HAW HAMBURG

Bachelorarbeit

Albert Sauer

Urbane Umgebungserkennung für Luftfahrzeuge mittels Datenfusion von Kamera- und Lidardaten

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department Computer Science

Albert Sauer

Urbane Umgebungserkennung für Luftfahrzeuge mittels Datenfusion von Kamera- und Lidardaten

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang *Bachelor of Science Informatik Technischer Systeme*
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Tim Tiedemann
Zweitgutachter: Prof. Dr. Stephan Pareigis

Eingereicht am: 21. Juni 2022

Albert Sauer

Thema der Arbeit

Urbane Umgebungserkennung für Luftfahrzeuge mittels Datenfusion von Kamera- und Lidardaten

Stichworte

Umgebungserkennung, Datenfusion, Kamera, Lidar, Maschinelles Lernen,
Instanz-Segmentierung

Kurzzusammenfassung

Diese Arbeit befasst sich mit der Datenfusion von Kamera- und Lidardaten. Ziel dabei ist es, auf den Daten der Kamera eine Instanz-Segmentierung mithilfe eines neuronalen Netzes anzuwenden und die daraus gewonnen Instanzen mit den Entfernungsdaten des Lidar zu fusionieren, um so eine Erkennung der Umgebung herzustellen.
Für das Training des neuronalen Netzes werden die Daten in einem urbanen Umfeld aufgenommen und anschließend evaluiert, wie sich dieses Verfahren unter Verwendung simulierten Daten von Luftfahrzeugen verhält. Die Ergebnisse dieser Evaluation zeigen, dass es möglich, ist eine Umgebungserkennung mittels Kamera und Lidar für den urbanen Raum herzustellen. Dabei sind einige Probleme entdeckt worden, die sich negativ auf die Genauigkeit der Ergebnisse ausgewirkt haben.

Albert Sauer

Title of Thesis

Urban environment recognition for aircraft using data fusion of camera and lidar data

Keywords

environment recognition, data fusion, camera, lidar, machine learning, instance segmentation

Abstract

This work focuses on data fusion of camera and lidar data with the aim to apply instance segmentation to the camera data with the help of a neural net. By fusion of the nets instances with range data of the lidar sensor it was possible to create environment recognition. The neural net was trained with data collected from urban areas. Subsequently, the performance of the net with simulated data of an airborne vehicle was evaluated. The evaluation showed it is possible to create environment recognition using camera and lidar in an urban area. Few problems had developed which influenced the accuracy of the results negatively.

Inhaltsverzeichnis

Abbildungsverzeichnis **vii**

Tabellenverzeichnis **viii**

Abkürzungen **ix**

1 Einleitung **1**
1.1 Zielsetzung 2
1.2 Struktur der Arbeit 2

2 Grundlagen **4**
2.1 Forschungsstand 4
2.2 Datenfusion 5
2.2.1 Early Fusion 6
2.2.2 Late Fusion 6
2.2.3 Combined Fusion 6
2.2.4 Instanz-Segmentierung 6
2.2.5 Mask R-CNN 7
2.3 Metriken 7
2.3.1 Recall 8
2.3.2 Precision 8
2.3.3 Intersection over Union 8
2.3.4 Average Precision 9
2.3.5 Mean Average Precision 9

3 Entwurf **10**
3.1 Auswahl der Software 10
3.1.1 Sensorik 10
3.1.2 Kalibrierungssoftware 11
3.1.3 Roboflow 11

3.1.4 Detectron2 . . . 11
3.2 Auswahl der Hardware . . . 11
3.2.1 Kamera . . . 12
3.2.2 Lidar . . . 12
3.3 Auswahl von Datenfusions Verfahren . . . 13
3.4 Vorbereitung . . . 13
3.4.1 Kamera Kalibrierung . . . 14
3.4.2 Lidar-Kamera Kalibrierung . . . 15
3.4.3 Erhebung der Daten . . . 16
3.4.4 Klassen Auswahl . . . 17
3.4.5 Annotation . . . 17
3.5 Implementation . . . 18

4 Durchführung **19**
4.1 Training des neuronalen Netzes . . . 19
4.2 Durchführung von Tests verschiedener Fusionsverfahren . . . 19
4.2.1 Early Fusion . . . 20
4.2.2 Combined Fusion . . . 22
4.3 Testen der Implementation . . . 24

5 Evaluation **25**
5.1 Trainingsergebnisse . . . 25
5.1.1 Instanz-Segmentierung auf simulierten UAV Daten . . . 28
5.2 Fusionsergebnisse . . . 30
5.2.1 Durchführungszeit . . . 32

6 Diskussion **33**
6.1 Probleme bei der Early Fusion . . . 33
6.2 Genauigkeit der Instanz-Segmentierung . . . 33
6.3 Auswertung der Fusionsergebnisse . . . 34
6.4 Auswertung der Durchführungszeit . . . 34

7 Fazit **36**
7.1 Ausblick . . . 37

Literaturverzeichnis **38**

Selbstständigkeitserklärung **41**

Abbildungsverzeichnis

2.1 Instanz-Segmentierung 7
2.2 Illustration für IoU 8

3.1 Verwendete Hardware, Konstruktion bestehend aus 3D-Druck, Lidar, Kamera und Kamerastativ 12
3.2 Unterschied der Verfahren. 13
3.3 Kamerakalibrierung mit OpenCV mittels Schachbrettmuster 14
3.4 Variationen von Ausrichtungen des Schachbrettmuster 15
3.5 Lidar Daten von Abbildung 3.2 15
3.6 Berliner Tor HAW Campus von oben. Rote Linien zeigen wo die Trainingsdaten gesammelt wurden. Grüner Punkt zeigt den Ort der simulierten Drohnen Daten. 16
3.7 Annotation von Klassen. 18

4.1 Early Fusion von Kamera- und Lidardaten. 20
4.2 Tests mit verschiedenen Positionen im Bild 21
4.3 Instanz-Segmentierung. 22
4.4 Instanz-Segmentierung mit Lidar Messpunkten. 22
4.5 Instanzen notiert mit durchschnittlichen Entfernungen. 23
4.6 Aufteilung der Instanz. 24

5.1 Trainings- und Validierungsloss Verlauf. 25
5.2 Vergleich von mAP der neuronalen Netze. 26
5.3 AP bei Segmentierung der einzelnen Klassen. 27
5.4 Segmentierung von UAV Daten mit vielen Bäumen. 29
5.5 Segmentierung von UAV Daten mit Personen. 29
5.6 Datenfusion auf UAV Daten. 30
5.7 Segmentierungsmasken mit Lidarmesspunkte. 31

Tabellenverzeichnis

2.1 Sensorvergleich von Sensoren, welche öfters in der Fahrzeugautonomie verwendet werden und wie zuverlässig diese in den unterschiedlichen Faktoren funktionieren. 5

3.1 Klassendefinition 17
3.2 Anzahl der Instanzen 17

4.1 Ermittelte Abstände zu Abbildung 4.5 23
4.2 Minimale Distanzen der Abschnitte 24

5.1 Ermittelte AP beider Netze durch Validierungsdaten 28
5.2 AP bei simulierten UAV Daten 28
5.3 Ermittelte Abstände zu Abbildung 4.5 31
5.4 Benötigte, durchschnittliche Zeit der Verfahren bei 2734 Durchführungen . 32

Abkürzungen

AP	Average precision
CF	Combined Fusion
EF	Early Fusion
FP	False Positive
FPN	Feature Pyramid Network
FN	False Negative
HAW	Hochschule für Angewandte Wissenschaften
i-LUM	innovative Luftgestützte Urbane Mobilität
IoU	Intersection over Union
Lidar	light detection and ranging
LF	Late Fusion
mAP	Mean Average Precision
TN	True Negative

ToF Time of Flight

TP True Positive

ROS Robot Operating System

UAV Unmanned aerial vehicle

1 Einleitung

Die Hochschule für Angewandte Wissenschaften (HAW) nimmt an einem Forschungsprojekt teil, beim das Thema innovative Luftgestützte Urbane Mobilität (i-LUM)[11] erforscht wird. Aufgabe der HAW ist es dabei, mögliche Lösungen für die Autonomie, der Sensorik und der Kommunikation eines Unmanned aerial vehicle (UAV) zu finden. Für Autonome Systeme in einem Urbanen Umfeld ist die Umgebungserkennung besonders wichtig. Da es dort viele Hindernisse gibt, müssen diese erfasst und klassifiziert werden, um so eine gute Wahrnehmung der Umgebung zu erreichen und sich so sicher in seinem Umfeld bewegen zu können.

Es gibt bereits viele Verfahren eine Umgebungserkennung im urbanen Umfeld umzusetzen. Viele verwenden dabei Sensoren wie Kameras oder Lidar in Kombination mit neuronalen Netzen. Einige UAV's [10] nutzen dabei einige dieser Sensoren, um Kollisionen bei geplanten Missionen zu vermeiden, wie zum Beispiel bei Inspektionen von Brücken oder Gebäuden. Um künftig den urbanen Luftraum für Paketdrohnen bis hin zu Lufttaxis zu nutzen, erfordert dies neue Lösungen, welche die Wahrnehmung der Umgebung durch eine Datenfusion erweitern.

In der Autonomie von Fahrzeugen gibt es bereits Lösungen, welche dabei helfen die Umgebung zu erkennen und es so ermöglichen, Fahrzeuge durch den urbanen Raum zu navigieren. Waymo[21] hat zum Beispiel bereits so eine Lösung umgesetzt, bei welcher der erste Ansatz funktioniert, auch wenn dieser noch nicht ausgereift ist und es dort öfter zu Problemen kommt. Bei diesem Ansatz wurden Sensoren wie Kameras und Lidar genutzt und ein Datensatz entwickelt, welcher dabei helfen soll ein neuronales Netz zu trainieren. Dieser und andere entwickelten Datensätze, wie zum Beispiel der Kitti-360 Datensatz[14], ermöglichten es, auf deren Basis weitere neuronale Netze zu erforschen, welche eine Umgebungserkennung ermöglichen.

1.1 Zielsetzung

Im Rahmen dieser Arbeit wird ein Verfahren der Datenfusion vorgestellt, welches ein neuronales Netz für eine Bildsegmentierung nutzt und anschließend mit den Entfernungsdaten des Lidar fusioniert. Zur Nutzung dieses Verfahrens soll des Weiteren ein mögliches Verfahren zur Kollisionsvermeidung implementiert werden.

Im Bereich des maschinellen Lernens gibt es für die Bildsegmentierung viele verschiedene Architekturen. Für diese Arbeit wurde sich auf die Mask R-CNN[15] Architektur beschränkt. Bei der Auswahl des neuronalen Netzes wurde darauf geachtet, dass ein guter Ausgleich zwischen kurzer Inferenzzeit und hoher mean average precision (mAP) besteht. Dies soll ermöglichen auch bei kleineren UAV's, wie Paketdrohnen, eine Umgebungserkennung in Echtzeit zu erreichen.

Im i-LUM Projekt könnte die Umgebungswahrnehmung für die Kommunikation zwischen den Drohnen genutzt werden, um Metadaten von erkannten Objekten auszutauschen und so eine kollektive Wahrnehmung zu generieren.
Auch in der Forschungsgruppe Autosys[22] könnte dieses Verfahren auf Roboterplattformen, wie zum Beispiel dem Husky, genutzt werden, um so mit der Umgebung zu interagieren.

1.2 Struktur der Arbeit

Zunächst werden in Kapitel 2 die nötigen Grundlagen für die Arbeit erläutert.

In Kapitel 3 wird ein Entwurf vorgestellt, welcher beschreibt, wie die Datenfusion umgesetzt werden kann und welche Hardware und Software benötigt wird, um diesen Entwurf zu realisieren. Des Weiteren wird die Vorbereitung erklärt, welche essentiell ist für die Datenfusion von Kamera- und Lidardaten. Abschließend wird eine Implementation vorgestellt, welche das Verfahren der Datenfusion nutzt.

In Kapitel 4 werden die Vorbereitungen genutzt, um das Training, die Datenfusion und die Implementation durchzuführen.

In dem Kapitel 5 werden die Ergebnisse des vorherigen Kapitels vorgestellt und evaluiert.

