Universität Bremen

Entwicklung eines neuronalen Netzes zur Pylonenerkennung und Fahrbahnberechnung für autonome Fahrzeugsysteme in der Formula Student.

Development of a neural network for cone recognition and lane calculation for autonomous vehicle systems in the Formula Student league.

Bachelorarbeit

im Rahmen des Studiengangs
Informatik
der Universität Bremen

vorgelegt von
Yannis Fynn Meyer
Matrikelnummer: 6114817

ausgegeben und betreut von
Prof. Dr.-Ing. Udo Frese
Prof. Dr.-Ing. Maren Petersen

Bremen, den 12. Mai 2025

Universität Bremen

Nachname	**Meyer**	Matrikelnr.	**6114817**
Vorname	**Yannis Fynn**		

Hinweise zu den offiziellen Erklärungen

1. Die folgende Seite mit den offiziellen Erklärungen

   A) Eigenständigkeitserklärung

   B) Erklärung zur Veröffentlichung von Bachelor- oder Masterarbeiten

   C) Einverständniserklärung über die Bereitstellung und Nutzung der Bachelorarbeit / Masterarbeit in elektronischer Form zur Überprüfung durch eine Plagiatssoftware

   ist entweder direkt in jedes Exemplar der Bachelor- oder Masterarbeit fest mit einzubinden oder unverändert im Wortlaut in jedes Exemplar der Bachelor- oder Masterarbeit zu übernehmen.

   Bitte achten Sie darauf, jede Erklärung in allen drei Exemplaren der Arbeit zu unterschreiben.

2. In der digitalen Fassung kann auf die Unterschrift verzichtet werden. Die Angaben und Entscheidungen müssen jedoch enthalten sein.

Zu B)

Die Einwilligung kann jederzeit durch Erklärung gegenüber der Universität Bremen, mit Wirkung für die Zukunft, widerrufen werden.

Zu C)

Das Einverständnis der dauerhaften Speicherung des Textes ist freiwillig.

Die Einwilligung kann jederzeit durch Erklärung gegenüber der Universität Bremen, mit Wirkung für die Zukunft, widerrufen werden.

Weitere Informationen zur Überprüfung von schriftlichen Arbeiten durch die Plagiatsoftware sind im Nutzungs- und Datenschutzkonzept enthalten. Diese finden Sie auf der Internetseite der Universität Bremen.

Universität Bremen

Nachname	**Meyer**	Matrikelnr.	**6114817**
Vorname	**Yannis Fynn**		

A) Eigenständigkeitserklärung

Ich versichere, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe. Alle Teile meiner Arbeit, die wortwörtlich oder dem Sinn nach anderen Werken entnommen sind, wurden unter Angabe der Quelle kenntlich gemacht. Gleiches gilt auch für Zeichnungen, Skizzen, bildliche Darstellungen sowie für Quellen aus dem Internet, dazu zählen auch KI-basierte Anwendungen oder Werkzeuge. Die Arbeit wurde in gleicher oder ähnlicher Form noch nicht als Prüfungsleistung eingereicht. Die elektronische Fassung der Arbeit stimmt mit der gedruckten Version überein. Mir ist bewusst, dass wahrheitswidrige Angaben als Täuschung behandelt werden.

☐ Ich habe KI-basierte Anwendungen und/oder Werkzeuge genutzt und diese im Anhang "Nutzung KI basierte Anwendungen" dokumentiert.

B) Erklärung zur Veröffentlichung von Bachelor- oder Masterarbeiten

Die Abschlussarbeit wird zwei Jahre nach Studienabschluss dem Archiv der Universität Bremen zur dauerhaften Archivierung angeboten. Archiviert werden:

1) Masterarbeiten mit lokalem oder regionalem Bezug sowie pro Studienfach und Studienjahr 10 % aller Abschlussarbeiten
2) Bachelorarbeiten des jeweils ersten und letzten Bachelorabschlusses pro Studienfach und Jahr.

☒ Ich bin damit einverstanden, dass meine Abschlussarbeit im Universitätsarchiv für wissenschaftliche Zwecke von Dritten eingesehen werden darf.

☐ Ich bin damit einverstanden, dass meine Abschlussarbeit nach 30 Jahren (gem. §7 Abs. 2 BremArchivG) im Universitätsarchiv für wissenschaftliche Zwecke von Dritten eingesehen werden darf.

☐ Ich bin nicht damit einverstanden, dass meine Abschlussarbeit im Universitätsarchiv für wissenschaftliche Zwecke von Dritten eingesehen werden darf.

C) Einverständniserklärung zur Überprüfung der elektronischen Fassung der Bachelorarbeit / Masterarbeit durch Plagiatssoftware

Eingereichte Arbeiten können nach § 18 des Allgemeinen Teil der Bachelor- bzw. der Master-prüfungsordnungen der Universität Bremen mit qualifizierter Software auf Plagiatsvorwürfe untersucht werden.
Zum Zweck der Überprüfung auf Plagiate erfolgt das Hochladen auf den Server der von der Universität Bremen aktuell genutzten Plagiatssoftware.

☑ Ich bin damit einverstanden, dass die von mir vorgelegte und verfasste Arbeit zum oben genannten Zweck dauerhaft auf dem externen Server der aktuell von der Universität Bremen genutzten Plagiatssoftware, in einer institutionseigenen Bibliothek (Zugriff nur durch die Universität Bremen), gespeichert wird.

☐ Ich bin nicht damit einverstanden, dass die von mir vorgelegte und verfasste Arbeit zum o.g. Zweck dauerhaft auf dem externen Server der aktuell von der Universität Bremen genutzten Plagiatssoftware, in einer institutionseigenen Bibliothek (Zugriff nur durch die Universität Bremen), gespeichert wird.

Mit meiner Unterschrift versichere ich, dass ich die obenstehenden Erklärungen gelesen und verstanden habe und bestätige die Richtigkeit der gemachten Angaben.

12.05.2025 Unterschrift

Kurzfassung Die Formula Student ist ein internationaler Konstruktionswettbewerb, bei dem studentische Teams Rennwagen im Formel-Stil selbstständig designen und konstruieren. Für die Teilnahme an den fahrerlosen Disziplinen in der Saison 2025 benötigt das Team Bremergy der Universität Bremen ein Pylonenerkennungssystem, das als zentrale Komponente der autonomen Fahrsoftware fungiert.
In dieser Arbeit wird ein neuronales Netz zur Detektion von Pylonen implementiert und durch eine Homographie-basierte Abstandsmessung ergänzt, um deren Position relativ zum Fahrzeug zu bestimmen. Untersucht wird die Forschungsfrage *„Ermöglicht ein minimalistisches Perzeptionssystem mit nur einer Kamera eine zuverlässige Pylonenerkennung und eine hinreichend präzise Fahrbahnberechnung, um autonomes Fahren in der Formula Student zu realisieren?“.*
Die Ergebnisse zeigen, dass das entwickelte System hohe Erkennungsgenauigkeit erreicht und damit die Anforderungen erfüllt. Die Forschungsfrage kann somit bestätigt werden. Das System bildet eine essentielle Grundlage für die erfolgreiche Wettbewerbsteilnahme.

