_{22} & \cdots & q_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ q_{n1} & q_{n2} & \cdots & q_{nn} \end{pmatrix} \tag{3.7}$$

Q modelliert die Unsicherheit in der Vorhersage des Systemzustands, wie Position, Geschwindigkeit und Orientierung der Objekte. Eine höhere Varianz in Q signalisiert größere Unsicherheiten in den Modellvorhersagen.

Messrauschen modelliert die Unsicherheiten und Fehler in den Sensordaten, verursacht durch Faktoren wie ungenaue Messungen, Sensorauflösung oder Störungen. Dieses Rauschen wird durch die Messrausch-Kovarianzmatrix R in 3.8 dargestellt.

$$R = \begin{pmatrix} r_{11} & r_{12} & \cdots & r_{1m} \\ r_{21} & r_{22} & \cdots & r_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ r_{m1} & r_{m2} & \cdots & r_{mm} \end{pmatrix} \tag{3.8}$$

R erfasst die Unsicherheit in den Messungen, die von den Sensoren erfasst werden. Eine höhere Varianz in R deutet auf größere Unsicherheiten in den Sensormessungen hin. [20]

3.6.3 Anwendung der Kovarianzmatrizen in Filtern

Kovarianzmatrizen Q und R sind essenziell für die Funktionsweise von Zustandsfiltern wie dem KF und dem IMM-Filter. Sie bestimmen, wie stark das System den Modellvorhersagen oder den Sensormessungen vertraut.

Kalman-Filter: Im KF beeinflussen Q und R die Gewichtung zwischen der Vorhersage und der Messung. Eine hohe Unsicherheit in Q führt dazu, dass der Filter stärker auf die Messungen vertraut, während eine hohe Unsicherheit in R das Gegenteil bewirkt.

IMM-Filter: Der IMM-Filter nutzt Q und R für jedes Zustandsmodell individuell, was eine flexible Anpassung an unterschiedliche Bewegungsmuster und Dynamiken der zu verfolgenden Objekte ermöglicht. Die Kovarianzmatrizen beeinflussen die Übergangswahrscheinlichkeiten und die Gewichtung der einzelnen Modelle in der Zustandsschätzung. [20]

4 Grundlagen der Metriken zur Bewertung von MOT

In diesem Teil der Arbeit werden die verschiedenen Metriken zur Bewertung von MOT Algorithmen vorgestellt und erläutert. Zu Beginn werden wesentliche Begriffe definiert, um ein besseres Verständnis für die Bewertungsmethoden zu schaffen. Es wird sowohl auf quantitative Metriken eingegangen, die numerische Maßstäbe für die Genauigkeit und Vollständigkeit der Verfolgung bieten, als auch auf qualitative Metriken, die eine visuelle oder heuristische Bewertung der Tracking-Qualität ermöglichen. Im Fokus steht die GOSPA Metrik, die durch ihre umfassende Berücksichtigung von Fehlerarten und flexibler Gewichtung eine präzise und differenzierte Bewertung der Tracking-Leistung ermöglicht. Diese Metrik wird später in der entwickelten Test-App hauptsächlich verwendet, um die Verfolgungs-Leistung von Sensoren zu bewerten.

4.1 Definition und Bedeutung von Metriken für MOT

Im Kontext von LiDAR und MOT bezeichnet der Begriff „Metrik“ ein quantitatives Maß oder Bewertungsverfahren, das die Genauigkeit und Effizienz von Verfolgungsalgorithmen objektiv bewertet. Dabei wird die Übereinstimmung zwischen erfassten Datenpunkten, Objekten oder Verfolgungspfaden gemessen, um die Qualität der Objektzuordnung und -verfolgung festzustellen.

Metriken spielen eine zentrale Rolle in der Entwicklung und Optimierung von MOT-Systemen, insbesondere bei der Verwendung von LiDAR Daten. Sie ermöglichen einen strukturierten Vergleich verschiedener Tracking-Algorithmen und helfen dabei, deren Stärken und Schwächen zu identifizieren. Geeignete Metriken unterstützen die gezielte Optimierung der Algorithmen, indem sie die Genauigkeit durch präzise Zuordnungs- und Distanzmessungen verbessern und so Mehrdeutigkeiten reduzieren.

Robuste Metriken, die gegenüber Rauschen und unvollständigen Daten unempfindlich sind, tragen zusätzlich zur Zuverlässigkeit von Verfolgungssystemen bei, indem sie die negativen Auswirkungen von Störungen minimieren. In Anwendungen wie der autonomen Fahrzeugnavigation ist eine Echtzeitverarbeitung essenziell, und effiziente Metriken fördern die Echtzeitfähigkeit der Systeme durch schnelle Berechnungen. Einheitliche Metriken unterstützen außerdem den Vergleich und die Standardisierung von Forschungsergebnissen, was die Weiterentwicklung und Integration von Tracking-Systemen in praxisnahe Anwendungen erleichtert. [26][27]

4.2 Grundwahrheiten/ Ground Truths

In der Künstliche Intelligenz (KI)-Modellierung bezeichnet der Begriff „Ground Truth“ reale Daten, die zur Schulung und Evaluierung von Modellen verwendet werden. Ground Truth Daten sind für zahlreiche KI-Anwendungen unerlässlich, beispielsweise für Systeme des autonomen Fahrens. Bei der Modelltestung dienen Ground Truth Daten als Testdatensatz, um die Genauigkeit von Algorithmen zu überprüfen.

Zur Erstellung von Ground Truth Daten ist eine präzise Kennzeichnung erforderlich. Dieser Kennzeichnungsvorgang umfasst die Charakterisierung der Rohdaten entsprechend ihrer semantischen Bedeutung. Die annotierten Ausgaben bilden die Grundlage für das Training überwachter Lernmodelle. Eine hohe Genauigkeit der Kennzeichnung führt direkt zu einer verbesserten Modellgenauigkeit. Allerdings kann die manuelle Annotation von Ground Truth Daten sehr zeitintensiv sein, da moderne KI-Modelle oft Tausende bis Millionen annotierter Datensätze benötigen, um zuverlässige Ergebnisse zu erzielen.

Um den Aufwand der manuellen Kennzeichnung zu reduzieren, bieten verschiedene Labeler-Apps von MATLAB voll- oder halbautomatische Annotierungsverfahren an. Diese Tools ermöglichen eine effizientere Datenkennzeichnung und verringern somit den zeitlichen Aufwand im Vergleich zur manuellen Methode erheblich.

Ein spezifisches Beispiel für ein solches Tool ist der LiDAR Labeler von MATLAB. Dieser erstellt Rahmen um 3D-Objekte und integriert Automatisierungstechniken wie das Clustern von Datenpunkten, das Entfernen der Grundfläche sowie das Nachverfolgen von Punktwolkendaten. Durch diese Funktionen unterstützt der LiDAR Labeler eine schnellere und präzisere Annotation von LiDAR-Daten.

Bei der **manuellen Annotation** werden Objekte in LiDAR-Daten durch Experten händisch markiert und verfolgt. Dies erfolgt üblicherweise durch das Zeichnen von Begrenzungsrahmen oder die manuelle Identifizierung relevanter Punkte in der Punktwolke. Diese Methode liefert präzise Ground Truth Daten, ist jedoch zeitaufwendig und anfällig für menschliche Fehler. Sie wird häufig verwendet, um qualitativ hochwertige Trainings- und Validierungsdatensätze für Tracking-Algorithmen zu erstellen.

Die **automatische Generierung** von Ground Truth Daten kann durch den Einsatz zusätzlicher Sensoren wie Kameras, RADAR oder Global Positioning System (GPS) erfolgen. Diese Sensoren bieten zusätzliche Informationen, um die Position und Bewegung von Objekten genauer zu erfassen. Durch die Fusion der Daten verschiedener Sensoren wird eine präzisere Grundwahrheit erreicht. Diese Methode ermöglicht eine schnelle und effiziente Erstellung von Ground Truth Daten, erfordert jedoch eine sorgfältige Sensor-Kalibrierung und Synchronisation.

In **simulierten oder virtuellen** Umgebungen werden synthetische LiDAR Daten generiert, bei denen die exakten Positionen und Bewegungen der Objekte bekannt sind. Tools wie Car Log Analyser (CARLA) oder „Gazebo“ bieten die Möglichkeit, realitätsnahe Szenarien zu erstellen, in denen eine vollständige Kontrolle über die Umgebung und die Objekte

besteht. Dies ermöglicht präzise und konsistente Ground Truth Daten, jedoch können die Ergebnisse aufgrund der eingeschränkten Realitätsnähe nicht immer direkt auf reale Anwendungen übertragen werden. [28]

4.3 Detektionsmetriken und Assoziationsmetriken

Bei der Bewertung von Objekt-Verfolgungs-Systemen ist es wichtig, sowohl die Detektion als auch die Assoziation von Objekten über die Zeit hinweg zu berücksichtigen. Ähnliche Kennzahlen werden in beiden dieser Metrikkategorien verwendet, um Leistungswerte für die jeweilige Anwendung bereitzustellen. In den folgenden Abschnitten werden die Detektionsmetriken, Assoziationsmetriken und deren Anwendung mit statistischen Kennzahlen näher erläutert.

Detektionsmetriken bewerten die Fähigkeit eines Systems, Objekte in einzelnen Bildern korrekt zu erkennen. Sie messen, wie gut das System Objekte identifiziert und klassifiziert.

Assoziationsmetriken hingegen, beurteilen die Fähigkeit eines Systems, Objekte über mehrere Bilder hinweg korrekt zu verfolgen und zuzuordnen. Diese Metriken messen, wie gut das System in der Lage ist, die Identität von Objekten durch den Verlauf einer Szene oder Sequenz aufrechtzuerhalten. Niedrige Werte in diesen Metriken zeigen an, dass das System effektiv in der Lage ist, Objekte konsistent über Zeiträume hinweg zu verfolgen und Fehler bei der Zuordnung minimal sind. Im Allgemeinen ähneln sich die beiden Metriken und ihre Kennzahlen stark. Jedoch beziehen sie sich auf unterschiedliche Aspekte des Verfolgungs-Prozesses. Zum weiteren Verständnis werden die Kennzahlen im folgenden Kapitel näher beleuchtet. [29]

4.3.1 Statistische Kennzahlen

Bei der Bewertung kommen statistische Kennzahlen zum Einsatz. Kennzahlen mit dem Präfix „ID“ beziehen sich auf die Assoziation und Identität von Objekten im Tracking. Sie messen, wie konsistent das System die Identität von Objekten über die Zeit hinweg hält und ob Fehler bei der Zuordnung auftreten. Kennzahlen ohne „ID“ beziehen sich auf die Detektion von Objekten in einzelnen Frames. Sie bewerten die Objekterkennung unabhängig von der Identität über mehrere Frames hinweg. Tabelle 4.1 zeigt die Kennzahlen und ihre zugehörigen Bedeutungen. [29]

Tabelle 4.1. Vergleich der Kennzahlen für Detektions- und Assoziationsmetriken [30][29]

Kennzahl	Detektion	Assoziation
True Positive	**TP**: Korrekt erkannte Objekte.	**IDTP**: Korrekt verfolgte Assoziationen.
False Positive	**FP**: Fälschlicherweise erkannte Objekte. Hohe FP beeinträchtigt die Zuverlässigkeit.	**IDFP**: Fälschlicherweise verknüpfte Assoziationen. Hohe IDFP mindert die Verfolgungsgenauigkeit.
False Negative	**FN**: Übersehene echte Objekte. Hohe FN zeigt Erkennungsprobleme.	**IDFN**: Verpasste oder falsch verfolgte Assoziationen. Hohe IDFN erschwert die konsistente Verfolgung.
True Negative	**TN**: Korrekt als nicht vorhanden erkannte Objekte. Weniger relevant im Tracking.	**Nicht anwendbar**: Keine direkte Entsprechung für Assoziationen.
Identitätenwechsel	**Nicht anwendbar**: IDSW betrifft die Detektion nicht.	**IDSW**: Anzahl der Identitätswechsel. Hohe IDSW zeigt Konsistenzprobleme.

Abbildung 4.1 zeigt eine schematische Darstellung eines Detektions- und Trackingszenarios, um die Bedeutung der jeweiligen Kennzahlen zu veranschaulichen. Im ersten Abschnitt der Darstellung werden 12 Detektionen (a) präsentiert, von denen 5 mit den insgesamt 7 vorhandenen Ground Truths (A) übereinstimmen. Daraus resultieren 5 True Positives (TP), 2 False Negatives (FN) und 7 False Positives (FP) für die Detektion. Im zweiten Abschnitt wird der Versuch eines Tracks (I) dargestellt, der mithilfe der Detektionen (a) ein Objekt verfolgt. Dabei trifft der Tracker in 5 Fällen auf die Ground Truths (A), was zu 5 TP, 2 FN und 2 FP führt.

Diese Darstellung verdeutlicht die Rolle und das Zusammenspiel der Kennzahlen TP, FN und FP sowohl für die Detektions- als auch für die Tracking-Performance.

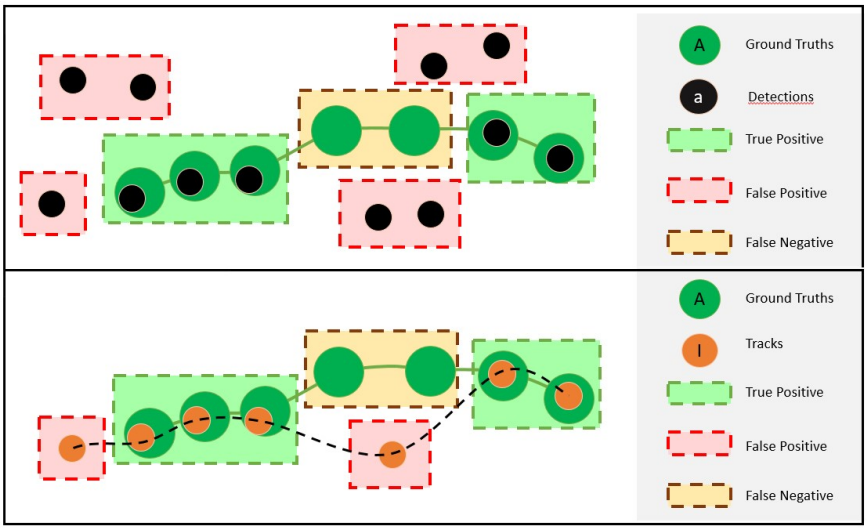


Abbildung 4.1. Vereinfachtes Detektions- und Trackingszenario

4.3.2 Recall und ID-Recall

Detection Recall (Recall) ist der Anteil der korrekt identifizierten Objekte an der Gesamtanzahl der tatsächlich vorhandenen Objekte. Ein hoher Detection-Recall zeigt an, dass der Algorithmus in der Lage ist, die meisten der tatsächlichen Objekte zu detektieren.

Association Recall (ID-Recall) misst den Anteil der korrekt erkannten Assoziationen zwischen Objekten an der Gesamtanzahl der tatsächlich vorhandenen Assoziationen. Ein hoher Tracking-Recall bedeutet, dass das Tracking-System in der Lage ist, einen großen Teil der tatsächlichen Objektverbindungen über die Zeit zu erkennen und zu verfolgen. Recall und ID-Recall lassen durch Gleichung 4.1 und 4.2 berechnen. [31]

$$\text{Recall} = \frac{|TP|}{|TP| + |FN|} \tag{4.1}$$

$$\text{ID-Recall} = \frac{|IDTP|}{|IDTP| + |IDFN|} \tag{4.2}$$

4.3.3 Precision und ID-Precision

Detection Precision (Precision) misst den Anteil der tatsächlich vorhandenen Objekte, die korrekt detektiert wurden. Eine hohe Precision weist darauf hin, das Objekte korrekt identifiziert werden und wenig Fehldetektionen erfolgen.

Association Precision (ID-Precision) misst den Anteil der korrekt erkannten Assoziationen im Verhältnis zu allen als Assoziationen klassifizierten Ereignissen. Eine hohe ID-Precision deutet darauf hin, dass wenige falsche Assoziationen produziert werden und

erkannte Verbindungen über die Zeit hinweg präzise und korrekt sind. Die Kennzahlen berechnen sich über Gleichung 4.4 und 4.3. [31]

$$\text{Precision} = \frac{|TP|}{|TP| + |FP|} \tag{4.3}$$

$$\text{ID-Precision} = \frac{|IDTP|}{|IDTP| + |IDFP|} \tag{4.4}$$

4.3.4 Accuracy

Accuracy zeigt den Anteil der korrekt identifizierten Objekte an der Gesamtanzahl der identifizierten Objekte. Ein hoher Accuracy Wert bedeutet, dass wenige Fehler sowohl bei der Erkennung als auch bei der Nicht-Erkennung von Objekten entstehen. Im Tracking-Kontext wäre eine Metrik wie Accuracy nicht besonders aussagekräftig. Der Hauptgrund dafür ist, dass diese Metrik nicht direkt die spezifischen Herausforderungen des Trackings adressiert. Accuracy ist beschrieben durch Gleichung 4.5.

$$\text{Accuracy} = \frac{|TP| + |TN|}{|TP| + |TN| + |FP| + |FN|} \tag{4.5}$$

Der nähere Zusammenhang der beiden Metriken Precision und Accuracy ist in Abbildung 4.2 schematisch dargestellt. [30][31]

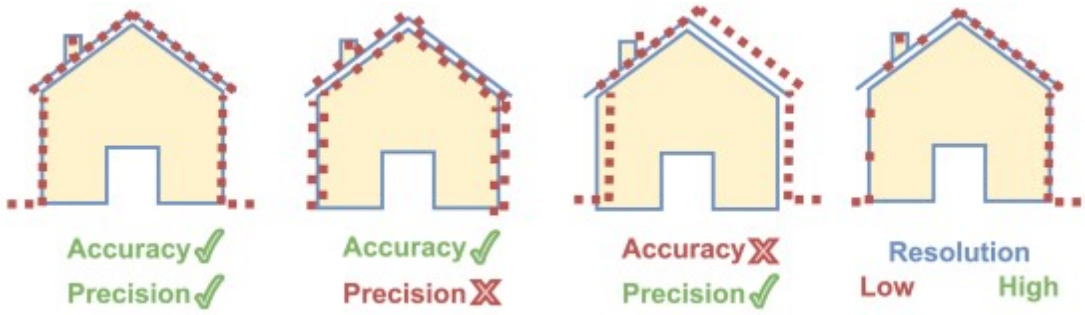


Abbildung 4.2. Zusammenhang von Precision und Accuracy [32]

4.3.5 F1-Score und ID-F1-Score

Detection F1-Score (F1) ist das harmonische Mittel von Detection Precision und Detection Recall.

Association F1-Score (IDF1) ist das harmonische Mittel von Association Precision und Association Recall. Beide Metriken können durch Gleichung 4.6 und 4.7 beschrieben werden. [30][31]

$$\text{F1} = \frac{|TP|}{|TP| + 0.5 \cdot |FN| + 0.5 \cdot |FP|} \tag{4.6}$$

$$\text{IDF1} = \frac{|IDTP|}{|IDTP| + 0.5 \cdot |IDFN| + 0.5 \cdot |IDFP|} \tag{4.7}$$

4.4 Klassische Metriken

Klassische Metriken konzentrieren sich auf die direkte Quantifizierung der Leistung eines Tracking-Algorithmus. Die erweiterten Metriken wie Multi Object Tracking Accuracy (MOTA), Multi Object Tracking Precision (MOTP) und Higher Order Tracking Accuracy (HOTA) kombinieren Aspekte der Detektionsmetriken als auch der Assoziationsmetriken, um eine umfassendere Bewertung der Tracking-Performance zu ermöglichen. Im weiteren Verlauf werden diese klassische Metriken zur Objektverfolgung erläutert.

4.4.1 MOTP Metrik

Die **MOTP** Metrik misst die Genauigkeit der Objektverfolgung, indem sie den Mittelwert der Distanz zwischen den vorhergesagten Positionen und den tatsächlichen Positionen von Objekten berechnet. Im Gegensatz zu MOTA, die Fehler bei der Erkennung und Identifikation von Objekten berücksichtigt, konzentriert sich MOTP auf die räumliche Genauigkeit der Verfolgung. MOTP wird berechnet mit der Gleichung 4.8.

$$\text{MOTP} = \frac{\sum_{t,i} d_{t,i}}{\sum_t c_t} \tag{4.8}$$

Hierbei ist c_t die Anzahl der korrekt zugeordneten Hypothesen in Zeitpunkt t. Die Überlappung des Begrenzungsrahmens des Ziels i mit dem dazugehörigen Ground Truth Objekt wird durch $d_{t,i}$ dargestellt. MOTP gibt die durchschnittliche Überlappung zwischen korrekt zugeordneten Hypothesen (Tracks) und Objekten (Truths) an. Sie misst die Lokalisierungspräzision, welche in der Praxis hauptsächlich die Lokalisierungsgenauigkeit des Detektors quantifiziert. MOTP bietet daher begrenzte Informationen über die tatsächliche Leistung des Trackers. [29]

4.4.2 MOTA Metrik

Die **MOTA** Metrik bietet eine umfassende Bewertung der Genauigkeit eines Tracking-Systems, indem sie die Anzahl der Fehler in Bezug auf FP, FN und Identitätswechsel ID Switches (IDSW) zusammenfasst. Auch die Gesamtanzahl der Ground Truths $gtDet$ wird berücksichtigt. MOTA kann mit der Gleichung 4.9 mathematisch beschrieben werden. Es

integriert sowohl Kennzahlen für Detektionsmetriken (FP und FN) als auch eine Kennzahl der Assoziationsmetrik (IDSW). [30][29]

$$\text{MOTA} = 1 - \frac{|FN| + |FP| + |IDSW|}{|gtDet|} \tag{4.9}$$

4.4.3 HOTA Metrik

HOTA baut auf der bestehenden und zuvor benannten MOTA Metrik auf, adressiert jedoch deren Schwächen. Das Hauptziel von HOTA ist es, eine umfassende Bewertung für Tracker bereitzustellen, die verschiedene Aspekte der Verfolgungsleistung fair kombiniert. Es evaluiert langfristige Verfolgungsassoziationen und zerlegt sich in Teilmetriken, die eine detaillierte Analyse der verschiedenen Komponenten der Verfolgungsleistung ermöglichen.

Die Definition von HOTA umfasst das Matching von Detektionen, ähnlich wie bei MOTA. Der HOTA-Score wird durch eine „doppelte Jaccard"-Formulierung berechnet, bei der Detektionskonzepte mit Assoziationskonzepten gewichtet werden.

Für dieses Konzept werden neue Metriken preDet und preID sowie FPA, FNA und TPA eingeführt welche in Tabelle 4.2 aufgeführt sind.

Tabelle 4.2. Beschreibung der Begriffe prDet, prID und der Assoziationsmetriken TPA, FNA und FPA

Begriff	**Beschreibung**
prDet (Predicted Detection)	Ein vorhergesagtes Objekt, das vom Tracking-Algorithmus erkannt wurde.
prID (Predicted ID)	Die vom Algorithmus zugewiesene Identität für eine prDet.
TPA (True Positive Associations)	Menge der korrekten Assoziationen für einen gegebenen True Positive (TP). Ein TPA liegt vor, wenn sowohl die Ground Truth ID (gtID) als auch die vorhergesagte ID (prID) des TPs übereinstimmen.
FNA (False Negative Associations)	Assoziationen, die für einen gegebenen TP fehlen. Eine FNA tritt auf, wenn ein Ground Truth Objekt zwar eine gtID hat, aber entweder eine falsche prID zugewiesen wurde oder gar keine Zuweisung erhalten hat (also „verfehlt" wurde).
FPA (False Positive Associations)	Fälschlicherweise erkannte Assoziationen für einen gegebenen TP. Eine FPA tritt auf, wenn ein vorhergesagtes Objekt (prDet) eine prID erhält, die entweder einer falschen gtID zugewiesen wird oder gar keinem echten Objekt entspricht.

Der Assoziationsscore A(c) berechnet sich durch Anwendung der Gleichung 4.10.

$$A(c) = \frac{|TPA(c)|}{|TPA(c)| + |FNA(c)| + |FPA(c)|} \tag{4.10}$$

A misst die Übereinstimmung zwischen der Ground-Truth-Trajektorie (gtTraj) und der vorhergesagten Trajektorie (prTraj), die bei dem TP_c übereinstimmen.

Die Geichung 4.11 für HOTA aggregiert den Assoziationsscore A(c) über alle TPs und normalisiert ihn durch die Gesamtanzahl der TPs, FNs und FPs, wodurch eine Metrik für die Gesamt-Tracking-Genauigkeit entsteht. Der Schwellenwert α in HOTA legt die Genauigkeitsschwelle fest, ab der eine Detektion als korrekt gewertet wird, und beeinflusst somit die Strenge der Metrik bei der Bewertung von Tracking-Genauigkeit. [30]

$$\text{HOTA}_\alpha = \frac{\sum_{c \in \text{TP}} A(c)}{|TP| + |FN| + |FP|} \tag{4.11}$$

4.5 Erweiterte Metriken

Erweiterte fortgeschrittene Metriken ergänzen die klassischen Bewertungsansätze für Tracking-Systeme, indem sie komplexere Aspekte der Tracking-Qualität berücksichtigen und speziellere Anforderungen adressieren. Diese Metriken bieten eine detailliertere Analyse der Tracking-Leistung und sind besonders nützlich in anspruchsvollen Szenarien. Die Optimal Subpattern Assignment (OSPA) und GOSPA Metrik gehören zu den komplexeren erweiterten Metriken, welche in MATLAB implementierbar sind. Im weiteren Verlauf der Arbeit wird die Funktionsweise und Berechnung der Metriken erläutert.

4.5.1 OSPA Metrik

Die OSPA Metrik wurde entwickelt, um die Leistung von MOT Systemen zu bewerten. Sie misst, wie gut ein Tracking-Algorithmus die Positionen mehrerer Objekte in einem Szenario schätzen kann, indem sie sowohl Positionsfehler als auch Kardinalfehler (d.h. Unterschiede in der Anzahl der verfolgten Objekte) berücksichtigt.

Die allgemeine Gleichung der OSPA Metrik ist in 4.12 definiert.

$$d_{OSPA} = \left(\frac{1}{\max(|X|, |Y|)} \left(\min_{\pi \in \Pi_{|Y|}} \sum_{i=1}^{|X|} d_c(x_i, y_{\pi(i)})^p + c^p \cdot (|Y| - |X|) \right) \right)^{\frac{1}{p}} \tag{4.12}$$

Die Parameter der OSPA Metrik sind wie folgt definiert:

X und Y repräsentieren die geschätzten und tatsächlichen Positionen der Objekte. π ist die Permutation, die die Objekte in den beiden Mengen optimal zuordnet, um den Fehler

zu minimieren. Durch das Anlegen einer Distanzmatrix $d_c(x_i, y_{\pi(i)})$ wird die Distanz zwischen einem geschätzten Objekt (Track) x_i und einem tatsächlichen Objekt (Ground Truth) $y_{\pi(i)}$ beschrieben. Diese Distanz ist durch den Cutoff-Parameter c begrenzt. Distanzen, die den Schwellenwert c überschreiten, werden auf c begrenzt. Der Cutoff-Parameter c legt also den maximalen Fehler fest, der für jede Distanz berücksichtigt wird. Der Parameter p steuert, wie stark Fehler gewichtet werden. Ein größerer Wert von p führt zu einer stärkeren Bestrafung von großen Fehlern (Ausreißern). Schließlich beschreiben $|X|$ und $|Y|$ die Kardinalität der Mengen X und Y, also die Anzahl der Objekte in den jeweiligen Mengen.

In der OSPA Metrik werden folgende **Fehlerkomponenten** berücksichtigt:

1. **Zuweisungsfehler (Positionsfehler)**: Der erste Teil 4.13 der Gleichung 4.12 sucht die beste Zuordnung der geschätzten Objekte zu den tatsächlichen Objekten, um den Gesamtabstand (Fehler) zwischen ihnen zu minimieren.

$$\min_{\pi \in \Pi_{|Y|}} \sum_{i=1}^{|X|} d_c(x_i, y_{\pi(i)})^p \tag{4.13}$$

Dieser Fehler wird durch den Cutoff-Parameter c begrenzt und durch Parameter p gewichtet. c verhindert, dass extrem große Distanzen die Metrik übermäßig beeinflussen. Bei einem größeren Wert von p werden größere Distanzen stärker bestraft und der Gesamtfehler wird empfindlicher gegenüber großen Abweichungen während ein kleinerer Wert von p größere Distanzen weniger stark gewichtet.

2. **Kardinalitätsfehler**:

Der Ausdruck 4.14 in Gleichung 4.12 stellt den Kardinalitätsfehler dar.

$$c^p \cdot (|Y| - |X|) \tag{4.14}$$

Wenn die Anzahl der geschätzten Objekte $|X|$ von der Anzahl der tatsächlichen Objekte $|Y|$ abweicht, wird ein zusätzlicher Fehler durch diese Differenz eingeführt. Der Exponent p hebt den Cutoff-Wert c auf eine Potenz, um den Einfluss der Kardinalitätsabweichung auf die Gesamtfehlerbewertung zu gewichten. Ein größerer Wert von p führt zu einer stärkeren Gewichtung der Kardinalitätsabweichung, indem der Fehler aufgrund der Differenz in der Anzahl der Objekte exponentiell erhöht wird.