In Kapitel 6 werden die aus von vorherigen Kapitel gewonnen Ergebnisse diskutiert und weitere mögliche Verbesserungen erläutert.

Abschließend wird in Kapitel 7 ein Fazit zu den gewonnen Erkenntnissen gezogen und ein Ausblick für zukünftige Arbeiten gegeben.

2 Grundlagen

In diesem Kapitel wird erläutert, welche Verfahren der Datenfusion es gibt und worin der wesentliche Unterschied zwischen ihnen besteht.
Des Weiteren wird erläutert, welche verschiedenen Variationen von Segmentierungsnetzen es gibt und wie sie sich voneinander unterscheiden. Zusammen mit den Segmentierungnetzen werden deren Metriken erläutert, welche später für die Evaluierung des Netzes wichtig sind.

2.1 Forschungsstand

Neben der Datenfusion gibt es auch andere Verfahren, die nur einzelne Sensoren verwenden, um eine Umgebungswahrnehmung mithilfe von neuronalen Netzen herzustellen. Das VoxelNet[26] war eines der ersten robusten Verfahren, welches alleine durch die Verwendung eines Lidar Sensor 3D-Objekte detektieren konnte. Dieses Verfahren verwendet für das Training und als Vergleichsmaßstab den Kitti Datensatz[14], welcher über 7000 annotierte Bilder und Punktwolken von Menschen, Radfahrern und Fahrzeugen enthält. Die Verfahren, welche diesen Datensatz für ein Training nutzen, sind allerdings auf diesen Datensatz beschränkt, sodass die Verwendung eines anderen Lidar Sensors nicht möglich ist. Zudem würde die Erstellung eines so großen Datensatz mit der hauseigenen Sensorik über den Rahmen dieser Arbeit hinausgehen, weshalb hier auf ein Verfahren der Datenfusion zurückgegriffen.

Bei einer Umgebungswahrnehmung, bei welcher Kameradaten genutzt werden, werden unterschiedlichste Verfahren der Bildsegmentierung genutzt. In der Miniaturautonomie [19] wird zum Teil die semantische Segmentierung genutzt, um Fahrbahnmarkierungen zu erkennen. Doch bei einer semantischen Segmentierung werden nur Klassifizierungen voneinander unterschieden. Für den Anwendungsfall in dieser Arbeit eignet sich dafür eine Instanz-Segmentierung deutlich besser. Diese ermöglicht es, erkannte Objekte der

gleichen Klassifizierung voneinander zu unterscheiden. Dies hat den Vorteil, dass jedem gefundenen Objekt seine eigene Distanz zugewiesen werden kann. Eine häufig genutzte Architektur der Instanz-Segmentierung ist das Mask R-CNN, welches in dieser Arbeit vorgestellt und auf den Anwendungsfall trainiert wird.

2.2 Datenfusion

Die Idee der Datenfusion ist es, Defizite eines Sensors mit einem anderen auszugleichen. Die Folgende Tabelle 2.1 zeigt einige Sensoren auf, unter welchen Faktoren deren Defizite auftreten können und mit welchen diese kombiniert werden können, um Defizite auszugleichen. Dabei zeigen Sensoren mit einem "✓" keine Probleme unter dem ausgewählten Faktor auf. Bei Sensoren mit einem " ∼ " funktionieren diese oft noch akzeptabel unter den spezifischen Faktoren. Die mit einem "x" zeigen auf, dass diese nicht zuverlässig unter den spezifischen Faktoren arbeiten.

Tabelle 2.1: Sensorvergleich von Sensoren, welche öfters in der Fahrzeugautonomie verwendet werden und wie zuverlässig diese in den unterschiedlichen Faktoren funktionieren.

Faktoren	**Kamera**	**Lidar**	**Radar**	**Fusion**
Reichweite	∼	∼	✓	✓
Auflösung	✓	∼	x	✓
Distanz Genauigkeit	∼	✓	✓	✓
Beschleunigung	∼	x	✓	✓
Farberkennung	✓	x	x	✓
Objekterkennung	∼	✓	✓	✓
Objekt Klassifizierung	✓	∼	x	✓
Fahbahnlienenerkennung	✓	x	x	✓
Hindernisserkennung	✓	✓	x	✓
Helligkeitsbeeinflussung	x	✓	✓	✓
Wetterbeeinflussung	x	∼	✓	✓

Quelle: [25]

Dabei gibt es unterschiedliche Verfahren, die Daten von mehreren Sensoren zu fusionieren. Im Folgenden werden die Verfahren early fusion (EF), late fusion (LF) und combined fusion (CF) genauer erläutert[13].

2.2.1 Early Fusion

Bei der EF werden die rohen Daten der Sensoren miteinander fusioniert. Die Fusion von Kamera- und Lidardaten ermöglicht es, für jeden gemessen Entfernungspunkt des Lidar den dazugehörigen Pixel im Kamerabild zu ermitteln. Die Ausführungszeit dieses Verfahrens ist somit nur von der Berechnungszeit der Transformation von den 3D-Koordinaten des Lidar in das Kamerabild abhängig.

2.2.2 Late Fusion

Bei Anwendung der LF arbeiten die Sensoren unabhängig voneinander. Dabei werden die erfassten Features aus den Daten der Sensoren extrahiert und auf diese zum Beispiel eine Objektklassifizierung angewendet. Die Ergebnisse der einzelnen Sensoren werden anschließend fusioniert. Dieses Verfahren ermöglicht, bei einer Objektklassifizierung eine höhere Genauigkeit zu erzielen. Dies hat zur Folge, dass der Schwellwert der Objektklassifizierung gesenkt werden kann und so mehr Objekte erkannt werden können [25]. Hier ist zu beachten, dass die Inferenzzeit beider Sensoren sich negativ auf die Ausführungszeit auswirken kann.

2.2.3 Combined Fusion

Die CF ist eine Kombination aus EF und LF. Bei diesem Verfahren werden Rohdaten des einen Sensor mit zum Beispiel der Objektklassifizierung des anderen Sensors kombiniert. Dies hat den Vorteil, dass dadurch eine kürzere Inferenzzeit erreicht werden kann.

2.2.4 Instanz-Segmentierung

Die Instanz-Segmentierung ordnet sich in der Kategorie des überwachten Lernens im Bereich des maschinellen Lernens ein. Das Verfahren unterteilt sich dabei in zwei wesentlich Schritte. Beim ersten Schritt geht es darum, individuelle Objekte in einem Bild zu klassifizieren und zu lokalisieren, so wie es auch in einem Objektdetektierungsnetz angewendet wird. Im zweiten Schritt wird für jede gefundene Instanz eine Segmentierung durchgeführt. Dabei wird jedem Pixel im Bild eine Klassifizierung zugeordnet und daraus eine Binärmaske erstellt (siehe Abbildung 2.1).

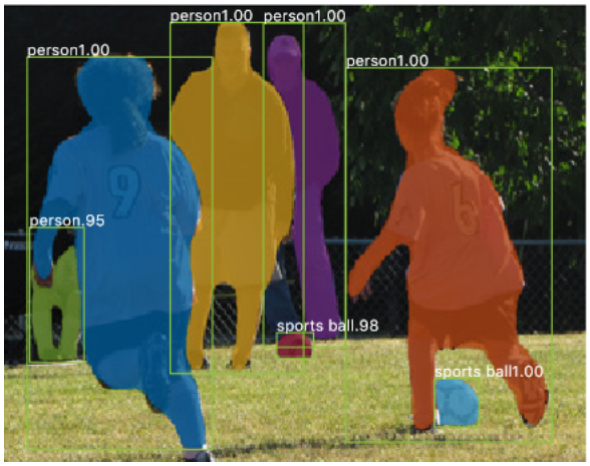


Abbildung 2.1: Instanz-Segmentierung
Quelle: [15]

2.2.5 Mask R-CNN

Die Architektur des Mask R-CNN ist eine Erweiterung von Faster R-CNN[18]. Das ursprüngliche Faster R-CNN ist ein Objekterkennungsnetz und besitzt zwei Ausgänge für jedes gefundene Objekt. Der erste Ausgang ist die Klasse des Objekts und der zweite der Offset für die Bounding-Box. Der Unterschied zwischen dem Faster R-CNN und dem Mask R-CNN besteht darin, dass dem Mask R-CNN ein weiterer Ausgang hinzugefügt wurde, welcher die Binärmaske des Objekts ausgibt.
Die Autoren des Mask R-CNN haben mit unterschiedlichen Backbone-Netzen experimentiert. Das in dieser Arbeit verwendete R50-FPN ist ein ResNet mit 50 Schichten und verwendet ein Feature Pyramid NEtwork (FPN)[17] für die Merkmalserkennung. Dieser Ansatz führte dazu, dass sich sowohl die Genauigkeit als auch die Geschwindigkeit des Netzes verbesserte.

2.3 Metriken

Die Metriken, die hier erläutert werden, sollen dabei helfen, das verwendete neuronale Netz evaluieren zu können. Schließlich werden dadurch die Bereiche des Trainings in Erfahrung gebracht, welche erfolgreich waren. Zusätzlich hilft es dabei, herauszufinden, bei welchen Klassifikationen mehr Daten und Training benötigt werden.

2.3.1 Recall

Der Recall setzt sich aus dem vom neuronalen Netz richtig erkannten true positive (TP) und dem falsch erkannten false negative (FN) zusammen. Damit wird berechnet, wie viele der tatsächlich richtigen Klassifikationen erkannt wurden.

$$recall = \frac{true\ positive}{true\ positive + false\ negative} \tag{2.3.1a}$$

2.3.2 Precision

Die Precision setzt sich aus dem TP und dem False Positiv(FP) zusammen. Es beschreibt das Verhältnis der richtig erkannten Klassifikationen.

$$precision = \frac{true\ positive}{true\ positive + false\ positiv} \tag{2.3.2a}$$

2.3.3 Intersection over Union

Die Intersection over Union (IoU) wird häufig bei Objektdetektoren genutzt, kann allerdings auch bei einer Instanz-Segmentierung genutzt werden. Dabei werden die Überschneidungen der von dem neuronalen Netz geschätzten mit den Tatsächlichen Pixeln eines Objekts überprüft und anschließend durch die gesamte Fläche der Pixel geteilt (siehe Abbildung 2.2). Je mehr Pixel sich überschneiden, desto besser ist der IoU Wert. Diese Werte werden generell im Bereich von 0 bis 1 normiert, wobei eine hundertprozentige Überschneidung den Wert 1 hat.

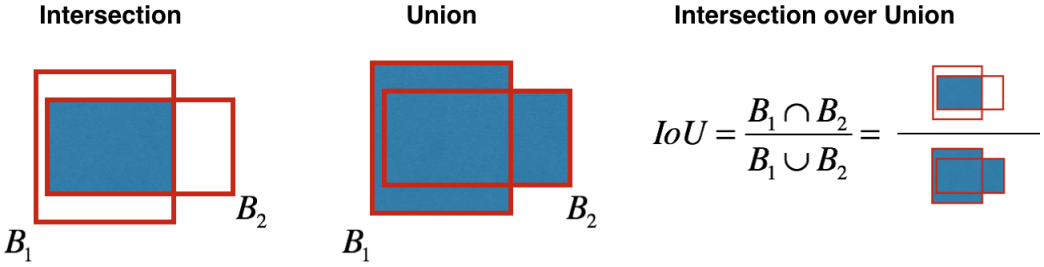


Abbildung 2.2: Illustration für IoU
Quelle: [16]

2.3.4 Average Precision

Die Average Precision (AP) wird durch die Precision und dem Recall berechnet. Dabei wird die Fläche unter einer Kurve berechnet, die sich aus dem Verhältnis von Precision und Recall ergibt.

2.3.5 Mean Average Precision

Mean Average Precision (mAP) wird als Metrik für die Lokalisierung und Klassifizierung eines erkannten Objektes bei Objektdetektoren genutzt. Sie ist die mittlere AP über alle Klassen, auf die das neuronale Netz trainiert wurde.