Abstract Formula Student is an international design competition in which student teams independently design and construct formula-style racing cars. To participate in the driverless disciplines in the 2025 season, the Bremergy team from the University of Bremen needs a cone detection system that acts as a central component of the autonomous driving software.
In this thesis, a neural network for the detection of cones is implemented and supplemented by a homography-based distance measurement to determine their position relative to the vehicle. The research question being investigated is *„Does a minimalist perception system with just one camera enable reliable cone detection and sufficiently precise lane calculation to realise autonomous driving in Formula Student league?“.*
The results show that the developed system achieves high detection accuracy and thus meets the requirements. The research question can therefore be confirmed. The system forms an essential basis for successful participation in the competition.

Inhaltsverzeichnis

1 Einleitung

Künstliche Intelligenz (KI) hat in den letzten Jahren enorme Fortschritte gemacht und ist heute in vielen Bereichen nicht mehr wegzudenken. Ob in der Medizin, der Industrie oder der Mobilität, KI-Systeme können komplexe Aufgaben lösen, die früher ausschließlich Menschen vorbehalten waren. Insbesondere im Bereich des autonomen Fahrens spielt KI eine entscheidende Rolle, da sie Fahrzeuge in die Lage versetzt, ihre Umgebung zu erfassen, zu analysieren und entsprechend zu reagieren [1], [2].

Auch im Motorsport gewinnt diese Technologie zunehmend an Bedeutung. Ein Beispiel dafür ist die Formula Student, ein internationaler Konstruktionswettbewerb, bei dem studentische Teams Rennwagen im Formel-Stil selbstständig designen und konstruieren. Neben den Kategorien für Verbrennungs- und Elektrofahrzeuge gibt es den sogenannten Driverless Cup, bei dem vollautonome Fahrzeuge gegeneinander antreten [3].

Bremergy ist ein studentischer Verein der Universität Bremen, der seit 2012 regelmäßig an der Formula Student teilnimmt. Für die Saison 2025 hat sich das Team ein ambitioniertes Ziel gesetzt: Zum ersten Mal wollen sie mit einem autonomen Fahrzeug am Driverless Cup teilnehmen. Um dieses Vorhaben zu realisieren, benötigt das Team eine leistungsfähige Software, die verschiedene Aspekte der autonomen Fahrzeugsteuerung abdeckt.

Diese Arbeit entstand mit der Motivation, Bremergy bei diesem Vorhaben zu unterstützen. Im Mittelpunkt steht dabei die Entwicklung und Implementierung eines Pylonerkennungssystems, das für die autonome Navigation und Fahrbahnberechnung essenziell ist. Dieses System wird in die bestehende Softwarearchitektur von Bremergy integriert und bildet einen wichtigen Baustein für die erfolgreiche Teilnahme am Driverless Cup (siehe Abbildung 1.1). Als Anforderungen an das System wurde eine Laufzeit von 20Hz gesetzt.

Da der Wettbewerb hohe Kosten verursacht und dem Team nur begrenzte Ressourcen zur Verfügung stehen, liegt der Fokus dieser Arbeit auf einem minimalistischen Monokamera-System. Die zentrale Forschungsfrage lautet: *„Ermöglicht ein minimalistisches Perzeptionssystem mit nur einer Kamera eine zuverlässige Pylonenerkennung und eine hinreichend präzise Fahrbahnberechnung, um autonomes Fahren in*

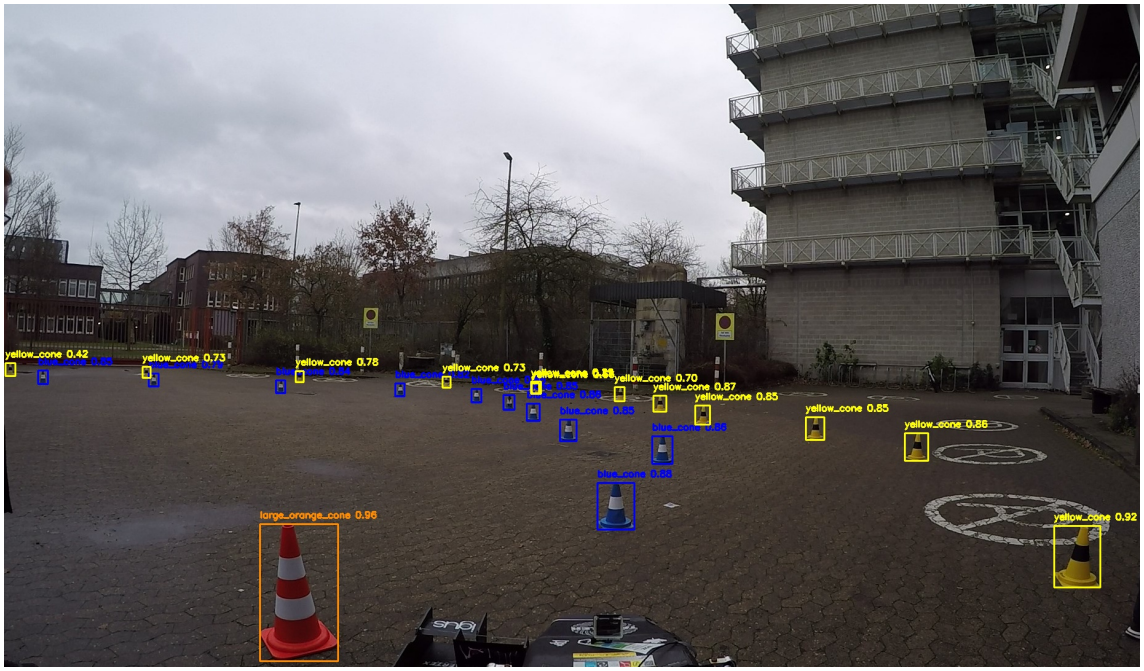

Abbildung 1.1: Ergebnis der Pylonenerkennung: Das entwickelte neuronale Netz detektiert die Pylonen der Rennstrecke.

der Formula Student zu realisieren?“. Diese wird durch methodische Evaluationen systematisch beantwortet.

Neben der praktischen Anwendung liegt die Relevanz dieser Arbeit in der Verbindung von Rennsport und Künstlicher Intelligenz, eine Schnittstelle mit wachsender Bedeutung. KI-basierte Lösungen sind nicht nur technisch faszinierend, sondern besitzen hohe gesellschaftliche Bedeutung. Die Formula Student bietet hier ein ideales, reales Testfeld, um autonomes Fahren unter Wettbewerbsbedingungen zu erforschen. Diese Arbeit trägt damit auch zur Weiterentwicklung kosteneffizienter und robuster KI-Systeme bei.

1.1 Aufbau der Arbeit

Neben der Einleitung und der Zusammenfassung am Ende gliedert sich diese Arbeit in die folgenden vier Kapitel.