3. **Normalisierung**: Die OSPA Metrik teilt den Gesamtfehler durch die maximale Anzahl der Objekte $\max(|X|, |Y|)$. Die Normalisierung ist als Term 4.15 in der Gleichung 4.12 wiederzufinden. Dies sorgt dafür, dass der Fehler pro Objekt berechnet wird.

$$\frac{1}{\max(|X|, |Y|)} \tag{4.15}$$

4. **Aggregation**: Schließlich wird der gesamte Fehler durch den Exponenten $\frac{1}{p}$ skaliert. Der Term 4.16 in der Gleichung 4.12 stellt diesen Zusammenhang dar. Durch die Aggregation

wird die Gewichtung großer Fehler beeinflusst.

$$(\cdots)^{\frac{1}{p}} \tag{4.16}$$

Durch die Berücksichtigung dieser Fehlerkomponenten kombiniert die OSPA Metrik zwei wichtige Aspekte eines MOT-Systems: die Genauigkeit der Positionen sowie die Korrektheit der Anzahl der verfolgten Objekte. Die Metrik gibt einen einzigen Fehlerwert zurück, der die Gesamtperformance des Systems darstellt. [30]

4.5.2 GOSPA Metrik

Die GOSPA erweitert die OSPA Metrik und bietet zusätzliche Flexibilität in der Bewertung von MOT Systemen. Die Hauptunterschiede zur OSPA Metrik sind die Einführung des α-Parameters und die Reduktion der Normalisierung, was eine präzisere Steuerung des Kardinalitätsfehlers ermöglicht.

Die GOSPA Metrik wird durch Gleichung 4.17 definiert.

$$d_{GOSPA} = \left(\min_{\pi \in \Pi_{|Y|}} \sum_{i=1}^{|X|} d_c(x_i, y_{\pi(i)})^p + \frac{c^p}{\alpha} \cdot (|Y| - |X|) \right)^{\frac{1}{p}} \tag{4.17}$$

Die GOSPA Metrik weißt dabei folgende Unterschiede zur OSPA Metrik auf:

1. **Kein Normalisierungsfaktor**: Im Gegensatz zur OSPA Metrik wird der Fehler in GOSPA **nicht** durch die Anzahl der Objekte geteilt. Dies bedeutet, dass der gesamte Fehler unabhängig von der Anzahl der Objekte bewertet wird, was in vielen Szenarien als realistischer angesehen wird.
2. α**-Parameter**: Dieser Parameter steuert, wie stark der Fehler durch eine falsche Anzahl von Objekten (Kardinalitätsfehler) gewichtet wird. Durch das Einführen des Faktors α kann man feiner justieren, wie wichtig die Anzahl der Objekte im Vergleich zur Positionsgenauigkeit ist:

Der Fehler, der durch eine unterschiedliche Anzahl von Objekten entsteht (Kardinalfehler), wird durch den Term 4.18 in Gleichung 4.17 dargestellt.

$$c^p \cdot \alpha \cdot (|Y| - |X|) \tag{4.18}$$

Hier ermöglicht der Parameter α eine feinere Steuerung dieses Fehlers. Ein kleinerer Wert von α erhöht das Gewicht des Kardinalitätsfehlers, während ein größerer Wert ihn verringert. Bei $\alpha = 1$ entspricht die GOSPA Metrik der OSPA Metrik (ohne Normalisierung).

GOSPA erweitert die OSPA Metrik durch die Möglichkeit, die Gewichtung zwischen Po-

sitionsfehlern und Kardinalitätsfehlern besser zu steuern. Mittels der Reduktion der Normalisierung durch die Anzahl der Objekte, wird der Fehler unabhängig von der Anzahl der Objekte bewertet. Der zusätzliche Parameter α erlaubt eine präzisere Anpassung des Kardinalitätsfehlers, sodass die Metrik auf verschiedene Tracking-Szenarien abgestimmt werden kann. Insgesamt ist die GOSPA Metrik eine allgemeinere und flexiblere Bewertungsmethode als die OSPA Metrik, die insbesondere für komplexere MOT Szenarien geeignet ist, bei denen sowohl die Positionsgenauigkeit als auch die richtige Anzahl von Objekten wichtig sind. [30]

5 Konzeption einer Test-App in MATLAB

Die Test-App wurde mithilfe des MATLAB-App-Designers entwickelt. Dieser Entwicklungsrahmen ermöglicht eine flexible, benutzerfreundliche Benutzeroberfläche sowie eine nahtlose Integration von MATLAB-Funktionen zur Datenverarbeitung und Analyse. Die App ist modular aufgebaut und umfasst mehrere spezialisierte Tabs, die jeweils spezifische Funktionen zur Dateneingabe, Modellparametrierung, Visualisierung und Analyse der Tracking-Ergebnisse bereitstellen. In den folgenden Kapiteln wird der Grundaufbau der Applikation sowie die Implementierung der Funktionalitäten erläutert.

5.1 Aufbau der App mit MATLAB-App-Designer

Die Applikation nutzt als Eingabedaten sowohl LiDAR-Messungen als auch Grundwahrheitsdaten, die innerhalb der Punktwolken der Messungen erstellt wurden. Ein im Code ausgelagertes Detektormodell wird verwendet, um Punktwolken zu segmentieren, Objekte zu erkennen und Begrenzungsrahmen um diese Objekte zu generieren. Anschließend wird ein in die Applikation integrierter JPDA-Tracker eingesetzt, der auf einen im Code ausgelagerten IMM-Filter zugreift. Dieser IMM-Filter generiert Zustandsschätzungen basierend auf den erkannten Detektionen und übermittelt diese an den Tracker. Darüber hinaus erfolgt eine Filterung nach den für die Analyse relevanten Objektmaßen. Der Tracker verfolgt die Objekte anhand der Zustandsschätzungen. Abschließend werden die Ergebnisse des Trackers sowie die vom Benutzer eingegebenen Grundwahrheitsdaten zur Durchführung einer Analyse unter Verwendung der GOSPA-Metrik und zusätzlich auswertender Abbildungen herangezogen.

Der schematische Aufbau der entwickelten MATLAB-App ist in Abbildung 5.1 dargestellt. In Anlage A des Anlagenverzeichnisses sind Abbildungen der Benutzeroberfläche der Applikation aufgeführt.

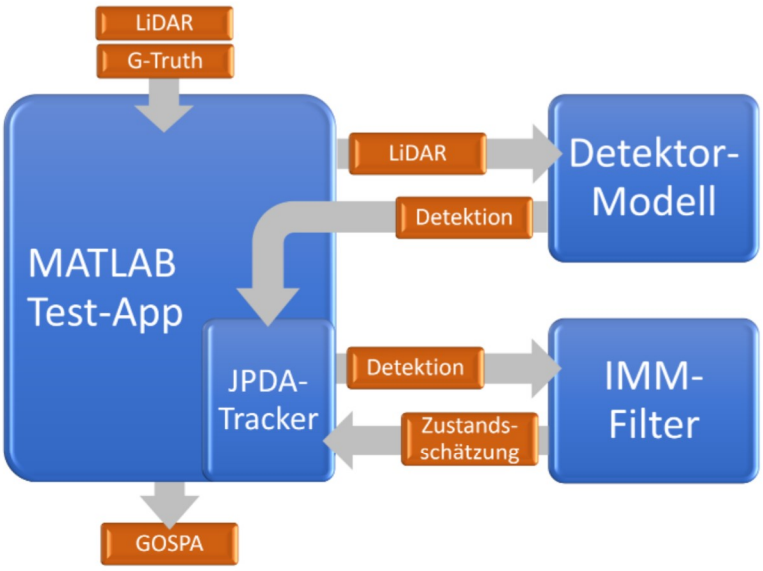


Abbildung 5.1. Schematischer Aufbau der entwickelten App

5.1.1 Dateneingabe

Der *Data Input Tab* dient der Eingabe und Überprüfung der Daten die benötigt werden. Der Benutzer hat die Möglichkeit Punktwolkendaten im `.pcd`-Format sowie Grundwahrheitsdaten als `.mat`-Datei zu importieren. Die LiDAR-Daten müssen aus einer Sequenz von Dateien innerhalb eines Ordners abgelegt sein. Diese müssen in einer logischen Reihenfolge organisiert sein, um eine korrekte Einlesung zu gewährleisten. Das kann beispielsweise durch eine fortlaufende Nummerierung (z.B. `0001.pcd`, `0002.pcd`) oder durch die Verwendung von Zeitstempeln im Dateinamen (z.B. `15-23-21.pcd`) erfolgen.

Die Grundwahrheitsdaten werden als `.mat`-Datei importiert und müssen ein MATLAB-Objekt vom Typ `groundTruthLidar` enthalten. Diese Datei dient als Referenz, um die Tracking-Ergebnisse mit den tatsächlichen Objektpositionen zu vergleichen. Eine wesentliche Voraussetzung für die korrekte Verarbeitung der Eingabedaten ist die Übereinstimmung der Sequenzlänge der Punktwolken mit der Anzahl der Reihen im `groundTruthLidar`-Objekt. Für jeden Punktwolken-Frame ist eine entsprechende Zeile im `groundTruthLidar`-Objekt erforderlich. Eine Leerzeile ist jedoch zulässig, wenn für ein bestimmten Frame keine Grundwahrheitsdaten verfügbar sind.

Die Grundwahrheitsdaten der Objekte werden in der App-Visualisierung mit Quadern dargestellt. Abbildung 5.2 veranschaulicht die Darstellung.

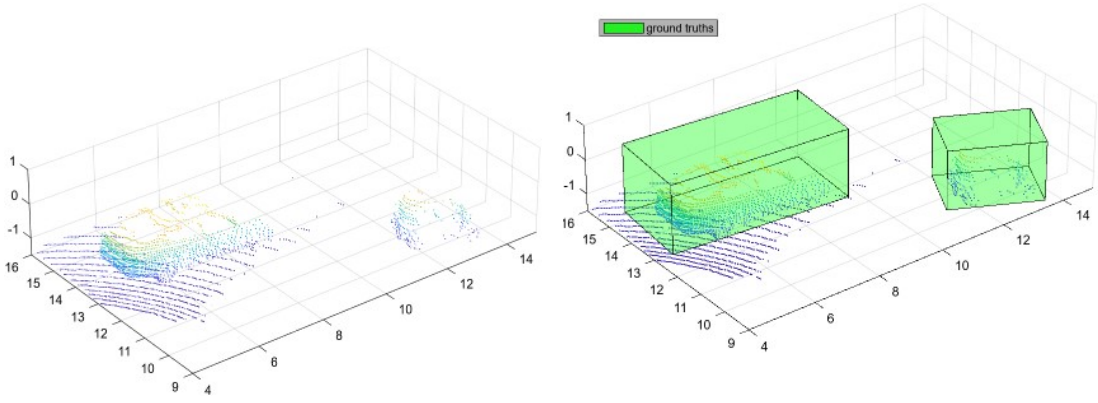


Abbildung 5.2. Vergleich zwischen Referenzpunktwolke und Grundwahrheiten

Des Weiteren bietet der Datenimport-Tab eine Fehlerkonsole, die potenzielle Fehler sowohl im Datenimportprozess, als auch während der Durchführung der Analyse erfasst und protokolliert. Über diese Konsole kann der Benutzer die Validität und Konsistenz der Daten überprüfen, da die Applikation vor dem eigentlichen Laden sicherstellt, dass die Dateitypen korrekt sind und die Datenmengen übereinstimmen. Dadurch wird verhindert, dass fehlerhafte oder unvollständige Datensätze in den Verarbeitungsprozess gelangen und die Analyse verfälschen. Bei erfolgreicher Importierung konsistenter und valider Daten liefert die Konsole eine entsprechende Meldung an den Benutzer.

In Tabelle 5.1 sind die Grundanforderungen an die Datensätze zusammengefasst, die zur Nutzung der App erforderlich sind.

Tabelle 5.1. Anforderungen der Datensätze für die App

Anforderung	**Datensatz 1: LiDAR-Messung**	**Datensatz 2: Ground Truth Daten**
Dateiformat	.pcd	.mat (MATLAB-Datei)
Dateistruktur	Ordner mit beliebig vielen Punktwolkendateien	Enthält ein MATLAB-Objekt vom Typ `groundTruthLidar`
Benennung der Dateien	Klare Nummerierung (z.B. 0001.pcd, 0002.pcd, ...) oder festgelegte Zeitstempel (z.B. LiDAR_2023-10-09_15-23-21.pcd)	Nicht relevant
Anzahl der Dateien/Reihen	Anzahl der Punktwolkendateien muss mit der Anzahl der Reihen im Ground Truth-Objekt übereinstimmen	Jede Punktwolkendatei muss eine entsprechende Zeile im Ground Truth-Objekt haben. Leere Zeilen sind erlaubt.

5.1.2 MOT-Tab

Der *MOT Tab* ist für die Visualisierung der Punktwolkendaten und die Ausführung des Detektor- und Trackermodells zuständig. Er bietet Funktionen zur Festlegung und Anpas-

sung einer Region of Interest (ROI), um die Verarbeitung der Punktwolke auf einen bestimmten Bereich zu beschränken. Der Benutzer kann außerdem die Achsenlimits der Punktwolke individuell festlegen und eine Animation der Punktwolkensequenz anzeigen lassen, welche die Ergebnisse des Detektor- und Trackermodells dynamisch darstellt. Die Frames der LiDAR-Messung können beliebig durch ein Slider-Element im unteren Bereich der App ausgewählt werden. Zusätzlich wird die Anzahl der Detektionen, Tracks und Grundwahrheiten in einem Balkendiagramm angezeigt, was eine schnelle Einschätzung der Verfolgungsgenauigkeit und -stabilität ermöglicht.

Die Visualisierung der Punktwolke erfolgt innerhalb von zwei Fenstern: dem „Pointcloud View“ und dem „Bird view“. Eine beispielhafte Visualisierung einer Punktwolke mit Trackingergebnissen sowie das Balkendiagramm zur Schnellanalyse ist in Anlage A dargestellt.

5.1.3 Settings-Tab

Der *Settings Tab* bietet detaillierte Konfigurationsoptionen für das Detektor- und Trackermodell. Hier können Parameter wie beispielsweise die Grenzwerte für die Segmentierung der Punktwolke oder die Zuweisung von Detektionen zu Tracks angepasst werden, um die Ergebnisse zu optimieren und an verschiedene Umgebungen oder Szenarien anzupassen. Diese Konfigurationsmöglichkeiten realisieren eine flexible Anpassung der Modelle und gewährleisten eine optimale Leistung in dynamischen Szenen.

5.1.4 Metric Analysis Tab

Der *Metric Analysis Tab* ist für die Berechnung und Analyse der GOSPA- und OSPA-Metriken zuständig. Diese Metriken liefern quantitative Informationen über die Genauigkeit und Effizienz des Tracking-Systems. Ferner werden auch die Komponenten-Kurven der Metriken visualisiert, um die Zusammensetzung des Metrik-Kostenwerten besser nachvollziehen zu können. Die Metriken werden innerhalb von Achsen-Fenstern angezeigt, um eine schnelle Bewertung der Trackingleistung vorzunehmen. Datenpunkte, die aufgrund fehlender Grundwahrheitsdaten nicht berechenbar sind, werden in der Graphendarstellung gekennzeichnet. Hierbei sind Anpassungen der Parameter *cutoff-distance* und *switching-penalty* möglich um die Metrik zu justieren. Der Benutzer kann die berechneten Metrikwerte als *.mat*-Datei speichern, um die Tracking-Performance objektiv zu bewerten und eine weiterführende Analyse der Ergebnisse vorzunehmen (vgl. Anlage A-4).

5.2 Implementierung der Detektionsmethode

Das Detektor-Modell verwendet die `pcsegdist`-Funktion, welche ein Clustering-Verfahren auf Basis der euklidischen Distanz implementiert, um Objekte in Punktwolken zu segmentieren. Die Wahl fiel auf das euklidische Clustering, da es sich durch Einfachheit und Effizi-

enz auszeichnet, insbesondere bei der Segmentierung von zusammenhängenden Strukturen in LiDAR-Punktwolken. Diese Methode ermöglicht es, Objekte wie Fahrzeuge, Fußgänger oder Hindernisse präzise zu trennen, indem sie Punkte basierend auf ihrer räumlichen Nähe zueinander gruppiert. Durch die Verwendung der euklidischen Distanz können Objekte, die klar voneinander abgegrenzt sind, zuverlässig unterschieden werden, was für eine exakte Umgebungserfassung unerlässlich ist. Somit eignet sich dieses Verfahren besonders gut für Anwendungen in der Umfeldwahrnehmung, bei denen die genaue Trennung und Erkennung von Objekten entscheidend ist.

Zur Detektion der Objekte wurde ein Skript erstellt, welches eine Reihe von Eigenschaften definiert um das Verhalten des Detektionsprozesses festzulegen. Das Prinzip der Detetktionsmethode wird in den folgenden Kapiteln erläutert.

5.2.1 Zuschnitt der Punktwolke

Im ersten Schritt wird die Punktwolke basierend auf den definierten X-, Y- und Z-Grenzen der Szene zugeschnitten. Dieser Schritt wird von der Funktion „cropPointCloud“ übernommen, die die Punkte außerhalb der festgelegten Limits verwirft und nur die relevanten Bereiche der Szene zur weiteren Verarbeitung überlässt. Darüber hinaus wird ein Radius um das Ego-Fahrzeug festgelegt, innerhalb dessen ebenfalls keine Objekte detektiert werden. Dadurch werden alle Punkte, die innerhalb dieses Radius liegen, entfernt, um eine klarere Detektion externer Objekte zu ermöglichen. Die vom Benutzer festgelegte Detektor-ROI im Settings-Tab wird an die Funktion übergeben um die Segmentierung auf einen gewünschten Bereich zu reduzieren. [25]

5.2.2 Segmentierung des Bodens

Nachdem die Punktwolke auf den interessierenden Bereich zugeschnitten wurde, erfolgt die Segmentierung des Bodens. Die Funktion „removeGroundPlane“ analysiert die Punktwolke, um Punkte zu identifizieren, die eine definierte maximale Entfernung zur Bodenebene haben. Hierfür wird der vom Benutzer im Settings-Tab übergebene Parameter „Ground max distance“ verwendet. Punkte, die der Bodenreferenz und dem festgelegten maximalen Winkelabstand entsprechen, werden als Bodenpunkte klassifiziert und aus der Punktwolke entfernt. Dieser Schritt ist essentiell, um Objekte von der Bodenebene zu trennen und so die Präzision bei der Detektion von Hindernissen zu erhöhen.

Die Funktion segmentiert die Bodenebene durch eine Kombination aus Distanz- und Winkelkriterien, welche sicherstellen, dass nur Punkte in unmittelbarer Nähe und Ausrichtung zur Bodenebene als Bodenpunkte klassifiziert werden. Die Methode zur Segmentierung der Bodenebene basiert auf der Funktion `pcfitplane`, die eine robuste Schätzung der Bodenebene durchführt. Hierbei verwendet MATLAB den RANSAC-Algorithmus (vgl. Kapitel 3.3.1). Dabei werden die Kriterien der Punktdistanz zur Ebene und des Winkelabstands zwischen den Ebenennormalen und einem Referenzvektor verwendet. [33][8]

Die Funktion `pcfitplane` verwendet die in Gleichung 5.1 definierte Ebenengleichung.

$$ax + by + cz + d = 0 \tag{5.1}$$

Hierbei stellt (a, b, c) den Normalvektor der Ebene und d den Abstand der Ebene zum Ursprung dar. Die `pcfitplane`-Funktion passt die Ebene so an die Punktwolke an, dass der Großteil der als Boden klassifizierten Punkte innerhalb eines definierten Abstandsbereichs liegt. Dieser max. Abstand zur Bodenebene wird durch den Parameter `GroundMaxDistance` festgelegt.

Für jeden Punkt $P = (x_p, y_p, z_p)$ in der Punktwolke wird anschließend die senkrechte Distanz d_p zur berechneten Ebene anhand der Ebenen-normalen (a, b, c) durch Gleichung 5.2 berechnet.

$$d_p = \frac{|ax_p + by_p + cz_p + d|}{\sqrt{a^2 + b^2 + c^2}} \tag{5.2}$$

Ein Punkt wird als Bodenpunkt klassifiziert, wenn seine Distanz zur Ebene d_p kleiner oder gleich dem festgelegten Schwellenwert `GroundMaxDistance` ist.

Zusätzlich zur Distanz wird auch der Winkelabstand zwischen dem Normalvektor der Ebene und einem Referenzvektor $\vec{r}$ (typischerweise $[0, 0, 1]^T$), um die Bodenebene als horizontal zu definieren) betrachtet. Der Winkel θ zwischen dem Normalvektor $\vec{n} = (a, b, c)$ und dem Referenzvektor $\vec{r}$ wird berechnet durch die Verwendung der Funktion 5.3.

$$\cos(\theta) = \frac{\vec{n} \cdot \vec{r}}{\|\vec{n}\|\|\vec{r}\|} \tag{5.3}$$

Das Skalarprodukt $\vec{n} \cdot \vec{r}$ misst die Ausrichtung der beiden Vektoren $\vec{n}$ und $\vec{r}$ zueinander. Es wird ermittelt, indem die jeweiligen Komponenten der Vektoren in X-, Y- und Z-Richtung miteinander multipliziert und anschließend summiert werden. Ist das Skalarprodukt maximal, liegen die Vektoren parallel zueinander. Ist das Skalarprodukt null, stehen sie orthogonal zueinander. Diese Information ist entscheidend, da sich mit ihr der Winkel zwischen dem Normalvektor der Ebene und einem Referenzvektor bestimmen lässt. Der mathematische Zusammenhang ist in Gleichung 5.4 beschrieben.

$$\vec{n} \cdot \vec{r} = a \cdot r_x + b \cdot r_y + c \cdot r_z \tag{5.4}$$

Die Normen der Vektoren $\vec{n}$ und $\vec{r}$, dargestellt in Gleichung 5.5, geben die Länge jedes Vektors im Raum an. Sie werden berechnet, indem die Quadrate der Komponenten des Vektors summiert und daraus die Quadratwurzel gezogen wird. Diese Normen spielen eine zentrale Rolle bei der Bestimmung des Winkels θ zwischen den Vektoren, da sie im Nenner des Skalarproduktes stehen.

$$\|\vec{n}\| = \sqrt{a^2 + b^2 + c^2} \quad \text{und} \quad \|\vec{r}\| = \sqrt{r_x^2 + r_y^2 + r_z^2} \tag{5.5}$$

Der Winkel θ zwischen den Vektoren $\vec{n}$ und $\vec{r}$ wird als Maß für die Übereinstimmung ihrer Ausrichtung verwendet. Ein Punkt wird nur dann als Bodenpunkt betrachtet, wenn der Winkel θ zwischen dem Ebenennormalen und dem Referenzvektor zur Bodenebene kleiner oder gleich einem bestimmten Schwellwert, dem Parameter `GroundMaxAngularDistance`, ist. [25][33]

Damit wird sichergestellt, dass nur Punkte die sowohl eine nahe Distanz zur Bodenebene aufweisen, als auch in einem flachen Winkel zur Ebene liegen, als Bodenpunkte klassifiziert werden. Diese zweistufige Bedingung verbessert die Genauigkeit der Bodenklassifikation und trennt Bodenpunkte verlässlich von Hindernissen.

In Abbildung 5.3 sind die als Bodenebene klassifizierten Punkte dargestellt. Für die Erkennung der Bodenebene wurden in der App zwei unterschiedliche Werte für den Parameter `GroundMaxDistance` getestet: 0,25 m und 0,5 m. Die Abbildung zeigt, dass je nach Einstellung dieses Parameters auch Punkte der Fahrzeuge fälschlicherweise als Bodenebene erkannt werden können. Daher ist es empfehlenswert, den Wert von `GroundMaxDistance` so zu wählen, dass möglichst viele Bodenpunkte korrekt erkannt werden, ohne dass zahlreiche Fahrzeugpunkte ebenfalls als Bodenebene klassifiziert werden.

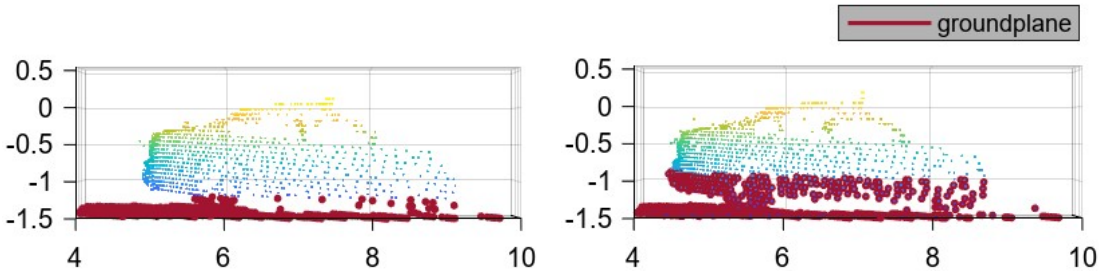


Abbildung 5.3. Klassifizierung der Bodenebene mit unterschiedlichen Parameterwerten

5.2.3 Segmentierung/ Clusterbildung

Die verbleibenden Punkte, die keine Bodenpunkte sind und sich innerhalb der Szenegrenzen befinden, werden in diesem Schritt gruppiert und mit Objekt-Begrenzungsrahmen versehen. Die Clusterbildung erfolgt mithilfe eines Parameters für den Mindestabstand (`SegmentationMinDistance`), welcher den minimalen Abstand zwischen Punkten innerhalb eines Clusters definiert. Punkte, die diesen Abstand unterschreiten, werden als zusammenhängend betrachtet und zu einem Cluster gruppiert.

Zwei Punkte werden als Teil des selben Clusters betrachtet, wenn ihr Abstand $d(P_i, P_j)$ kleiner oder gleich dem Schwellenwert `SegmentationMinDistance` ist.

Für jedes Cluster werden grundlegende statistische Berechnungen durchgeführt, um die durchschnittliche Position, Größe und Orientierung des Clusters zu bestimmen.

Die Höhe eines Clusters C wird dabei als Mittelwert der z-Koordinate berechnet mit Gleichung 5.6. Dabei ist n die Gesamtanzahl der Punkte im Cluster, k der Index jedes Punktes

und z_k die z-Koordinate des k-ten Punktes. Der Mittelwert der z-Koordinaten ermöglicht eine zuverlässige Schätzung der durchschnittlichen Höhe des Clusters, um diesen anhand der vorgegebenen Höhenbeschränkungen zu filtern.

$$z_{\text{mean}} = \frac{1}{n} \sum_{k=1}^{n} z_k \tag{5.6}$$

Ein Cluster wird nur dann akzeptiert, wenn der berechnete Durchschnitt der z-Koordinaten, also die mittlere Höhe des Clusters, innerhalb eines vorgegebenen Höhenbereichs liegt. Dieser Höhenbereich wird durch die Einstellung der Parameter `MinZDistanceCluster` und `MaxZDistanceCluster` festgelegt und dient dazu, sicherzustellen, dass nur Cluster in einer realistischen Höhe als relevante Objekte erkannt werden.

Diese Filterung verhindert die Detektion von unbedeutenden Clustern und sichert, dass nur relevante Objekte detektiert werden. [34][25]

5.2.4 Objekt-Begrenzungsrahmen-Erstellung

Für die bereinigten Cluster werden Begrenzungsrahmen (Bounding Boxes) berechnet, um deren genaue Position, Dimensionen und Ausrichtung zu bestimmen. Drei wesentliche Parameter beschreiben die Begrenzungsrahmen: der Schwerpunkt des Clusters, die Dimensionen (Länge, Breite, Höhe) sowie der Gierwinkel, der die Orientierung in der horizontalen Ebene angibt.

Berechnung des Schwerpunkts: Der Schwerpunkt eines Clusters repräsentiert die zentrale Lage aller Punkte im Cluster und wird durch den Mittelwert der Koordinaten in den x-, y- und z-Richtungen berechnet. Angenommen, ein Cluster enthält n Punkte, die durch die Koordinaten (x_k, y_k, z_k) für jeden Punkt k beschrieben sind. Der Mittelwert jeder Koordinate wird durch die Summe der entsprechenden Koordinaten aller Punkte, geteilt durch die Gesamtanzahl der Punkte n, berechnet.

Die Berechnungen für die Mittelwerte in jeder Richtung kann über den Zusammenhang in Gleichung 5.7 beschrieben werden.

$$x_{\text{mean}} = \frac{1}{n} \sum_{k=1}^{n} x_k, \quad y_{\text{mean}} = \frac{1}{n} \sum_{k=1}^{n} y_k, \quad z_{\text{mean}} = \frac{1}{n} \sum_{k=1}^{n} z_k \tag{5.7}$$

Diese Mittelwerte beschreiben zusammen den Punkt $(x_{\text{mean}}, y_{\text{mean}}, z_{\text{mean}})$, der den Schwerpunkt bzw. das Zentrum der Bounding Box angibt. Dieses Zentrum dient als Referenzpunkt für die Lage der Bounding Box und erleichtert die Beschreibung der räumlichen Ausdehnung des Clusters in den nachfolgenden Berechnungen.