3 Entwurf

Hier wird ein Entwurf vorgestellt, beim dem eine mögliche Lösung unter Verwendung von bestehenden Komponenten für eine Datenfusion entwickelt wird. Diese soll es ermöglichen, anhand der Daten von Kamera und Lidar eine Umgebungserkennung zu realisieren. Des Weiteren soll die Implementierung dieser Lösung zeigen, wie es dadurch möglich ist, Hindernissen in der urbanen Umgebung auszuweichen. Dafür werden außerdem die Auswahl von Hardware, Software, Bibliotheken und Architekturen erläutert, sowie die Vorbereitung dieser Komponenten.

3.1 Auswahl der Software

Bei der Software wird Unterschieden zwischen Software, die benötigt wird für die Ansteuerung der Sensoren, der Kalibrierung der Sensoren und der Annotation der Daten, und nötige Software, die für Training und Ausführung des maschinellen Lernens verantwortlich ist.

3.1.1 Sensorik

Um die Daten sammeln zu können, wurde die Software Robot Operating System (ROS) Noetic [8] verwendet. Zudem wurden die ROS-Pakete *realsense-ros*[2] und *rslidar_sdk*[6] von den Sensorherstellern genutzt, um die Sensoren auslesen zu können. Bei der Aufnahme der Daten wurden ROS-Bags verwendet, welche sich auf die publizierten Nachrichten der jeweiligen Sensoren anmelden und so die Daten empfangen und abspeichern. Des Weiteren wurde das ROS-Paket *rviz*[9] genutzt, um die Daten bei der Aufnahme zu visualisieren.

3.1.2 Kalibrierungssoftware

Für die Kalibrierung der Kamera wird die OpenCV Bibliothek mit Python 3.7 verwendet [4]. Diese ermöglicht es, durch wenig Aufwand eine Kamerakalibrierung durchzuführen. Bei der Kalibrierung der Kamera mit dem Lidar wird ein ROS-Paket genutzt, welches aus einem wissenschaftlichen Bericht entstanden ist. Die Autoren dieses Berichtes haben dafür eine schrittweise Anleitung in einem Git-Repo hochgeladen[23].

3.1.3 Roboflow

Um die Daten zu annotieren, wurde die online Software Roboflow [5] verwendet. Diese ermöglicht es, Daten sowohl mit Boxen als auch mit Polygonen zu versehen und anschließend in das benötigte COCO-Format zu konvertieren.

3.1.4 Detectron2

Detectron2 [24] ist ein von Meta Research [3] erstelltes Softwaresystem, welches state-of-the-art Architekturen verwendet, um eine Objekterkennung zu erzielen. Es ermöglicht das Training für Objekterkennung, semantische Segmentierung, Instanz-Segmentierung, panoptische Segmentierung und eine Schlüsselpunkterkennung. Dieses Softwaresystem wird in dieser Arbeit für das Training, die Ausführung und die Evaluierung der Instanz-Segmentierung genutzt.

3.2 Auswahl der Hardware

Um Daten mit der Hardware aufnehmen zu können, wurde ein 3D-Druck angefertigt. Dadurch war es möglich, Lidar und Kamera in die gleiche Richtung auszurichten. Anschließend wurde die Konstruktion auf einem Stativ befestigt (siehe Abbildung 3.1), um die Daten mobil aufzunehmen.

Abbildung 3.1: Verwendete Hardware, Konstruktion bestehend aus 3D-Druck, Lidar, Kamera und Kamerastativ

3.2.1 Kamera

Die in dieser Arbeit verwendet Kamera ist eine Intel Realsense D435 [1]. Mit der Kamera ist es möglich, sowohl RGB als auch Stereodaten aufzunehmen. Allerdings wurde hier nur die RGB-Kamera als Monokamera genutzt. Die Auflösung der RGB-Kamera beträgt 1980 x 1080 Pixel und sie besitzt eine Field of View (FoV) von 69° x 42°. Videos können mit bis zu 30 Bildern pro Sekunde aufgenommen werden.

3.2.2 Lidar

Der verwendete Lidar Sensor ist ein Robosense RS-Lidar-16 [7]. Dieser arbeitet nach dem Time of Flight (ToF) Prinzip. Dabei werden gepulste Lichtwellen in die Umgebung gestrahlt und für jede Lichtwelle die Zeit gemessen, die sie für das Auftreffen auf eine Oberfläche und die folgende Reflexion benötigt. Anhand dieser Zeit kann die Distanz zur bestrahlten Oberfläche berechnet werden.
Der Sensor hat eine horizontale Sichtweite von 360° und misst dabei in 16 verschiedenen Ebenen. Er hat so eine vertikale Sichtweite von 30° bei einer Auflösung von 2° pro Ebene. Die maximale Reichweite des Sensors beträgt 150 m bei einer Genauigkeit von bis zu 2 cm. Der Sensor wurde in dieser Arbeit so eingestellt, dass er eine Bildrate von 10 Hz erreicht und sich mit 600 Umdrehungen pro Sekunde dreht, wodurch er eine Horizontale

Auflösung von 0.2° erreicht. Dabei werden pro Sekunde 300k Punkte in einer Punktwolke generiert.

3.3 Auswahl von Datenfusions Verfahren

Bei dem Verfahren der Datenfusion mit der CF wurde eine Instanz-Segmentierung einer klassischen Objekterkennung mit Begrenzungsrahmen vorgezogen. Durch die Begrenzungsrahmen würden sich bei einer Datenfusion viel mehr Datenpunkte des Lidar innerhalb des Begrenzungsrahmen befinden und sich überschneiden. Diese Überschneidungen führen dazu, dass Objekte, die hinter einem erfassten Objekt sind, mit in die Messungen aufgenommen werden und die Distanz zum erfassten Objekt schwerer zu ermitteln ist. Dadurch wird die Genauigkeit verschlechtert. Die Abbildung 3.2 verdeutlicht diesen Unterschied noch einmal. Die linke Abbildung stellt die Objekterkennung mittels Begrenzungsrahmen dar und die rechte Abbildung die Segmentierung. Die Roten Linien stellen die Lidar Messpunkte dar, die Blauen Rechtecke den Begrenzungsrahmen und die Blaue Einfärbung auf der rechten Abbildung zeigt dabei die Segmentierung. Wie zu sehen sind hier auf der linken Abbildung die Messpunkte von dem dahinter liegenden Gebäude innerhalb des Begrenzungsrahmen und auf der rechten Abbildung sind diese nur auf der Segmentierungmaske.

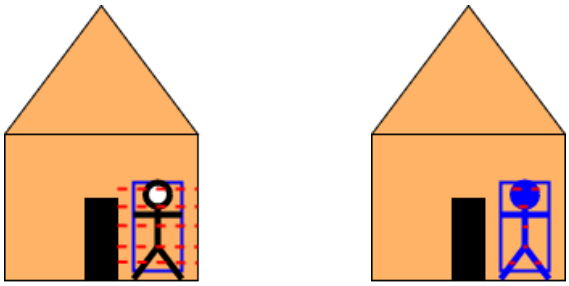


Abbildung 3.2: Unterschied der Verfahren.

3.4 Vorbereitung

Zur Vorbereitung ist die Kalibrierung für die Datenfusion der wichtigste Schritte. Dabei gibt es unterschiedliche Verfahren, eine Kamera mit einem Lidar zu kalibrieren. Hier wurde ein zielbasiertes Verfahren ausgewählt, bei dem ein Schachbrettmuster genutzt wird,

um die extrinsischen Parameter festzulegen. Es ist zu beachten, dass für den ausgewählten Lidar ein großes Schachbrettmuster gebraucht wird, damit später genug Datenpunkte für die Berechnung der Kalibrierung erfasst werden. Die Maße des Schachbrettmuster sind 840 x 595 mm und hat eine Muster von 7 x 6 Kacheln.

3.4.1 Kamera Kalibrierung

Bevor der Lidar mit der Kamera kalibriert werden kann, muss erst die Kamera selbst kalibriert werden, um die intrinsischen Parameter P festzulegen. Diese werden im Anschluss für die Lidarkalibrierung benötigt. Die intrinsischen Parameter ergeben sich aus der Brennweite f_x und f_y und dem optischem Zentrum c_x und c_y (siehe Formel 3.3.1a). Diese werden durch die Kamerakalibrierung von OpenCV berechnet[4].

$$P = \begin{bmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{3.3.1a}$$

Für die Kamerakalibrierung mit OpenCV werden 20 Aufnahmen von verschiedenen Posen des Schachbrettmusters gemacht, bei denen die Eckpunkte der Muster erfasst werden (siehe Abbildung 3.3). Da die Maße des Schachbrettmusters bekannt sind, ist es möglich, die Transformation von den Punkten der 3D-Umgebung in das 2D-Bild zu berechnen und so die intrinsischen Parameter festzulegen.

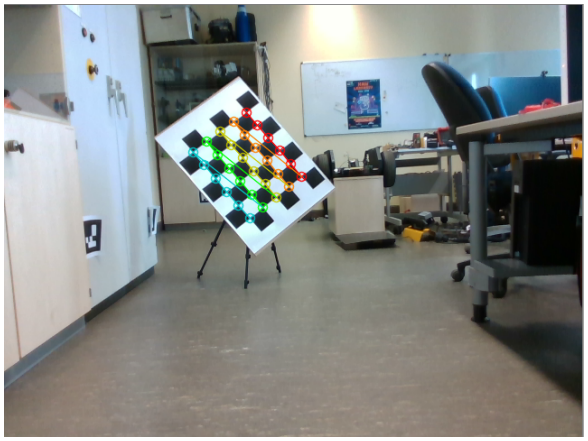


Abbildung 3.3: Kamerakalibrierung mit OpenCV mittels Schachbrettmuster

3.4.2 Lidar-Kamera Kalibrierung

Bei dieser Kalibrierung wird zeitgleich der Lidar und die Kamera verwendet, um die extrinsischen Parameter festzulegen. Dabei wurden 40 verschiedene Aufnahmen von verschiedenen Ausrichtungen im Abstand von 1.5 m bis 3.5 m des Schachbrett aufgenommen. Bei der Ausrichtung ist es wichtig, viele Variationen von Drehung und Neigung des Schachbrettmusters herzustellen, um so bei der späteren Berechnung eine Überanpassung zu vermeiden. Zudem ist darauf zu achten, dass das Schachbrettmuster um 45° zum Boden gedreht wird (Abbildung 3.4). Dadurch wird erreicht, dass der Lidar mehr Messpunkte auf der Fläche des Schachbrettmusters bekommt Tsai u. a. [23].

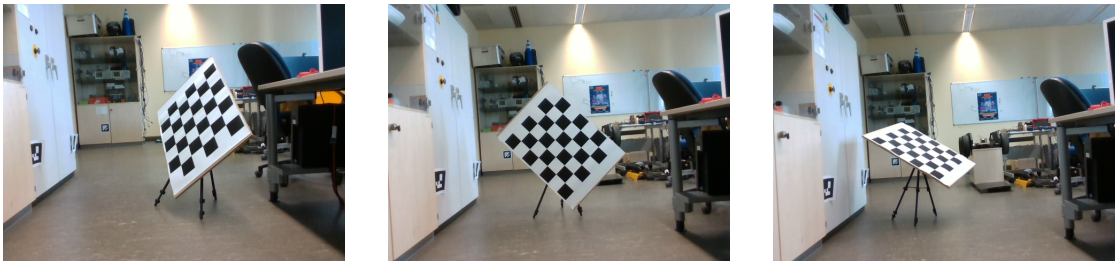
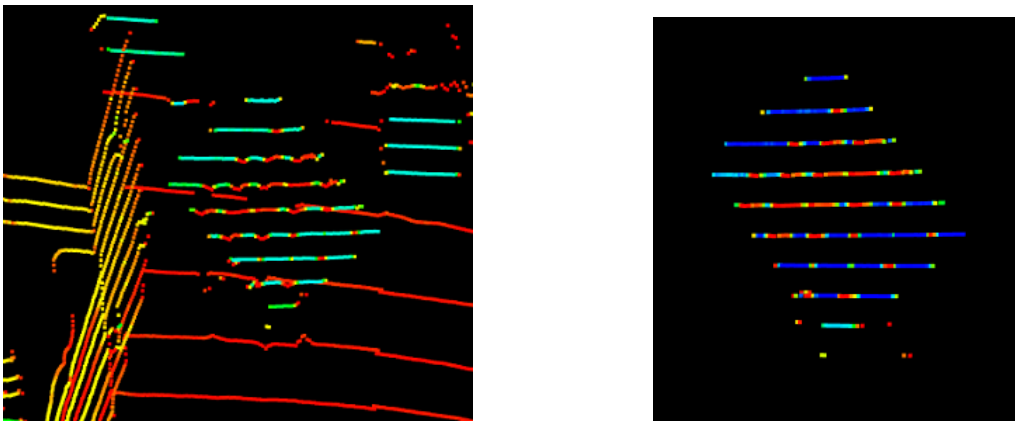


Abbildung 3.4: Variationen von Ausrichtungen des Schachbrettmuster

Bevor die Berechnung der Transformation von Lidar zu Kamera durchgeführt werden kann, muss in der Punktwolke der Lidardaten das Schachbrettmuster von der Umgebung isoliert werden (siehe Abbildung 3.5). Dies wird durch Regler in Rviz eingestellt, welche die Punktwolke auf die maximale und minimale X,Y und Z-Richtung begrenzen.