Kapitel 2 vermittelt die für diese Arbeit relevanten Grundlagen. Es bietet einen Überblick über den aktuellen Stand der Technik und führt die wesentlichen Konzepte ein, die als Basis für die folgenden Kapitel dienen.

Kapitel 3 behandelt die Entwicklung und Bewertung eines neuronalen Netzes zur Pylonenerkennung.

Kapitel 4 beschreibt die Implementierung und Evaluation einer Kamerakalibrierung zur Abstandsberechnung von Pylonen relativ zum Fahrzeug.

Kapitel 5 erläutert die Integration des Gesamtsystems in die Bremergy-Architektur.

1.2 Beitrag der Arbeit

Die zentralen Beiträge dieser Arbeit lassen sich wie folgt zusammenfassen:

- Entwicklung eines neuronalen Netzes zur Pylonenerkennung auf Basis von YOLOv11 mit spezifischem Training für den Formula Student Anwendungsfall.
- Berechnung einer Kamerakalibrierung mittels Homographie, um präzise Abstände der erkannten Pylonen zum Fahrzeug zu bestimmen.
- Umfassende Evaluation der Erkennungsrate und Präzision zur Validierung des entwickelten Systems.
- Entwicklung eines Python-Skripts zur automatisierten Performance-Analyse neuronaler Netze durch Vergleich von Ground-Truth-Daten mit Modellvorhersagen.
- Analyse über die beste Position der Kamera am Fahrzeug für ein Monokamera-System.
- Integration des Gesamtsystems in die Bremergy-Architektur zur Nutzung im autonomen Fahrzeug.
- Beantwortung der Forschungsfrage *„Ermöglicht ein minimalistisches Perzeptionssystem mit nur einer Kamera eine zuverlässige Pylonenerkennung und eine hinreichend präzise Fahrbahnberechnung, um autonomes Fahren in der Formula Student zu realisieren?“*.

2 Grundlagen

Dieses Kapitel beschreibt alle für die Arbeit notwendigen Grundlagen.

2.1 Stand der Technik

Autonome Fahrzeuge haben in den letzten Jahren sowohl in der industriellen Entwicklung als auch in der akademischen Forschung erhebliche Fortschritte gemacht. Aktuelle Surveys [2], [4] zeigen jedoch, dass trotz dieser Fortschritte grundlegende Herausforderungen bestehen bleiben: Die Wahl der optimalen Sensorik ist weiterhin Gegenstand der Forschung, Algorithmen benötigen höhere Effizienz für Echtzeitanwendungen, und die Robustheit unter realen Bedingungen, wie wechselnden Wetterverhältnissen oder unstrukturierten Umgebungen, bleibt ungelöst [4]. Insbesondere der hohe Ressourcenbedarf etablierter Systeme, darunter teure LiDAR-Sensoren und rechenintensive HD-Maps, limitiert deren Einsatz in kostensensitiven Anwendungen [2].

Formula Student Driverless hat sich als relevante Testplattform für ressourceneffiziente autonome Systeme etabliert, da sie Echtzeitfähigkeit und Robustheit unter strikten Leistungsbedingungen erfordert. Führende Teams wie der Academic Motorsports Club Zurich (AMZ), Gewinner des Driverless Cups 2017 und 2018, setzen dabei auf rechenintensive Multisensor-Ansätze, typischerweise bestehend aus LiDAR und multiplen Kameras [5]. Dieser Ansatz wurde von zahlreichen weiteren Teams adaptiert [3], [6], was ihn zum Standard in der Formula Student macht.

Diese Arbeit untersucht, inwieweit sich die Perzeption in der Formula Student mit minimaler Komplexität über ein Monokamera-System lösen lässt. Während es bereits Lösungen solcher Systeme gibt [7], liegt der Fokus dieser Arbeit auf der Reduktion der Systemkomplexität und der Untersuchung der Frage, ob ein solches System präzise genug arbeiten kann, um autonomes Fahren zu ermöglichen. Die technischen Details dieses minimalistischen Ansatzes werden in den jeweiligen Grundlagenabschnitten der Kapitel erläutert.

2.2 Der Formula Student Driverless Cup

Der Driverless Cup ist eine seit 2017 bestehende Erweiterung der Formula Student, bei der der Fokus auf autonomem Fahren liegt. In diesem Wettbewerb treten Teams mit elektrisch angetriebenen Fahrzeugen in verschiedenen Disziplinen gegeneinander an [8].

Die Disziplinen sind Acceleration, Skidpad, Autocross und Trackdrive:

- Bei Acceleration geht es um das möglichst schnelle Beschleunigen auf einer 75 Meter langen Geraden.
- Skidpad fordert die Fahrzeuge heraus, eine Strecke in Form einer Acht präzise und wiederholt zu durchfahren.
- Autocross und Trackdrive stellen die größte Herausforderung dar: Hier muss ein zuvor unbekannter, geschlossener Kurs absolviert werden. Die Wertung erfolgt auf Basis der Gesamtzeit nach einer beziehungsweise zehn gefahrenen Runden. Die Schwierigkeit besteht darin, dass das Streckendesign nicht bekannt ist, es unterliegt lediglich allgemeinen Vorgaben wie:

  > *Miscellaneous: Chicanes, multiple turns, decreasing radius turns, hairpin turns, etc.* [9]

Diese Disziplinen erfordern eine zuverlässige Pylonenerkennung, eine präzise Fahrzeugsteuerung und effiziente Routenplanung, sowohl bei niedrigen als auch bei hohen Geschwindigkeiten [6].

Die Strecken aller autonomen Disziplinen sind durch Pylonen abgegrenzt. Die Fahrbahn ist Minimum 3 Meter breit, wobei die linke Seite mit blauen Pylonen und die rechte Seite mit gelben Pylonen begrenzt ist [9] (siehe Abbildung 2.1). Darüber hinaus gibt es große und kleine orangene Pylonen, welche für die Start/Ziellinie verwendet werden. Insgesamt gibt es damit vier verschiedene Pylonentypen (siehe Abbildung 2.2), welche die Teams zuverlässig erkennen müssen.

2.3 Das Fahrzeug

Das Fahrzeug von Bremergy (Bremo) wird jedes Jahr neu gefertigt. Das letztjährige Fahrzeug, der Bremo24, hat ein Gewicht von 224kg und ist vollkommen elektrisch betrieben (siehe Abbildung 2.3). Dies ermöglicht eine Beschleunigung von 0-100km/h in 2,6 Sekunden sowie eine maximale Geschwindigkeit von 126km/h [11].

Abbildung 2.1: Beispiel einer Formula Student Driverless Strecke, aufgenommen während Formula Student Germany 24 [10]. Zu sehen sind die verschiedenen Arten von Pylonen, welche die Strecke begrenzen.

Abbildung 2.2: Alle Arten von Pylonen einer Formula Student Driverless Strecke. Zugeschnitten aus dem Regelbuch der Saison 2025 [9].

Abbildung 2.3: Der Bremo24, da der Bremo25 noch in Fertigung ist.