Bestimmung der Dimensionen: Die Länge L, Breite W und Höhe H der Bounding Box werden anhand der maximalen und minimalen Werte der Punkte im Cluster in jeder Rich-

tung berechnet 5.8.

$$L = \max(x) - \min(x), \quad W = \max(y) - \min(y), \quad H = \max(z) - \min(z) \tag{5.8}$$

Berechnung des Gierwinkels (Yaw-Winkel): Der Gierwinkel θ beschreibt die Orientierung des Clusters in der horizontalen Ebene, also die Rotation um die z-Achse. Dieser Winkel wird so berechnet, dass die Längsachse der Bounding Box entlang der Hauptorientierung des Clusters verläuft. Auf diese Weise passt sich der Begrenzungsrahmen optimal an die Form und Ausrichtung des Clusters an, was insbesondere bei Objekten wie Fahrzeugen wichtig ist.

Die Berechnung des Winkels erfolgt anhand der Hauptorientierung des Clusters durch Gleichung 5.9.

$$\theta = \arctan\left(\frac{\sum_{k=1}^{n}(y_k - y_{\text{mean}})(x_k - x_{\text{mean}})}{\sum_{k=1}^{n}(x_k - x_{\text{mean}})^2}\right) \tag{5.9}$$

Dabei stellt θ den Winkel dar, der die beste Ausrichtung der Bounding Box entlang der Hauptachse des Clusters bietet.

Um sicherzustellen, dass die Orientierung der Bounding Box stabil bleibt und keine abrupte Richtungsänderung auftritt, wird der Winkel θ angepasst, wenn er größer als 45 Grad beträgt. Ein solch hoher Wert kann auf eine zu starke Neigung hindeuten, die die Ausrichtung der Bounding Box in eine ungünstige Richtung lenken würde. Falls $|\theta| > 45°$, wird die Orientierung der Bounding Box durch Hinzufügen oder Subtraktion eines 90°-Schritts angepasst. Dadurch wird gewährleistet, dass die Längsachse der Bounding Box nicht in einer instabilen oder unpassenden Richtung verläuft und die Ausrichtung des Begrenzungsrahmens konstant und konsistent bleibt, selbst bei leichten Veränderungen in der Clusterstruktur. [25]

Die Darstellung der Bounding Box innerhalb der App-Visualisierung erfolgt über Quader. In Abbildung 5.4 ist dies dargestellt.

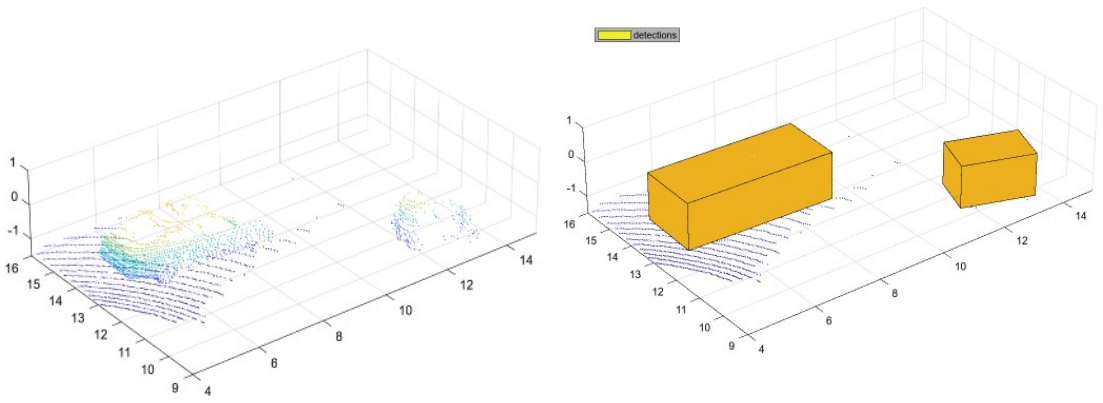


Abbildung 5.4. Vergleich zwischen Referenzpunktwolke und Detektionen

5.2.5 Umwandlung in das Detektionsformat

Durch die berechneten Parameter kann der Begrenzungsrahmen eines Clusters eindeutig definiert werden. Für jeden Cluster werden der Schwerpunkt $(x_{\text{mean}}, y_{\text{mean}}, z_{\text{mean}})$, die Dimensionen (Länge L, Breite W und Höhe H) sowie der Gierwinkel (θ) ermittelt und gespeichert. Diese Parameter ermöglichen eine präzise Abbildung der räumlichen Lage und Ausrichtung des Clusters.

Im nächsten Schritt werden die ermittelten Bounding Box-Parameter in das MATLAB-Detektionsformat `objectDetection` überführt. Innerhalb des `objectDetection`-Objekts werden die Informationen im `Measurement`-Property gespeichert, das den gesamten Messvektor 5.10 des Clusters umfasst:

$$\texttt{Measurement} = \begin{bmatrix} x_{\text{mean}} \\ y_{\text{mean}} \\ z_{\text{mean}} \\ \theta \\ L \\ W \\ H \end{bmatrix} \tag{5.10}$$

Dieser Vektor enthält die Positionsdaten, die Orientierung sowie die Dimensionen des Clusters in einer strukturierten Form, die eine optimale Verarbeitung in den nachfolgenden Tracking-Algorithmen unterstützt. Damit ist der Messvektor für die Zustandsschätzung des Verfolgungsalgorithmus gebildet.

Die Umwandlung in das `objectDetection`-Format erfolgt durch die MATLAB-Funktion `assembleDetections`, die für jede Bounding Box eine Detektion erstellt und die berechneten Positions- und Orientierungsinformationen einbindet. Zusätzlich wird das im System definierte Messrauschen (`MeasurementNoise`) hinzugefügt, um die Unsicherheiten der Bounding Box-Detektion zu modellieren. Dieses Rauschmodell bildet die unvermeidlichen Ungenauigkeiten in der Detektion ab und ist entscheidend für die Zuverlässigkeit der nachfolgenden Verfolgungsschritte. Damit lässt sich die Rauschproblematik von LiDAR-Messungen (vgl. Kapitel 3.2) weitestgehend berücksichtigt. [35]

5.2.6 Übersicht der Parameter des Detektormodells

Die Parameter der Klasse `HelperBoundingBoxDetector` ermöglichen eine präzise Kontrolle über die Punktwolkenverarbeitung und die Detektionsqualität. Alle Parameter welche das Verhalten des Detektormodells beeinflussen sind in Tabelle 5.2 aufgezählt. Im *MOT* bzw. *Settings*-Tab der App können die Parameter vom Nutzer übergeben werden.

Tabelle 5.2. Parameter zur Konfiguration der Bounding Box-Detektion

Parameter	**Beschreibung**
`XLimits`, `YLimits`, `ZLimits`	-begrenzen den Detektionsbereich in den jeweiligen Achsen
`GroundMaxDistance`, `GroundMaxAngularDistance`	-maximale Entfernung und Winkel zur Bodenreferenz für Bodenklassifikation
`SegmentationMinDistance`	-Mindestabstand zwischen Punkten innerhalb eines Clusters
`MinDetectionsPerCluster`	-Mindestanzahl von Punkten für gültige Cluster-Detektion
`MeasurementNoise`	-Messrauschen der Bounding Box zur Unsicherheitsmodellierung
`MaxZDistanceCluster`, `MinZDistanceCluster`	-Höhenbereich zur Filterung irrelevanter Cluster

5.3 Implementierung und Verwendung des JPDA-Trackers mit IMM

Für die Verarbeitung der Detektionen, welche vom Detektormodell geliefert werden, wurde sich für ein JPDA-Tracker entschieden. Dieser Tracker ist in MATLAB einfach zu implementieren und bietet eine robuste Verfolgung von Objekten. In MATLAB kann der Tracker mit dem Befehl *trackerJPDA* initialisiert werden. Er benötigt nach der Initialisierung die Übergabe einer Filter-Initialisierungs-Funktion und die Parametrierung einer Verfolgungslogik. Für die Filterfunktion wird ein ausgelagertes Skript verwendet, welches einen IMM-Filter initialisiert. Die Vorgehensweise wird folgend in Kapitel 5.4 näher beschrieben. Für die Verfolgungslogik des JPDA-Trackers stehen zwei Optionen zur Verfügung: „*History*“ und „*Integrated*“. In der App kommt die deutlich einfacher aufgebaute History-Logik zum Einsatz.

5.3.1 Verfolgungslogik

In diesem Abschnitt wird die Verfolgungslogik des JPDA Trackers erläutert, der in Verbindung mit einer „History“-Tracking-Logik in der vorliegenden Arbeit verwendet wurde. Der JPDA-Tracker ist speziell auf Situationen ausgelegt, in denen es zu Mehrdeutigkeiten in den Messzuordnungen kommt, und nutzt diese Wahrscheinlichkeitslogik, um die korrekte Zuordnung der Detektionen zu den bestehenden Tracks zu optimieren.

Die „History“-Tracking-Logik des JPDA-Trackers ermöglicht eine kontinuierliche Verfolgung der Objekte auch bei temporären Unterbrechungen der Detektionsverfügbarkeit. Dabei speichert der Tracker Informationen über frühere Detektionszuordnungen, um die Stabilität der Tracks bei Unsicherheiten zu erhöhen und Übergangslösungen in Fällen temporärer

Detektionsverluste anzubieten. Um diese Verfolgungslogik besser verständlich zu machen, wird ein kurzes Beispiel einer Verfolgung angeführt.

Die Funktionsweise der „History“-Logik wird durch eine Matrix dargestellt. Die Matrix repräsentiert eine Sequenz von Treffern (1) und Verfehlungen (0) über mehrere Iterationsschritte. Angenommen, die Schwellenwerte für das *confirmation gate* und das *deletion gate* sind wie in 5.11 festgelegt.

$$\begin{aligned} \textbf{Confirmation Gate} &: [M, N] = [3, 5] \\ \textbf{Deletion Gate} &: [P, R] = [4, 5] \end{aligned} \tag{5.11}$$

Ein Track wird basierend auf den letzten fünf Iterationen entweder bestätigt, wenn er mindestens drei Treffer erzielt, oder gelöscht, falls er vier oder mehr Verfehlungen aufweist. Eine Treffer- und Verfehlungssequenz kann beispielhaft durch folgende History-Matrix 5.12 dargestellt werden.

$$\text{Logic} = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{5.12}$$

Die ersten drei Elemente der Matrix sind 0, was auf anfängliche Verfehlungen hinweist. Ab dem vierten bis zum siebten Schritt treten jedoch aufeinanderfolgende Treffer (1) auf. Da in den letzten 5 Schritten (Schritte 3 bis 7) mindestens 3 Treffer verzeichnet wurden, erreicht der Track die Schwelle des *confirmation gate* und wird als bestätigt eingestuft.

Nach der Bestätigung bleibt der Track aktiv, selbst wenn in den folgenden Schritten Verfehlungen (0) auftreten. In den Schritten 8 und 9 werden keine weiteren Treffer verzeichnet, doch da die Verfehlungen noch nicht die Schwelle des *deletion gate* erreichen, bleibt der Track bestätigt.

Ab Schritt 8 bis zum Ende der Matrix gibt es eine Serie von Verfehlungen (0). In den letzten 5 Schritten (Schritte 7 bis 11) werden insgesamt 4 Verfehlungen verzeichnet, womit die Schwelle des *deletion gate* erreicht wird. Der Track wird daher gelöscht.

Diese Matrix zeigt, wie die „History“-Logik durch die Treffer- und Verfehlungsabfolge eine zuverlässige Entscheidung über die Bestätigung und Löschung von Tracks trifft. Ein Track bleibt nur dann bestehen, wenn ausreichend Treffer in den letzten Iterationen verzeichnet wurden, und wird gelöscht, wenn er eine definierte Anzahl an Verfehlungen aufweist. [36]

5.3.2 Detektierbare Verfolgungen

Eine weitere Optimierung des JPDA-Trackers kann durch den optionalen Input-Parameter `detectableTrackingInput` erfolgen. Dieser Parameter gibt dem Tracker Hinweise darauf, ob ein Objekt in einem bestimmten Iterationsschritt detektierbar ist. Dies ist entscheidend, um in Szenarien mit unregelmäßigen Detektionen eine sinnvolle Berechnung der

Zuordnungswahrscheinlichkeiten zu gewährleisten und dabei übermäßige Identitätswechsel oder Track-Löschungen zu vermeiden.

Um den Parameter `detectableTrackingInput` effektiv zu nutzen, wurde die Funktion `helperCalcDetectability` implementiert. Diese Funktion berechnet die Detektionswahrscheinlichkeit (P_d) für jeden Track basierend auf der Entfernung r des Tracks vom Sensor. Ziel ist es, die Wahrscheinlichkeit der Detektion in Abhängigkeit von der Entfernung zu modellieren, um die Effizienz und Genauigkeit des Trackers zu verbessern. [37]

Die Detektionswahrscheinlichkeit $P_d(r)$ wird durch die stückweise definierte Funktion 5.13 dargestellt.

$$P_d(r) = \begin{cases} 0.9 & \text{für } r \leq 40 \text{ m}, \\ 0.4 & \text{für } 40 \text{ m} < r \leq 75 \text{ m}, \\ 0.99 & \text{für } r > 75 \text{ m}. \end{cases} \tag{5.13}$$

Diese Funktion ermöglicht es dem Tracker, die Wahrscheinlichkeit der Detektion dynamisch anzupassen. Innerhalb der zulässigen Entfernung von ($r \leq 40$ m) bleibt die Wahrscheinlichkeit hoch, was eine zuverlässige Verfolgung unterstützt. Im Bereich zwischen 40 m und 75 m wird die Wahrscheinlichkeit reduziert, um Unsicherheiten und potenzielle Fehlzuweisungen zu minimieren. Über 75 m hinaus wird die Wahrscheinlichkeit stark erhöht, um sicherzustellen, dass Tracks, die sich außerhalb der effektiven Reichweite befinden, schnell gelöscht werden und somit die Tracking-Performance optimiert wird.

Diese mathematische Modellierung trägt dazu bei, die Tracking-Logik flexibel und anpassungsfähig zu gestalten, indem sie die Detektionswahrscheinlichkeit basierend auf der Entfernung dynamisch anpasst. [25]

5.3.3 TrackingObject und State-Eigenschaft

Nach der Verfolgung der Objekte gibt der JPDA-Tracker gibt zwei zentrale Ergebnisgruppen aus: Vorläufige (tentative) und bestätigte (confirmed) Tracks. Vorläufige Tracks sind Verfolgungen, die sich noch in der vorläufigen Phase befinden und weitere Bestätigungen benötigen. Bestätigte Tracks sind die Verfolgungen, die eine bestimmte Anzahl von Bestätigungen erreicht haben und somit als bestätigt gelten. Nur diese Tracks werden als zuverlässige Verfolgungen betrachtet und für die GOSPA-Metrik verwendet.

In MATLAB wird das `TrackingObject` verwendet, um die Eigenschaften eines verfolgten Objekts in einem strukturierten Format zu speichern. Besonders wichtig ist hierbei das `State`-Property, das die dynamischen Parameter eines Tracks enthält, wie Position, Geschwindigkeit und Objektlänge und -breite. Das `State`-Property ermöglicht eine effiziente Speicherung und Abfrage der Track-Daten im Verlauf der Tracking-Iterationen. Die numerischen Werte für die dynamischen Parameter des Objekts werden innerhalb der Zustandsschätzung des Trackers berechnet.

Die bestätigten Tracks werden in der App als Quader innerhalb der Punktwolke visualisiert. In Abbildung 5.5 sind die Tracks mit den Modell-Wahrscheinlichkeiten (Konst. Geschw. und Konst. Drehrate) erkennbar.

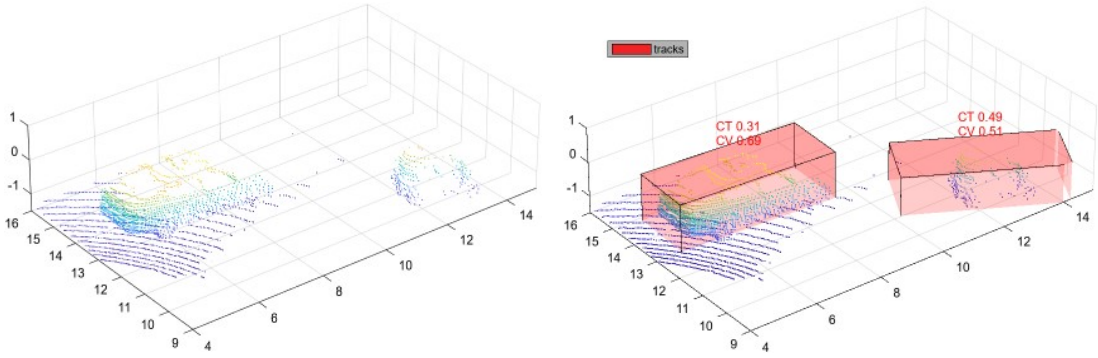


Abbildung 5.5. Vergleich zwischen Referenzpunktwolke und Tracks

5.3.4 Übersicht der Parameter des Trackermodells

Die Parameter des JPDA-Trackers spielen eine entscheidende Rolle bei der effektiven Verfolgung von Objekten in einer Umgebung, in der Mehrdeutigkeiten und unklare Zuordnungen auftreten. Mithilfe der „History“-Logik werden diese Parameter angepasst, um die Robustheit des Trackings zu erhöhen und die Detektionszuverlässigkeit über mehrere Zeitschritte hinweg zu sichern. [36]

In Tabelle 5.3 wurden alle Parameter, die auch in der entwickelten App anpassbar sind, festgehalten.

Tabelle 5.3. Übersicht der JPDA-Tracker Parameter in der „History“-Logik

Parameter	**Beschreibung**
`MaxNumDetections`	Maximale Anzahl der Detektionen pro Iteration, um die Rechenlast zu kontrollieren und Überläufe zu vermeiden.
`MaxNumTracks`	Obergrenze für gleichzeitig verfolgte Tracks, um Ressourcen zu optimieren.
`ClutterDensity`	Erwartete Fehlalarmdichte, verwendet als Dichteparameter für ein Poisson’sches Cluttermodell.
`AssignmentThreshold`	Zweistufige Zuordnungslogik [C1, C2] über den Normalisierungsdistanz-Grenzwert zwischen Tracks und Detektionen.
`InitializationThreshold`	Wahrscheinlichkeitsgrenze zur Initialisierung neuer Tracks, um übermäßige Trackvermehrung zu vermeiden.
`ConfirmationThreshold`	Zweistufige Logik [M, N] für Trackbestätigung bei mindestens M Treffern in den letzten N Updates.
`DeletionThreshold`	Zweistufige Löschlogik [P, R]: Track wird entfernt, wenn in den letzten R Schritten mindestens P Verfehlungen auftreten.
`HitMissThreshold`	Schwelle zur Registrierung eines „Treffers“ oder „Fehlers“. Registriert „Miss“ bei zu geringer Wahrscheinlichkeit für die Trackzuweisung.

5.4 Zustandsschätzung durch IMM-Filter

Für die Zustandsschätzung der Objekte muss dem Tracker eine Filter-Funktion übergeben werden. Hierbei wird ein IMM-Filter mit zwei Zustandsmodellen verwendet. Für die Initialisierung des Filters sind mehrere Code-Skripte notwendig. Dabei nutzt das Hauptskript `InitIMMFilter` weitere Unterskripte, um die zwei Bewegungsmodelle zu verwenden. Desweiteren wird die Schrumpfproblematik durch die Verwendung von weiteren Skripten korrigiert.

In Tabelle 5.4 sind die Hauptaufgaben der Unterskripte veranschaulicht.

Tabelle 5.4. Übersicht der MATLAB-Skripte für IMM-Filter und deren Aufgaben

Funktion	**Aufgaben**
`helperInitIMMFilter`	• initialisiert den IMM-Filter • kombiniert CV- und CT-Filtermodelle
`helperInverseLidarModel`	• rekonstruiert Objektposition und -dimensionen aus LiDAR-Messungen • kompensiert Sensorschrumpfungseffekte
`helperLidarModel`	• simuliert LiDAR-Messungen basierend auf Objektparametern • modelliert Messschrumpfung mit Entfernung
`helperCvmeasCuboid`	• definiert die Messfunktion für das CV-Modell • übersetzt den CV-Zustand in den Messraum
`helperCtmeasCuboid`	• definiert die Messfunktion für das CT-Modell • übersetzt den CT-Zustand in den Messraum
`helperConstvelCuboid`	• implementiert das CV-Bewegungsmodell für Quader • beschreibt die Zustandsvorhersage bei konstanter Geschwindigkeit
`helperConstturnCuboid`	• implementiert das CT-Bewegungsmodell für Quader • beschreibt die Zustandsvorhersage bei konstanter Drehung

Ablauf des IMM-Filters:

1. Initialisierung: Beide Modelle werden mit initialen Zuständen und Kovarianzen gestartet.
2. Vorhersage: Jedes Modell prognostiziert den nächsten Zustand basierend auf seinem Bewegungsmodell und fügt Prozessrauschen hinzu.
3. Update: Die Messungen werden verwendet, um die Zustände der Modelle zu aktualisieren.
4. Modellwahrscheinlichkeiten: Die Wahrscheinlichkeit jedes Modells wird basierend auf der Übereinstimmung zwischen Vorhersage und Messung berechnet.
5. Kombination: Die Zustände der Modelle werden entsprechend ihrer Wahrscheinlichkeiten gewichtet und zu einer Gesamtzustandsschätzung kombiniert. [37][38]

5.4.1 Korrektur der Schrumpf-Problematik

Wenn die Detektions-Bounding Box eines Objekts kleiner wird, z.B. weil das Objekt weiter vom Sensor entfernt ist oder durch andere Sensorbedingtheiten, kann dies den Tracker dazu bringen, die Schätzung der tatsächlichen Position des Objekts zu verfälschen (vgl. Kapitel 3.2). Der Tracker könnte sich zu stark auf die neue, ungenauere Detektion verlassen

und somit den Track verschieben oder falsch anpassen. Um diese Problematik innerhalb des Tracking-Prozesses zu korrigieren, berechent der IMM-Filter die eigentliche Größe und Position der Objekte anhand der Entfernung zum Sensor.

Die Funktion `helperInverseLidarModel` adressiert dieses Problem durch eine mathematische Anpassung der erfassten Positionen. Im Folgenden wird der mathematische Zusammenhang dieser Anpassung erläutert.

Zunächst werden zwei Schrumpfraten 5.14 definiert:

$$\begin{aligned} s &= \frac{3}{50} \quad \text{(Schrumpfrate für Länge und Breite)} \\ sz &= \frac{2}{50} \quad \text{(Schrumpfrate für Höhe)} \end{aligned} \tag{5.14}$$

Diese Raten bestimmen, wie stark die Positionen in den jeweiligen Dimensionen angepasst werden sollen, um die Schrumpfung der Bounding Boxes zu kompensieren.

Die Funktion `helperInverseLidarModel` verwendet sphärische Koordinaten, um die Positionen der erkannten Objekte zu berechnen. Dabei werden die kartesischen Koordinaten (x, y, z) in sphärische Koordinaten (θ, ϕ, r) umgewandelt, wobei θ den Azimutwinkel, ϕ den Elevationswinkel und r die Entfernung darstellt. Mit der Umrechnung $(\theta, \phi, r) = \text{cart2sph}(x, y, z)$ in MATLAB können die sphärischen Koordinaten berechnet werden. Basierend auf den sphärischen Koordinaten werden die Schrumpfungen in den drei Dimensionen mit Gleichung 5.15 berechnet. [39]

$$\begin{aligned} L_{\text{shrink}} &= |s \cdot r \cdot \cos(\theta)| \\ W_{\text{shrink}} &= |s \cdot r \cdot \sin(\theta)| \\ H_{\text{shrink}} &= sz \cdot r \end{aligned} \tag{5.15}$$

Hierbei repräsentieren L_{shrink}, W_{shrink} und H_{shrink} die Schrumpfungen in Länge, Breite und Höhe. Die Schrumpfungen sind proportional zur Entfernung r und hängen vom Azimutwinkel θ ab.

Die ursprünglichen Positionen werden um die berechneten Schrumpfungen mit Gleichung 5.16 angepasst:

$$\begin{aligned} x_{\text{neu}} &= x + \text{sign}(x) \cdot \frac{L_{\text{shrink}}}{2} \\ y_{\text{neu}} &= y + \text{sign}(y) \cdot \frac{W_{\text{shrink}}}{2} \\ z_{\text{neu}} &= z + \text{sign}(z) \cdot \frac{H_{\text{shrink}}}{2} \end{aligned} \tag{5.16}$$

Diese Anpassungen verschieben die Positionen der Bounding Boxes, um die durch die Schrumpfraten verursachte Verkleinerung zu kompensieren. Der Faktor $\frac{1}{2}$ sorgt dafür, dass die Verschiebung gleichmäßig auf beide Seiten des Ursprungs verteilt wird. Nach dieser

Anpassung ergibt sich eine neue Position für die Tracks gemäß der Berechnung des neuen Schwerpunktes $[x_{\text{neu}}, y_{\text{neu}}, z_{\text{neu}}]$.

5.4.2 Aufbau der Bewegungsmodelle CV und CT

In der Objektverfolgung mit LiDAR-Daten werden Bewegungsmodelle verwendet, um die Position und Bewegung eines Objekts vorherzusagen. Zwei gängige Modelle sind das **Constant Velocity (CV)** Modell und das **Constant Turn (CT)** Modell. Beide Modelle teilen viele Gemeinsamkeiten, unterscheiden sich jedoch in der Behandlung der Drehbewegung. Im Folgenden werden beide Modelle gemeinsam vorgestellt, wobei auf ihre Gemeinsamkeiten und Unterschiede eingegangen wird. [25]

5.4.3 Gemeinsamer Zustandsvektor

Der Zustandsvektor enthält insgesamt 10 Komponenten welche in Tabelle 5.5 festgehalten sind.

Tabelle 5.5. Übersicht der Zustandskomponenten

Komponente	**Variablen**	**Beschreibung**
Position	x, y, z	Die aktuellen Koordinaten des Objekts.
Geschwindigkeit	$\dot{x}, \dot{y}, \dot{z}$	Die Geschwindigkeit in jeder Achse.
Orientierung	θ	Der Gierwinkel (Yaw) des Objekts.
Dimensionen	L, W, H	Länge, Breite und Höhe des Objekts.

Für das CT-Modell wird zusätzlich die Drehgeschwindigkeit ω berücksichtigt. Somit werden die Zustandsvektoren $\mathbf{x}_{\text{CV}}$ und $\mathbf{x}_{\text{CT}}$ wie in 5.17 definiert.

$$\mathbf{x}_{\text{CV}} = \begin{bmatrix} x \\ \dot{x} \\ y \\ \dot{y} \\ z \\ \dot{z} \\ \theta \\ L \\ W \\ H \end{bmatrix} \qquad \mathbf{x}_{\text{CT}} = \begin{bmatrix} x \\ \dot{x} \\ y \\ \dot{y} \\ \omega \\ z \\ \dot{z} \\ \theta \\ L \\ W \\ H \end{bmatrix} \tag{5.17}$$

5.4.4 Gleichungen der Zustandsvorhersage

Beide Modelle basieren auf der Vorhersage des nächsten Zustands gemäß dem aktuellen Zustand und dem Bewegungsmodell. Die allgemeinen Komponenten, wie die vertikale Bewegung und die Dimensionen werden in beiden Modellen gleich behandelt.

Die Dimension des Objektes wird laut Gleichung 5.18 über die gesamte Laufzeit der Zustandsvorhersage als konstant betrachtet.

$$\begin{aligned} L_{k+1} &= L_k \\ W_{k+1} &= W_k \\ H_{k+1} &= H_k \end{aligned} \tag{5.18}$$

Die neue Positionskomponente z_{k+1} wird berechnet, indem zur aktuellen Position z_k das Produkt der momentanen Geschwindigkeit $\dot{z}_k$ und der Zeitdifferenz Δt addiert wird, während die Geschwindigkeit $\dot{z}_{k+1}$ konstant bleibt und der aktuellen Geschwindigkeit $\dot{z}_k$ entspricht 5.19.

$$\begin{aligned} z_{k+1} &= z_k + \dot{z}_k \Delta t \\ \dot{z}_{k+1} &= \dot{z}_k \end{aligned} \tag{5.19}$$

Die Unterschiede zwischen den Modellen liegen in der Behandlung der Bewegung in der horizontalen Ebene (x und y).