(a) Rohe Lidar Daten mit Schachbrettmuster (b) Schachbrettmuster isoliert

Abbildung 3.5: Lidar Daten von Abbildung 3.2

Anschließend wird die Aufnahme für die Ausrichtung bestätigt. Daraufhin wird für die jeweilige Aufnahme in dem Kamerabild mittels OpenCV die Eckpunkte der Muster erfasst und so der Mittelpunkt und der Normalvektor n_{cam} auf der Ebene des Schachbrettmuster ermittelt. Bei den Lidardaten wird dies mittels eines random sample consensu (RANSAC) Algorithmus durchgeführt und so der Normalvektor n_{lidar} berechnet. Abschließend werden für alle Normalvektoren der Kamera $N_{cam} = [n^0_{cam}, ..., n^i_{cam}]^T$ und des Lidar $N_{lidar} = [n^0_{lidar}, ..., n^i_{lidar}]^T$ die Rotationsmatrix R^{cam}_{lidar} berechnet Tsai u. a. [23].

3.4.3 Erhebung der Daten

Da es in dieser Arbeit nicht möglich war, die verwendete Sensorik auf einer Drohne zu montieren, um Daten aufzunehmen, wurden teilweise Daten auf der Terrasse des BT21 aufgenommen. Diese Terrasse hat eine Höhe von ungefähr 30 m und die Daten sollen so einen kurzen Flug einer Drohne simulieren. Des Weiteren wurde für das Training des neuronalen Netz, Daten mobile vom Boden aus aufgenommen (Abbildung 3.6), welche dann anschließend annotiert wurden.

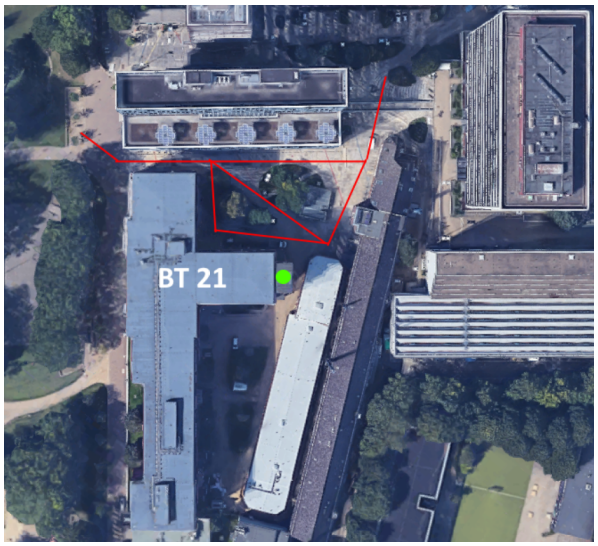


Abbildung 3.6: Berliner Tor HAW Campus von oben. Rote Linien zeigen wo die Trainingsdaten gesammelt wurden. Grüner Punkt zeigt den Ort der simulierten Drohnen Daten.

Quelle: Google Maps

3.4.4 Klassen Auswahl

Für die Instanz-Segmentierung des neuronalen Netz, werden vorher Klassen ausgewählt, auf die es trainiert wird. Dafür wurden Klassen gewählt, welche in einer urbanen Umgebung, sowohl im Luftraum als auch bei einer Landung, ein Hindernis darstellen können. Im Luftraum treten hier zum Beispiel Hindernisse auf, wie Bäume, Gebäude, Kabel, Vögel, Luftfahrzeuge oder auch hohe, schmale Objekte, wie Straßenbeleuchtungen oder Schornsteine. Davon treten auch einige bei der Landung als Hindernis auf. Dazu kommen beispielsweise noch mehr Hindernisse hinzu wie Menschen, Radfahrer und Kraftfahrzeuge. Aus dieser Sammlung von Hindernissen wurden davon vier ausgewählt, welche auch häufig in den gesammelten Daten vorkommen. Diese Auswahl beschränkt sich auf Bäume, Menschen, Gebäude, Straßenbeleuchtungen und Schornsteine, wobei die letzten beiden als eine Klasse angesehen werden. Tabelle 3.1 zeigt, die sich daraus ergebenden Klassendefinitionen.

Tabelle 3.1: Klassendefinition

Klassifizierung	**Bezeichnung**	**Klassen Nr.**
Gebäude	building	1
Menschen	person	2
Hohe, schmale Objekte	pole	3
Bäume	tree	4

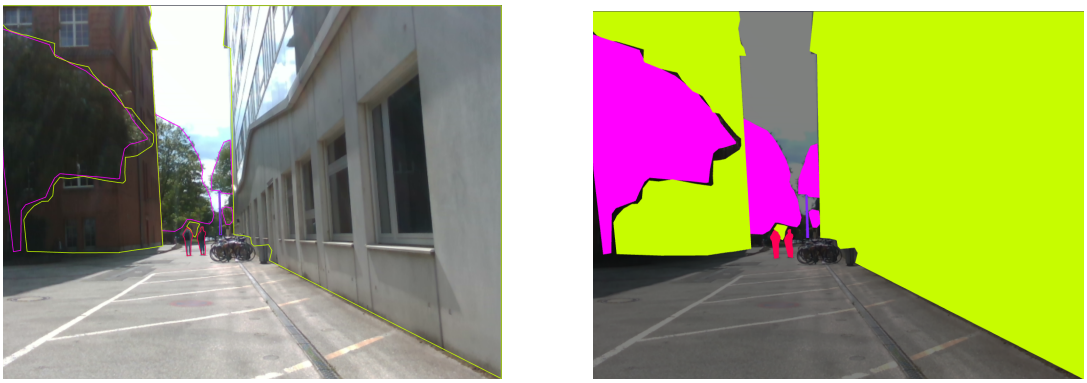
3.4.5 Annotation

Es wurden insgesamt 133 Bilder annotiert. Diese wurden anschließend durch Augmentation erweitert, sodass der daraus folgende Datensatz für das Training eine Anzahl von 313 Bildern umfasst. Durch die Augmentation wurde so auch die Anzahl der zu trainierenden Instanzen gesteigert (siehe Tabelle 3.2), welche zu einer Verbesserung des Trainings führen.

Tabelle 3.2: Anzahl der Instanzen

Klassifikation	**Ohne Augmentation**	**Mit Augumentation**
Person	300	705
Gebäude	510	1200
Hohe, schmale Objekte	186	438
Bäume	366	861

Bei der Annotation wurden die ausgewählten Klassen mit Polygonen annotiert. Dabei wurden die Klassen so annotiert, dass es keine Überschneidungen der Polygone gibt (siehe Abbildung 3.7). Dies soll beim späteren Training die Genauigkeit und die Trainingsdauer verbessern. Da die Überschneidungen dazu führen können, dass sich segmentierte Pixel der Klassen überlappen, was durch die IoU eine negative Auswirkung auf die jeweiligen Klassen hat. Außerdem können die Überschneidungen bei der späteren Datenfusion zu weiteren Problemen führen.



(a) Eingezeichnete Polygone. (b) Schichten der Polygone

Abbildung 3.7: Annotation von Klassen.

3.5 Implementation

Bei der Implementation wird ein einfaches Verfahren vorgestellt, welches die Datenfusion nutzt, um mit den erhaltenen Daten eine mögliche Kollisionsvermeidung durchzuführen. Bei diesem Verfahren wird der Begrenzungsrahmen der erkannten Objekte jeweils vertikal in 3 Abschnitte aufgeteilt. Anschließend wird innerhalb jedes Abschnittes geprüft, ob sich die Lidar Messpunkte mit den Pixeln der Segmentierungsmaske überschneiden. Bei einer Überschneidung wird die Distanz in einer Liste für den jeweiligen Abschnitt gespeichert. Bei genügend Überschneidungen werden jeweils 5% vom Anfang und Ende der Listen entfernt, um so mögliche Messfehler zu entfernen. Anhand der drei entstandenen Listen kann nun zum Beispiel jeweils der minimale Abstand für jede Liste ermittelt werden. Daraus resultierend kann ein UAV so gesteuert werden, dass es in die Richtung lenkt, wo der Abstand am höchsten ist, um nicht direkt mit dem Objekt zu kollidieren. Dies ermöglicht es, im urbanen Raum an Fassaden von Häusern entlang zu fliegen.

4 Durchführung

Für die Durchführung wird zuerst das neuronale Netz mit der Instanz-Segmentierung trainiert, welches anschließend für die Tests der Fusionsverfahren und zuvor vorgestellten Implementation genutzt wird.

4.1 Training des neuronalen Netzes

Für das Training wurde der Datensatz aufgeteilt in Trainings-, Test- und Validierungsdaten. Dabei wurden 80% der 313 Bilder als Trainings-, 10% als Test- und 10% als Validierungsdaten genutzt. Dies ergibt 250 Trainingsbilder und 31 Bilder für jeweils Test und Validierung.
Das Training wurde auf einem Computer mit einer AMD Ryzen 9 3900X CPU mit zwölf Kernen, einer GTX 3070 mit 8 GB Grafikspeicher und einem 32 GB Arbeitsspeicher durchgeführt. Wie bereits in vorherigen Kapiteln erwähnt, werden hier Detectron2 mit deren jeweiliger Model Zoo Implementierung für die Durchführung des Trainings und die von Detectron2 vortrainierte R50-FPN Architektur verwendet. Anschließend wird diese Architektur zweimal auf den vorgestellten Datensatz trainiert. Dabei werden unterschiedliche, zufällig initiale Gewichte benutzt und das Training mit 15000 Lernschritten und einer BatchSize von vier durchgeführt.

4.2 Durchführung von Tests verschiedener Fusionsverfahren

Bei der Datenfusion der Kamera- und Lidardaten wird zuerst das Fusionsverfahren der EF erläutert und getestet. Die EF dient als die Basis, um die anschließende CF durchzuführen.

4.2.1 Early Fusion

Um die rohen Daten von Kamera und Lidar zu fusionieren, werden die intrinsischen Parameter der Kamera P und der Transformation von Lidar zu Kamera T_{lidar}^{cam} verwendet. Die Transformation ergibt sich dabei aus der Rotationsmatrix R_{lidar}^{cam} und der Translation t_{lidar}^{cam} von Lidar zu Kamera wie in Formel 4.2.1a Geiger u. a. [14] zu sehen ist.

$$T_{lidar}^{cam} = \begin{pmatrix} R_{lidar}^{cam} & t_{lidar}^{cam} \\ 0 & 1 \end{pmatrix} \tag{4.2.1a}$$

Die folgende Berechnung (Formel 4.2.1b) Geiger u. a. [14] ermöglicht es, anschließend einen 3D-Punkt x in den Lidar-Koordinaten auf einen Pixel y in den Bildkoordinanten zu berechnen. Damit diese Matrizen miteinander multipliziert werden können, müssen sie in die gleiche Form gebracht werden. Hierfür werden die euklidischen Koordinaten in homogene Koordinaten umgewandelt und anschließend miteinander multipliziert. Das Ergebnis daraus ist eine 3x1 Matrix. Der gesuchte Pixel y wird anschließend über eine Division des letzten Wertes in der Matrix berechnet, wobei der erste Eintrag für die x-Koordinate und der zweite für die y-Koordinate steht.

$$y = P\ T_{lidar}^{cam}\ x \tag{4.2.1b}$$

Dieses Verfahren wird für die EF über alle 3D-Punkte des Lidar, welche im Sichtfeld des Kamerabildes sind, berechnet. Die Abbildung 4.1 zeigt auf wie so eine EF aussieht. Dabei wird hier für jede unterschiedliche Entfernung ein anderer Farbverlauf dargestellt.