Im diesjährigen Boliden, dem Bremo25, sollen Sensoren für das Driverless System verbaut werden. Diese sind eine Intel RealSense Depth Kamera D435, sowie ein Ouster OS1 LiDAR. Damit hat man eine einfache aber robuste Konfiguration, mit der man sowohl Bildverarbeitung wie auch Tiefenwahrnehmung umsetzen kann. Als zentrale Recheneinheit wird ein reComputer J4012 eingesetzt. Dabei handelt es sich um ein lüfterloses Edge-KI-Gerät, welches mit einem NVIDIA Jetson Orin NX-Modul ausgestattet ist.

Die Tiefensensoren der Kamera sowie der LiDAR wurden in dieser Arbeit bewusst nicht verwendet, da, wie bereits erwähnt, ein möglichst minimalistisches System im Fokus steht. Dieses soll auf Basis reiner RGB-Bilddaten fungieren, um eine einfache Kamera zu simulieren. Auch die genaue Positionierung der Kamera am Fahrzeug war zum Zeitpunkt der Arbeit noch nicht final festgelegt und wird im Rahmen dieser Arbeit diskutiert.

2.4 Die Driverless Pipeline

Die Driverless Pipeline beschreibt den Ablauf der einzelnen Funktionsblöcke einer autonomen Steuergunssoftware. Konkrete Umsetzungen können sich von Team zu Team deutlich unterscheiden (siehe in [3], [5], [6]), dennoch lassen sich die folgenden Hauptkomponenten identifizieren (siehe Abbildung 2.4):

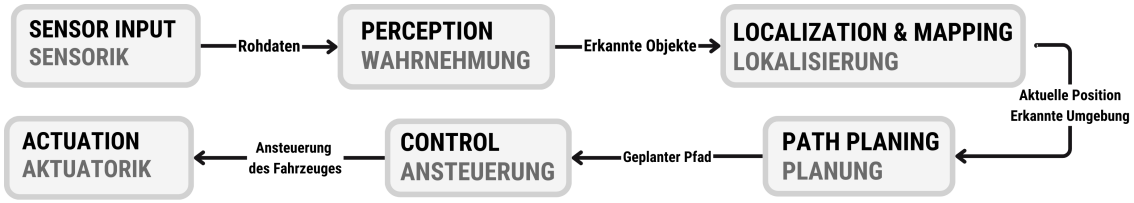

Abbildung 2.4: Die Driverless Pipeline in vereinfachter Form. In jeder Sekunde werden alle Schritte mehrmals durchlaufen.

- **Sensor Input**: Beinhaltet die verschiedenen verbauten Sensoren, welche Umgebungsdaten ermitteln.
- **Perception**: Nutzt die Rohdaten der Sensoren, um umliegende Objekte wie z.B. Pylonen zu erfassen.
- **Localization & Mapping**: Verarbeitet Informationen über die Umgebung und berechnet daraus die aktuelle Position des Fahrzeugs, sowie eine Karte der Umgebung.
- **Path Planing**: Bestimmt den optimalen Pfad durch die ermittelte Umgebung.
- **Control**: Errechnet aus dem geplanten Pfad die Ansteuerung des Fahrzeugs.
- **Actuation**: Nimmt die Befehle zur Ansteuerung entgegen und setzt diese um.

Innerhalb dieser Arbeit wird sich dabei, wie bereits beschrieben, ausschließlich auf den Teil der Perzeption beschränkt.

3 Erstellung eines neuronalen Netzes zur Pylonenerkennung

In diesem Kapitel wird die Entwicklung und Implementierung eines neuronalen Netzes zur Pylonenerkennung beschrieben. Grundlegende Kenntnisse über neuronale Netze werden dabei vorausgesetzt und können hier [12] nachgelesen werden.

3.1 Grundlagen

3.1.1 Das Framework

Für die Erstellung des neuronalen Netzes standen im Wesentlichen zwei Ansätze zur Auswahl, welche beide auf Convolutional Neural Networks basieren: Single-Stage-Detection-Frameworks und Region-Proposal-Frameworks. Der zentrale Unterschied besteht darin, dass Single-Stage-Methoden die Objekterkennung und Klassifikation in einem einzigen Verarbeitungsschritt durchführen, während Region-Proposal-Ansätze diese Aufgaben in separaten Phasen abwickeln [2]. Aufgrund ihrer Effizienz hinsichtlich Rechenzeit und Speicherbedarf sind Single-Stage-Algorithmen besonders gut für Anwendungen im Bereich des autonomen Fahrens geeignet:

> *Meanwhile, single stage detection algorithms tend to have fast inference time and low memory cost, which is well-suited for real-time driving automation.* [2]

Ein besonders populärer Vertreter der Single-Stage-Algorithmen ist You Only Look Once (YOLO). Dieser Ansatz wurde erstmals 2015 von Redmon et al. vorgestellt [13] und zählt zu den ersten Verfahren, die Objekterkennung in einem einzigen Verarbeitungsschritt umsetzen [14], [2]. Die aktuellste Version, YOLOv11, gilt als eines der schnellsten und ressourcenschonendsten Frameworks auf dem Markt und eignet sich daher besonders gut für den in dieser Arbeit verfolgten minimalistischen Anwendungsfall [14]. Ein weiterer Vorteil liegt in der hohen Anwenderfreundlichkeit, da das Framework von der Firma Ultralytics aktiv weiterentwickelt und als Open Source zur Verfügung gestellt wird [15].

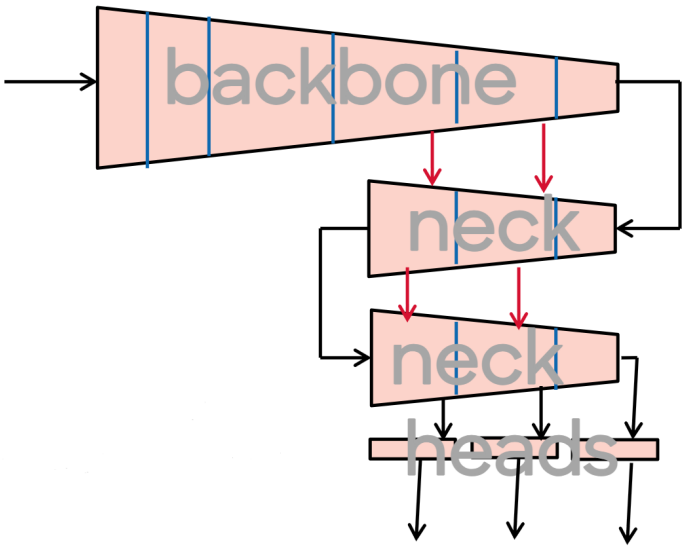

Abbildung 3.1: Übersicht über die YOLO Architektur. Hier von YOLOv8 aus der Vorlesung Deep-Learning- und 3D-Bildverarbeitung der Universität Bremen [17].

Die Verwendung der YOLO-Architektur zur Pylonenerkennung in der Formula Student hat dabei durchaus Bestand, da bereits einige Teams ihre Systeme darauf aufbauen [5], [7]. Beide dieser Teams gewannen den Driverless Cup zuvor mit der Nutzung von YOLO.