Positionsaktualisierung

Das **CV**-Modell geht von einer geradlinigen Bewegung mit konstanter Geschwindigkeit aus. Die neue Position (x_{k+1}, y_{k+1}) eines Objekts wird berechnet, indem zur aktuellen Position (x_k, y_k) das Produkt der momentanen Geschwindigkeitskomponenten $(\dot{x}_k, \dot{y}_k)$ und der Zeitdifferenz Δt addiert wird. Der Gierwinkel θ_k wird als konstant angenommen 5.20.

$$\begin{aligned} \textbf{CV:} \quad x_{k+1} &= x_k + \dot{x}_k \Delta t \\ y_{k+1} &= y_k + \dot{y}_k \Delta t \end{aligned} \tag{5.20}$$

Das **CT**-Modell hingegen berücksichtigt eine konstante Drehgeschwindigkeit ω für kurvenförmige Bewegungen. Die Positionsänderung wird durch die rotatorische Bewegung des Objekts beschrieben, wobei die neuen Positionen von der momentanen Winkelgeschwindigkeit ω_k, der Geschwindigkeit v_k in der Ebene und der aktuellen Orientierung θ_k abhängen 5.21.

$$\begin{aligned} \textbf{CT:} \quad x_{k+1} &= x_k + \frac{\dot{v}_k}{\omega_k} \left(\sin(\theta_k + \omega_k \Delta t) - \sin(\theta_k)\right) \\ y_{k+1} &= y_k - \frac{\dot{v}_k}{\omega_k} \left(\cos(\theta_k + \omega_k \Delta t) - \cos(\theta_k)\right) \end{aligned} \tag{5.21}$$

Geschwindigkeitsaktualisierung

Im **CV**-Modell bleiben die Geschwindigkeitskomponenten in der Ebene konstant 5.22.

$$\textbf{CV:} \quad \dot{x}_{k+1} = \dot{x}_k, \quad \dot{y}_{k+1} = \dot{y}_k \tag{5.22}$$

Im **CT**-Modell berechnen sich die neuen Geschwindigkeitskomponenten, gemäß Gleichung 5.23, basierend auf der aktuellen Geschwindigkeit $\dot{v}_k$ und der Ausrichtung $\theta_k + \omega_k \Delta t$.

$$\begin{aligned} \textbf{CT:} \quad \dot{x}_{k+1} &= \dot{v}_k \cos(\theta_k + \omega_k \Delta t) \\ \dot{y}_{k+1} &= \dot{v}_k \sin(\theta_k + \omega_k \Delta t) \end{aligned} \tag{5.23}$$

Orientierung und Drehgeschwindigkeit

Im **CV**-Modell wird die Orientierung bzw. der Gierwinkel θ konstant gehalten 5.24.

$$\textbf{CV:} \quad \theta_{k+1} = \theta_k \tag{5.24}$$

Im **CT**-Modell wird die Drehgeschwindigkeit ω als konstant angenommen, während die Orientierung θ mit der Winkeländerung des Objekts innerhalb des Zeitintervalls Δt berechnet wird5.25.

$$\begin{aligned} \textbf{CT:} \quad \omega_{k+1} &= \omega_k \\ \theta_{k+1} &= \theta_k + \omega_k \Delta t \end{aligned} \tag{5.25}$$

5.4.5 Prozessrauschen

In beiden Modellen wird Prozessrauschen hinzugefügt, um Unsicherheiten in der Bewegung zu berücksichtigen. Das Prozessrauschen $\mathbf{v}_k$ wird als normalverteilte Zufallsvariable gemäß Gleichung 5.26 modelliert.

$$\mathbf{v}_k \sim \mathcal{N}(0, \mathbf{Q}) \tag{5.26}$$

Hierbei stellt $\mathbf{v}_k$ den Vektor des Prozessrauschens zum Zeitpunkt k dar. Der Opterator $\sim$ steht für „ist verteilt wie“, und $\mathcal{N}(0, \mathbf{Q})$ bezeichnet eine mehrdimensionale Normalverteilung mit Mittelwert Null und Kovarianzmatrix $\mathbf{Q}$. Die Kovarianzmatrix $\mathbf{Q}$ ist die Prozessrauschkovarianzmatrix, die die Varianzen der Unsicherheiten in den Zustandsgrößen enthält. Für beide Zustandsmodelle kann eine Prozessrauschkovarianzmatrix $\mathbf{Q}_{\text{CV}}$ und $\mathbf{Q}_{\text{CT}}$ wie in 5.27 definiert werden. Hierbei sind $q_{\dot{x}}$, $q_{\dot{y}}$ und $q_{\dot{z}}$ die Varianzen der Beschleunigungen in x-, y- und z-Richtung, q_θ die Varianz des Gierwinkels θ, und q_ω die Varianz der Winkelbeschleunigung ω (nur im CT Modell).

$$\mathbf{Q}_{\text{CV}} = \begin{bmatrix} q_{\dot{x}} & 0 & 0 & 0 \\ 0 & q_{\dot{y}} & 0 & 0 \\ 0 & 0 & q_{\dot{z}} & 0 \\ 0 & 0 & 0 & q_\theta \end{bmatrix} \qquad \mathbf{Q}_{\text{CT}} = \begin{bmatrix} q_{\dot{x}} & 0 & 0 & 0 & 0 \\ 0 & q_{\dot{y}} & 0 & 0 & 0 \\ 0 & 0 & q_\omega & 0 & 0 \\ 0 & 0 & 0 & q_{\dot{z}} & 0 \\ 0 & 0 & 0 & 0 & q_\theta \end{bmatrix} \tag{5.27}$$

Die diagonale Struktur der Kovarianzmatrizen impliziert, dass die Prozessrauschkomponenten unkorreliert sind. Durch geeignete Wahl der Varianzen kann gesteuert werden, wie stark das Modell auf unerwartete Änderungen reagiert.

5.4.6 Messmodell

Da die Detektionen in beiden Modellen identisch sind, teilen sie das gleiche Messmodell. Die Detektionen, die als Messvektor $\mathbf{z}_k$ in das Messmodell einfließen, liefern die erforderlichen Informationen zur Position, Orientierung und Dimension des Objekts, jedoch keine Geschwindigkeiten.

Das Messmodell verknüpft den geschätzten Zustand $\mathbf{x}_k$, der eine Annäherung an die tatsächlichen, aber unbekannten Positions-, Orientierungs- und Dimensionsdaten des Objekts darstellt, mit den Messungen gemäß der Gleichung 5.28.

$$\mathbf{z}_k = \mathbf{H}\,\mathbf{x}_k + \mathbf{w}_k \tag{5.28}$$

Dabei ist $\mathbf{x}_k$ der geschätzte Zustand, also die interne Modellvorstellung der Position, Orientierung und Dimension des Objekts, die der Verfolgungsalgorithmus basierend auf bisherigen Daten und dem Modell berechnet. Die Messung $\mathbf{z}_k$ stellt die tatsächlichen Detektionen dar, die von den Sensoren erfasst wurden und die Informationen zur aktuellen Position, Orientierung und Dimension enthalten.

Das Messmodell nutzt die Beobachtungsmatrix $\mathbf{H}$ (5.29), um aus dem geschätzten Zustand $\mathbf{x}_k$ die Komponenten auszuwählen, die mit den Messungen verglichen werden. Der

Messrauschvektor $\mathbf{w}_k$ modelliert zufällige Fehler in den Messungen und erklärt Abweichungen zwischen der Schätzung und den tatsächlichen Detektionen. Auf diese Weise können die Messungen $\mathbf{z}_k$ zur Korrektur und Aktualisierung des geschätzten Zustands $\mathbf{x}_k$ verwendet werden, um die Zustandsschätzung kontinuierlich zu verbessern. [18]

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 & \dots \\ 0 & 0 & 1 & 0 & \dots \\ 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & 0 & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} \tag{5.29}$$

Die genauen Positionen der Einsen in $\mathbf{H}$ hängen vom jeweiligen Zustandsvektor ab, aber in beiden Fällen werden die Geschwindigkeitskomponenten und die Drehgeschwindigkeit nicht gemessen und daher nicht durch das Messmodell aktualisiert.

Obwohl Geschwindigkeit und Drehgeschwindigkeit nicht direkt gemessen werden, ist der IMM-Filter vorteilhaft, weil er durch die Kombination von CV- und CT-Modellen unterschiedliche Bewegungsmuster abdeckt. Er ermöglicht eine flexible Anpassung an geradlinige und kurvenförmige Bewegungen, indem er die nicht gemessenen Zustandsgrößen indirekt aus den Positionsänderungen schätzt. Dies führt zu einer verbesserten Genauigkeit und Robustheit in der Mehrzielverfolgung, da der Filter besser auf Bewegungsänderungen reagieren kann und die Wahrscheinlichkeit von Fehlassoziationen zwischen Zielen reduziert wird.

5.4.7 Kombination beider Modelle im IMM-Filter

Die beiden Zustandsmodelle (CV und CT) werden im IMM-Filter als zwei separate Filter in Form von UKF implementiert. Die Filter werden in einem Zellarray gespeichert und anschließend dem IMM-Filter übergeben.

Der IMM-Filter schätzt, welches Modell das aktuelle Bewegungsverhalten des Objekts besser beschreibt, und passt die Modellwahrscheinlichkeiten entsprechend an. Durch die Kombination beider Modelle können sowohl geradlinige als auch kurvenförmige Bewegungen effektiv verfolgt werden. Um den Wechsel zwischen den Modellen zu ermöglichen, wird die Funktion `switchimmCuboid` verwendet. Diese Funktion konvertiert die Zustandsvektoren und Kovarianzmatrizen zwischen den Modellen, sodass die Zustände konsistent bleiben. Sie stellt sicher, dass beim Übergang zwischen CV- und CT-Modell die kinematischen Zustandsgrößen korrekt transformiert werden, während die Dimensionsparameter (Länge, Breite, Höhe) des Objekts unverändert bleiben.

Die Übergangswahrscheinlichkeiten zwischen den Modellen werden durch die *Transition-Probabilities* festgelegt. Diese Wahrscheinlichkeiten bestimmen, wie wahrscheinlich es ist, dass das Bewegungsverhalten des Objekts vom einen zum anderen Modell wechselt. Die Übergangswahrscheinlichkeiten werden in Form einer Matrix 5.30 definiert.

$$\mathbf{\Pi_{ij}} = \begin{bmatrix} \pi_{11} & \pi_{12} \\ \pi_{21} & \pi_{22} \end{bmatrix} \tag{5.30}$$

Hierbei repräsentiert π_{ij} die Wahrscheinlichkeit, vom Modell i zum Modell j zu wechseln. Durch geeignete Wahl der Übergangswahrscheinlichkeiten kann gesteuert werden, wie sensibel der IMM-Filter auf Änderungen im Bewegungsverhalten reagiert. Höhere Diagonalelemente π_{ij} bedeuten, dass der Filter dazu neigt, im aktuellen Modell zu bleiben, während höhere Off-Diagonalelemente π_{ij} einen häufigeren Wechsel zwischen den Modellen ermöglichen. [25]

6 Auswertung von LiDAR-Messungen und Validierung der Test-App

In diesem Kapitel erfolgt eine Leistungsbewertung verschiedener LiDAR-Sensoren anhand spezifischer Versuchsszenarien sowie die Validierung der Funktionalität der entwickelten Applikation. Für die vorliegende Diplomarbeit wurden Messdaten herangezogen, die im Rahmen eines Gaststudentenprogramms von drei Studierenden der German International University (GIU) in Kooperation mit B. Sc. Mohammad Ammad erhoben wurden. [40]

Die anschließende Analyse der Ergebnisse soll Aufschluss darüber geben, in welchem Umfang die GOSPA Metrik zur Bewertung der Verfolgungsleistung geeignet ist. Neben den Ergebnissen der GOSPA Metrik wird auch die Lokalisierungs-Komponente der Metrik untersucht. Darüber hinaus wird betrachtet, ob signifikante Unterschiede in der Leistung zwischen den verschiedenen Sensoren bestehen.

6.1 Verwendete Sensorik

Verwendet wurden drei verschiedenen LiDAR-Sensoren: Ouster OS1-64, Livox Horizon und Leishen S1. Die Sensoren sind in den Abbildungen 6.1, 6.2 und 6.3 dargestellt.

Abbildung 6.1. Ouster OS1-64 LiDAR [41]

Abbildung 6.2. Livox Horizon LiDAR [42]

Abbildung 6.3. LS-S1 LiDAR [43]

Der **Ouster OS1** realisiert eine 360°-Umgebungsabtastung durch den Einsatz eines mechanisch rotierenden Spiegels. Im Gegensatz zu herkömmlichen mechanischen Oszillatoren, die zur Strahllenkung verwendet werden, kombiniert der OS1 eine kontinuierliche horizontale Rotation mit fixierten vertikalen Strahlen in 64 Ebenen. Diese Vorgehensweise gewährleistet eine gleichmäßige Verteilung der Laserstrahlen über das gesamte horizontale Sichtfeld. [44]

In Abbildung 6.4 wird die Rotation mittels einer oszillierenden optischen Komponente sowie das entsprechende Scanmuster dargestellt.

Der **Livox Horizon** verwendet eine Wedge-Mirror-Technologie mit einem daraus resultierenden nicht-repetitiven horizontalen Scanmuster. Durch den Einsatz keilförmiger Spiegel werden die Laserstrahlen in variablen Winkeln und Abständen ausgesendet, was ein gleichmäßiges Scanmuster erzeugt. [45]

Das Prinzip des Wedge-Mirror-Scanning sowie das daraus resultierende Scanmuster sind in Abbildung 6.5 dargestellt.

Der **Leishen LS-S1** setzt auf eine MEMS-basierte Scanmethode auf 128 Ebenen. MEMS (vgl. Kapitel 2.4.1) ermöglichen eine präzise und flexible Steuerung der Laserstrahlen. [46]

In Abbildung 6.6 wird das MEMS-Prinzip sowie das resultierende Scanmuster veranschaulicht.

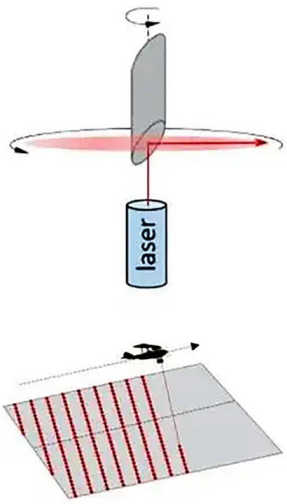


Abbildung 6.4. Rotierendes Scanning [47]

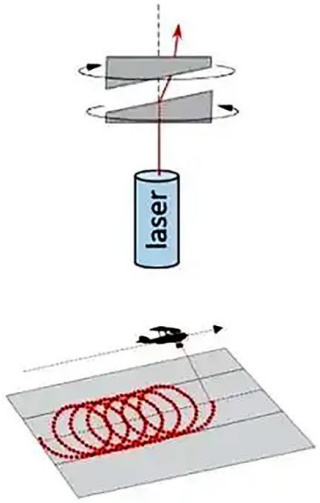


Abbildung 6.5. Wedge-Mirror Scanning [47]

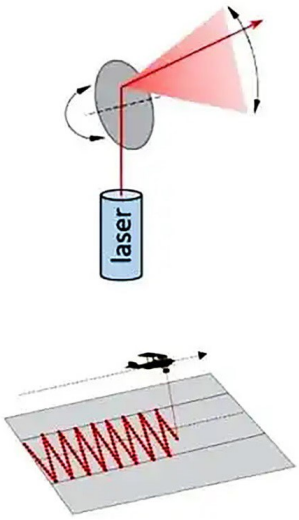


Abbildung 6.6. MEMS-Scanning [47]

Aus der unterschiedlichen Scan-Technologien resultieren unterschiedliche Scan-Muster die vorallem in der Bodenebene der Punktwolken ersichtlich werden.

Einige Leistungsmerkmale der Sensoren sind in der Tabelle 6.1 vergleichend festgehalten.

Tabelle 6.1. Vergleich der Eigenschaften verschiedener LiDAR-Sensoren [44][45][46]

Eigenschaft	**Ouster OS1**	**Livox Horizon**	**LS-S1**
Datenpunkt-Generierungsrate (pts/sec)	1.310.720	240.000/ 480.000	1.600.000
Sichtfeld Horizontal	360°	70,4°	120°
Sichtfeld Vertikal	42,4°	25°	25°
Laserwellenlänge	865 nm	905 nm	1550 nm

6.2 Anwendung der entwickelten Applikation

Um den Validierungsprozess mit der entwickelten Applikation zu veranschaulichen, wird im Folgenden eine detaillierte Anleitung zur Analyse der Messdaten anhand eines Beispiel-Szenarios präsentiert.

Schritt 1: Dateneingabe

Im ersten Schritt erfolgt die Übertragung der LiDAR-Messdaten sowie der Ground Truth Daten in die Applikation. Dies wird über den *Data Input*-Tab der Benutzeroberfläche durchgeführt. Nach erfolgreicher Eingabe der Daten überprüft die Applikation die Validität der Daten sowie deren Koinzidenz. Bei erfolgreicher Validierung werden entsprechende Meldungen in der Konsole ausgegeben, wodurch die weiteren Verarbeitungsschritte eingeleitet werden können.

Schritt 2: Starten der Visualisierung und Vorbereitung des MOT-Algorithmus

Nach der erfolgreichen Dateneingabe kann die Visualisierung im *MOT-Tab* der Applikation gestartet werden. Hierbei besteht die Möglichkeit, die Achsen auf das gewünschte Sichtfeld der Punktwolke anzupassen. Im unteren Bereich des Bildschirms wird die Anzahl der Ground Truth Objekte visualisiert. Anschließend kann das Szenario im Panel *Frame Limits* auf einen spezifischen zeitlichen Abschnitt eingeschränkt werden. Es empfiehlt sich, einen Abschnitt mit kohärenten Ground Truth Daten zu wählen, um eine konsistente Analyse zu gewährleisten.

Zusätzlich ist im Panel *Detector Region* ein relevanter Bereich auszuwählen, um den Messraum zu begrenzen. Durch diese Einschränkung wird die Punktwolke auf den definierten Bereich reduziert, in dem der MOT-Algorithmus operiert. Dies trägt dazu bei, Störungen innerhalb der Messdaten auf ein Minimum zu reduzieren.

Die Detekor-Region wurde anhand des Sensors mit der kleinsten horizontalen Sichtweite ausgewählt. Der Livox Sensor ist auf 70,4° beschränkt.

Bei einer festgelegte Messstrecke von $d = 50m$ und eine horizontale Sichtweite von $\alpha = 70.4°$ ergibt sich die Sichtbreite b mit der Gleichung 6.1.

$$b = 2 \times d \times \sin\left(\frac{\alpha}{2}\right) = 2 \times 50m \times \sin\left(\frac{70.4°}{2}\right) \approx 57.64m \tag{6.1}$$

Der maximale Sichtfeldpegel des Livox Horizon innerhalb einer Punktwolke wurde in Abbildung 6.7 veranschaulicht.

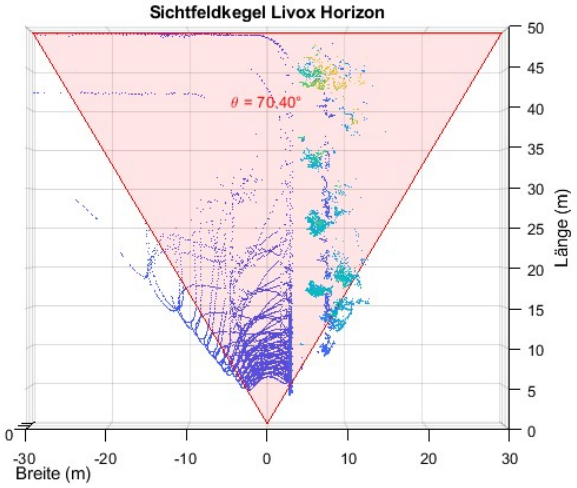


Abbildung 6.7. Sichtfeldkegel des Livox Horizon

Da durch die Referenzversuche festgestellt wurde, dass Detektionen bei $50m$ Sichtweite spätestens bei $40m$ Sichtbreite nicht mehr detektierbar sind, wurde der Messraum dementsprechend begrenzt. Der real nutzbare horizontale Sichtfeldwinkel des Livox-Sensors beträgt nach Gleichung 6.2 also nur etwa $47.2°$.

$$\alpha = 2 \times \arcsin\left(\frac{b}{2 \times d}\right) = 2 \times \arcsin\left(\frac{40m}{2 \times 50m}\right) \approx 47.2° \tag{6.2}$$

Um gleiche Rahmenbedingungen zu schaffen, wurde in jeder Messung ein gleich großer Messraum definiert. Nach der Betrachtung des kleinst möglichen Sichtbereichs wurde ein definierter Messraum von x = 22 m, y = 49 m und z = 8 m definiert. Die Messräume, die durch den Detektor-ROI definiert sind, wurden in Abbildung 6.8 dargestellt.

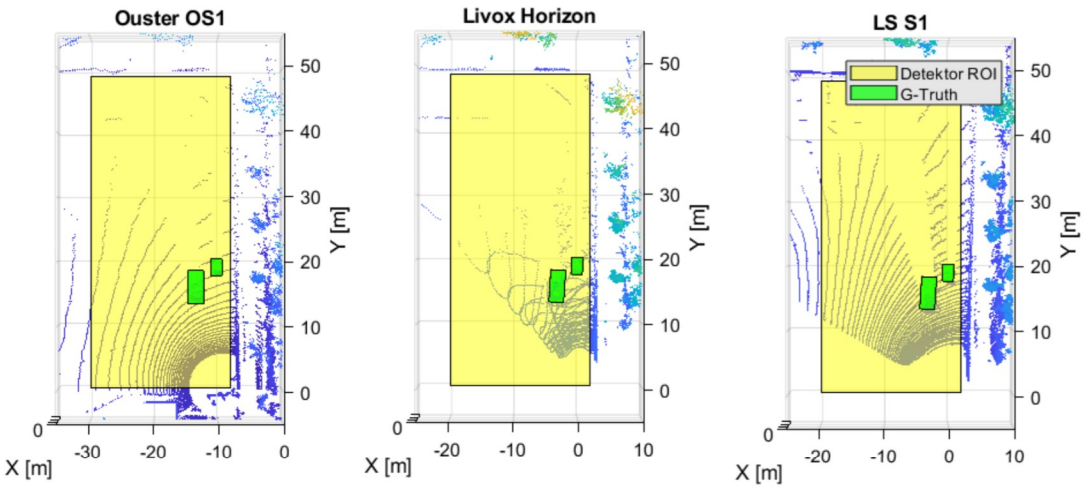


Abbildung 6.8. Festgelegter Messraum für alle Sensoren

Um vergleichbare Bedingungen für alle Sensoren zu gewährleisten, wurden bei der Mehrfachobjektverfolgung identische Parameter für den Detektor und den Tracker verwendet. Dadurch werden gleiche Anforderungen an die Punktwolken der verschiedenen Sensoren gestellt, was eine faire Bewertung ermöglicht. Es ist jedoch wichtig zu berücksichtigen, dass Abweichungen oder Unterschiede in der Leistung der Sensoren auch durch die Wahl der Parametrierung beeinflusst werden können. Jeder Sensor könnte unterschiedliche optimale Einstellungen benötigen, um unter den jeweiligen Bedingungen seine maximale Performance zu erreichen.

Die hier vorgestellte Analyse basiert bewusst auf gleichbleibenden Algorithmusbedingungen, um die Eignung der Sensoren für Mehrfachobjektverfolgungsanwendungen zu bewerten. Die GOSPA Metrik gibt dabei weniger Aufschluss darüber, wie präzise die Sensoren an sich arbeiten, sondern vielmehr darüber, wie gut sie für Anwendungen der Mehrfachobjektverfolgung geeignet sind.

Signifikante Unterschiede zwischen den Sensoren können hauptsächlich auf unterschiedliche Rauschstärken oder auf verschiedene Messmethodiken zurückgeführt werden, wie beispielsweise rotierende gegenüber stationären Sensoren. Diese Faktoren beeinflussen die Beschaffenheit der generierten Punktwolken und somit die Leistungsfähigkeit bei der Objekterkennung und -verfolgung. Selbst in einfachen Szenarien zeigt sich eine deutliche Abhängigkeit der Detektionsergebnisse von der verwendeten LiDAR-Technologie. Vorangegangene Testungen eines rasterkartenbasierten Detektionsalgorithmus zeigen eine signifikante Variation der Ergebnisse, die auf die Wahl des LiDAR-Messverfahrens zurückzuführen ist. Somit beeinflusst das Messverfahren maßgeblich die Genauigkeit und Zuverlässigkeit der Objektdetektion. [48][32]

Durch die quantitative Bewertung mithilfe der GOSPA Metrik können Stärken und Schwächen der einzelnen Sensoren hinsichtlich ihrer Eignung für die Mehrfachobjektverfolgung herausgearbeitet werden.

Für die Erkennung von Fahrzeugen wurden erwartete Dimensionen von $L = 4{,}7\,\text{m}$, $B =$

$1{,}8\,\text{m}$ und $H = 1{,}4\,\text{m}$ verwendet, während für Fußgänger Dimensionen von $L = 0{,}6\,\text{m}$, $B = 0{,}6\,\text{m}$ und $H = 1{,}7\,\text{m}$ herangezogen wurden. Die Auswahl zwischen den Objekttypen erfolgt durch die Einstellung *ObjType* im *Settings Tab*.

Tabelle 6.2 und 6.3 listet die verwendeten Parameter auf, die im *Settings Tab* konfiguriert wurden.

Tabelle 6.2. Parametereinstellungen für den Detektor

Parameter	**Wert**
Segmentation min. distance	0.5 m
min. detections per Cluster	10
ground max. distance	0.25 m
max. ground angular distance	5 m
max. Z dist. Cluster	3 m
min Z dist. Cluster	-3 m
ego vehicle radius	3 m

Tabelle 6.3. Parametereinstellungen für den JPDA-Tracker mit IMM

Parameter	**Wert**
assignmentGate (C_1, C_2)	50, 100
confirmationGate (M, N)	7, 10
deletionGate (P, R)	9, 12
ClutterDensity	1×10^{-9}
HitMissThreshold	0.5
MaxNumDetections	∞
MaxNumTracks	50
ObjType	Car / Human

Schritt 3: Starten der Modelle und Validierung der Ergebnisse

Nach der Parametrierung kann das Detektor- bzw. Trackermodell gestartet werden. Die Sequenz der Punktwolken wird animiert dargestellt, wobei die Detektionen und Tracks frameweise visualisiert werden. Im unteren Bereich der Applikation wird nach Abschluss der Modellberechnungen eine erste Einschätzung der Ergebnisse ermöglicht. Die gelbe Kurve im Diagramm stellt die Anzahl der Detektionen dar (vgl. Anlage A-2). Sollten hierbei zahlreiche Extremwerte oder generell zu hohe Detektionszahlen auftreten, ist eine erneute Parametrierung und Anpassung des Detektormodells erforderlich.

Die Anzahl der Tracks wird durch rote Balken visualisiert. Eine gute Übereinstimmung zwischen den Ground Truth Daten (grün) und den Tracks (rot) deutet auf eine plausible und effektive Verfolgung der Objekte hin (vgl. Anlage A-2). Sind die Ergebnisse des Verfolgungsalgorithmus zufriedenstellend, kann im nächsten Schritt die Berechnung der Metrik erfolgen.

Schritt 4: Metrikanalyse

Im finalen Schritt besteht die Möglichkeit, im *Metric Analysis* Tab die GOSPA- und OSPA-Metriken berechnen zu lassen. Zusätzlich werden die Komponenten beider Metriken als

Graphen dargestellt. Über das Panel *Metric Settings* kann der Benutzer festlegen, welche Komponenten angezeigt oder verborgen werden sollen. Zudem besteht die Option, den Frame-Intervall der Achsen zu bestimmen. Sollte die berechnete Metrik plausibel erscheinen, kann der Benutzer die Ergebnisse als MATLAB-Datei speichern. Hierzu müssen ein Speicherordner sowie ein Dateiname festgelegt werden. Desweiteren ist die Erstellung von unterstützenden MATLAB-Visualisierungen über den Knopf *Visualize MOT results* möglich. Die Abbildungen können als MATLAB-Figuren durch Betätigung eines Knopfes in der Benutzeroberfläche in einen beliebigen Ordner gespeichert werden. Das Dateiformat ermöglicht eine einfache Nachbearbeitung und kann später in Bilddateien umgewandelt werden. Diese Abbildungen werden im folgenden auch zur Auswertung herangezogen.

Durch die strukturierte Anwendung der entwickelten Applikation wird eine systematische und nachvollziehbare Analyse der Messdaten ermöglicht, welche essenziell für die Validierung der LiDAR-Sensoren und der Verfolgungsalgorithmen ist.

6.3 Auswertung

Für die Leistungsbewertung der verschiedenen LiDAR-Sensoren wurden zwei unterschiedliche Validierungsszenarien herangezogen, welche durch B. Sc. Mohammad Ammad im Rahmen eines Gaststudentenprogramms auf dem Prüffeld der HTW Dresden aufgenommen wurden. Die Aufnahmen sowie die annotierten Datensätze für die Grundwahrheiten dienen als Grundlage für die in dieser Arbeit durchge- führten Auswertung mittels der entwickelten Applikation. [40]

6.3.1 Beschreibung der Validierungs-Szenarien

Vor der Durchführung der eigentlichen Analyse war eine sorgfältige Sortierung der Ground Truth Daten und der LiDAR-Messungen notwendig. Die Dateien waren ursprünglich nicht im erwarteten numerischen Format (z. B. 0001, 0002, 0003) benannt, sondern folgten einem fehlerbehafteten Schema (z.B.1, 2, 3, ...). Diese abweichende Benennung führte dazu, dass die Daten beim Einlesen in MATLAB in der ASCII-Reihenfolge sortiert wurden (10, 100, 101, . . . , 109, 11, 112). Konkret kam es dadurch zu einer fehlerhaften Zuordnung der Datensätze, sodass die Ground Truth Daten sowie die Punktwolken-Frames an falschen Positionen im Datenbestand landeten. Um die korrekte Reihenfolge sicherzustellen und fehlerhafte Zuordnungen zu vermeiden, wurde die Funktion `natsort` in MATLAB verwendet, um die Dateien in natürlicher Reihenfolge einzulesen. [49]

Diese Vorverarbeitung war essenziell, um die Integrität der Daten sicherzustellen und die Grundlage für eine valide Leistungsbewertung der LiDAR-Sensoren sowie die anschließende Validierung der Verfolgungsalgorithmen zu schaffen. Ohne diese Anpassung wären die Ergebnisse der Analyse durch fehlende oder falsch zugeordnete Daten verfälscht worden, was die Aussagekraft der gesamten Auswertung erheblich beeinträchtigt hätte. Für

zukünftige Arbeiten ist es von wesentlicher Bedeutung, die Datensätze in einem standardisierten numerischen Format oder unter Verwendung üblicher Zeitstempel-Formate zu benennen, die eine eindeutige Reihenfolge der Datensätze garantieren.