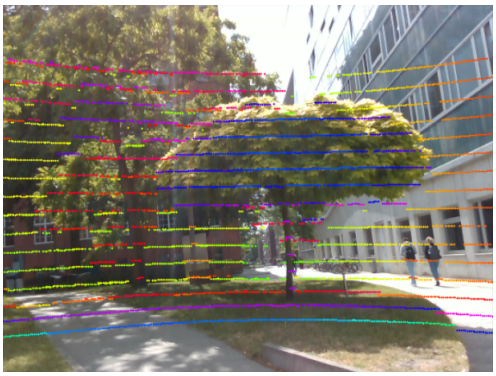


Abbildung 4.1: Early Fusion von Kamera- und Lidardaten.

Bei weiteren Tests sind bereits erste Probleme der EF aufgetreten. Eines der Problem besteht darin, dass die Messpunkte des Bildes nicht perfekt mit den Kanten der Objekte übereinstimmen und diese des Öfteren etwas verschoben sind. Für weitere Tests wurde geprüft, wann dieser Fall eintritt. Dafür bewegte sich eine Person von links nach rechts durch das Bild, sowie bei kurzen und weiter entfernten Distanzen. Die Abbildung 4.2 zeigt, dass bei der kurzen Distanz die eingezeichneten Lidar Messpunkte etwas nach rechts verschoben sind. Bei der weiter entfernten Distanz tritt dies weniger auf, dafür sind die Messpunkte hier etwas nach unten verschoben, sodass sich die Messpunkte des Baumes mit der davor stehenden Person überschneiden. Auch bei den seitlichen Bewegungen der Person sind hier die Messpunkte leicht nach rechts verschoben. Dieses Problem kann bei der anschließenden CF zu Abweichungen in der Genauigkeit führen.

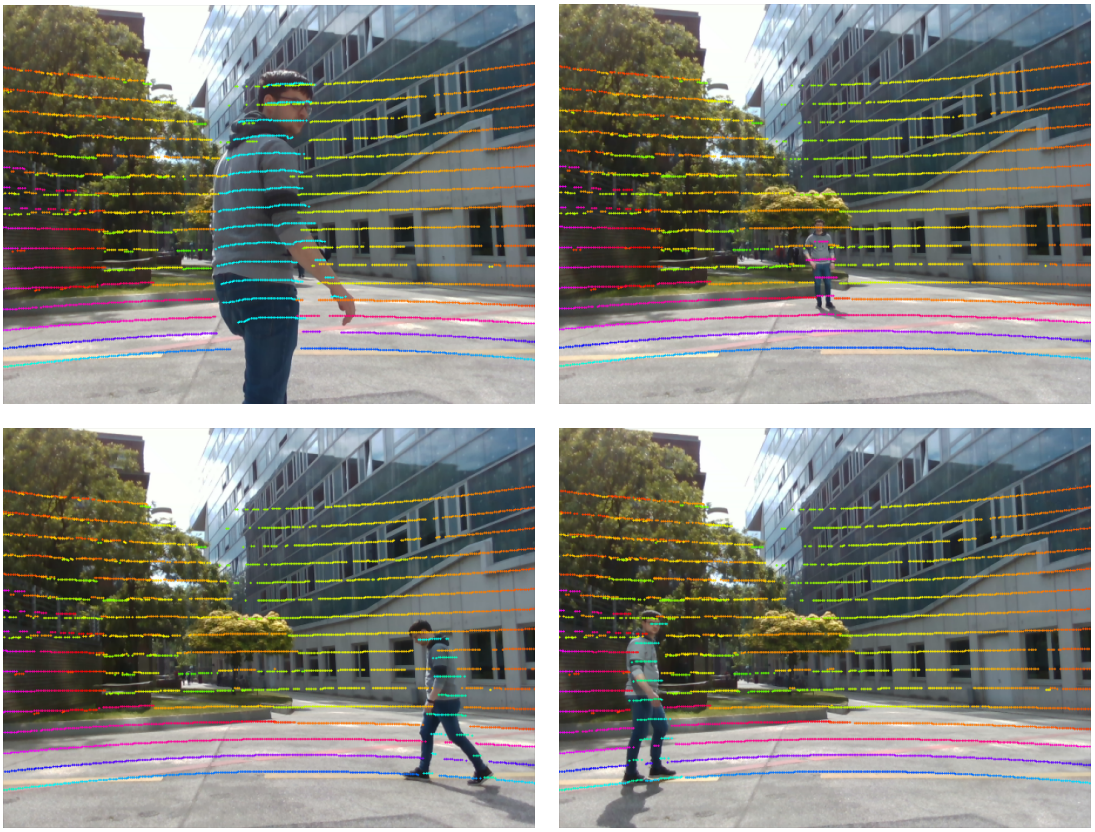


Abbildung 4.2: Tests mit verschiedenen Positionen im Bild

4.2.2 Combined Fusion

Da ein Teil der CF die EF beinhaltet, wird erneut die Berechnung in Formel 4.2.1b verwendet, um die Lidar Daten auf das Bild übertragen zu können. Bei der CF wird auf den Daten der Kamera die Instanz-Segmentierung durchgeführt und die Masken der erkannten Instanzen abgespeichert (siehe Abbildung 4.3). Die Grenze, ab wann eine Instanz erkannt werden soll, wurde dabei auf 85% festgelegt. An den Masken kann anschließend für alle Pixel geprüft werden, ob diese sich mit den Lidardaten überschneiden, wie in Abbildung 4.4 dargestellt wird.

Abbildung 4.3: Instanz-Segmentierung.

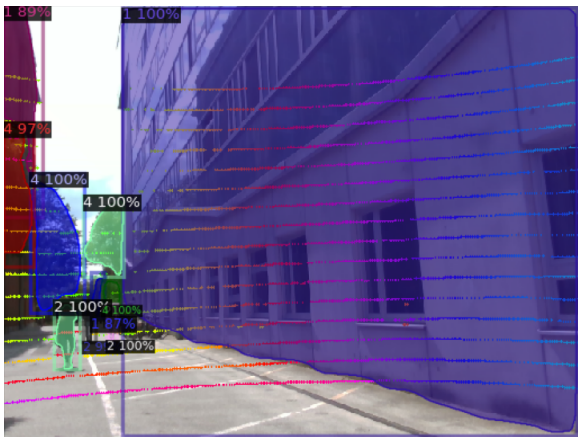


Abbildung 4.4: Instanz-Segmentierung mit Lidar Messpunkten.

Dabei wird für jede Überschneidung die Distanz in einer Liste für die jeweilige Instanz abgespeichert. Anhand dieser Liste können nun verschiedene Distanzen der erkannten In-

stanzen ermittelt werden, wie zum Beispiel die durchschnittliche Distanz zu einer Instanz. Dies soll die Abbildung 4.5 veranschaulichen. Für die Übersicht wurden die Instanzen hier mit einem farbigen Begrenzungsrahmen für die jeweilige Klasse eingerahmt und deren Entfernungswerte in Metern in der oberen linken Ecke des Begrenzungsrahmen notiert.

Abbildung 4.5: Instanzen notiert mit durchschnittlichen Entfernungen.

Die Abweichung, welche bereits beim Test in der vorherigen EF erwähnt wurde, führte auch hier zu einer größeren Abweichung in der Abstandsmessung zu den erkannten Instanzen. Die folgende Tabelle 4.1 zeigt dabei die Messungen zu den Instanzen aus Abbildung 4.5 mit den jeweiligen minimalen, maximalen und durchschnittlichen Abständen. Die Instanzen werden dabei von links nach rechts nummeriert.

Tabelle 4.1: Ermittelte Abstände zu Abbildung 4.5

Instanz	**Durchschnitt**	**Minimum**	**Maximum**
1. Gebäude	12,26 m	5,39 m	98,89 m
2. Gebäude	35,63 m	9,15 m	53,92 m
1. Baum	27,81 m	9,15 m	45,33 m
2. Baum	63,1 m	28,07 m	78,72 m
3. Baum	97,75 m	75,71 m	105,24 m
4. Baum	45,31 m	45,31 m	45,31 m
1. Person	12,86 m	12,74 m	12,99 m
2. Person	24,81 m	24,81 m	24,81 m
3. Person	47,4 m	25,16 m	67,46 m

Die Tabelle verdeutlicht, dass zum Beispiel bei der 2. Person alle Distanzen gleich sind. Dies liegt daran, dass nur ein Messpunkt des Lidar mit der Maske der Instanz übereinstimmt. Des Weiteren hat die 3. Person, welche direkt neben der 2. Person in der Abbildung steht, einen großen Unterschied in den Abstandsmessungen.

4.3 Testen der Implementation

Zum Testen der Implementation wird als Beispiel die Fassade eines Gebäudes genutzt, wie in Abbildung 4.6 zu sehen ist. Die Grüne Maske wurde von dem Segmentierungsnetz erstellt und die rot eingezeichneten Linien stellen die Trennung der jeweiligen Abschnitte dar.

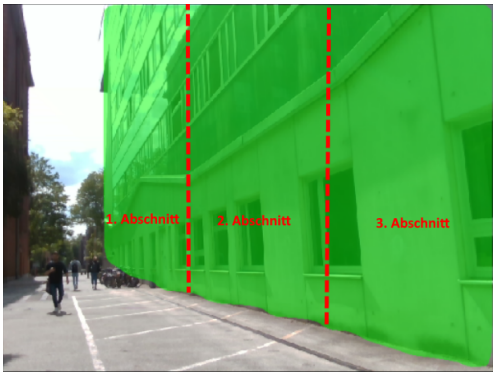


Abbildung 4.6: Aufteilung der Instanz.

Nachdem die CF durchgeführt wurde, wird die vorgestellte Implementation genutzt und für jeden Abschnitt die minimale Entfernung ermittelt. Wie in Tabelle 4.2 zu sehen ist, ist der kleinste, gemessene Abstand im 3. Abschnitt. Dieser Wert ist auch um 20 cm größer als der minimale Abstand, welcher im Test der zuvor gezeigten CF ermittelt wurde. Anhand dieser drei Abstände kann nun vorausgesagt werden, in welche Richtung ein UAV steuern muss, um nicht mit der Fassade zu kollidieren.

Tabelle 4.2: Minimale Distanzen der Abschnitte

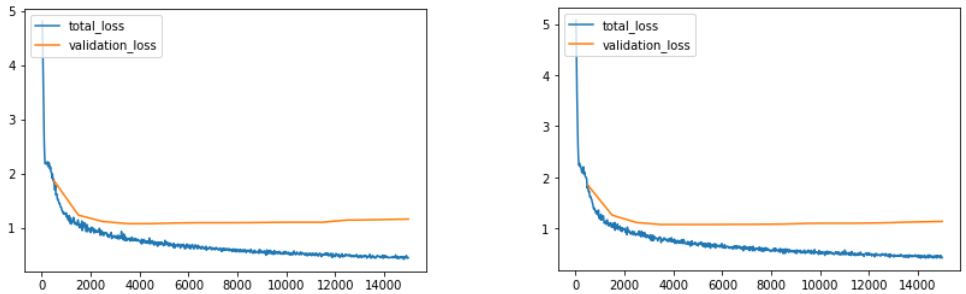
Abschnitt	**Distanz**
1. Abschnitt	11.54 m
2. Abschnitt	7.79 m
3. Abschnitt	5.59 m

5 Evaluation

In diesem Kapitel wird das vorgestellte neuronale Netz und die Datenfusion von Kamera- und Lidardaten evaluiert. Für die Trainingsergebnisse werden die vorgestellten Metriken aus Kapitel 2.3 verwendet, um Aussagen über den Verlauf des Trainings zu treffen. Außerdem wird auf simulierten UAV Daten geprüft, wie gut das neuronale Netz bei diesen Daten eine Instanz-Segmentierung durchführt. Anschließend werden bei der Datenfusion die ermittelten Entfernungen der erkannten Instanzen mit den realen Entfernung verglichen und evaluiert. Zudem wird geprüft, wie lange diese Datenfusion bei der Durchführung des Algorithmus braucht.