Technisch gesehen ist YOLO grob in drei Teile unterteilbar: Dem *backbone*, dem *neck* und dem *head* (siehe Abbildung 3.1). Dabei wird im *backbone* in jedem Schritt (markiert durch vertikale Striche) die Auflösung der Eingabe halbiert bis zu 1/32 der Ursprungsauflösung. Im *neck* wird diese dann wieder hochskaliert auf bis zu 1/8 und dann erneut herunterskaliert, wobei es währenddessen auch Querverbindungen gibt für Informationen aus früheren Layern (rote Pfeile). Zum Schluss werden diese Informationen im *head* ausgegeben. Ein detaillierter Einblick in die komplexe Software sowie wesentliche Unterschiede zu Vorgängerversionen sind hier [14], [16] zu finden.

Dabei ist es wichtig zu erwähnen, dass YOLOv11 in verschiedenen Versionen verfügbar ist (Abbildung 3.2). Diese unterscheiden sich teils stark in der Komplexität und damit in der Anzahl an Parametern und der Geschwindigkeit. Da das Netz für den in dieser Arbeit beschriebenen Anwendungsfall mit mindestens 20Hz laufen können soll, wurde sich für eine Gegenüberstellung der Medium- und der Nano-Variante entschieden. Ziel ist es, im Verlauf der Evaluation eines der beiden Modelle hinsichtlich Laufzeit und Genauigkeit als die geeignetere Wahl zu identifizieren.

Model	size (pixels)	mAP^{val} 50-95	Speed CPU ONNX (ms)	Speed T4 TensorRT10 (ms)	params (M)	FLOPs (B)
YOLO11n	640	39.5	56.1 ± 0.8	1.5 ± 0.0	2.6	6.5
YOLO11s	640	47.0	90.0 ± 1.2	2.5 ± 0.0	9.4	21.5
YOLO11m	640	51.5	183.2 ± 2.0	4.7 ± 0.1	20.1	68.0
YOLO11l	640	53.4	238.6 ± 1.4	6.2 ± 0.1	25.3	86.9
YOLO11x	640	54.7	462.8 ± 6.7	11.3 ± 0.2	56.9	194.9

Abbildung 3.2: Die verschiedenen Versionen von YOLOv11: Nano, Small, Medium, Large und Extra-Large (Screenshot von der offiziellen Website [15]). Grundsätzlich gilt, dass mit steigender Anzahl an Parametern die Genauigkeit zunimmt, jedoch auch die Laufzeit entsprechend länger wird.

3.1.2 Der Datensatz

Diese Arbeit basiert auf dem Formula Student Object in Context (FSOCO) Datensatz [18], [19]. Dieser wird von der Formula Student Driverless Community verwaltet und beinhaltet ca. 11.500 Aufnahmen von Pylonen mit dazugehörigen Labels, wobei im Durchschnitt 19 annotierten Pylonen pro Bild vorkommen (Abbildung 3.3). Der Datensatz ist Open Source und jedes Team kann zur Erweiterung beitragen. Zum Zeitpunkt dieser Arbeit haben insgesamt 18 verschiedene Teams ihre Aufnahmen dort geteilt, was für eine große Vielfalt an Umgebungen, Kamerawinkeln sowie Licht- und Wetterverhältnissen sorgt. Somit bietet dies eine optimale Grundlage zum Trainieren eines neuronalen Netzes.
Im Datensatz wird grundsätzlich unterschieden zwischen den Klassen *Blue cone*, *Yellow cone*, *Orange cone*, *Large orange cone* und *Unknown cone*. Damit werden alle Pylonentypen abgedeckt, die es auf einer Formula Student Driverless Strecke gibt (siehe Abschnitt 2.2).
Zusätzlich sind die Pylonen mit Tags versehen, die situative Bedingungen wie *knocked over*, *tape modified* oder *truncated* kennzeichnen. Da diese Bedingungen für das in dieser Arbeit entwickelte Erkennungssystem nur von geringer Relevanz sind und sich das Tagging-Konzept nur schwer in die YOLO-Architektur integrieren lässt, wurden diese Informationen bei der Entwicklung nicht berücksichtigt.

3.2 Konzept

Die Entwicklung eines neuronalen Netzes lässt sich grundsätzlich in zwei Phasen unterteilen: die Trainingsphase zur Modelloptimierung und die Inferenzphase für Vorhersagen.

Abbildung 3.3: Ein Beispielbild aus dem FSOCO Datensatz mit annotierten Pylonen.

Beim Training wird das Netz mithilfe der annotierten Bildern des FSOCO Datensatz auf die Eigenschaften der verschiedenen Pylonenklassen trainiert. Dieser Prozess basiert auf einem iterativen Optimierungsverfahren, bei dem die Bilder mehrfach durchlaufen und die internen Gewichte des Netzes schrittweise angepasst werden. Ziel ist es, dass das Netz die relevanten Bildmerkmale zuverlässig erkennt und korrekt zuordnet.
Ein zentraler Aspekt dabei ist die Generalisierungsfähigkeit des Netzes. Es soll in der Lage sein, Pylonen unter verschiedenen Bedingungen zu erkennen, unabhängig von Lichtverhältnissen, Perspektiven oder Hintergrundvariationen. Der FSOCO-Datensatz unterstützt diesen Anspruch aufgrund seiner zuvor beschriebenen Vielseitigkeit.

Nach Abschluss des Trainings wird das Modell zur Inferenz genutzt. Dabei verarbeitet es neue, zuvor nicht gesehene Bilder, womit sich die Erkennungsleistung testen lässt. Die Geschwindigkeit und Zuverlässigkeit der Inferenz sind dabei entscheidend, da das System zeitkritisch und präzise Pylonen erkennen muss.

3.3 Umsetzung

3.3.1 Modelltraining

Vor dem Training muss zunächst der Datensatz angepasst werden, genauer gesagt die Label-Dateien. Dies sind Dateien, welche zu jedem Bild wichtige Metadaten enthalten wie die Klassenzugehörigkeit der annotierten Pylonen oder die Koordinaten der dazugehörigen Bounding Boxes. Standardmäßig liegen diese Dateien im Supervisely-Format vor (siehe Quelltext 3.1), welches von YOLO nicht unterstützt wird. Da YOLO ein eigenes Format für Annotationen verwendet, müssen alle Label-Dateien zunächst konvertiert werden (siehe Quelltext 3.2).

```
"classTitle": "orange_cone",
"points": {
        "exterior": [
                [
                        313,
                        822
                ],
                [
                        457,
                        978
                ]
        ],
        "interior": []
}
```

Quelltext 3.1: Metadaten im Supervisely Format. Die Zahlenpaare in „exterior“ stehen für die Bildpixelkoordinaten der oberen linken sowie der unteren rechten Ecke der Bounding Box eines Pylonen der Klasse *Orange Cone*.

```
8 0.175 0.661764705882 0.0654545454545 0.1147058823529
```

Quelltext 3.2: Metadaten im YOLO Format. Die einzelnen Zahlen stehen dabei für Klasse (8 steht für *Orange Cone*), x-Koordinate des Mittelpunktes der Bounding Box, y-Koordinate des Mittelpunktes der Bounding Box, Breite der Bounding Box und Höhe der Bounding Box. Alle Koordinaten sind normiert.