Im Folgenden werden die beiden betrachteten Szenarien detailliert beschrieben:

Szenario 1: Überholmanöver mit Spurwechsel bei Fahrzeugen

In diesem Szenario wurde ein Überholmanöver zwischen zwei Fahrzeugen auf einer definierten Messstrecke aufgezeichnet. Zu Beginn des Versuchs befand sich Fahrzeug A auf der linken Fahrspur und bewegte sich mit konstanter Geschwindigkeit. Fahrzeug B näherte sich von hinten auf derselben Spur mit einer höheren Geschwindigkeit. Um eine Behinderung des nachfolgenden Verkehrs zu vermeiden, wechselte Fahrzeug A die Fahrspur nach rechts. Dieser Spurwechsel ermöglichte es Fahrzeug B, ohne Verzögerung auf der linken Spur weiterzufahren und Fahrzeug A zu überholen. Nach Abschluss des Überholvorgangs setzte Fahrzeug B seine Fahrt auf der linken Spur fort, während Fahrzeug A auf der rechten Spur weiterfuhr. Beide Fahrzeuge bewegten sich somit aneinander vorbei, wobei Fahrzeug B danach vor Fahrzeug A lag.

Eine Darstellung der Grundwahrheiten kann mit der erweiterten Analyse erfolgen. Die ungefähre Trajektorie der Fahrzeuge lassen sich durch Abbildung der Cuboid-Daten abbilden. Anzumerken ist hierbei, das die Grundwahrheiten des rechten Fahrzeuges ab 35 m in einer Messung nicht mehr annotiert wurden. Dies wird in der weiteren Auswertung berücksichtigt. Der Ablauf des Fahrmanövers ist schematisch für jeden Sensor in Abbildung 6.9 dargestellt.

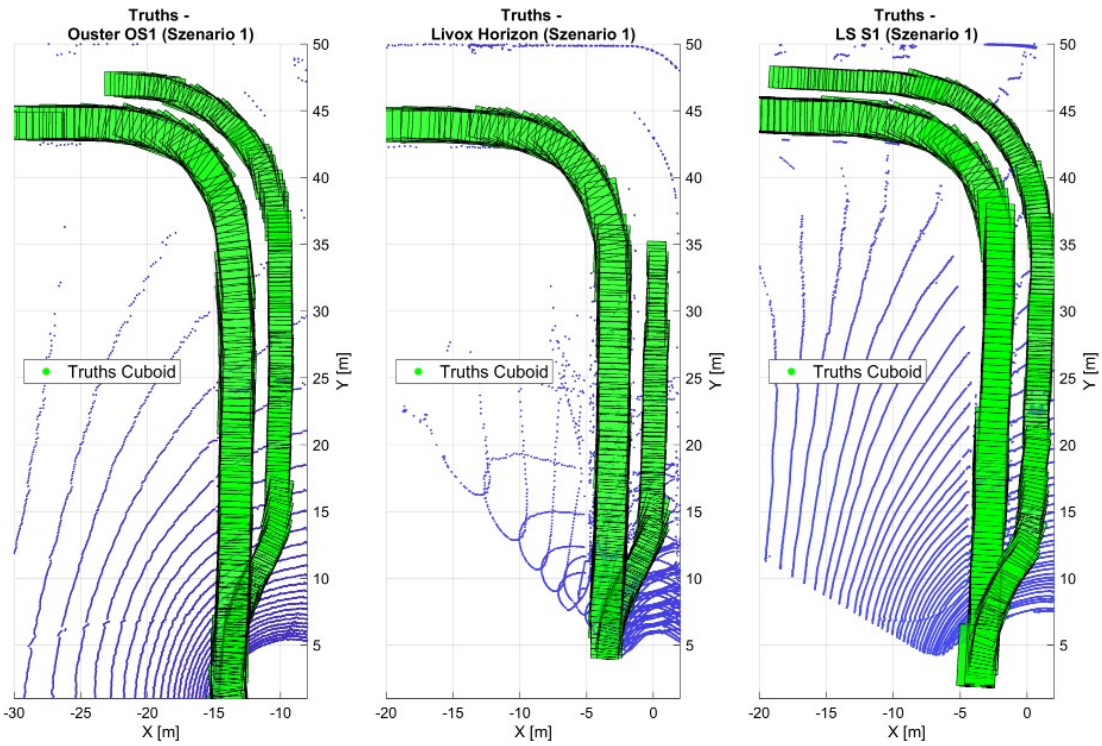


Abbildung 6.9. Grundwahrheiten (Szenario 1)

Szenario 2: Überholmanöver bei Fußgängern

Das zweite Szenario konzentriert sich auf das Bewegungsverhalten von drei Fußgängern auf der Fahrbahn. Die Personen starteten in einer Reihe hintereinander und bewegten sich gemeinsam in eine Richtung. Während des Gehens blieb die vorderste Person plötzlich stehen. Daraufhin entschieden sich die beiden hinteren Personen, die stehende Person zu überholen. Sie passierten diese, indem sie ihre Position seitlich versetzten und setzten ihren Weg fort. Während sich die zwei Personen vom Sensor entfernen, vergrößern und verringern sie dabei ständig ihren seitlichen Abstand zueinander. Nach einer bestimmten Distanz kehrten die beiden überholenden Personen um und liefen zurück in Richtung des Ausgangspunkts. Auf dem Rückweg ordneten sie sich wieder hinter der zunächst stehengebliebenen Person ein, wodurch die ursprüngliche Reihenfolge der Gruppe wiederhergestellt wurde. Anschließend setzten alle drei Personen ihren Weg fort und beendeten das Szenario in der ursprünglichen Formation.

Der Ablauf des Fußgänger-Manövers ist schematisch in Abbildung 6.10 durch die Darstellung der Grundwahrheiten visualisiert.

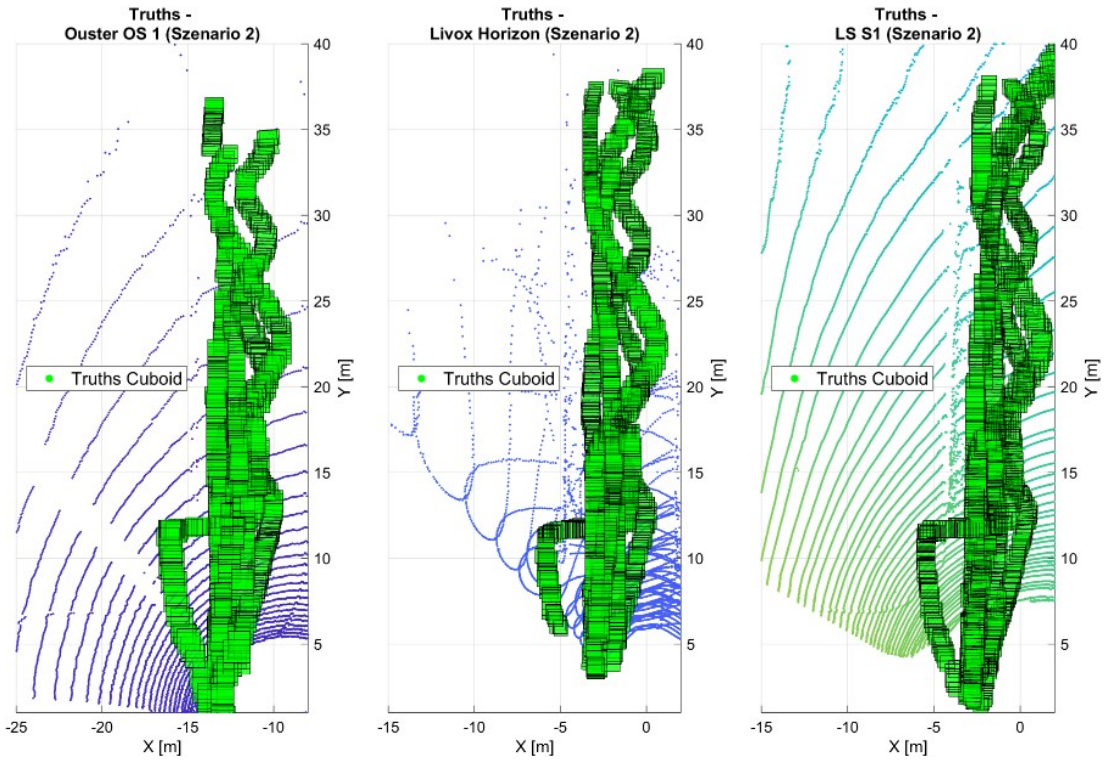


Abbildung 6.10. Grundwahrheiten (Szenario 2)

Beide beschriebenen Szenarien repräsentieren realitätsnahe Manöver, bei denen die Objekte (Fahrzeuge bzw. Fußgänger) eng beieinander agieren. Diese enge räumliche Nähe stellt besondere Anforderungen an die Objektverfolgung durch einen MOT-Algorithmus. Insbesondere die Fähigkeit, einzelne Objekte trotz Überlappungen und schnellen Bewegungsänderungen präzise zu verfolgen, ist entscheidend für die Bewertung der Leistungsfähigkeit der eingesetzten LiDAR-Sensoren.

Die Verwendung dieser Szenarien ermöglicht eine Analyse der Tracking-Leistung unter realistischen Bedingungen und trägt somit zur Validierung der entwickelten Applikation bei. Durch den Vergleich der Messergebnisse mit den Ground Truth Daten kann die Genauigkeit der Sensoren sowie die Effektivität der Verfolgungsalgorithmen beurteilt werden.

6.3.2 Detektierbarkeit und Verfolgung: Szenario 1

Die Visualisierung der Detektierbarkeit sowie der Verfolgung der Fahrzeuge kann in der Applikation mit den erweiterten Analyse-Darstellungen geschehen. Abbildung 6.11 zeigt die Detektionen der Objekte über den gesamten Messzeitraum. Diese Darstellung verdeutlicht die Problematik des Schrumpfens (vgl. Kapitel 3.2 und 5.4.1) der Detektionsdimensionen bei steigender Entfernung der Objekte zum Sensor. Auch scheint das Fahrzeug was sich auf der rechten Fahrbahn befindet in allen drei Punktwolken der Sensoren bei einer Entfernung ca. 40 m schlechter detektierbar zu sein. Dies kann vor Allem durch Überdeckung des anderen Fahrzeugen erklärt werden. Desweiteren sind zahlreiche Detektionen der Bodenebene bei Ouster und LS ersichtlich.

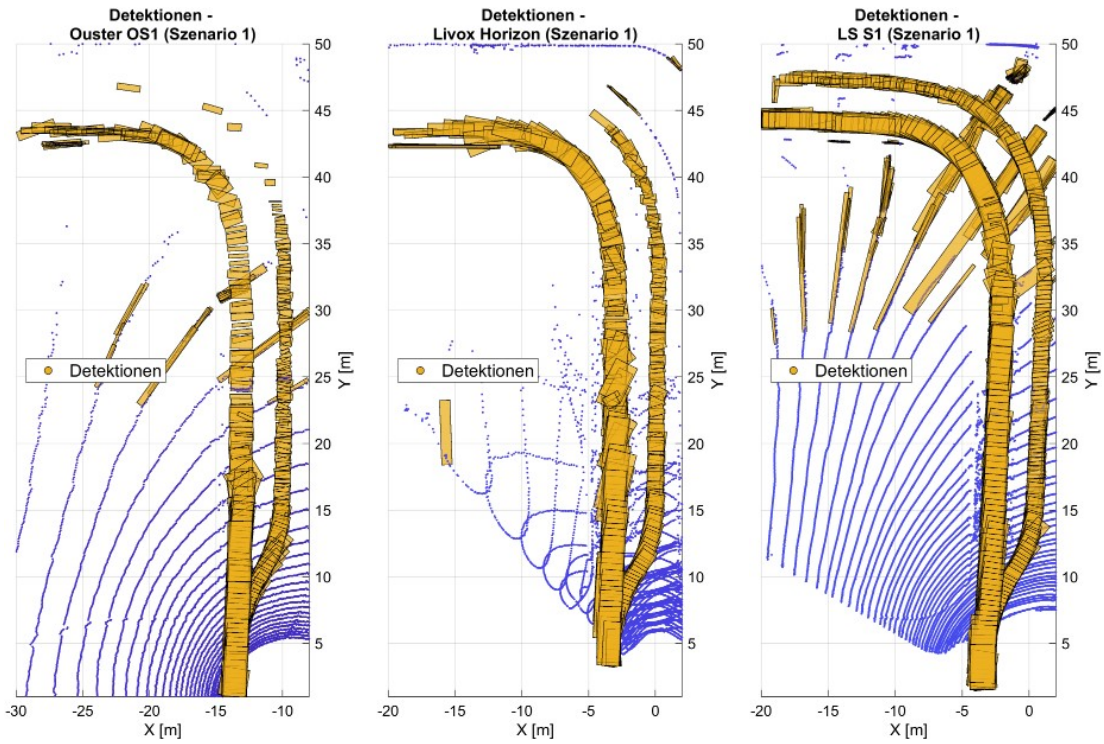


Abbildung 6.11. Ergebnisse der Detektionen (Szenario 1)

Nach der Detektion der Objekte erfolgt die Verfolgung mittels JPDA-Trackers mit IMM-Filter. Die Begrenzungsrahmen bzw. Cuboid-Daten der Tracks wurden in Abbildung 6.12 dargestellt. Desweiteren lassen sich die ungefähren Trajektorien der Tracks anhand der Cuboid-Daten darstellen. Bei der Punktwolke des Livox-Sensors ist eine starke Überlappung der Quader bei etwa 25 m bis 35 m zu verzeichnen. Dies lässt darauf schließen, das der Algorithmus hier Probleme hat, die Objekte aufgrund der örtlichen Nähe zueinander klar zu trennen. Bei allen drei Punktwolken tritt das Phänomen auf, das die Tracks nach Kurven-

ausfahrt falsch orientiert sind. Der Algorithmus detektiert hierbei die linke Fahrzeugflanke und interpretiert dies als Heck.

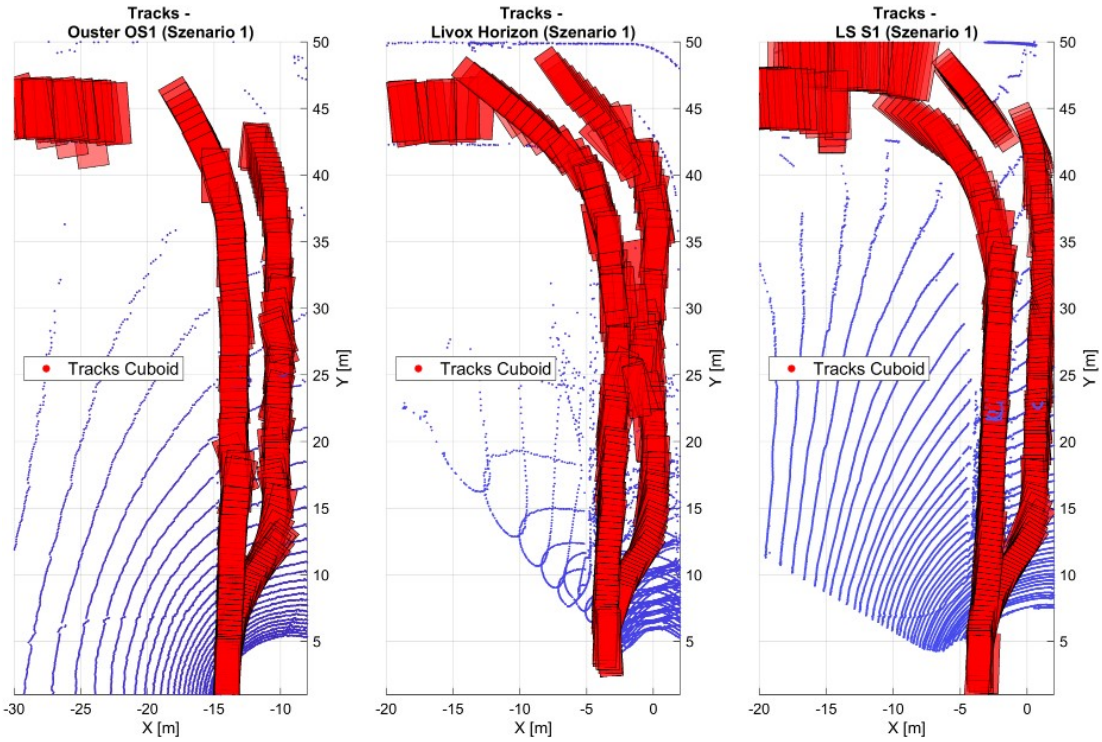


Abbildung 6.12. Ergebnisse der Tracks (Szenario 1)

Die Trajektorien der Tracks und Truths, dargestellt durch die Schwerpunkte der Quader, können nun übereinander gelegt werden um diese mit einander zu vergleichen. In Abbildung 6.13 ist zu erkennen, dass die Verfolgung in allen 3 Punktwolkenmessungen zufriedenstellend ist. Besonders im Messbereich ab 40 m sind größere Abweichung der Trajektorien zu erkennen.

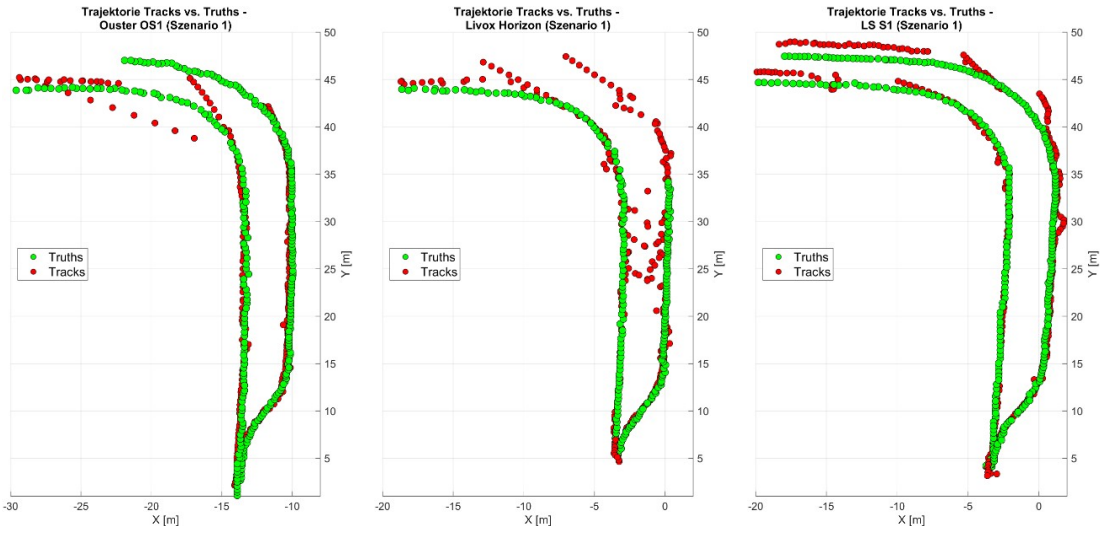


Abbildung 6.13. Vergleich der Trajektorien (Szenario 1)

Die erweiterte Analyse liefert in Abbildung 6.14 ein Balkendiagramm mit Informationen

über die Gesamtanzahlen der Detektionen, Verfolgungen und Grundwahrheiten. Hier kann deutlich erkannt werden, dass der LS-Sensor die meisten Detektionen hervorbringt. Desweiteren sind die Defizite der Grundwahrheiten bei der Livox-Messung klar erkennbar. Dennoch sind die Gesamtanzahlen der Tracks bei allen drei Sensoren ähnlich groß. Dies Unterstreicht die Plausibilität der Tracking-Ergebnisse.

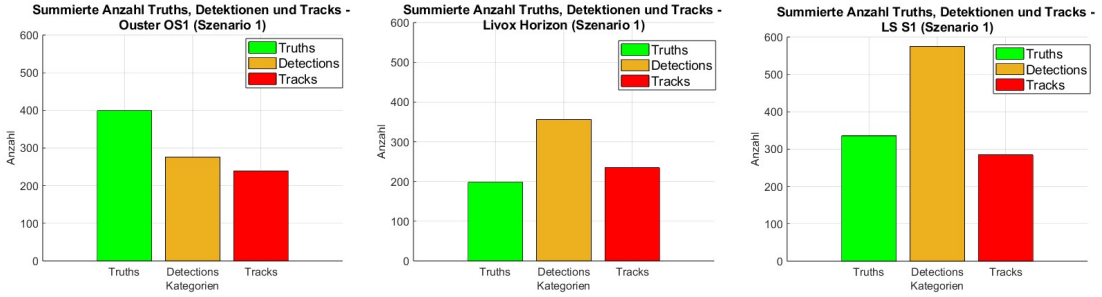


Abbildung 6.14. Vergleich der summ. Anzahlen (Szenario 1)

6.3.3 Detektierbarkeit und Verfolgung: Szenario 2

Die Detektierbarkeit der Personen aus Szenario 2 kann mithilfe der Abbildung 6.15 der erweiterten Analyse visualisiert werden. Ähnlich wie bei Szenario 1 nehmen die Detektionen bei Entfernung der Fußgänger vom Sensor ab. Teilweise werden auch Bereiche der Bodenebene als Objekte detektiert. Dies wird besonders bei den Detektions-Ergebnissen von Ouster und LS beobachtet.

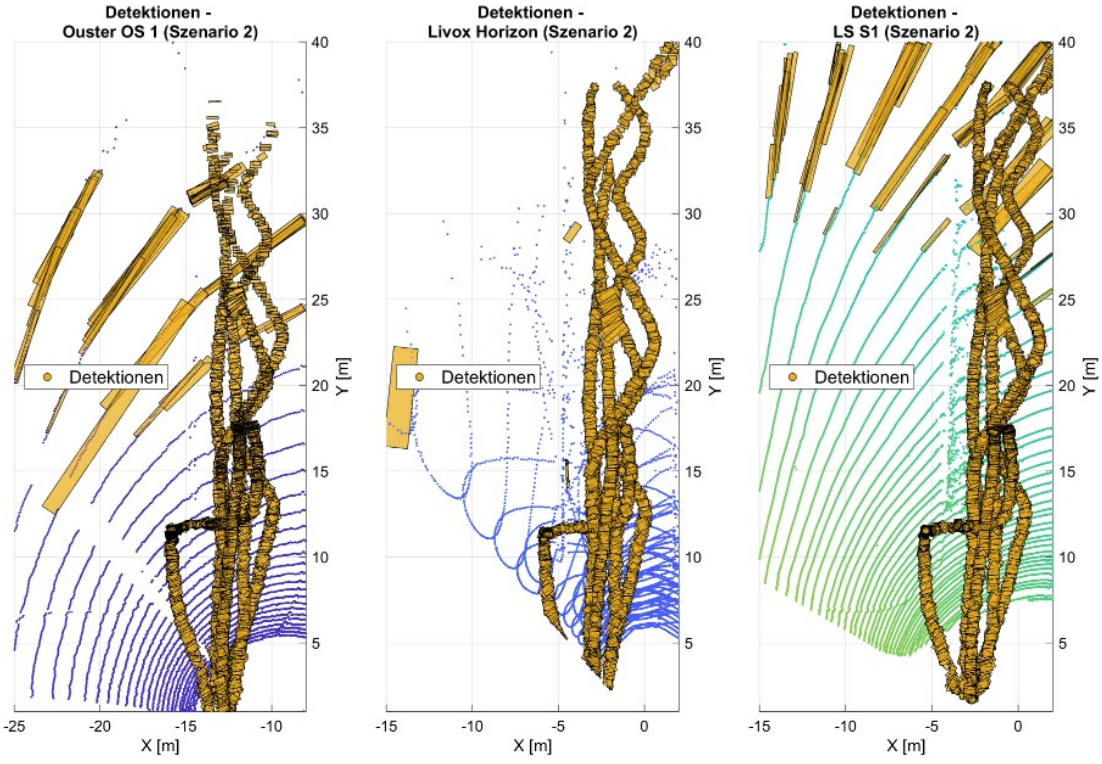


Abbildung 6.15. Ergebnisse der Detektionen (Szenario 2)

Die Tracking-Ergebnisse für die Verfolgung der Personen ist in Abbildung 6.16 dargestellt.

Die Begrenzungsrahmen sind analog zum vorangegangenen Beispiel visualisiert.

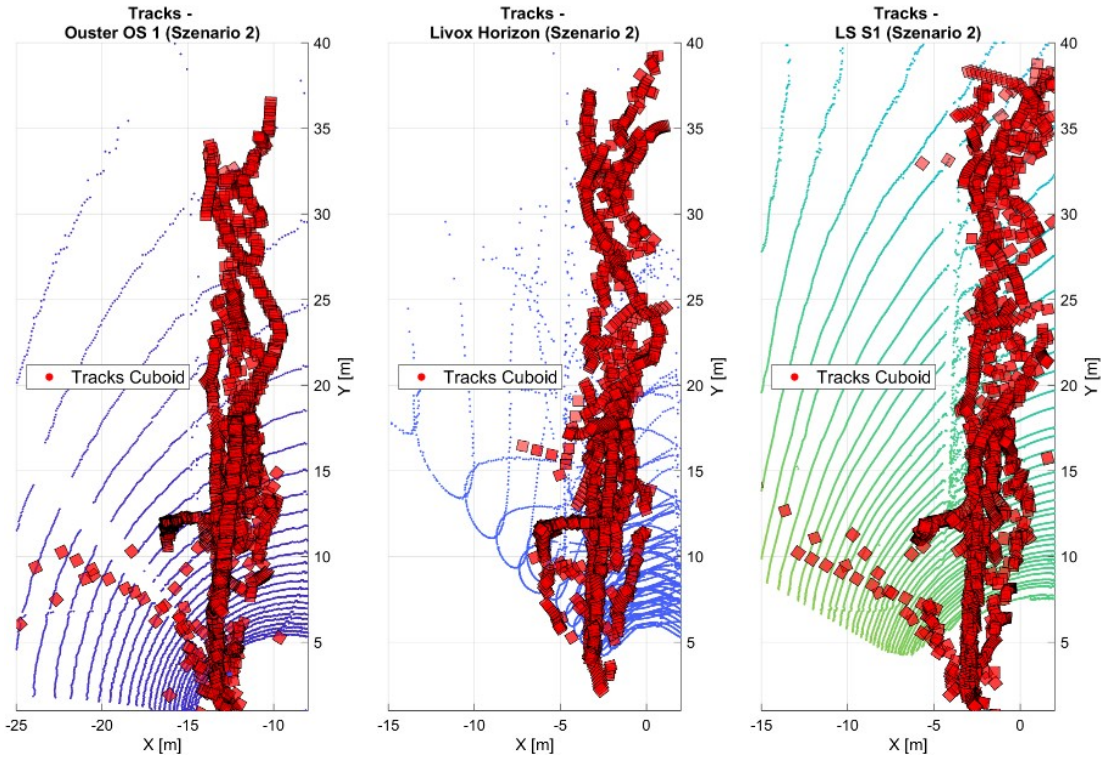


Abbildung 6.16. Ergebnisse der Tracks (Szenario 2

Der Vergleich der Trajektorien der Tracks und Truths in Abbildung 6.17 veranschaulicht die Verfolgungsleistung mit Personen. Hier sind starke Ausreißer mit zunehmender Entfernung der Objekte zum Sensor zu verzeichnen.

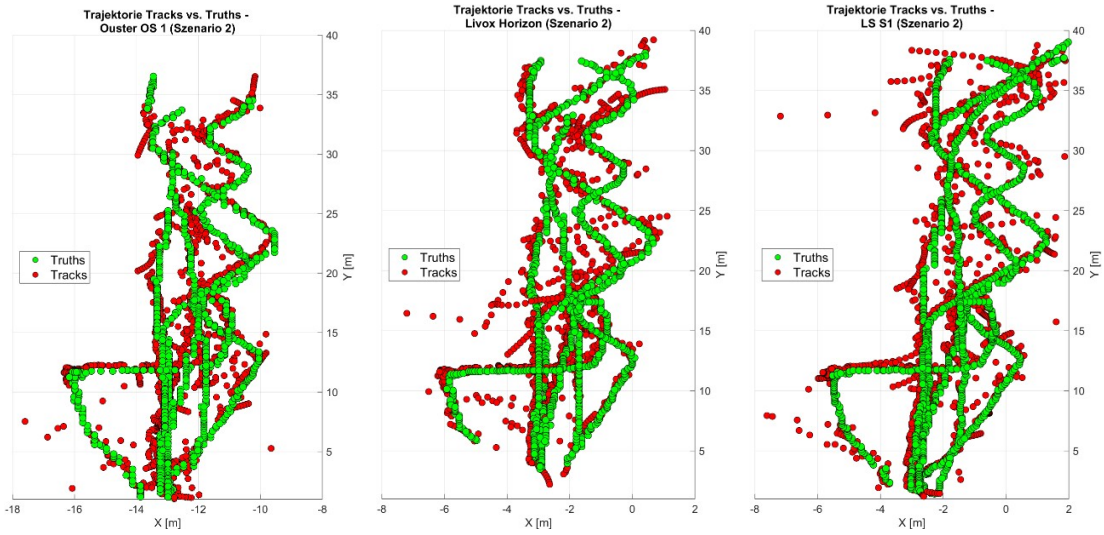


Abbildung 6.17. Vergleich der Trajektorien (Szenario 2)

Die summierten Anzahlen aller Wahrheiten, Detektionen und Verfolgungen sind in Abbildung 6.18 visualisiert. Erneut verzeichnet der LS-Sensor die meisten Detektionen. Dies ist hauptsächlich mit der Vielzahl an Fehldetektionen durch Bodenebenen-Bereiche zu erklären. Die Anzahlen der Grundwahrheiten ist bei allen Messungen ähnlich hoch. Das lässt

darauf schließen, dass in allen Messungen eine gute Annotation der Grundwahrheiten statt gefunden hat. Die Anzahlen der Tracks sind in allen Messungen ähnlich, was erneut für die Plausibilität des Verfolgungsalgorithmus steht.