5.1 Trainingsergebnisse

Für die Trainingsergebnisse wird zunächst der Trainingsverlauf analysiert und ausgewertet. Dabei wird der Loss von Training und Evaluierung anhand der beiden trainierten Netze zunächst genauer betrachtet und geprüft, wie sehr sich die Ergebnisse voneinander unterscheiden. Da beide Netze mit den gleichen Einstellungen, bis auf die zufälligen Gewichte, trainiert wurden, wird erwartet, dass die Verlaufskurven beider Netze in etwa gleich sind.

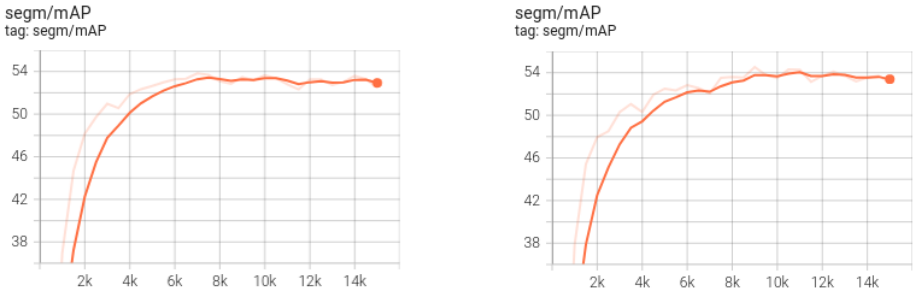


(a) 1. neuronales Netz (b) 2. neuronales Netz

Abbildung 5.1: Trainings- und Validierungsloss Verlauf.

Wie Abbildung 5.1 zeigt, ist der Trainingsverlauf bei den beiden neuronalen Netzen relativ gleich. Bei beiden Abbildungen ist zu sehen, dass der Validierungsloss bereits ab ungefähr 3000 Schritten nicht mehr weiter sinkt und sich so der Unterschied zum Trainingsloss stetig vergrößert. Zudem ist bei Abbildung 5.1 (a) ab Schritt 12000 bereits ein leichter Anstieg des Validierungsloss zu sehen. Diese Anzeichen weisen darauf hin, dass die Netze anfangen, sich auf die Trainingsbilder einzustellen wodurch eine Überanpassungen an den Trainingsbildern entsteht. Dies führt dazu, dass sich die Erkennung von Instanzen bei unbekannten Bildern verschlechtert.

Für weitere Analysen wird die mAP und die AP des Trainigsverlaufs der beiden trainierten Netze verglichen und anschließend die erreichten Ergebnisse der Evaluierung. Der Vergleich soll weiter verdeutlichen, ob die Ergebnisse der neuronalen Netze durch die zufällig gewählten Gewichte stark variieren.



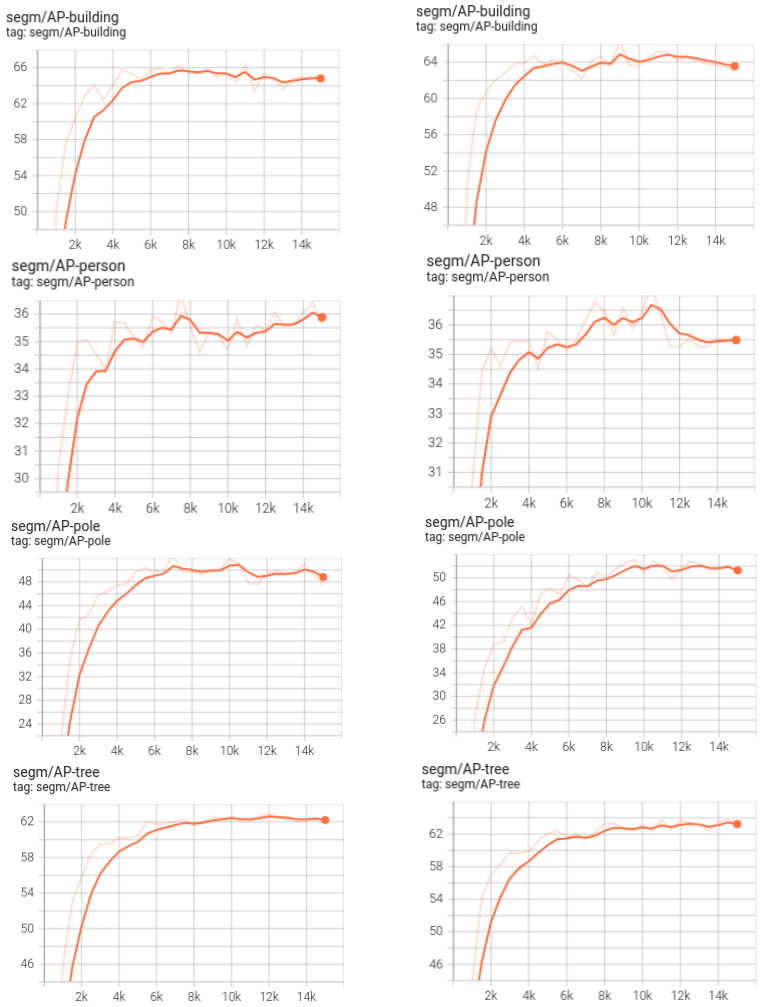
(a) 1. neuronales Netz (b) 2. neuronales Netz

Abbildung 5.2: Vergleich von mAP der neuronalen Netze.

Anhand der Abbildung 5.2 wird deutlich, dass die Kurven fast gleich verläufen. Dennoch ist zu sehen, dass das erste neuronale Netz sein Maximum bei bereits ungefähr 7000 Schritten erreicht und es sich ab da etwas verschlechtert. Bei dem zweiten neuronalen Netz hingegen wird das Maximum erst nach ungefähr 9000 Schritten erreicht und fällt zum Ende des Trainings nicht mehr so stark ab wie bei dem ersten Netz. Ein Abfall dieser Kurve deutet auch hier auf eine Überanpassung der Trainingsbilder hin.

Die Abbildung 5.3 zeigt die AP der einzelnen Klassen. Dabei fällt auf, dass sich die Ergebnisse der beiden Netze minimal voneinander Unterscheiden. Bei der Klasse *building* wird hier zum Beispiel eine höhere AP des ersten neuronalen Netzes erreicht während bei der Klasse *pole* wiederum das zweite neuronale Netz besser ist. Anhand der Klasse

person fällt auf, dass beide Netze leichte Probleme beim Training hatten, da bei beiden Netzen die Kurve anders als bei den anderen Klassen verläuft und die AP etwas stärker abfällt während des Trainings. Dies zeigt, dass bei dieser Klasse mehr Trainingsdaten erforderlich sind, um die Überanpassung zu vermeiden.



(a) 1. neuronales Netz (b) 2. neuronales Netz

Abbildung 5.3: AP bei Segmentierung der einzelnen Klassen.

In der folgenden Tabelle 5.1 sind die verschiedenen AP der Segmentierungsnetze zu sehen, welche mithilfe der Validierungsdaten ermittelt wurden. Dabei wurde die Segmentierung so eingestellt, dass nur Instanzen mit einer Erfolgsrate von 85% oder mehr erkannt wurden. Hier ist zu erkennen, dass die Werte beider Netze nahe beieinander liegen. Die mAP des ersten neuronalen Netz ist nur um 0.5% besser als die des zweiten Netzes, wird aber

dadurch als das bessere gewertet. Die mAP50 und mAP75 gibt den prozentualen Anteil der Pixel an, die eine Überschneidung von mindestens 50% beziehungsweise 75% haben. Weiterhin schafft es das erste Netz,insgesamt bessere Ergebnisse zu erzielen als das zweite, da es bei einer Überschneidung von mindestens 50% um 2% besser ist und bei der Überschneidung von mindestens 75% nur um 0.8% schlechter.
Die mAP-large, mAP-medium und mAP-small Metrik zeigt hier den prozentualen Anteil von Instanzen, die erkannt wurden, mit unterschiedlichen Größen an. Dabei geht es bei mAP-large um Instanzen, die eine Größe von über 96x96 Pixeln haben, bei mAP-medium eine Größe von 32x32 bis 96x96 Pixel und bei mAP-small eine Größe bis zu 32x32 Pixel. Anhand dieser Metriken kann erkannt werden, wie gut die Netze unterschiedliche große Instanzen erkennen können. Es fällt auf, dass bei großen Instanzen beide annähernd gleiche Ergebnisse erreichen und jeweils besser darin sind, größere als kleinere Instanzen zu erkennen. Ein weiterer Grund dafür kann sein, dass die annotierten Instanzen in den Trainingsdaten oft größere Instanzen sind. Zuletzt werden die verschiedenen AP der Klassen betrachtet, wobei auffällt, dass die AP von Gebäuden und Personen wie bei den Trainingsdaten vom ersten Netz besser erkannt werden und das zweite Netz die anderen beiden Klassen besser erkennt.

Tabelle 5.1: Ermittelte AP beider Netze durch Validierungsdaten

Name	mAP	mAP50	mAP75	mAP-large	mAP-medium	mAP-small	AP-building	AP-person	AP-pole	AP-tree
1. seg. Netz	**41.436**	**72.771**	39.573	58.228	**37.588**	31.168	**54.018**	**34.167**	31.485	46.074
2. seg. Netz	40.962	70.614	**40.307**	**58.236**	36.124	**31.607**	51.421	32.833	**31.930**	**47.663**

5.1.1 Instanz-Segmentierung auf simulierten UAV Daten

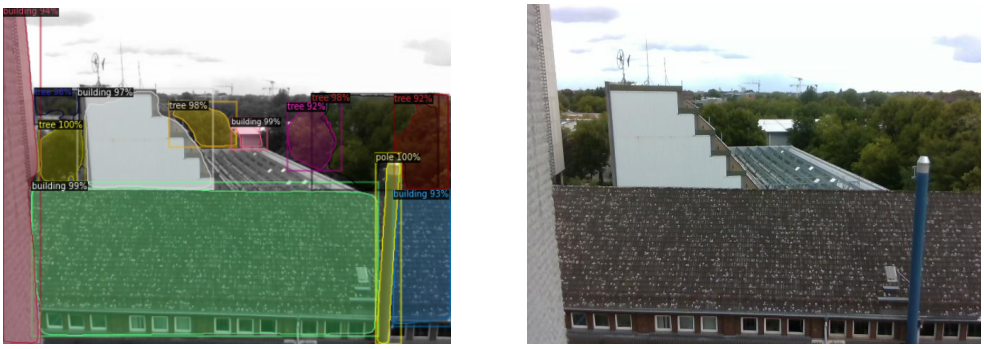
Mit der Segmentierung auf UAV Daten soll geprüft werden, wie gut der Transfer vom neuronalen Netz ist, wenn es auf den mobilen Daten trainiert wurde und dann auf den simulierten UAV Daten angewendet wird. Dafür wurde eine weitere Evaluierung mit dem ersten neuronalen Netz auf UAV Daten durchgeführt.

Tabelle 5.2: AP bei simulierten UAV Daten

Name	mAP	mAP50	mAP75	mAP-large	mAP-medium	mAP-small	AP-building	AP-person	AP-pole	AP-tree
1. seg. Netz	25.960	52.802	19.666	32.278	24.406	7.618	42.953	30.459	17.438	12.987

Anhand der Tabelle 5.2 wird deutlich, dass die AP in allen Bereichen deutlich geringer ist als vorher. Der kleinste Unterschied ist hier bei der AP-person. Da in den simulierten UAV Daten nur wenige Personen vorkommen und diese zudem richtig vom Netz segmentiert

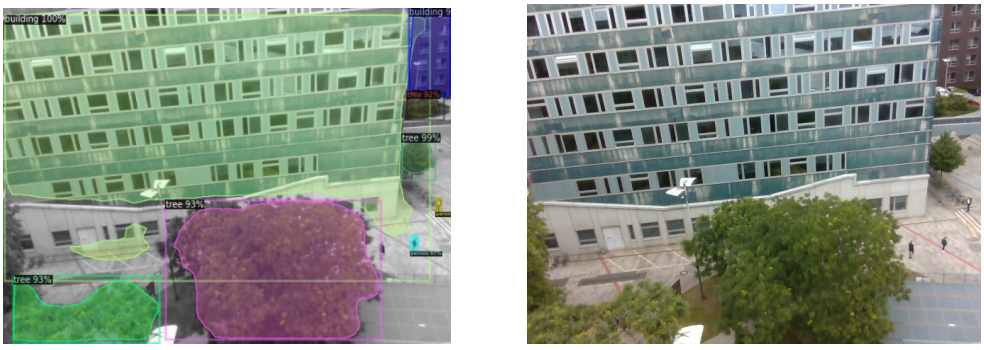
wurden. Der niedrige Werte von AP-tree entsteht dadurch, dass das Netz bei Bäumen die Baumkronen nicht richtig voneinander differenzieren konnte und diese als einen Baum erkannte, wie Abbildung 5.4 (a) verdeutlicht. Selbst für das menschliche Auge ist dies schwer voneinander zu differenzieren.