Ebenfalls angepasst wurden die Bilder des Datensatzes. Standardmäßig haben diese einen schwarzen Rand, auf dem das Logo des jeweiligen Teams abgebildet ist, von dem das Bild stammt. Da diese Informationen jedoch keine Relevanz für die Erkennung von Pylonen haben, wurden alle schwarzen Ränder entfernt (siehe Abbildung 3.4). Zusätzlich mussten die Koordinaten der Bounding Boxes an die resultierende

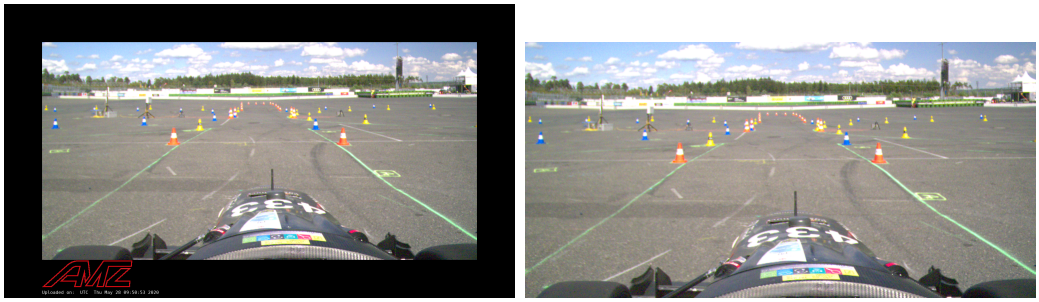

Abbildung 3.4: Beispiel eines Bildes des FSOCO Datensatz vor (links) und nach (rechts) dem Zuschnitt.

Bildgröße angepasst werden. Mit diesem Schritt wurde die Qualität der Eingabebilder hinsichtlich des Anwendungsfalles verbessert.

Als abschließender Vorbereitungsschritt wurde der Datensatz wie folgt aufgeteilt:

- 80% Trainingsdaten: Diese dienen zur Optimierung der internen Modellparameter.
- 20% Validierungsdaten: Diese dienen zur Überprüfung der Generalisierung des Modells während des Trainings.

Dies ist eine gängige Aufteilung und wird so in [12] beschrieben. Dabei wurde besonders darauf geachtet, dass die Aufteilung kein Ungleichgewicht in den Daten verursacht. So wurde die 80/20-Aufteilung zunächst separat für die Bilder jedes Teams durchgeführt und anschließend zu einem gemeinsamen Datensatz zusammengeführt. Somit konnte das Verhältnis der äußeren Umstände zwischen Trainings- und Validierungsdaten gewahrt werden.

Nun sind alle Vorbereitungen getroffen und das Training kann beginnen. Wie in Unterabschnitt 3.1.1 erläutert, wurden zwei unterschiedliche Modellvarianten ausgewählt, welche anschließend miteinander verglichen werden: das Nano-Modell und das Medium-Modell. Beim Training bietet YOLO die Möglichkeit, verschiedene Parameter anzupassen. Um den Vergleich der beiden Modelle in der Evaluation fair zu gestalten, wurden diese mit den selben Parametern trainiert (siehe Quelltext 3.3). Da diese nicht selbsterklärend sind, werden sie im Folgenden erläutert:

```
# Train the model
model.train(data="../fsoco_yolo_format/data.yaml",
   epochs=15, imgsz=1024, batch=-1, mosaic=True,
   patience=5)
```

Quelltext 3.3: Python-Code zum Trainieren des neuronalen Netzes.

- **data**: Der bearbeitete Datensatz als Input.
- **epochs**: Die Anzahl der Durchläufe durch den gesamten Datensatz. Während jedes Durchlaufs werden die Eigenschaften der Pylonen gelernt und das Modell iterativ optimiert. Eine Anzahl von 15 Epochen wurde gewählt, da höhere Werte keinen signifikanten Leistungszuwachs gezeigt haben.
- **imgsz**: Die Eingabegröße der Bilder. YOLO skaliert während des Trainings alle Bilder quadratisch auf diese angegebene Größe, wobei das Seitenverhältnis beibehalten und Padding hinzugefügt wird, um Verzerrungen zu vermeiden [20]. Da der Datensatz jedoch Bilder unterschiedlicher Auflösungen durch die verschiedenen Teams enthält, wurde um Qualitätsverluste zu minimieren eine Eingabegröße von 1024 Pixeln gewählt, da dies die durchschnittliche Größe der Bilder ist.
- **batch**: Bestimmt die Anzahl der Bilder, die das Modell gleichzeitig während des Trainings verarbeitet. Eine größere Batch-Größe kann die Trainingsstabilität verbessern, erfordert jedoch mehr Speicher. Wird der Wert auf `-1` gesetzt, berechnet das System automatisch eine optimale Batch-Größe basierend auf der verfügbaren Hardware.
- **mosaic**: Aktiviert die Mosaic Augmentation, bei der vier Trainingsbilder zu einem kombiniert werden. Diese Technik kann helfen, die Generalisierungsfähigkeit des Modells zu verbessern, indem sie eine größere Variabilität in den Trainingsdaten erzeugt [20]. Dadurch sollte das Modell robuster gegenüber unterschiedlichen Perspektiven, Hintergrundvariationen und Bildverzerrungen werden, die bei realen Rennsituationen auftreten.
- **patience**: Legt den Schwellenwert für ein Early Stopping fest. Bleibt die Modellleistung über eine definierte Anzahl von Epochen (hier: 5) unverändert, wird das Training vorzeitig beendet, um Overfitting zu vermeiden. Diese Entscheidung basiert auf der Entwicklung interner Validierungsmetriken wie des Mean Average Precision (mAP).

3.3.2 Modellinferenz

Die trainnierten Modelle werden nun zur Inferenz genutzt. Dabei werden den Modellen Eingabedaten in Form von Bildern oder Videos übergeben, auf welchen dann Vorhersagen basierend auf den gelernten Mustern und Erkenntnissen aus dem Training getätigt werden. Analog zum Training bietet YOLO auch in dieser Phase eine Vielzahl von Parametern zum Einstellen an, wobei der wichtigste Parameter die Bildgröße **imgsz** ist.
Die Größe des Eingabebildes hat einen erheblichen Einfluss auf die Geschwindigkeit

Abbildung 3.5: Beispiel einer Aufnahme vor (links) und nach (rechts) dem Zuschnitt.