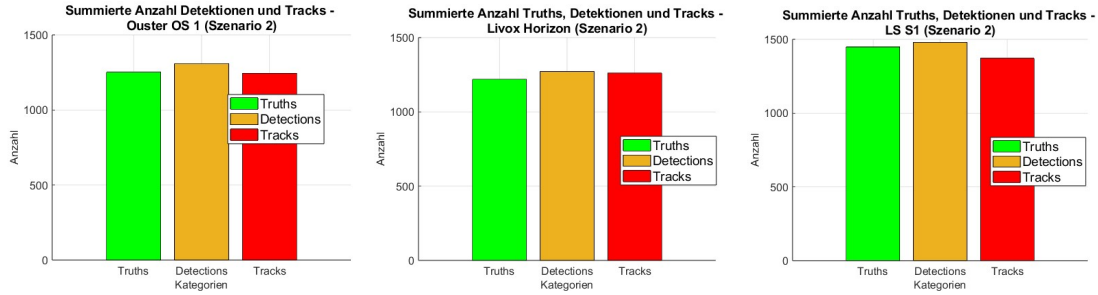


Abbildung 6.18. Vergleich der summ. Anzahlen (Szenario 2)

6.3.4 Analyse der GOSPA-Metrik

Nach der Durchführung der Mehrfachobjektverfolgung für beide Szenarien wurden die Metrik-Kurven für die jeweiligen Sensoren berechnet und in Abbildung 6.19 dargestellt. Die Berechnung der Metrik und deren Komponenten erfolgte unter Verwendung der Parameter *cutoff-distance* = 1 m und *switching-penalty* = 0.25, welche in der App parametriert werden können.

Der Switch-Penalty-Parameter bestimmt, wie stark Identitätswechsel in die GOSPA Metrik eingehen. Höhere Werte erhöhen die Bestrafung für solche Wechsel, wodurch die Metrik empfindlicher auf Tracking-Fehler reagiert. Der Cutoff-Distanz-Parameter legt fest, ab welcher Distanz Fehler nicht weiter ansteigen. Er begrenzt den Einfluss großer Lokalisierungsfehler, indem er deren Beitrag zur Gesamtkostenfunktion deckelt. Wenn der Cutoff-Wert beispielsweise auf 1 gesetzt wird, entspricht dies einer maximalen Fehlerdistanz von 1 Meter. Durch die Anpassung dieser Parameter kann die Bewertung flexibler gestaltet werden, da sie es ermöglichen, die Sensitivität der Metrik gegenüber bestimmten Fehlerarten zu steuern. So kann die Metrik an spezifische Anforderungen angepasst werden.

Bei der Auswertung aller Sensoren wurden die GOSPA-Kurven über eine normalisierte Zeitachse dargestellt. Aufgrund der unterschiedlichen Anzahl von Frames in jeder Messung war es notwendig, die Zeit zu normalisieren, um eine vergleichbare und aussagekräftige Darstellung zu erhalten.

Wie zu erwarten, schließen alle Sensoren ähnlich bei der Bewertung mit der GOSPA Metrik ab. Hierbei weist ein niedrigerer Metrik-Kostenwert auf eine höhere Verfolgungsqualität hin.

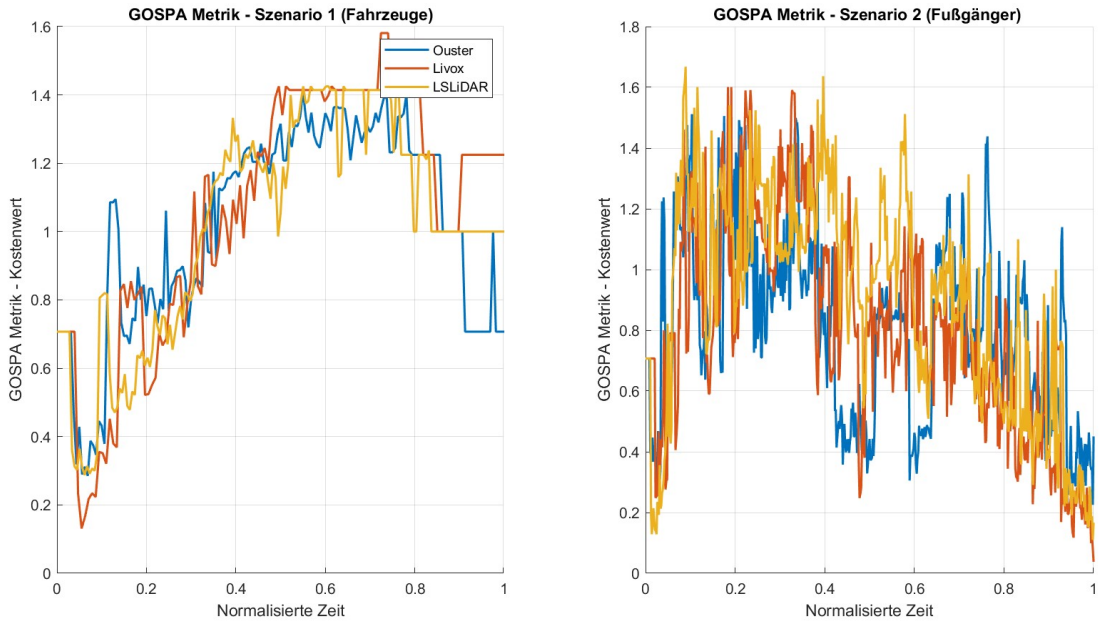


Abbildung 6.19. Vergleich der GOSPA-Metrik von Szenario 1 und 2

Um die Unterschiede zwischen den Sensoren deutlicher hervorzuheben und die Auswirkungen von Ausreißern zu reduzieren, wurden die Kurven zusätzlich stark geglättet. Hierbei kam ein gaußsches Glättungsverfahren zum Einsatz, das durch die Anwendung einer Gauß-Funktion als Gewichtungsfunktion nahegelegene Datenpunkte stärker berücksichtigt und so eine effektive Glättung erreicht. Die interpolierte und geglättete Kurve ermöglicht es, die GOSPA-Metrik vergleichend in Abbildung 6.20 zu betrachten.

Die Betrachtung der geglätteten Metrik-Kurven suggeriert, das die Messung des Ouster-Sensor in Szenario 1 die beste Leistung im Verfolgungsprozess erreicht. In Szenario 2 weist der Ouster bis zur norm. Zeit 0.6 ebenfalls das beste Ergebnis auf. Es ist anzumerken, dass die Metrik-Kurve des Livox-Sensors in Szenario 1 durch die fehlenden Grundwahrheiten negativ beeinflusst wurde. Bei vollständiger Annotation der Wahrheiten hätte evtl. ein besseres Ergebnis für die Livox-Messung erzielt werden können.

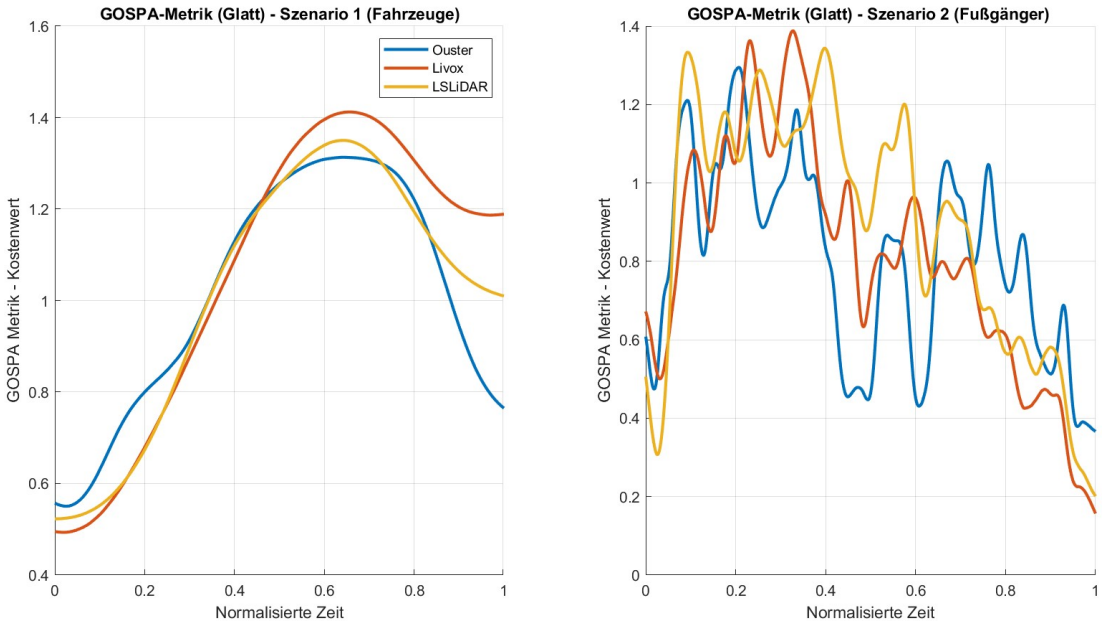


Abbildung 6.20. Vergleich der interpolierten GOSPA-Metrik von Szenario 1 und 2

In **Szenario 1** ist festzustellen, dass die GOSPA Metrik mit zunehmender Entfernung der Objekte zunächst ansteigt und gegen Ende der Messung wieder abfällt. Die Maximalwerte der Metrikkurven treten während der Phasen der Kurvenein- und -ausfahrt der Fahrzeuge auf. Dieses Verhalten kann darauf zurückgeführt werden, dass der Tracking-Algorithmus während der Kurvenfahrt Schwierigkeiten hat, die Fahrzeuge präzise zu detektieren, was zu erhöhten Fehlerwerten in der Metrik führt. Nach Abschluss der Kurvenfahrt sinken die Werte der GOSPA Metrik, da der Algorithmus die Fahrzeuge an der linken Fahrbahnseite wieder zuverlässig erfassen kann. Dies deutet darauf hin, dass die Performance des Trackings stark von der Bewegungsdynamik der Objekte und der damit verbundenen Sichtbarkeit beeinflusst wird.

In **Szenario 2** zeigt die GOSPA Metrik einen ansteigenden Verlauf, der auf die zunehmende Distanz zwischen Sensor und Fußgängern zurückzuführen ist. In den Bereichen der normalisierten Zeit zwischen 0,4 und 0,6 ist ein Abfall der Werte zu beobachten. Diese Schwankung kann dadurch erklärt werden, dass sich zwei der Fußgänger außerhalb des Sichtfeldes des Sensors befanden und keine Ground Truth Daten zur Verfügung standen. Der dritte Fußgänger befand sich hingegen in einer näheren Distanz von 20 bis 30 Metern zum Sensor. Da entfernte Objekte tendenziell zu höheren Fehlerwerten führen, sinken die Metrik-Werte signifikant, sobald sich die Fußgänger außerhalb der Detektierbarkeit aufhalten und keine Referenzdaten bzw. Grundwahrheiten mehr vorhanden sind. Beim Eintreten der zwei entfernten Personen steigt der Wert wieder kurz an. Beim Rückweg aller Personen in Richtung des Sensors nehmen die Metrik-Kostenwerte wieder ab, was auf eine verbesserte Lokalisierung der Fußgänger im Bereich von 0 bis 30 Metern zurückzuführen ist. Dieses Verhalten unterstreicht die Bedeutung der Objektdistanz für die Genauigkeit des Trackings.

Um die Vergleichbarkeit der Sensoren zu erhöhen, wurde der durchschnittliche GOSPA-

Wert pro Sensor kategorisiert nach Szenario ermittelt. Die Ergebnisse dieser Berechnungen sind in Abbildung 6.21 dargestellt und deuten darauf hin, dass alle Sensoren bei der Verfolgung der Fahrzeuge bessere Leistungswerte erziehlten.

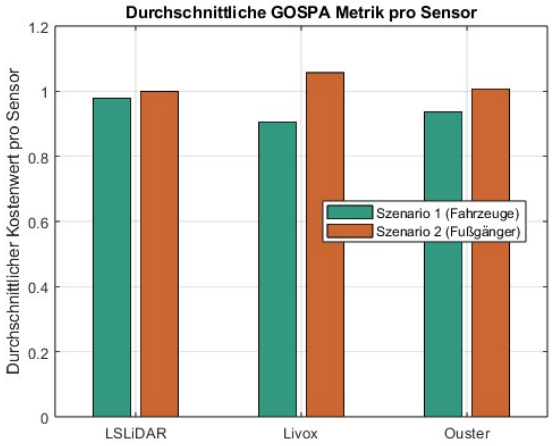


Abbildung 6.21. Vergleich der durchschnittlichen GOSPA Metrik pro Sensor

Bei der Berechnung des durschnittlichen Kostenwertes wurden alle fehlenden Werte (NaN) aus den Datensätzen ausgeschlossen, da sie ungültige oder nicht vorhandene Messungen darstellen. Diese fehlenden Werte treten zum Beispiel an Stellen auf, wo keine Ground Truth Daten vorhanden sind. Die Durchschnittsberechnung erfolgte somit ausschließlich über die gültigen und vorhandenen Messwerte. Diese Vorgehensweise stellt sicher, dass die berechneten Durchschnittswerte die tatsächliche Performance der Sensoren reflektieren, ohne durch fehlende Daten verzerrt zu werden.

Es ist zu beachten, dass Durchschnittswerte von interpolierten und geglätteten Kurven abweichen können. Bei der Interpolation und Glättung werden fehlende oder ungültige Datenpunkte durch geschätzte Werte ersetzt, wobei ein gaußsches Glättungsverfahren verwendet wurde, das nahegelegene Datenpunkte stärker gewichtet. Während die Durchschnittsberechnung nur auf den gemessenen gültigen Werten basiert, enthalten die geglätteten Kurven auch interpolierte Daten.

Die geglätteten Kurven ermöglichen eine Einschätzung der Sensorleistung zu spezifischen Zeitpunkten, indem sie lokale Trends und Schwankungen abbilden. Der Durchschnittswert hingegen liefert eine Gesamtbewertung über die gesamte Messdauer ohne detaillierte zeitliche Variationen. Daher können Sensoren je nach Bewertungsmethode unterschiedlich abschneiden, da geglättete Ergebnisse und Durchschnittswerte voneinander abweichen. Dies verdeutlicht, dass die Wahl der Bewertungsmethode die Interpretation der Sensorleistung erheblich beeinflusst. Es ist daher wichtig, sowohl geglättete Kurven als auch Durchschnittswerte in die Analyse einzubeziehen, um ein umfassendes und ausgewogenes Verständnis der Messergebnisse zu erlangen.

6.3.5 Analyse der Lokalisierungskomponente

Eine Fehlerkomponente die in der GOSPA Metrik inkludiert ist, ist der Lokalisierungsfehler. Dieser wird zusätzlich zur Metrik betrachtet um eine Bewertung der Lokalisierungsgenauigkeit zu liefern.

In Abbildung 6.22 sind die Kurven der Lokalisierungskomponente für beide Szenarien dargestellt. Es ist dabei zu berücksichtigen, dass die Nullwerte in den Kurven auf fehlende Tracks oder Grundwahrheiten zurückzuführen sind. Fehlen in einem Frame entweder eine oder beide dieser Komponenten, kann der Lokalisierungsfehler zwischen Track und Ground Truth nicht berechnet werden. Diese Nullwerte wurden bei der weiteren Auswertung der Lokalisierungsergebnisse ausgeschlossen, da sie keine Aussage über die tatsächliche Leistungsfähigkeit der Lokalisierung zulassen.

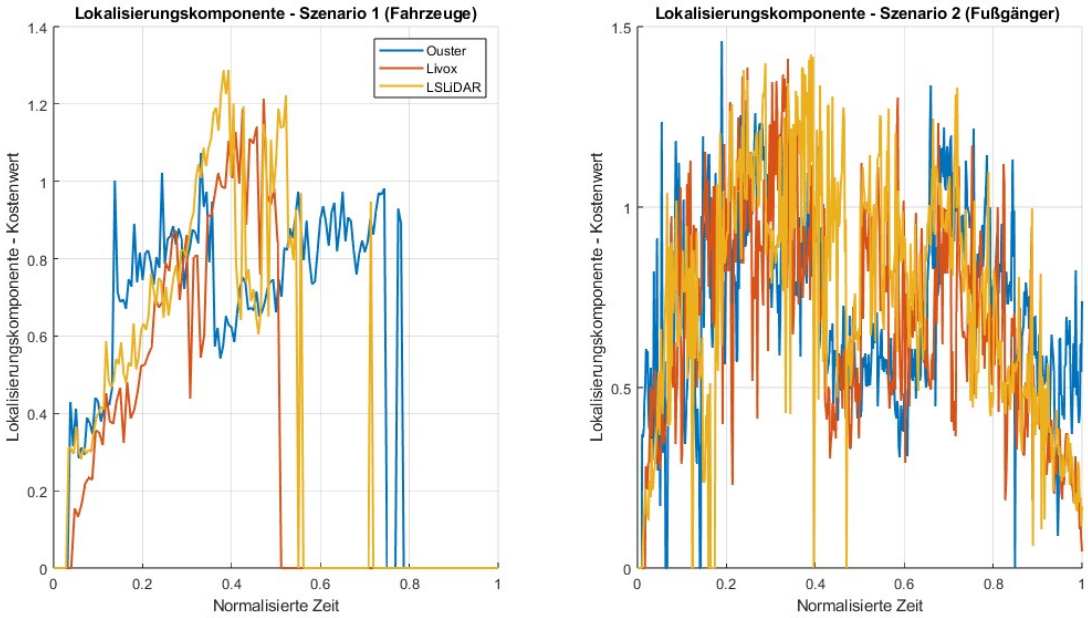


Abbildung 6.22. Vergleich der Lokalisierungskomponente von Szenario 1 und 2

Nach Interpolation über fehlende Datenpunkte (Nullwerte) und einer Glättung der Kurve kann die Lokalisierungskomponente vergleichend in Abbildung 6.23 betrachtet werden.

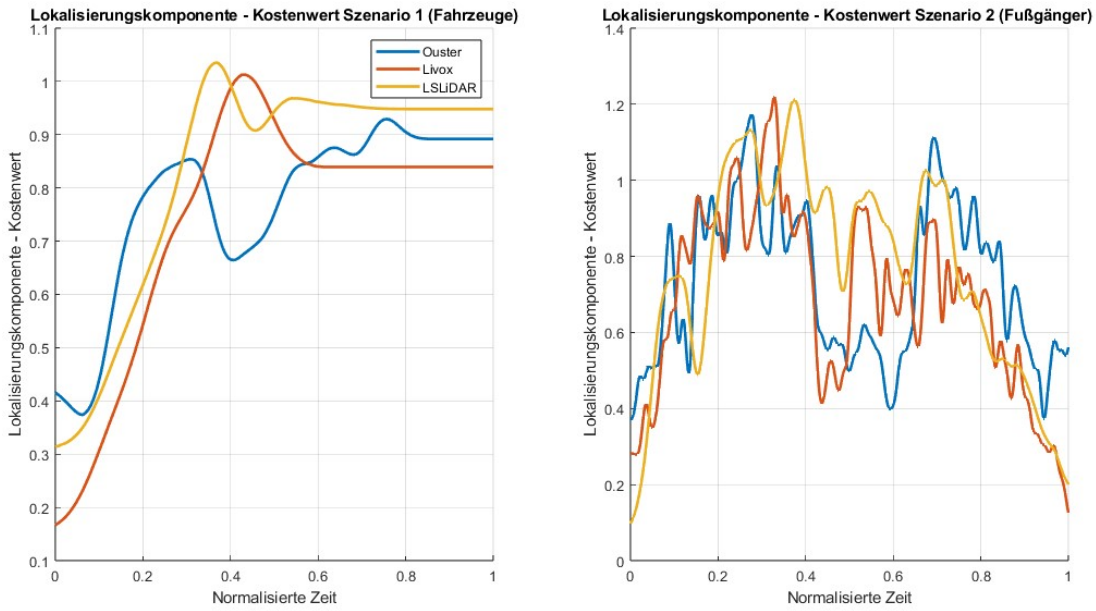


Abbildung 6.23. Vergleich der interpolierten Lokalisierungskomponente von Szenario 1 und 2

In **Szenario 1** steigt der Lokalisierungsfehler bei Entfernung der Objekte zum Sensor. Im Bereich der normalisierten Zeit von 0.2 bis 0.6 treten Schwankunken des Fehlers auf, die unmittelbar auf das Überholmanöver und die entstehende Nähe der Objekte in diesem Bereich zurückzuführen ist sind. Bei der Fehlerkurve des Livox-Sensors ist diese Schwankung kaum zu beobachten. Ein Grund hierfür könnten erneut die fehlenden Grundwahrheiten sein.

In **Szenario 2** steigt der Lokalisierungsfehler am Anfang erneut an. Im Bereich der normalisierten Zeit von 0,4 bis 0,6 treten die größten Schwankungen der Fehlerkomponente auf. Diese Schwankungen sind erneut auf das Verlassen des Sichtfeldes durch einige Fußgänger und das Fehlen von Ground Truth Daten zurückzuführen.

Der durchschnittliche Kostenwert der Lokalisierungskomponente pro Sensor kategorisiert nach Szenario ist in Abbildung 6.24 dargestellt. Die Ergebnisse weisen hier auf einen gleichbleibenden Kostenwert der Lokalisierung sowohl im Fahrzeug- als auch im Fußgängerszenario hin, wobei der Livox insgesamt den niedrigsten Lokalisierungskostenwert aufweist.

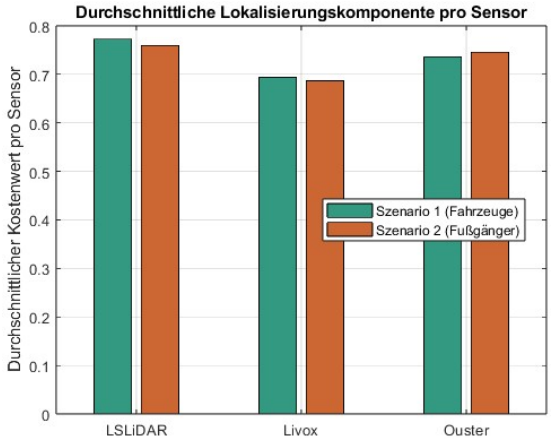


Abbildung 6.24. Vergleich der durchschnittlichen Lokalisierungskomponente pro Sensor

6.3.6 Analyse der Falschen Tracks, Verpassten Ziele und Identitätenwechsel

Die Komponenten falscher Tracks, verpasster Ziele und Identitätswechsel werden nicht anhand der Metrik-Kurven ausgewertet, da diese Werte stets sporadisch auftreten und keine kontinuierliche Kurve wie die GOSPA- und Lokalisierungskomponenten bilden. Das kurze Auftreten dieser Ereignisse führt zu sprunghaften Anstiegen und Abstiegen der Werte. Diese können nicht durch Interpolation adäquat abgebildet werden. Die Darstellungen der Rohdaten sind in den Anlagen A-5 bis A-7 zu finden. Die Analyse dieser Komponenten erfolgt direkt anhand der Durchschnittswerte pro Frame.

Der Durchschnittswert wurde über alle verfügbaren Frames berechnet, wobei lediglich die fehlenden Werte (NaN) ausgeschlossen wurden. Die Null-Werte in diesen Komponenten bleiben erhalten, da sie aussagen, dass keine entsprechenden Fehler aufgetreten sind. Dies ist entscheidend, um die Häufigkeit oder den Schweregrad dieser Fehler korrekt in den Durchschnittswerten widerzuspiegeln. Durch die Einbeziehung der Null-Werte wird eine umfassende und präzise Bewertung aller relevanten Fehlerarten gewährleistet.

Die Ergebnisse aus 6.25 und 6.26 zeigen, dass alle Sensoren im Fahrzeug- und Fußgängerszenario ähnliche große Kostenwerte liefern.

Im Fahrzeug-Szenario weisen alle Sensoren bei den Komponenten „falsche Tracks“ und „verpasste Ziele“ eine vergleichbar niedrige Fehlerquote auf, was auf eine stabile Erkennung hindeutet. Im Fußgänger-Szenario hingegen zeigt sich eine höhere Fehleranfälligkeit bei allen Sensoren. Hier deuten die erhöhten Kostenwerte darauf hin, dass die Sensoren bei der Erkennung von dynamischen und kleinen Objekten weniger zuverlässig arbeiten.

Bei „Track Switching“ zeigt sich, dass die Sensoren im Fahrzeug-Szenario häufiger Track-Wechsel verzeichnen als im Fußgänger-Szenario, was auf mögliche Schwierigkeiten in der Verfolgung von schnellen Objekten mit Überlappungen hinweist. Insgesamt scheint das Fahrzeug-Szenario stabilere Ergebnisse zu liefern, während das Fußgänger-Szenario die Sensoren stärker herausfordert, insbesondere bei der Verfolgung und Detektion kleinerer oder sich unregelmäßig bewegender Objekte.

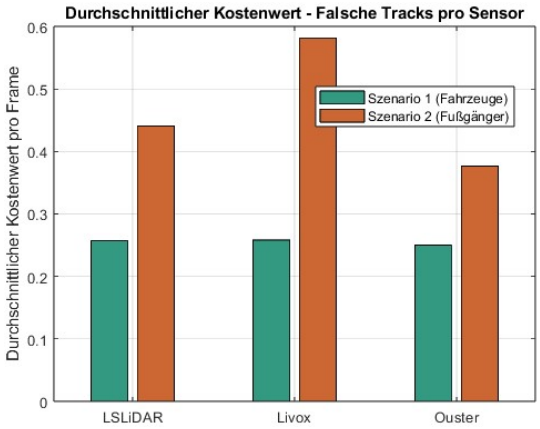


Abbildung 6.25. Vergleich der durchschnittlichen Falschen Tracks pro Frame

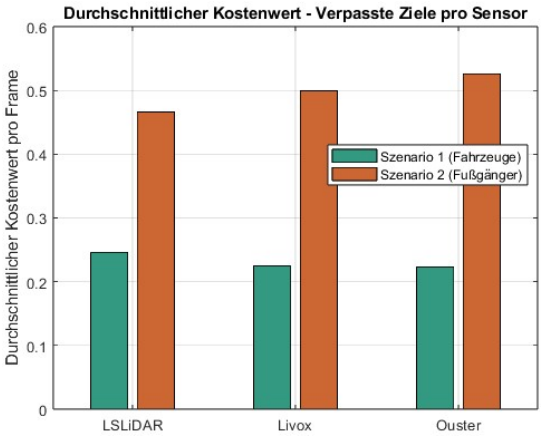


Abbildung 6.26. Vergleich der durchschnittlichen verpassten Ziele pro Frame

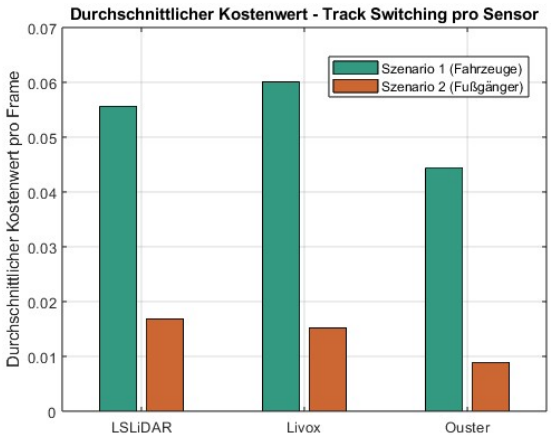


Abbildung 6.27. Vergleich der durchschnittlichen Identitätenwechsel pro Frame

6.3.7 Gesamtbewertung der Sensoren

Um eine abschließende Gesamt-Bewertung der Sensoren zu leisten, wurden die Durchschnittswerte über beide Szenarien zusammengefasst.