(a) Instanz-Segmentierung (b) Referenzbild zu a

Abbildung 5.4: Segmentierung von UAV Daten mit vielen Bäumen.

Bei den AP-pole entsteht der niedrige Wert vermutlich dadurch, dass das neuronale Netz nie die Straßenbeleuchtungen von oben in den Trainigsbildern gesehen hat und deshalb eher besser darin ist, die Schornsteine zu erkennen. Dies wird in Abbildung 5.5 (a) noch einmal aufgezeigt. Hier sieht man, dass alle drei möglichen Straßenbeleuchtungen nicht erkannt wurden. Außerdem ist zusehen, dass ein Gebäude nicht vollständig segmentiert und eine Person nicht erkannt wurde.



(a) Instanz-Segmentierung (b) Referenzbild zu a

Abbildung 5.5: Segmentierung von UAV Daten mit Personen.

Das nicht Erkennen von Instanzen beziehungsweise das nicht komplett Segmentieren von Instanzen wird dazu führen, dass bei der Datenfusion die Datenpunkte von Kamera und Lidar sich weniger überschneiden und dadurch die Genauigkeit verringert wird.

5.2 Fusionsergebnisse

Für die Evaluierung der Fusionsergebnisse wird hier verglichen, wie groß der Entfernungsunterschied zwischen echten Daten und den erfassten Instanzen ist. Dabei wurde für die Instanzsegmentierung das erste neuronale Netz aus den vorherigen Ergebnissen ausgewählt, da es eine insgesamt höhere mAP erzielt hat. Des Weiteren wird ermittelt, wie viel Zeit die Algorithmen für die Ausführung der Fusion benötigen, um anschließend bewerten zu können, ob dieses Verfahren auf UAV's durchführbar ist.
Für die Erfassung von Instanzen werden auch hier die simulierten UAV Daten genutzt, wie in Abbildung 5.6 zu sehen ist.

Abbildung 5.6: Datenfusion auf UAV Daten.

Der Entfernungsunterschied zwischen den realen und erfassten Instanzen wird anhand der minimalen, maximalen und der durchschnittlichen Distanz verglichen. Dabei wurden die realen Instanzen durch selektive Auswahl der Messpunkte in den Punktwolken des Lidar Sensors erstellt und die jeweiligen Distanzen ermittelt. Die folgende Tabelle 5.3 zeigt dabei auf, welche Distanzen für die jeweiligen Instanzen gemessen wurden und wie groß der absolute Distanzunterschied zwischen den realen und den ermittelten Instanzen ist.

Bei einigen Distanzen fällt dabei auf, dass hier ein relativ großer Unterschied entstanden ist. Grund dafür ist zum Teil, dass die Segmentierung der Instanzen nicht auf den Pixel genau ist und so zum Beispiel die Segmentierungsmaske des ersten Gebäudes sich leicht mit der des zweiten Gebäudes überschneidet. Dadurch wurden die Messpunkte des Lidar

Tabelle 5.3: Ermittelte Abstände zu Abbildung 4.5

Instanz	**Durchschnitt**			**Minimum**			**Maximum**		
	real	ermittelt	abs. Differenz	real	ermittelt	abs. Differenz	real	ermittelt	abs. Differenz
1. Gebäude	41,70 m	40,30 m	**1,40 m**	23,4 m	36,73 m	**13,33 m**	46,68 m	91,99 m	**45,31 m**
2. Gebäude	92,01 m	84,74 m	**5,27 m**	90,61 m	40,37 m	**50,24 m**	92,54 m	92,54 m	**0 m**
1. Baum	20,43 m	33,51 m	**13,08 m**	19,96 m	19,96 m	**0 m**	21,35 m	40,58 m	**19,23 m**
2. Baum	26,36 m	29,32 m	**2,96 m**	20,12 m	22,98 m	**2,86 m**	36,76 m	47,15 m	**10,39 m**
3. Baum	50,09 m	50,90 m	**0,81 m**	49,26 m	43,65 m	**5,61 m**	51,44 m	57,62 m	**6,18 m**
1. Person	39,76 m	44,56 m	**4,80 m**	39,64 m	44,56 m	**4,92 m**	39,88 m	44,56 m	**4,68 m**
2. Person	40,71 m	43,66 m	**2,95 m**	40,56 m	43,66 m	**3,10 m**	40,88 m	43,66 m	**2,78 m**
3. Person	46,20 m	48,74 m	**2,54 m**	46,14 m	48,74 m	**2,60 m**	46,28 m	48,74 m	**2,46 m**

vom zweiten Gebäude mit in die Liste der maximalen Distanzen des ersten Gebäudes aufgenommen, womit eine größere absolute Differenz entsteht. Des Weiteren führt eine nicht vollständige Segmentierung einer Instanz dazu, dass einige Messpunkte des Lidar sich nicht mit der Segmentierungsmaske überschneiden und so die Messpunkte bei der Datenfusion nicht mit in die Distanzlisten aufgenommen werden. Auf der Abbildung 5.7 sind einige dieser Probleme zu sehen. Hier fällt besonders auf, dass das Segmentierungsnetz die Segmentierungsmaske des ersten Gebäudes auf der rechten unteren Seite zu groß wurde und auf der linken Unteren Seite zu klein, wodurch diese großen Differenzen entstehen. Ein weiteres Problem kann bei Bäumen entstehen. Da der Lidar zum Teil durch kleine Öffnungen in den Baumkronen hindurch misst, werden Distanzen der dahinter liegenden Instanz erfasst, welches zu weiteren Messabweichungen führt.

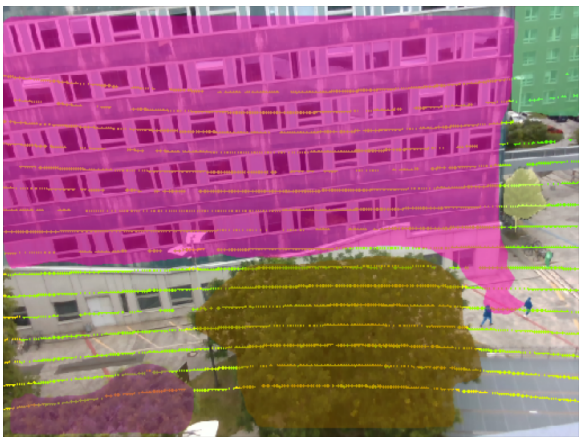


Abbildung 5.7: Segmentierungsmasken mit Lidarmesspunkte.

5.2.1 Durchführungszeit

Um die Durchführungszeit der genutzten Verfahren evaluieren zu können, werden die Verfahren auf den simulierten UAV Daten durchgeführt. Bei diesen Daten sind 2834 Bilder mit den dazugehörigen Punktwolken entstanden. Während der Ausführung der ersten 100 Iterationen wird keine Zeit gemessen, damit sich das System erst einmal aufwärmen kann. Danach wird für jede Iteration die Zeit für die jeweiligen Verfahren gemessen und schließlich die durchschnittliche benötigte Zeit jedes Verfahrens berechnet (siehe Tabelle 5.4). Anhand dieser Tabelle sieht man nun deutlich, bei welchen Verfahren es zu längeren Verzögerungen kommen kann. Dabei ist zu beachten, dass selbst ohne die Implementation das Verfahren zu lange braucht, um dies auf einem kleinem UAV durchzuführen, da diese oft durch die Leistung der Hardware limitiert sind. Durch diese lange Ausführungszeit kann nicht gewährleistet werden, dass ein UAV sicher in der urbanen Umgebung manövrieren kann.

Tabelle 5.4: Benötigte, durchschnittliche Zeit der Verfahren bei 2734 Durchführungen

Verfahren	**Durchschnittliche Zeit**
Ohne Implementation	
Instanz-Segmentierung	120,48 ms
Early Fusion	1,02 ms
Combined Fusion	441,21 ms
Gesamt	**562,71 ms**
Mit Implementation	
Implementation	398,91 ms
Gesamt	**840,12 ms**

6 Diskussion

In diesem Kapitel werden die Ergebnisse der vorgestellten Verfahren diskutiert und eingeordnet. Dabei haben die Ergebnisse der Datenfusion gezeigt, dass es möglich ist, mit diesen Verfahren eine Umgebungserkennung im urbanen Umfeld zu erstellen. Um das Verfahren zu verbessern, muss dennoch geklärt werden, welche Änderungen die Genauigkeit und die Ausführungszeiten verbessern können.

6.1 Probleme bei der Early Fusion

Die Tests der EF in Kapitel 4.2.1 haben gezeigt, dass bei der Zuordnung der Lidarmesspunkte auf das Bild zu leichten Verschiebungen kommen kann. Um dieses Problem zu beheben, wurden mehrere Kalibrierungen von Kamera und Lidar durchgeführt, welche jedoch zu keiner Verbesserung führten. Bei Weiteren Nachforschungen hat sich herausgestellt, dass der Grund dafür die zu hohen Fehlerraten sind, welche bei der Kalibrierung zwischen Kamera und Lidar entstanden sind. Dabei sind unter 30 mm noch akzeptable Fehlerraten, die bei der Kalibrierung[23] vorkommen können. Dennoch war es nicht möglich, mit der verwendeten Hardware diese Fehlerraten zu minimieren.

6.2 Genauigkeit der Instanz-Segmentierung

Wie die Ergebnisse der Evaluierung zeigen, ist für die Datenfusion eine hohe Genauigkeit der Instanz-Segmentierung und der daraus entstehenden Segmentierungsmasken erforderlich. Des Weiteren zeigen die Ergebnisse, dass die mobilen Daten nicht ausreichen, um eine hohe Genauigkeit bei den simulierten UAV Daten zu erreichen. Für eine Verbesserung der Genauigkeit ist es somit notwendig, mehr Daten und vor allem Daten auf einem UAV zu sammeln, die dann für das Training des neuronalen Netzes verwendet

werden. Zudem ist es erforderlich zusätzliche Klassifikationen einzuführen, damit weitere Hindernisse erkannt werden können, wie zum Beispiel andere UAV's oder Vögel.

6.3 Auswertung der Fusionsergebnisse

Anhand der Evaluierung der Fusionsergebnisse ist klar zu erkennen, dass die Genauigkeit verbessert werden kann. Dabei gibt es mehrere Möglichkeiten, die zu einer Verbesserung beitragen können.
Wie bereits erläutert trägt eine genauere Segmentierungsmaske etwas dazu beitragen. Allerdings ist es auch möglich diese Segmentierungsmaske kleiner zu skalieren, damit das Problem der sich überschneidenden Ränder minimiert wird. Dabei ist jedoch nachteilig, dass sich bei bereits kleinen Segmentierungsmasken, wie die Personen in Abbildung 5.7, eventuell keine Messpunkte des Lidar mit der Segmentierungsmaske überschneiden. Die Tabelle 5.3 zeigt sogar auf, dass bei der zweiten und der dritten Person die ermittelten Entfernungen der Datenfusion alle dieselbe Entfernung haben. Dies trifft auf, weil sich nur ein Messpunkt bei der Datenfusion mit der Segmentierungsmaske überschneidet. Um dem entgegenzuwirken, ist es möglich einen Lidar mit einer höheren, vertikalen Auflösung zu verwenden. Dies erzielt allgemein genauere Ergebnisse, da es zu mehr Messpunkten führt.
Des Weiteren ist es möglich, Messpunkte aus den Distanzlisten herauszufiltern, wenn diese eine große Abweichung von den gesammelten Distanzen haben. Dies ist allerdings nur realisierbar, falls genug Messpunkte für die jeweilige Instanz vorhanden sind.