der Vorhersage. Dies liegt daran, dass YOLO das Bild in mehrere Abschnitte unterteilt, um nach den trainierten Merkmalen für Pylonen zu suchen. Eine kleinere Bildgröße führt zu weniger Abschnitten und damit zu schnelleren Berechnungen.
Im Gegensatz zum Training bietet YOLO bei der Inferenz mehr Flexibilität in der Festlegung der Eingabegröße des Bildes. Während beim Training nur eine feste Größe übergeben werden kann, ermöglicht YOLO bei der Inferenz die genaue Angabe der Pixelwerte in x- und y-Richtung. Dies ist möglich, da YOLO bei der Inferenz stets ein Bild zur Zeit verarbeitet. Beim Training wird hingegen mit mehreren Bildern gleichzeitig gearbeitet, weshalb diese auf eine einheitliche quadratische Größe skaliert werden. Diese Flexibilität bei der Inferenz ermöglicht es, eine optimierte Eingabegröße zugunsten der Erkennungsleistung festzulegen.
Dabei gibt es verschiedene Ansätze um dies zu tun. Eine Möglichkeit besteht darin, die Bilder in der Skalierung zu verkleinern. Dabei können jedoch Details verloren gehen. Eine andere Option ist das Zuschneiden der Bilder. Dabei können unwichtige Abschnitte eines Bildes entfernt werden ohne wichtige Details zu verlieren. Bei genauer Betrachtung der Aufnahmen fällt auf, dass sich die Pylonen ausschließlich in einem bestimmten Bereich des Bildes befinden. Dieser Bereich lässt sich definieren durch einen horizontalen Streifen zwischen dem Fahrzeug und dem Ende des Asphalts. Der Himmel und der Hintergrund sind für die Pylonenerkennung irrelevant und können deshalb abgeschnitten werden (siehe Abbildung 3.5).

Detaillierte Analysen der vorgestellten Ansätze zur Optimierung der Erkennungsleistung und Laufzeit sind in den folgenden Kapiteln der Evaluation zu finden.

3.4 Evaluation

Um die neuronalen Netze umfassend zu evaluieren, wurden verschiedene Analysen durchgeführt. Diese umfassen eine Bewertung der Gesamtleistung, eine Laufzeitanalyse und eine Analyse der Erkennungsleistung in Abhängigkeit zur Distanz. Darüber

hinaus wird über die Kapitel hinweg eine Analyse über die Positionierung der Kamera am Fahrzeug durchgeführt.

Für die Evaluation wurde ein Test mit dem Bremo24 durchgeführt. Dazu wurde auf dem Betriebshof der Universität Bremen eine Strecke gemäß des Formula Student Regelwerks [9] aufgebaut. Die Gestaltung orientierte sich am Trackdrive-Szenario, dessen Regeln wie folgt definiert sind:

> *The trackdrive layout is a closed loop circuit built to the following guidelines:*
>
> - *Straights: No longer than 80 m*
> - *Miscellaneous: Chicanes, multiple turns, decreasing radius turns, hairpin turns, etc.*
> - *The minimum track width is 3 m*
> - *The minimum required turning diameter is 9 m* [9]

Zusätzlich wird in [21] erwähnt, dass der maximale Abstand zwischen Pylonen derselben Farbe 5 Meter nicht überschreiten darf.

Um die optimale Kameraposition zu bestimmen, wurde das Fahrzeug mit drei Kameras ausgestattet: einer GoPro Hero 4 am Mainhoop, einer GoPro Hero 3 auf der Motorabdeckung (im Folgenden Engine Cover genannt) und einer Intel RealSense im Frontflügel (im Folgenden Frontwing genannt), welche final im Fahrzeug verbaut werden soll (siehe Abbildung 3.6).

Das Fahrzeug wurde anschließend bemannt durch den Parcours gefahren, um möglichst viele Daten für die Evaluation zu sammeln. Dabei traten jedoch einige Probleme auf: Die Frontwing-Kamera zeichnete aufgrund einer fehlerhaften Einstellung in einem falschen Format auf. Zudem stellte sich nachträglich heraus, dass die Speicherkarte einer der GoPro-Kameras beschädigt war, weshalb auch diese in einem alternativen Format aufnehmen musste. Dadurch liegen die finalen Aufnahmen in den folgenden Auflösungen vor:

- **Mainhoop:** 1920×1080 Pixel
- **Engine Cover:** 1280×960 Pixel
- **Frontwing:** 640×480 Pixel

Diese Varianz in den Auflösungen ist suboptimal und wird in der folgenden Evaluation entsprechend berücksichtigt.

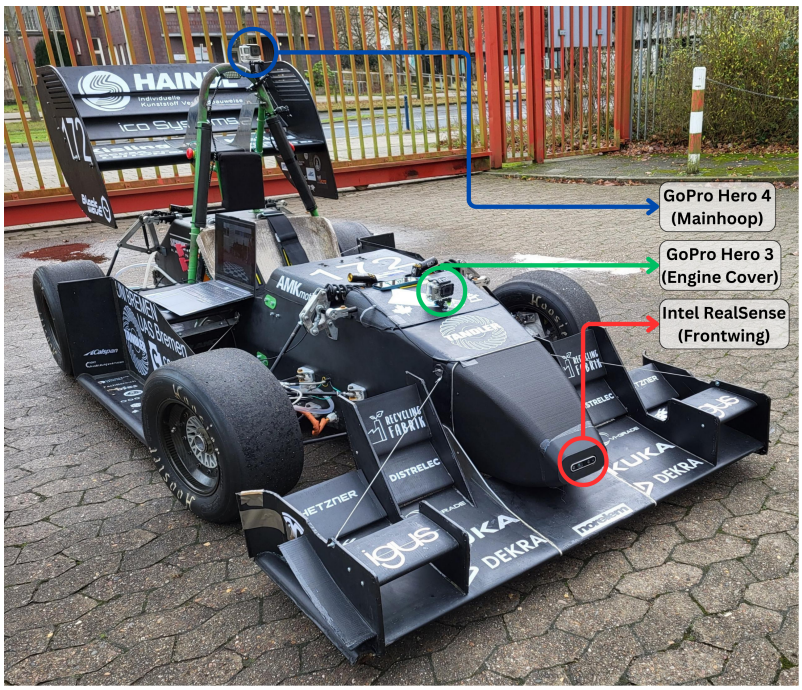

Abbildung 3.6: Der Bremo24 während des Tests, ausgestattet mit drei Kameras.

3.4.1 Analyse der Erkennungsleistung

Um die Netze fair evaluieren zu können und einen aussagekräftigen Vergleich zwischen den Kameras zu ermöglichen, mussten zunächst Ground Truths in einem einheitlichen Rahmen erzeugt werden. Dazu wurden die Aufnahmen des Tests zu einem komprimierten, repräsentativen Video zusammengeschnitten (siehe Abbildung 3.7). Insbesondere folgende Kriterien wurden berücksichtigt:

- Fahrzeuggeschwindigkeit
- Lichtverhältnisse
- Streckenabschnitte

Diese Bedingungen sollten für alle Kameras möglichst identisch sein, um eine faire Evaluation zu gewährleisten. Zudem war es wichtig, dass die Videos typische Streckenabschnitte einer Formula Student Driverless Strecke, wie zuvor beschrieben, realistisch widerspiegeln. Letztendlich wurden alle brauchbaren Szenen zu etwa einminütigen Videos für jede Kamera zusammengestellt.