Diese Gegenüberstellung stellt dar, dass der Ouster- sowie der Livox-Sensor die geringeren Durchschnittswerte gegenüber dem Leishen-Sensor liefern. Die Gegenüberstellung des durchschnittlichen Lokalisierungs-Kostenwertes zeigt, dass der Livox-Sensor am besten abschneidet und somit den geringsten Lokalisierungsfehler aufweist, wobei der Unterschied jedoch nur marginal ist (6.29). Die größten Differenzen der Kostenwerte treten bei dem Identitätenwechsel (Track Switching) auf. Die beste Identitätskontinuität weist hier der Ouster-Sensor auf (6.28). Die geringsten Detektionsausfälle bzw. verpassten Ziele gewährleistet der Livox (6.30). Jedoch bringt er die meisten falschen Tracks im vergleich zu den anderen Sensoren hervor (6.25).

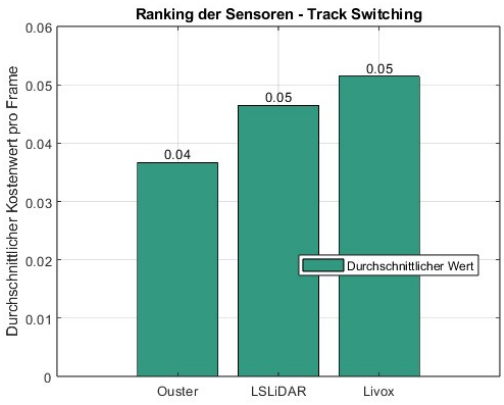


Abbildung 6.28. Ranking der Sensoren - Identitätenwechsel

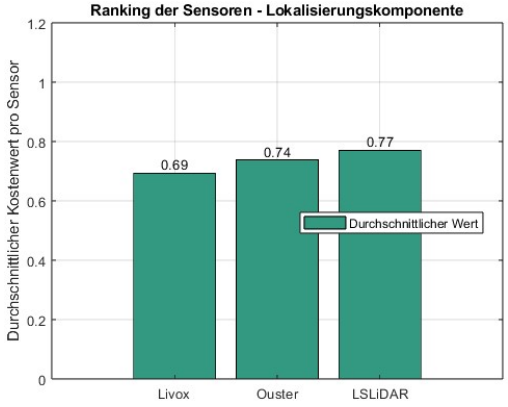


Abbildung 6.29. Ranking der Sensoren - Lokalisierungskomponente

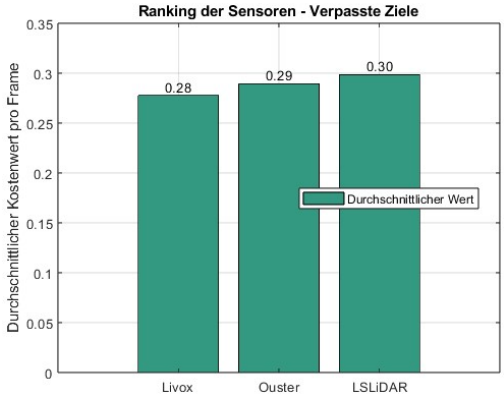


Abbildung 6.30. Ranking der Sensoren - Verpasste Ziele

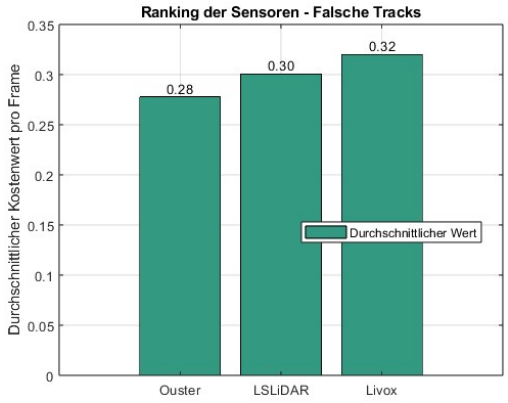


Abbildung 6.31. Ranking der Sensoren - Falsche Tracks

In der Gesamtbewertung, nominiert durch die GOSPA Metrik, schließt der Livox-Sensor mit dem geringsten Kostenwert ab wobei der LS-Sensor das schlechteste Ergebnis liefert. Die Unterschiede zwischen den Ergebnissen sind jedoch minimal und wenig aussagekräftig. Dieser Zusammenhang lässt sich auch in der Metrik-Kurve interpretieren (6.20).

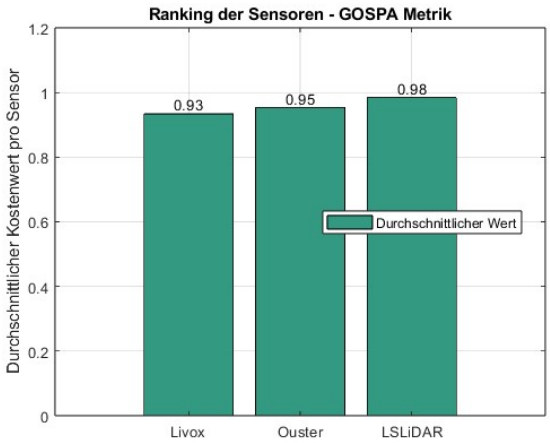


Abbildung 6.32. Ranking der Sensoren mit GOSPA

7 Konzept für die Implementierung in ecu.test von tracetronic

7.1 Anbindung von ecu.test an MATLAB

Die Integration einer Anwendung aus der MATLAB Umgebung in ecu.test wird durch die Nutzung der MATLAB Engine API für Python realisiert. Zunächst ist eine ordnungsgemäße Installation der MATLAB-Software sowie die Konfiguration der Engine innerhalb der Python-Umgebung erforderlich. Dies erfolgt durch die Installation der Engine über das MATLAB-Installationsverzeichnis mittels eines Pip-Befehls, wodurch die notwendigen Schnittstellen und Bibliotheken bereitgestellt werden. Damit wird eine reibungslose Kommunikation zwischen Python und der MATLAB-Plattform gewährleistet. [50]

Nach der erfolgreichen Einrichtung ermöglicht die Engine das Starten einer MATLAB-Sitzung direkt aus einem Python-Skript heraus. Dies geschieht durch die Initialisierung der Engine und das anschließende Aufrufen von Funktionen der MATLAB-Umgebung innerhalb des Python-Codes. Dabei können Daten bidirektional übertragen werden, was den Austausch zwischen den beiden Programmiersprachen erleichtert. Numerische Berechnungen, Datenanalysen oder Visualisierungen, die in der MATLAB-Umgebung erstellt wurden, lassen sich somit nahtlos in Python-Anwendungen integrieren.

Ein weiterer wesentlicher Aspekt der Integration ist die Ausführung von MATLAB Anwendungen im Batch-Modus. Dieser Modus ermöglicht die Ausführung von Skripten und Funktionen ohne eine grafische Benutzeroberfläche. Durch den Aufruf einer MATLAB Anwendung aus Python heraus kann diese im Hintergrund betrieben werden, wodurch die Systemressourcen effizient genutzt werden.

Im Rahmen des Implementierungskonzeptes der entwickelten Anwendung in ecu.test wurde ein Python-Skript erstellt, das als Schnittstelle zwischen der MATLAB-Umgebung und ecu.test fungiert. Der Ordner „UserPyModules“ innerhalb der ecu.test Ordnerstruktur dient dabei als zentraler Speicherort für das Python-Skript. Dieses passt sich automatisch der vorhandenen Dateisystemstruktur an, vorausgesetzt, die relative Struktur des ecu.test-Workspace bleibt unverändert. Dadurch wird gewährleistet, dass keine zusätzlichen Anpassungen der im Skript festgelegten Pfade erforderlich sind, was die Portabilität des Projekts auf verschiedenen Rechnern und in unterschiedlichen Umgebungen erleichtert.

Innerhalb eines Berechnungsschritts in ecu.test besteht die Möglichkeit, ein Python-Skript auszuwählen. Über die Parameteroberfläche kann die gewünschte Python-Datei bestimmt

werden. Nach Abschluss aller Berechnungsschritte in der MATLAB-Anwendung können die Ergebnisse in einer Variablenstruktur gespeichert werden. Diese Struktur wird anschließend mittels des Python-Skripts an ecu.test übermittelt, wo die Variablen im Workspace gespeichert und in der Trace-Analyse ausgewertet werden können. Ein beispielhafter Berechnungsschritt in ecu.test mit dem übergebenen Python-Skript ist in Abb. 7.1 dargestellt.

Abbildung 7.1. Aufruf des Python-Skriptes in ecu.test

Darüber hinaus besteht die Möglichkeit, in der MATLAB-Umgebung erstellte Visualisierungen über Python an ecu.test zu übermitteln. Diese Funktion unterstützt insbesondere die Überprüfung der Plausibilität der bereitgestellten Daten und erhöht somit die Zuverlässigkeit der Analyseergebnisse. Durch die Einbindung der generierten Abbildungen in den Testprozess von ecu.test kann eine zusätzliche Validierungsebene geschaffen werden, die sicherstellt, dass die erfassten Daten konsistent und nachvollziehbar sind. Dies fördert die Identifikation potenzieller Anomalien und trägt zur Verbesserung der Gesamtqualität der Datenverarbeitung bei.

Das Konzept der Applikationskette zwischen ecu.test, Python und MATLAB garantiert eine reibungslose Übergabe von Parameterwerten und das Einbinden von Berechnungsschritten der entwickelten GOSPA-Analyse-Applikation. Dennoch fehlt es im aktuellen Zustand von ecu.test an der Funktionalität zur Generierung von Ground Truth Daten innerhalb von LiDAR-Punktwolken. Dieser Schritt in der Applikationskette erfordert eine fortlaufende und kontinuierliche Weiterentwicklung, um die Generierung von Wahrheitswerten in LiDAR-Punktwolken zu ermöglichen und die effektive Bewertung von Objektverfolgungen zu garantieren.

In Abbildung 7.2 sind die Arbeitsschritte und Datenübergaben visualisiert.

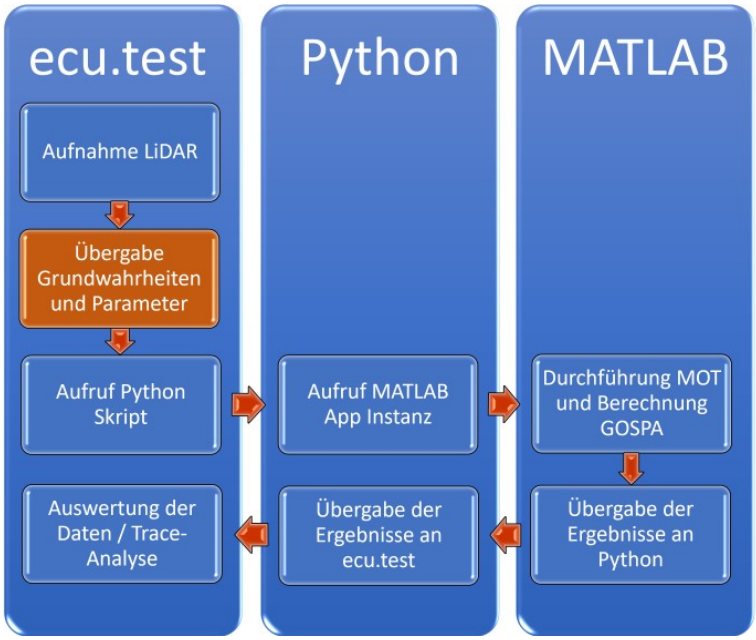


Abbildung 7.2. Schematische Darstellung der Applikationskette

Alle Daten- und Benutzereingaben die standardgemäß über die Graphical User Interface (GUI) der App gesteuert wurden, werden nun durch die Phyton-Instanz übergeben. Alle Parameter des Detektormodells (vgl. 5.2.6) sowie des Trackermodells (vgl. 5.3.4) und die erwarteten Objektabmessungen müssen im Python-Skrip definiert sein und übergeben werden.

Um die Übergabe der erforderlichen Parameter, LiDAR-Messungen und Wahrheitsdaten von dem Python-Skript an die MATLAB-App zu ermöglichen, war eine Erweiterung zu den App-internen privaten Funktionen erforderlich. Hierfür wurde eine öffentliche Funktion `getGOSPA()` implementiert. Diese Funktion ist für externe Instanzen zugänglich und führt sämtliche Berechnungsschritte zur Ermittlung der GOSPA Metrik durch. Der Programmablaufplan des Python-Skriptes ist in Anlage B aufgeführt.

7.2 Mögliche Konzepte für die Erstellung von Wahrheitsdaten

Die Generierung von Wahrheitsdaten (Ground Truth) spielt eine entscheidende Rolle bei der Verwendung der Applikation und der Berechnung der Metriken. Insbesondere für Testschritte ist es von großer Bedeutung, gleichbleibende Bedingungen und konsistente Abläufe sicherzustellen, um zuverlässige und reproduzierbare Ergebnisse zu erzielen. Manuelles Annotieren der Grundwahrheiten stellt hierbei eine erhebliche Herausforderung dar, da es äußerst zeitaufwendig und fehleranfällig ist. Der Aufwand steigt exponentiell mit der Komplexität und der Anzahl der zu annotierenden Objekte, was die Skalierbarkeit solcher Ansätze stark einschränkt.

Für effektive und effiziente Testprozesse ist daher ein einfacher und schneller Zugang zu präzisen Ground Truth Daten unerlässlich. Automatisierte Methoden zur Erzeugung die-

ser Daten bieten eine vielversprechende Lösung, da sie den manuellen Aufwand erheblich reduzieren und gleichzeitig die Konsistenz und Genauigkeit der Ground Truth sicherstellen können. Zudem ermöglichen automatisierte Verfahren eine schnellere Iteration und Anpassung der Testumgebungen, was die Entwicklung und Validierung von Multi-Object Tracking-Systemen beschleunigt.

Im Folgenden werden verschiedene Ansätze diskutiert, die darauf abzielen, Ground Truth Daten entweder automatisch zu generieren oder speziell für Testzwecke zu erstellen.

1. Pfadfolger/ Trajektorienplanung

Ein etabliertes Konzept zur Generierung von Wahrheitsdaten besteht darin, Pfadfolger an den zu verfolgenden Objekten zu installieren. Diese Pfadfolger erfassen präzise Bewegungsparameter wie Lenkwinkel und Gierrate (Yaw Rate) in Bezug auf einen bekannten Referenzpunkt zum LiDAR-Sensor. Durch die genaue Steuerung und Überwachung der Objektbewegungen können exakte Trajektorien erstellt werden, die als Ground Truth dienen. Dieses Verfahren gewährleistet eine hohe Genauigkeit der Bewegungsdaten, erfordert jedoch eine aufwändige Hardware-Integration und sorgfältige Kalibrierung der Pfadfolgersysteme. [51]

2. Erkennung hochreflektiver Anbringungen am Objekt

Ein weiteres Konzept zur automatischen Generierung von Wahrheitsdaten basiert auf der Erkennung hochreflektiver Markierungen an den zu verfolgenden Objekten. Beispielsweise können spezielle Anbringungen wie Fahrzeugkennzeichen oder reflektierende Aufkleber mittels fortschrittlicher Algorithmen identifiziert werden. Diese Algorithmen analysieren die LiDAR-Daten, um die Position und Bewegung der markierten Objekte präzise zu bestimmen. Der Vorteil dieses Ansatzes liegt in der Automatisierung und der Reduzierung manueller Eingriffe, allerdings hängt die Genauigkeit stark von der Qualität und Platzierung der reflektiven Markierungen ab.

3. Erstellung der Daten durch Positionsbestimmung mit Sensorfusion

Die Kombination verschiedener Sensortechnologien zur Positionsbestimmung stellt ein weiteres effektives Konzept zur Generierung von Wahrheitsdaten dar. Durch die Fusion von Daten aus RADAR- und Kamera-Systemen sowie der Verwendung von Real-Time Kinematic (RTK)-GPS-Empfängern können die Positionen und Bewegungen der Objekte mit höchster Genauigkeit ermittelt werden. Diese multimodale Herangehensweise nutzt die Stärken der einzelnen Sensoren und ermöglicht es, eine robuste und verlässliche Ground Truth zu erstellen. Mithilfe eines GNN-Filters werden dabei die präzisen Positionsdaten aus RTK, die hohe Reichweite des RADAR und die Auflösung der Kamera optimal kombiniert. Diese Fusion kompensiert Schwächen einzelner Systeme und verbessert so die Gesamtgenauigkeit der Wahrheitsdaten für Validierungs- und Testzwecke. [52]

4. Verwendung bekannter Ground Truths mit präzisen Trajektorien

Ein weiteres Konzept beinhaltet die Nutzung von bekannten Ground Truths, die durch präzise definierte Trajektorien charakterisiert sind. Dies kann durch den Einsatz fahrender

Dummys oder speziell ausgestatteter Fahrzeuge erfolgen, die exakte Trajektorien mithilfe von Pfadfolgern fahren. Eine Weiterentwicklung dieser Idee besteht darin, feste Trajektorien als Ground Truth zu bestimmen und diese kontinuierlich mit einem Fahrzeug abzufahren. Dieser Ansatz erfordert geringe Ungenauigkeiten bei der Trajektorienverfolgung, um verlässliche Wahrheitsdaten zu gewährleisten. Die präzise Kontrolle der Fahrzeugbewegungen ermöglicht eine konsistente und wiederholbare Erstellung von Ground Truth Daten, die ideal für die Validierung und Kalibrierung von Tracking-Systemen sind.

7.3 Mögliche Erweiterungen

Die Konzeption, Testung und Validierung der Applikation erfolgte unter Verwendung von Messdaten, die durch stationär montierte Sensoren erhoben wurden. Infolgedessen ist die Applikation primär auf die Verfolgung von Objekten und die anschließende Metrikbewertung unter der Annahme statischer Sensorpositionen optimiert. Eine potenzielle Erweiterung der Applikation bestünde in der Anpassung für Messungen innerhalb dynamischer Detektor-Regionen. Anstelle eines statischen Messbereichs könnte eine an die Geschwindigkeit gekoppelte, variable Erfassungsregion (Fahrschlauch) implementiert werden. Dadurch könnte die Applikation besser auf reale Einbaubedingungen der Sensoren abgestimmt werden.

Darüber hinaus wäre es erforderlich, bei Tests in denen sich die Sensoren selbst in Bewegung befinden, die Visualisierungen der erweiterten Analyse anzupassen. Eine mögliche Methode zur Darstellung der Messstrecken wäre die Nutzung eines Simultaneous Localization and Mapping (SLAM)-Verfahrens, um die abgefahrene Strecke als eine zusammenhängende Punktwolke zu rekonstruieren. Dies würde ermöglichen, die Detektions- und Verfolgungsergebnisse innerhalb dieser rekonstruierten Punktwolke zu visualisieren und somit eine präzisere Analyse der Daten zu ermöglichen. [53]

Durch diese erweiterten Möglichkeiten könnten Testanwendungen realistischer gestaltet werden, was insbesondere die Testung von Fahrerassistenzsystemen positiv beeinflusst. Die genaue Darstellung und Analyse der Sensorleistungen in dynamischen Szenarien ermöglicht eine verbesserte Bewertung und Optimierung der Systeme für den realen Einsatz.

8 Zusammenfassung und Ausblick

In dieser Diplomarbeit wurde eine MATLAB-Testapplikation zur Bewertung der Objektverfolgung mittels LiDAR-Sensoren entwickelt und implementiert. Die Applikation wurde im MATLAB App Designer erstellt und ermöglicht eine benutzerfreundliche Oberfläche zur Analyse und Visualisierung der Verfolgungsergebnisse. Die Punktwolkendaten werden von einem Detektormodell verarbeitet, das eine Segmentierungsmethode nutzt, die auf der euklidischen Distanz basiert, um robuste Detektionen zu erzeugen. Die anschließende Verfolgung dieser Detektionen wird durch einen JPDA-Tracker mit Zustandsschätzung realisiert, der auf zwei Bewegungsmodellen – einem konstanten Geschwindigkeitsmodell und einem konstanten Drehratenmodell – basiert.

Zur fundierten Bearbeitung des Themas erfolgte eine umfassende Einarbeitung in die Grundlagen der LiDAR-Technik sowie in die Methoden der Detektion, Verfolgung und Zustandsschätzung. Zudem wurden die Metriken für die Mehrzielverfolgung eingehend untersucht, um eine solide Basis für die Bewertung der Verfolgungsergebnisse zu schaffen.

Durch die Verwendung von LiDAR-Messdaten und den zugehörigen Grundwahrheitsdaten konnte die Funktionsfähigkeit der entwickelten Testapplikation validiert werden. Die Applikation ermöglichte außerdem eine detaillierte Analyse der Detektionsergebnisse verschiedener LiDAR-Sensoren, wodurch Leistungsunterschiede identifiziert werden konnten. Die GOSPA Metrik wurde effektiv eingesetzt, um spezifische Aspekte der Verfolgungsergebnisse der Sensoren zu untersuchen, wie beispielsweise Lokalisierungsfehler und Identitätswechsel. Dadurch konnten Unterschiede in der Verfolgungsgenauigkeit und Zuverlässigkeit fundiert bewertet werden.

Ferner stellt diese Arbeit ein Konzept zur Anbindung der MATLAB-App an die Python-basierte Software ecu.test vor, um eine Integration der Auswertelösungen zu ermöglichen. Darüber hinaus wurden Konzepte zur Weiterentwicklung von Bewertungsverfahren für die LiDAR-Objektverfolgung präsentiert. Diese basieren auf der effizienten Erstellung von Grundwahrheitsdaten durch halb- bis vollautomatisierte Systeme, beispielsweise mithilfe von Sensorfusion.

Die in dieser Arbeit durchgeführte Auswertung dient als beispielhafte Anleitung für zukünftige Studien im Bereich der LiDAR-basierten Objektverfolgung. Die Ergebnisse des Sensorvergleichs zeigten, dass zwar leichte Unterschiede in der Tauglichkeit für die Mehrzielverfolgung bestehen, diese jedoch marginal sind. Um eindeutige Aussagen über die optimalen Anwendungsgebiete der verschiedenen Sensoren und deren messtechnische Ansätze treffen zu können, sind weitere Prüfungen und Validierungsprozesse erforderlich.

Eine mögliche Erweiterung besteht darin, eine umfassendere Studie durchzuführen, die auf umfangreichen Messungen mit mehreren Sensoren basiert. Durch die Durchführung zahlreicher Messungen und die Variation von MOT-Einstellungen sowie GOSPA-Parametern können verschiedene Aspekte der Verfolgung detaillierter untersucht werden. Unterschiedliche GOSPA-Einstellungen ermöglichen beispielsweise eine gezielte Untersuchung spezifischer Aspekte wie Identitätswechsel bei Überlappung von Objekten. Anpassungen der MOT-Einstellungen, beispielsweise bei der Segmentierung hinsichtlich der Punktdichte für Objekte oder durch Anpassung der Verfolgungslogik, erlauben es, die Modelle realistischer an echte Szenarien anzupassen. Dadurch können verschiedene technische Aspekte unterschiedlicher LiDAR-Technologien untersucht und miteinander verglichen werden.

Die Applikation unterstützt dabei eine schnelle und präzise Testung der Sensoren und bildet somit die Grundlage für eine fundierte Analyse und Optimierung der Sensorleistung. Zukünftig wäre es sinnvoll, die Testapplikation auf zusätzliche Sensor- und Verfolgungsmodelle zu erweitern, um die Kompatibilität mit verschiedenen LiDAR-Technologien weiter zu verbessern. Auch der Einsatz von maschinellen Lernverfahren zur Erkennung und Vermeidung von Identitätswechseln sowie zur Steigerung der Detektionsqualität könnte weiter untersucht werden.

Diese Optimierungen könnten die Grundlage für detailliertere Analysen schaffen und die Effizienz zukünftiger Untersuchungen verbessern. Insgesamt bietet diese Arbeit damit nicht nur eine konkrete Anwendung, sondern auch eine Grundlage und Orientierung für weiterführende, groß angelegte Studien im Bereich der LiDAR-Technologie.

Literatur- und Quellenverzeichnis

[1] Hermann Winner et al. *Handbuch Fahrerassistenzsysteme - Grundlagen, Komponenten und Systeme für aktive Sicherheit und Komfort*. Berlin Heidelberg New York: Springer-Verlag, 2015. ISBN: 978-3-658-05734-3.

[2] Thomas Tille. *Automobil-Sensorik 2: Systeme, Technologien und applikationen*. Springer Vieweg, 2018.

[3] Dingkang Wang, Connor Watkins und Huikai Xie. "MEMS mirrors for LiDAR: A review". en. In: *Micromachines (Basel)* 11.5 (Apr. 2020), S. 456.

[4] Han Woong Yoo et al. *MEMS-based Lidar for autonomous driving - e+i elektrotechnik und Informationstechnik*. Juli 2018. URL: https://link.springer.com/article/10.1007/s00502-018-0635-2#citeas.

[5] Christopher Vincent Poulton et al. "Long-range LIDAR and free-space data communication with high-performance optical phased arrays". In: *IEEE Journal of Selected Topics in Quantum Electronics* 25.5 (Sep. 2019), S. 1–8. DOI: 10.1109/jstqe.2019.2908555.

[6] Schott. "Anwendungen von hermetischen Gehäusen für LiDAR-Sensoren". In: *SCHOTT* (Okt. 2024). Zugriff am 15.10.2024. URL: https://www.schott.com/de-de/products/hermetic-packages-for-lidar-sensors-p1000281/applications.

[7] P. Wang et al. "Research on 3D Point Cloud Data Preprocessing and Clustering Algorithm of Obstacles for Intelligent Vehicle". In: *World Electric Vehicle Journal* 13.7 (2022), S. 130. DOI: 10.3390/wevj13070130.

[8] M.A. Fischler und R.C. Bolles. "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography". In: *Communications of the ACM* 24.6 (1981), S. 381–395. DOI: 10.1145/358669.358692.

[9] J.M. Martínez-Otzeta et al. "RANSAC for Robotic Applications: A Survey". In: *Sensors* 23.1 (2023), S. 327. DOI: 10.3390/s23010327.

[10] L. Liao et al. "A Supervoxel-Based Random Forest Method for Robust and Effective Airborne LiDAR Point Cloud Classification". In: *Remote Sensing* 14.6 (2022), S. 1516. DOI: 10.3390/rs1406151. URL: https://doi.org/10.3390/rs1406151.

[11] Hamid Mahmoudabadi, Timothy Shoaf und Michael Olsen. "Superpixel Clustering and Planar Fit Segmentation of 3D LIDAR Point Clouds". In: Juli 2013, S. 1–7. DOI: 10.1109/COMGEO.2013.2.

[12] Yiming Zeng et al. “RT3D: Real-Time 3-D Vehicle Detection in LiDAR Point Cloud for Autonomous Driving”. In: *IEEE Robotics and Automation Letters* 3.4 (2018), S. 3434–3440. DOI: 10.1109/LRA.2018.2852843.

[13] Dimitris Zermas, Izzat Izzat und Nikolaos Papanikolopoulos. “Fast Segmentation of 3D Point Clouds: A Paradigm on LiDAR Data for Autonomous Vehicle Applications”. In: Mai 2017. DOI: 10.1109/ICRA.2017.7989591.

[14] Daniel Müllner. *Modern hierarchical, agglomerative clustering algorithms*. 2011. arXiv: 1109.2378 [stat.ML]. URL: https://arxiv.org/abs/1109.2378.

[15] Pavlina Konstantinova, Alexander Udvarev und Tzvetan Semerdjiev. “A study of a target tracking algorithm using global nearest neighbor approach”. In: (Jan. 2003). DOI: 10.1145/973620.973668. URL: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=8697e4a283f9fd59ddcbcfe0071d764ff1ccc3c4.

[16] H. W. Kuhn. “The Hungarian Method for the Assignment Problem”. In: *50 Years of Integer Programming 1958-2008*. Hrsg. von M. Jünger et al. Springer, Berlin, Heidelberg, 2010, S. 29–47. DOI: 10.1007/978-3-540-68279-0_2. URL: https://doi.org/10.1007/978-3-540-68279-0_2.

[17] T. Bodrumlu, M. M. Gozum und A. Semiz. “Extended Object Tracking Performance Comparison for Autonomous Driving Applications”. In: *Engineering Proceedings* 58 (2023), S. 35. DOI: 10.3390/ecsa-10-16201. URL: https://doi.org/10.3390/ecsa-10-16201.

[18] Lennart Svensson et al. “Set JPDA Filter for Multitarget Tracking”. In: *IEEE Transactions on Signal Processing* 59.10 (2011), S. 4677–4691. DOI: 10.1109/TSP.2011.2161294.

[19] Xiangyang Liu et al. “Improved JPDA Algorithm with Measurements Adaptively Censored”. In: *2012 International Conference on Industrial Control and Electronics Engineering*. 2012, S. 207–211. DOI: 10.1109/ICICEE.2012.62.

[20] R. Marchthaler und S. Dingler. *Kalman-Filter: Einführung in die Zustandsschätzung und ihre Anwendung für eingebettete Systeme*. Springer Fachmedien Wiesbaden, Springer Vieweg, 2024. DOI: https://doi.org/10.1007/978-3-658-43216-4.

[21] G. Welch und G. Bishop. “An Introduction to the Kalman Filter”. In: *ACM SIGGRAPH 2001 Course Notes*. ACM, 2001, S. 19–24.

[22] Rufi Lindl. “Tracking von Verkehrsteilnehmern im Kontext von Multisensorsystemen”. Zugriff am 05.09.2024. Dissertation. Technische Universität München, 2009, S. 78–88. URL: https://mediatum.ub.tum.de/doc/667321/667321.pdf.