6.4 Auswertung der Durchführungszeit

Im Hinblick auf die benötigte Durchführungszeit laut Tabelle 5.4 wird deutlich, dass auch hier Verbesserungspotential besteht, damit dieses Verfahren bei UAV's einsetzbar ist. Die benötigte Zeit für die CF lässt sich daher erklären, dass bei der Prüfung der Überschneidung, jeder Pixel der Segmentierungsmasken mit den Messpunkten des Lidar verglichen wird. Dabei variiert die benötigte Zeit je nachdem, wie viele Instanzen gefunden wurden und wie groß deren Masken sind. Um diese Zeit zu reduzieren, ist es möglich, die Auflösung des Kamerabildes zu reduzieren, damit über weniger Pixel Iteriert werden muss.

Eine weitere Alternative ist es, einen schnellen Objekdetektor wie YOLOv4 [12] für das Kamerabild und dazu einen Objektdetektor, wie zum Beispiel Complex-YOLO [20], für die Lidardaten zu verwenden. Anschließend wird eine LF anhand der ermittelten 2D- und 3D-Objekte durchgeführt. Die Autoren des YOLOv4 Papers geben dabei an, dass deren Verfahren eine FPS von über 40 FPS erreicht, während die Autoren von Complex-YOLO angeben, dass ihr Verfahren auf Lidardaten sogar über 60 FPS erreicht. Die Kombination dieser Verfahren könnte somit zu einer Verbesserung der Durchführungszeit führen. Dieses Verfahren erfordert einen umfangreichen Datensatz an annotierten Bildern und Lidar Punktwolken, welcher über den Rahmen dieser Bachelorarbeit hinausgeht.

7 Fazit

In dieser Arbeit wurde eine Umgebungserkennung im urbanen Raum mittels Datenfusion von Kamera- und Lidardaten entworfen und implementiert.

Bei dem Entwurf wurde vorgestellt, wie sich Hardware und Software kombinieren lassen, um Implementation und Datenfusion durchzuführen. Zudem wurden die notwendigen Vorbereitung für die Kalibrierung der Sensorik erläutert. Des Weiteren wurde für die Vorbereitung des verwendeten Mask R-CNN mobile und aus der Luft simulierte Daten im urbanen Umfeld gesammelt, auf die das neuronale Netz anschließend nur mit den mobilen Daten trainiert wurde. Anschließend wurde eine Implementierung vorgestellt, welche die Datenfusion nutzt, um bei einem Flug eines UAV's ein mögliches Ausweichszenario bei Hindernissen zu ermitteln.

Mit der Durchführung der Verfahren ließen sich unterschiedliche Probleme der einzelnen Verfahren feststellen, welche sich negativ auf die Genauigkeit der Datenfusion auswirkten. Diese Auswirkungen wurden im laufe dieser Arbeit identifiziert, konnten aber nicht behoben werden. Allerdings zeigten diese Probleme auf, dass es notwendig ist, eine optimale Kalibrierung der Hardware zu erzielen, sowie ein robustes neuronales Netz für die Datenfusion zu verwenden, um anschließend bessere Ergebnisse erzielen zu können.

Die Evaluierung zeigte, dass trotz der geringen Datenmenge das Instanz-Segmentierungsnetz eine mAP von 41,44% auf den mobilen Daten erreichte. Doch zeigt sich anhand der simulierten Daten, dass das neuronale Netz auch auf diesen Daten trainiert werden sollte, da es ansonsten nur eine mAP von 25,96% erreichte. Des Weiteren wurde bei der Evaluierung der Datenfusion gezeigt, dass es möglich ist, Objekte in der urbanen Umgebung mit den jeweiligen Distanzen zu erfassen. Diese Entfernungen benötigen zum Teil eine höhere Genauigkeit, damit die vorgestellte Implementierung darauf angewendet werden kann. Die benötigte Ausführungszeit der vorgestellten Verfahren ist zudem mit einer

Gesamtzeit von 840 ms zu hoch, um dies auf einem UAV anzuwenden, da nicht sichergestellt werden kann, dass mögliche Echtzeitanforderungen eingehalten und Kollisionen verhindert werden können.

7.1 Ausblick

Zur Verbesserung der Datenfusion können noch andere Ansätze verfolgt werden, welche zum Beispiel am Ende von Kapitel 6.4 vorgestellt wurden. Diese würden dazu beitragen, die Fusion in der Ausführungszeit zu verbessern und so einsetzbar auf einem UAV mit Echtzeitanforderungen zu machen.

Des Weiteren kann die Genauigkeit der Datenfusion durch Optimierung der Kalibrierung sowie durch weitere Algorithmen verbessert werden. Dabei könnten Algorithmen wie eine Filterung angewendet werden, um Abweichungen bei der Datenfusion zu filtern. Im Anschluss muss eine Validierung mit unterschiedlichen Tests bei verschiedenen Szenarien vorgenommen werden.

Interessant wäre es, einen Datensatz zu erstellen, welcher so ähnlich wie der Kitti-Datensatz ist. Statt für autonomes Fahren wird dann ein Datensatz für autonomes Fliegen erstellt. Mithilfe so eines Datensatzes wäre es möglich, weitere Verfahren zu entwickeln und zu testen.

Ausblickend kann man feststellen, dass eine weitere Optimierung der vorgestellten Verfahren und eine Umsetzung auf einem UAV eine Lösung anbietet, damit eine Umgebungswahrnehmung im urbanen Umfeld realisiert werden kann.

Literaturverzeichnis

[1] : *Intel Realsense D435 Datasheet*. Web. – URL https://www.intel.com/content/dam/support/us/en/documents/emerging-technologies/intel-realsense-technology/Intel-RealSense-D400-Series-Datasheet.pdf

[2] : *Intel Realsense D435 ROS-package*. Github. – URL https://github.com/IntelRealSense/realsense-ros

[3] : *Meta Research*. Web. – URL https://opensource.fb.com/projects/

[4] : *OpenCV*. Web. – URL https://docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html

[5] : *Roboflow annotation Tool*. Web. – URL https://roboflow.com/

[6] : *Robosense ROS-package*. Github. – URL https://github.com/RoboSense-LiDAR/rslidar_sdk

[7] : *Robosense RS-Lidar-16 Datasheet*. Web. – URL https://cdn.robosense.cn/20200723161715_42428.pdf

[8] : *Robot Operating System*. Web. – URL https://www.ros.org/

[9] : *rviz*. Web. – URL http://wiki.ros.org/rviz

[10] : *DJI*. Web. 2021. – URL https://www.dji.com/de/matrice-300

[11] : *Innovative Urban Air Mobility (i-LUM)*. Web. Mar. 2021. – URL https://i-lum.de/

[12] Bochkovskiy, Alexey ; Wang, Chien-Yao ; Liao, Hong-Yuan M.: YOLOv4: Optimal Speed and Accuracy of Object Detection. In: *CoRR* abs/2004.10934 (2020). – URL https://arxiv.org/abs/2004.10934

[13] Darshan Bhanushali, Karan Manghi Abhishek Vashist Clark Hochgraf Amlan Ganguly Andres Kwasinski Michael E. Kuhl Raymond P.: LiDAR-Camera Fusion for 3D Object Detection. In: *on Electronic Imaging* (2020). – URL https://www.rit.edu/academicaffairs/facultyscholarship/submit/download_file.php?id=71433

[14] Geiger, Andreas ; Lenz, Philip ; Stiller, Christoph ; Urtasun, Raquel: Vision meets Robotics: The KITTI Dataset. In: *International Journal of Robotics Research (IJRR)* (2013). – URL http://www.cvlibs.net/publications/Geiger2012CVPR.pdf

[15] He, Kaiming ; Gkioxari, Georgia ; Dollár, Piotr ; Girshick, Ross B.: Mask R-CNN. In: *CoRR* abs/1703.06870 (2017). – URL http://arxiv.org/abs/1703.06870

[16] Iffat Zafar, Richard Burton Nimesh Patel Leonardo A.: *Hands-On Convolutional Neural Networks with TensorFlow*. Packt Publishing, 2018. – ISBN 9781789130331

[17] Lin, Tsung-Yi ; Dollár, Piotr ; Girshick, Ross ; He, Kaiming ; Hariharan, Bharath ; Belongie, Serge: Feature Pyramid Networks for Object Detection. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, S. 936–944

[18] Ren, Shaoqing ; He, Kaiming ; Girshick, Ross B. ; Sun, Jian: Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In: *CoRR* abs/1506.01497 (2015). – URL http://arxiv.org/abs/1506.01497

[19] Riege, Daniel: *Segmentierung von Straßenmarkierungen durch maschinelles Lernen für die Miniaturautonomie*. 2022. – URL https://autosys.informatik.haw-hamburg.de/papers/2021DanielRiegeBA.pdf

[20] Simon, Martin ; Milz, Stefan ; Amende, Karl ; Gross, Horst-Michael: *Complex-YOLO: Real-time 3D Object Detection on Point Clouds*. 2018. – URL https://arxiv.org/abs/1803.06199

[21] Sun, Pei ; Kretzschmar, Henrik ; Dotiwalla, Xerxes ; Chouard, Aurelien ; Patnaik, Vijaysai ; Tsui, Paul ; Guo, James ; Zhou, Yin ; Chai, Yuning ; Caine, Benjamin ; Vasudevan, Vijay ; Han, Wei ; Ngiam, Jiquan ; Zhao, Hang ; Timofeev, Aleksei ; Ettinger, Scott ; Krivokon, Maxim ; Gao, Amy ; Joshi, Aditya ; Zhang, Yu ; Shlens, Jonathon ; Chen, Zhifeng ; Anguelov, Dragomir:

Scalability in Perception for Autonomous Driving: Waymo Open Dataset. In: *CoRR* abs/1912.04838 (2019). – URL http://arxiv.org/abs/1912.04838

[22] Tiedemann Tim, Pareigis Stephan Becke Martin Meisel A.: autosys. . – URL https://autosys.informatik.haw-hamburg.de/

[23] Tsai, Darren ; Worrall, Stewart ; Shan, Mao ; Lohr, Anton ; Nebot, Eduardo: Optimising the selection of samples for robust lidar camera calibration. In: *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, URL https://gitlab.acfr.usyd.edu.au/its/cam_lidar_calibration, 2021, S. 2631–2638

[24] Wu, Yuxin ; Kirillov, Alexander ; Massa, Francisco ; Lo, Wan-Yen ; Girshick, Ross: *Detectron2*. https://github.com/facebookresearch/detectron2. 2019

[25] Yeong, De J. ; Velasco-Hernandez, Gustavo ; Barry, John ; Walsh, Joseph: Sensor and Sensor Fusion Technology in Autonomous Vehicles: A Review. In: *Sensors* 21 (2021), Nr. 6. – URL https://www.mdpi.com/1424-8220/21/6/2140. – ISSN 1424-8220

[26] Zhou, Yin ; Tuzel, Oncel: VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection. In: *CoRR* abs/1711.06396 (2017). – URL http://arxiv.org/abs/1711.06396

Erklärung zur selbstständigen Bearbeitung einer Abschlussarbeit

Gemäß der Allgemeinen Prüfungs- und Studienordnung ist zusammen mit der Abschlussarbeit eine schriftliche Erklärung abzugeben, in der der Studierende bestätigt, dass die Abschlussarbeit „— bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit [(§ 18 Abs. 1 APSO-TI-BM bzw. § 21 Abs. 1 APSO-INGI)] — ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt wurden. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich zu machen."

Quelle: § 16 Abs. 5 APSO-TI-BM bzw. § 15 Abs. 6 APSO-INGI

Erklärung zur selbstständigen Bearbeitung der Arbeit

Hiermit versichere ich,

Name: _____

Vorname: _____

dass ich die vorliegende Bachelorarbeit – bzw. bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit – mit dem Thema:

Urbane Umgebungserkennung für Luftfahrzeuge mittels Datenfusion von Kamera- und Lidardaten

ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

_____	_____	_____
Ort	Datum	Unterschrift im Original