[23] Sebastian Schneider. “Fusion von Kamera und LiDAR zur modellbasierten autonomen Navigation”. Zugriff am 18.10.2024. Diss. 2023, S. 38–41. URL: https://athene-forschung.unibw.de/144973.

[24] Marco Kruse. *Mehrobjekt-Zustandsschätzung mit verteilten Sensorträgern am Beispiel der Umfeldwahrnehmung im Straßenverkehr* -. Karlsruhe: KIT Scientific Publishing, 2014, S. 149–152. ISBN: 978-3-866-44982-4.

[25] MathWorks. *Track Vehicles Using Lidar*. Zugriff am 14.09.2024. 2024. URL: https://de.mathworks.com/help/fusion/ug/track-vehicles-using-lidar.html.

[26] Branko Ristic et al. “A Metric for Performance Evaluation of Multi-Target Tracking Algorithms”. In: *IEEE Transactions on Signal Processing* 59.7 (2011), S. 3452–3457. DOI: 10.1109/TSP.2011.2140111.

[27] Ángel F. García-Fernández, Marcel Hernandez und Simon Maskell. “An analysis on metric-driven multi-target sensor management: GOSPA versus OSPA”. In: *2021 IEEE 24th International Conference on Information Fusion (FUSION)*. 2021, S. 1–8. DOI: 10.23919/FUSION49465.2021.9626837.

[28] MathWorks. *What Is Ground Truth?* Zugriff am 18.10.2024. 2024. URL: https://de.mathworks.com/discovery/ground-truth.html.

[29] Anton Milan et al. “MOT16: A Benchmark for Multi-Object Tracking”. In: (2016). arXiv: 1603.00831 [cs.CV]. URL: https://arxiv.org/abs/1603.00831.

[30] Jonathon Luiten et al. “HOTA: A Higher Order Metric for Evaluating Multi-object Tracking”. In: *International Journal of Computer Vision* 129.2 (Feb. 2021). DOI: 10.1007/S11263-020-01375-2.

[31] David M. W. Powers. *Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation*. 2020. arXiv: 2010.16061 [cs.LG]. URL: https://arxiv.org/abs/2010.16061.

[32] Ricardo Roriz, Jorge Cabral und Tiago Gomes. “Automotive LiDAR Technology: A Survey”. In: *IEEE Transactions on Intelligent Transportation Systems* 23.7 (2022), S. 6282–6297. DOI: 10.1109/TITS.2021.3086804.

[33] MathWorks. *pcfitplane - Fit plane to point cloud*. https://de.mathworks.com/help/vision/ref/pcfitplane.html. Zugriff am 19.08.2024. 2024.

[34] MathWorks. *Clustering*. https://de.mathworks.com/discovery/clustering.html?s_tid=srchtitle_support_results_3_cluster. Zugriff am 02.06.2024. n.d.

[35] MathWorks. *Convert Detections into ObjectDetection Format*. https://de.mathworks.com/help/fusion/ug/convert-detections-into-objectDetection-format.html. Zugriff am 21.07.2024. n.d.

[36] MathWorks. *trackerJPDA System Object*. https://de.mathworks.com/help/fusion/ref/trackerjpda-system-object.html. Zugriff am 29.06.2024. n.d.

[37] MathWorks. *Tracking und Tracking-Filter*. https://de.mathworks.com/help/fusion/ug/tracking-and-tracking-filters.html. Zugriff am 14.07.2024. n.d.

[38] MathWorks. *trackingIMM System Object*. https://de.mathworks.com/help/fusion/ref/trackingimm.html. Zugriff am 14.06.2024. n.d.

[39] MathWorks. *cart2sph - Konvertierung von kartesischen in sphärische Koordinaten*. https://de.mathworks.com/help/matlab/ref/cart2sph.html?s_tid=doc_ta. Zugriff am 30.07.2024. n.d.

[40] Mechlab. *Erste Messungen // First Measurements*. Zugriff am 22.07.2024. 2023. URL: https://mechlab.de/erste-messungen-first-measurements/.

[41] Generation Robots. *Ouster OS1-32 Lidar Ver. 6 mit mittlerer Reichweite*. Zugriff am 03.07.2024. 2024. URL: https://www.generationrobots.com/de/403994-ouster-os1-32-lidar-ver-6-mit-mittlerer-reichweite.html.

[42] DJI ARS Madrid. *Sensor LiDAR Livox Horizon Image*. Zugriff am 05.07.2024. 2024. URL: https://djiarsmadrid.com/6104-large_default/sensor-lidar-livox-horizon.jpg.

[43] Leishen LiDAR. *Archiv für neue Produktversionen*. Zugriff am 05.07.2024. URL: https://www.leishenlidar.com/de/kategorie/neue-produkt-release/.

[44] Ouster, Inc. *OS1 Sensor Product Specifications, Revision D, Version 2.0*. Zugriff am 09.06.2024. 2023. URL: https://data.ouster.io/downloads/datasheets/datasheet-revd-v2p0-os1.pdf.

[45] Livox Technology Company. *Livox Horizon User Manual, Version 1.0*. Zugriff am 16.06.2024. 2023. URL: https://www.livoxtech.com/3296f540ecf5458a8829e01cf429798e/assets/horizon/Livox%20Horizon%20user%20manual%20v1.0.pdf.

[46] EMT Savunma. *LS LiDAR Katalog*. Zugriff am 09.06.2024. 2022. URL: https://emtsavunma.com.tr/wp-content/uploads/2022/09/ls-lidar-katalog.pdf.

[47] Leishen Intelligent System Co., Ltd. *Hybrid Solid-State LiDAR*. Zugriff am 10.06.2024. URL: https://www.lslidar.com/products/hybrid-solid-state-lidar/.

[48] T. Trautmann und F. Mendt. "LiDAR-Objekterkennung – Wie das Messverfahren die Detektion beeinflusst". In: *ATZ Elektron* 19 (2024), S. 40–45. DOI: 10.1007/s35658-023-1562-5. URL: https://doi.org/10.1007/s35658-023-1562-5.

[49] Sean de Wolski. *Natural Order Sorting*. Zugriff am 24.06.2024. 2014. URL: https://blogs.mathworks.com/pick/2014/12/05/natural-order-sorting/.

[50] MathWorks. *Install the MATLAB Engine API for Python*. Zugriff am 09.09.2024. 2024. URL: https://de.mathworks.com/help/matlab/matlab_external/install-the-matlab-engine-for-python.html.

[51] Changhee Kim et al. "Trajectory Planning and Control of Autonomous Vehicles for Static Vehicle Avoidance in Dynamic Traffic Environments". In: *IEEE Access* PP (Jan. 2023), S. 1–1. DOI: 10.1109/ACCESS.2023.3236816.

[52] Hatem Hajri und Mohamed-Cherif Rahal. *Real Time Lidar and Radar High-Level Fusion for Obstacle Detection and Tracking with evaluation on a ground truth*. 2019. arXiv: 1807.11264 [cs.RO]. URL: https://arxiv.org/abs/1807.11264.

[53] MathWorks. *Simultaneous Localization and Mapping (SLAM) – Überblick*. Zugriff am 13.10.2024. 2024. URL: https://de.mathworks.com/discovery/slam.html.

Eidesstattliche Erklärung

Das vorliegende Dokument wurde an der Hochschule für Technik und Wirtschaft Dresden unter der Leitung von Prof. Dr. rer. nat. Toralf Trautmann und Dipl.-Ing (FH) Franziskus Mendt angefertigt.

Hiermit erkläre ich, dass ich die vorliegende Arbeit zum Thema

“Implementierung und Test von Bewertungsverfahren für die Objekterkennung”

selbstständig und ohne Benutzung anderer Quellen und Hilfsmittel als angegeben angefertigt habe. Insbesondere versichere ich, dass ich alle wörtlichen und sinngemäßen Übernahmen aus anderen Werken als solche kenntlich gemacht habe.

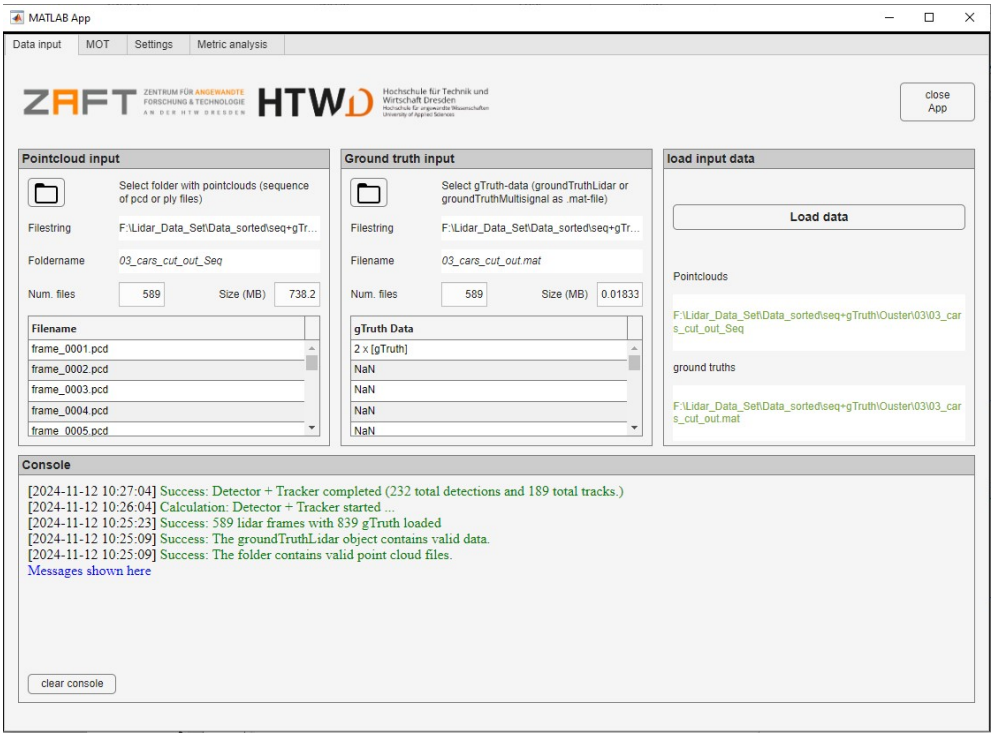
Ferner gestatte ich der Hochschule für Technik und Wirtschaft Dresden, die vorliegende Diplomarbeit unter Beachtung insbesondere urheber-, datenschutzund wettbewerbsrechtlicher Vorschriften für Lehre und Forschung zu nutzen.

Ort, Datum: _____ Unterschrift: _____

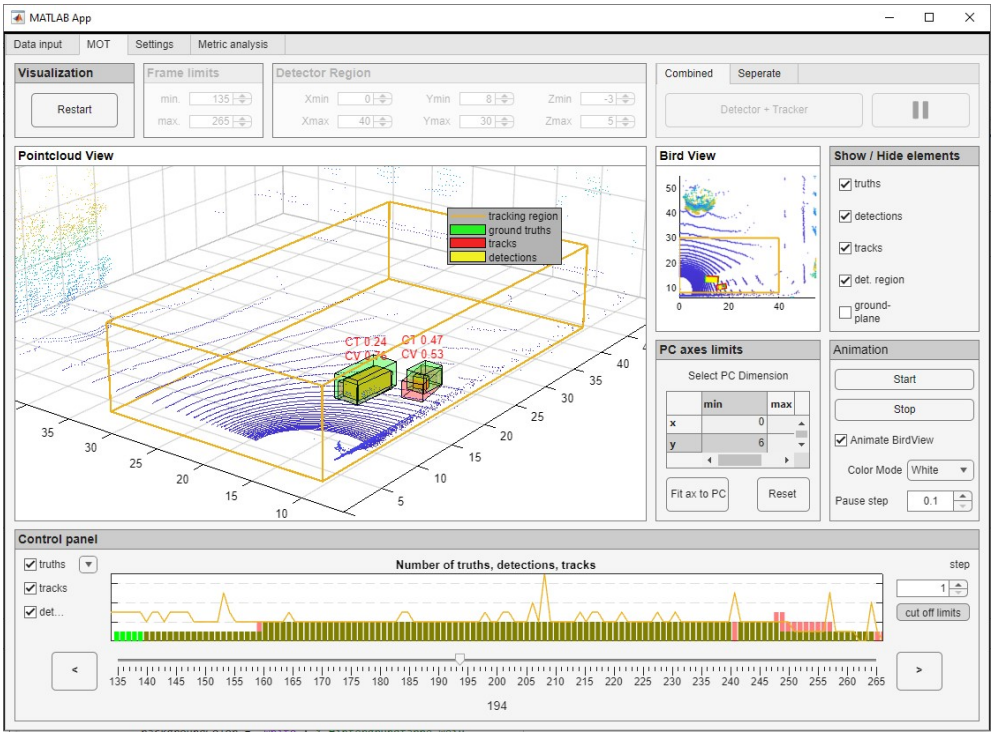
Anlagenverzeichnis

A Abbildungen

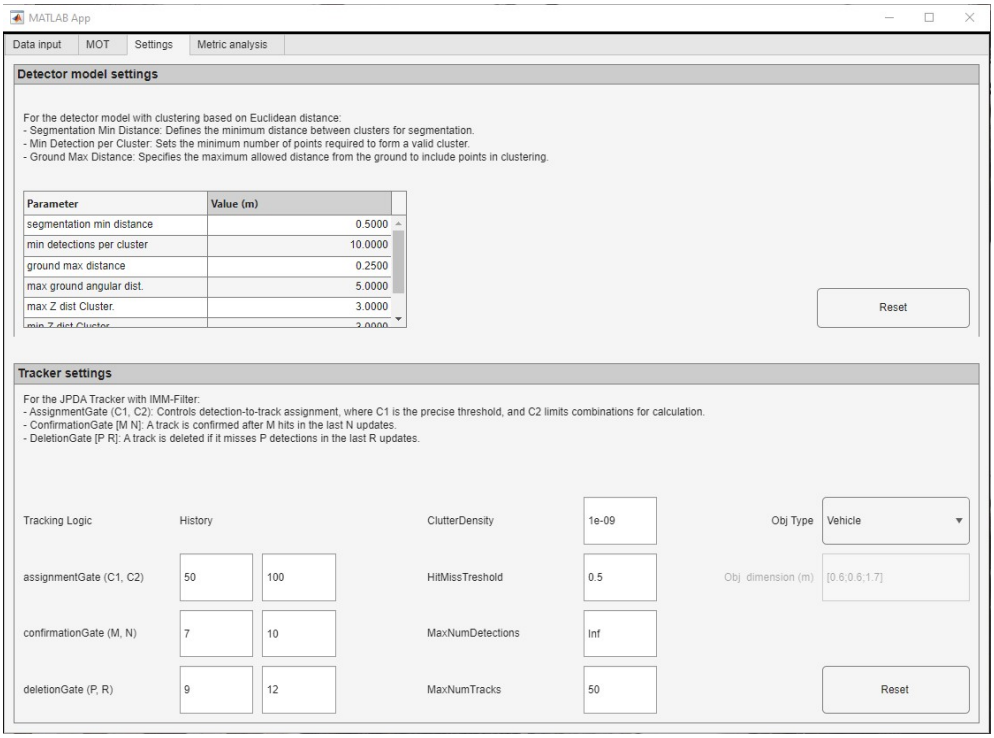
Anlage A - 1 MATLAB Test-Applikation Tab *Data input*



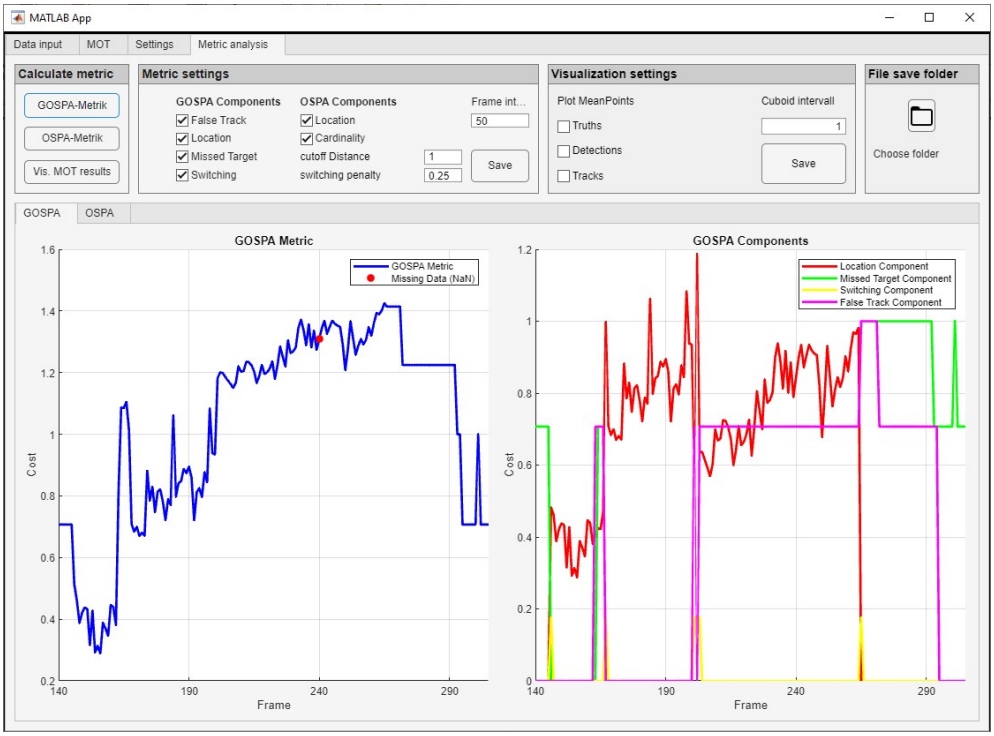
Anlage A - 2 MATLAB Test-Applikation Tab *MOT*



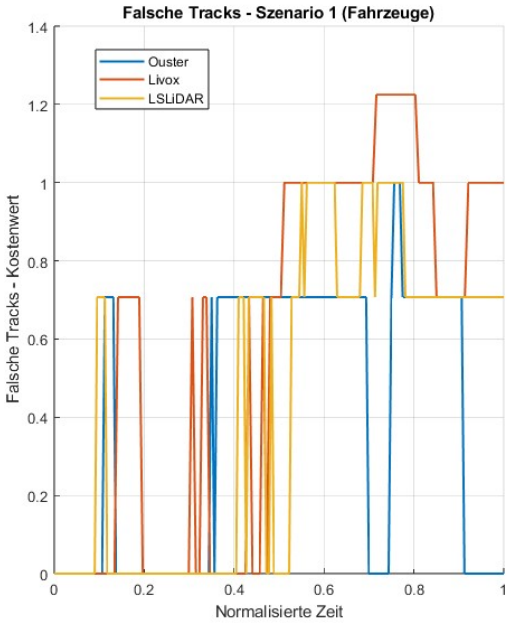
Anlage A - 3 MATLAB Test-Applikation Tab *Settings*

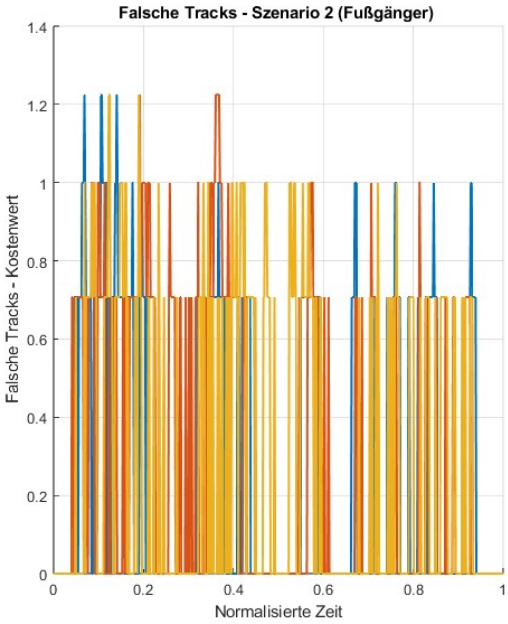


Anlage A - 4 MATLAB Test-Applikation Tab *Metric analysis*

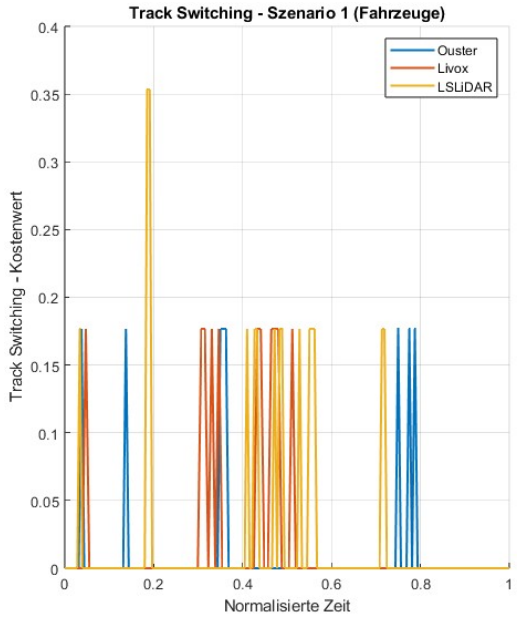


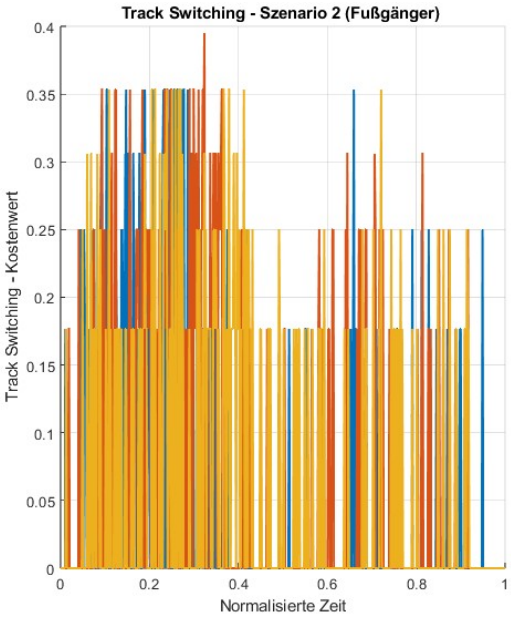
Anlage A - 5 Falsche Tracks GOSPA-Komponente



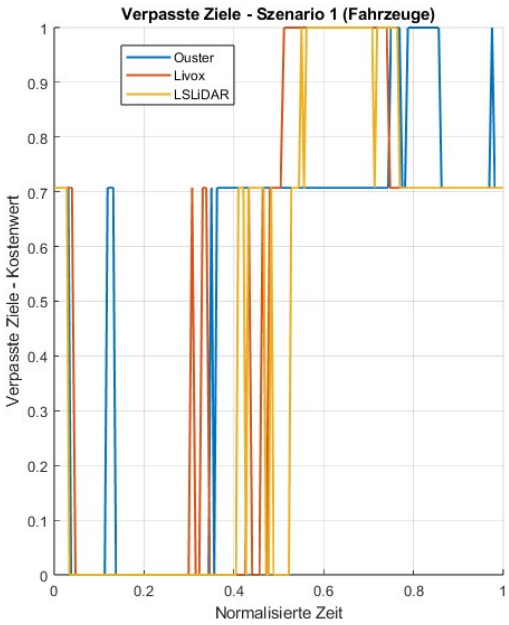


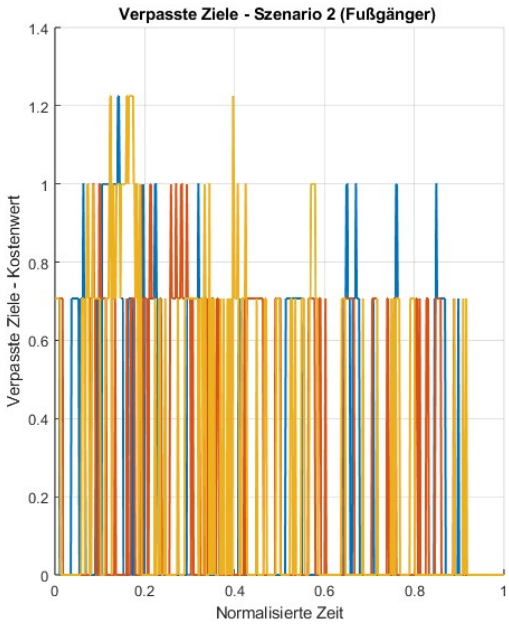
Anlage A - 6 Track Switching GOSPA-Komponente





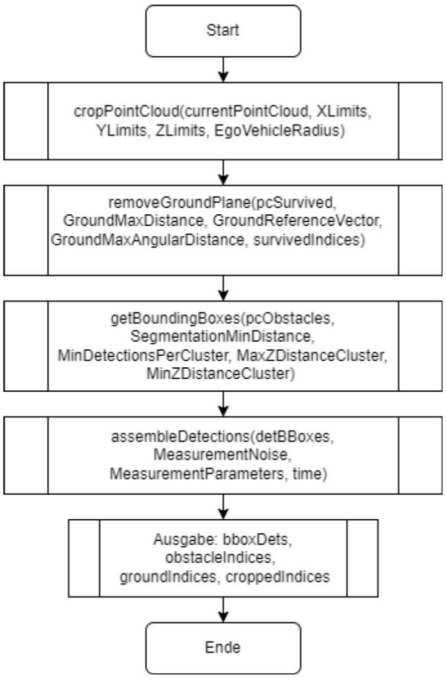
Anlage A - 7 Verpasste Ziele GOSPA-Komponente



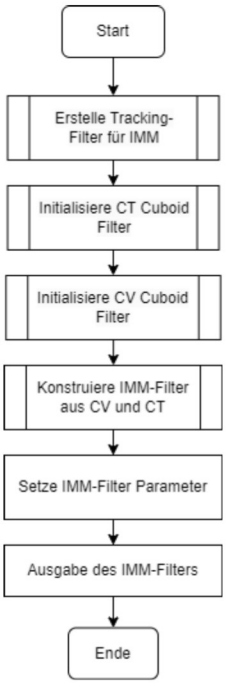


B Programmablaufpläne

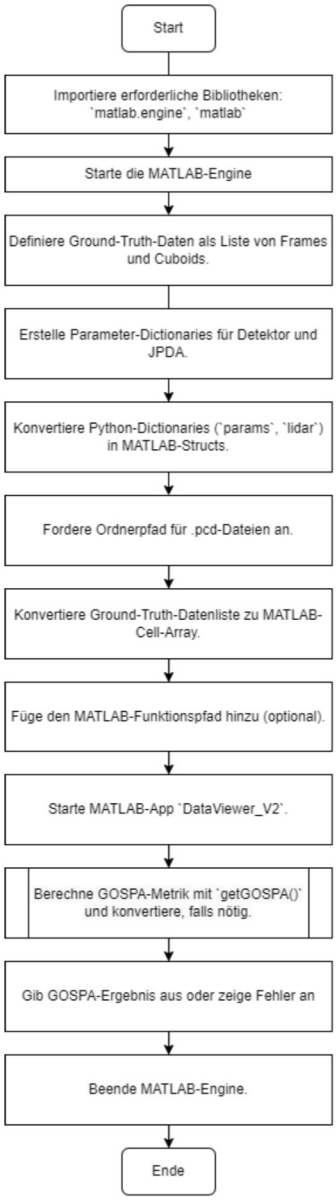
Anlage B - 1 Programmablaufplan BoundingBoxDetector



Anlage B - 2 Programmablaufplan InitIMMFilter



Anlage B - 3 Programmablaufplan Python Skript



C Tabellen

Anlage C - 1 Verwendete MATLAB Toolboxen

Toolbox	Version (Release)
Automated Driving Toolbox	23.2 (R2023b)
Computer Vision Toolbox	23.2 (R2023b)
Image Processing Toolbox	23.2 (R2023b)
Lidar Toolbox	23.2 (R2023b)
Sensor Fusion and Tracking Toolbox	23.2 (R2023b)
Statistics and Machine Learning Toolbox	23.2 (R2023b)

D Datenträger Ordnerstruktur

Anlage D - 1 Ordnerstruktur

```
Diplomarbeit_Lenke
├── Diplomarbeit_Final
│   └── Diplomarbeit_MaxLenke.pdf
├── MATLAB_App
│   ├── App_install_package
│   │   └── GOSPA-TestApp.mlapinstall
│   └── App_designer_files
│       ├── icons
│       ├── DataViewer_V2.mlapp
│       ├── GOSPA-TestApp.prj
│       ├── HelperBoundingBoxDetector.m
│       ├── helperCalcDetectability.m
│       ├── helperConstturnCuboid.m
│       ├── helperConstvelCuboid.m
│       ├── helperCtmeasCuboid.m
│       ├── helperCvmeasCuboid.m
│       ├── helperInitIMMFilter.m
│       ├── helperInverseLidarModel.m
│       ├── helperLidarModel.m
│       └── plotSensorData.m
├── MATLAB_und_Python_Skripte
│   ├── Auswertung_Metriken.m
│   ├── rotateGtruth.m
│   ├── rotatePointCloud.m
│   └── App_getGospa.py
├── Metrik-Dateien
│   ├── Metriken_Szenario1
│   └── Metriken_Szenario2
└── LiDAR_Messung+GTruth
    ├── Livox
    │   ├── GroundTruth_Data
    │   └── LiDAR_pointclouds
    ├── LSLiDAR
    │   ├── GroundTruth_Data
    │   └── LiDAR_pointclouds
    └── Ouster
        ├── GroundTruth_Data
        └── LiDAR_pointclouds
```