Im nächsten Schritt wurde aus diesen Videos jeweils um die 100 Frames extrahiert, welche im Nachgang händisch annotiert wurden. Dies geschah, um ein hochwertigen

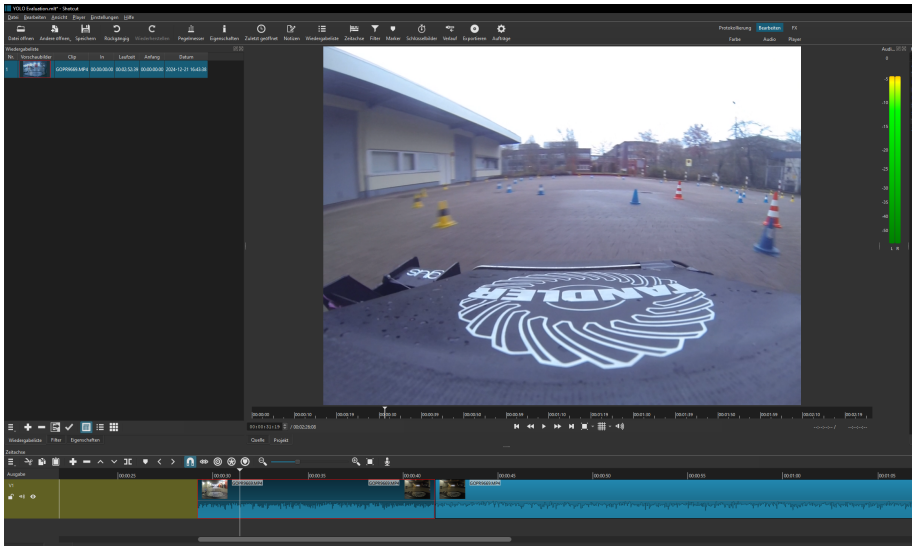

Abbildung 3.7: Schneiden der Evaluationsvideos mithilfe von Shotcut [22].

und konsistenten Ground Truth Datensatz zu erstellen (siehe Abbildung 3.8).

Während der Annotation wurden strikte Kriterien eingehalten. Es wurden ausschließlich Pylonen annotiert, die eindeutig zu erkennen und klar zuzuordnen waren (siehe Abbildung 3.9). Es ist jedoch anzumerken, dass dieser Prozess eine gewisse Subjektivität beinhaltet.

Resultierende Ground Truth Daten sind in Tabelle 3.1 dargestellt. Bereits auf den ersten Blick sind signifikante Unterschiede zwischen den Kameras erkennbar. Während die Engine Cover-Kamera eine große Anzahl an Pylonen erfasst hat, ist diese Zahl bei der Mainhoop-Kamera etwas geringer. Die Frontwing-Kamera weist mit Abstand die geringste Anzahl an Pylonen auf. Die Ursache dieser Unterschiede wird folglich untersucht.

Ein entscheidender Faktor ist die Herkunft der Aufnahmen. Während die Engine Cover- und Frontwing-Kameras die gleiche Fahrt zeigen, musste für die Mainhoop-Kamera auf eine alternative, aber vergleichbare Fahrt zurückgegriffen werden. Wie zuvor erwähnt, waren nicht alle Aufnahmen aufgrund technischer Probleme verwendbar, was diese Maßnahme erforderlich machte. Durch diesen Unterschied befinden sich insgesamt weniger Pylonen im Sichtfeld der Mainhoop-Kamera.
Besonders interessant sind die Unterschiede zwischen der Engine Cover- und der Frontwing-Kamera, da diese, wie zuvor erläutert, die selbe Fahrt dokumentieren. Der Unterschied in der Anzahl der extrahierten Frames lässt sich dabei auf die unterschiedlichen Aufnahmefrequenzen der Kameras zurückführen. Der signifikante Unterschied in der erkannten Pylonenanzahl ist jedoch bemerkenswert und lässt sich durch drei wesentliche Faktoren erklären:

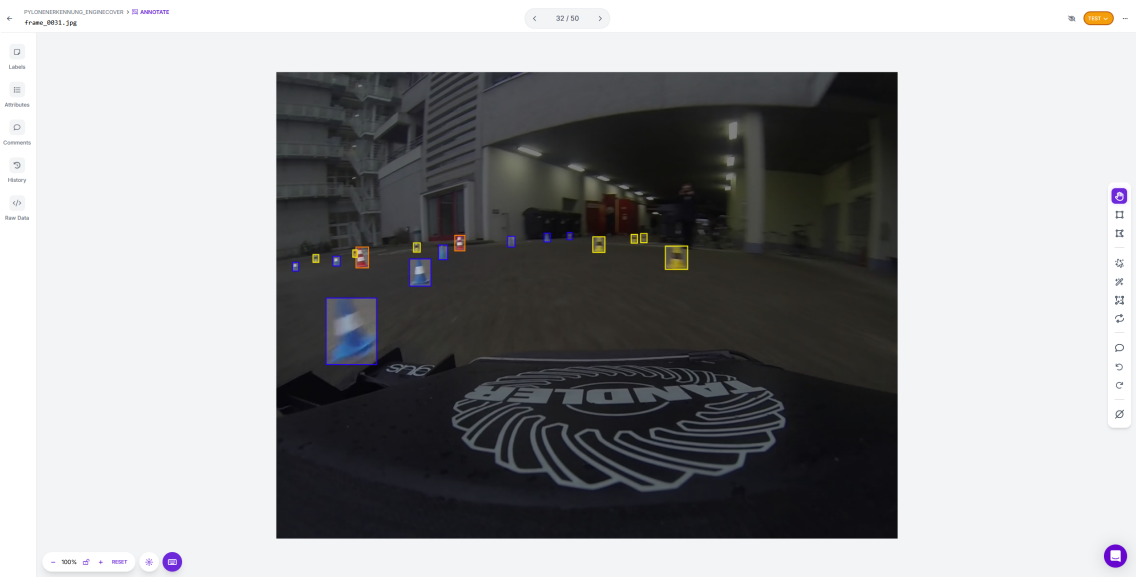

Abbildung 3.8: Manuelle Annotation der extrahierten Frames mithilfe von Roboflow [23].

Abbildung 3.9: Beispiele der manuellen Annotation. Unerkennliche Pylonen wurden nicht annotiert.

Tabelle 3.1: Die resultierenden Ground Truths pro Kamera.

Kamera	Yellow Cone	Blue Cone	Large Orange Cone	Frames
GoPro Mainhoop	584	625	51	101
GoPro Engine Cover	605	650	66	113
RealSense Frontwing	256	231	27	105
Gesamt	**1445**	**1506**	**144**	**319**

Abbildung 3.10: Vergleich des selben Frames der Kameras am Engine Cover (links) vs. Frontwing (rechts).

1. Die Frontflügel-Kamera erfasst aufgrund ihrer geringeren Auflösung einen deutlich kleineren Bildausschnitt, was dazu führt, dass weniger Pylonen im Bild sichtbar sind.
2. Die Positionierung der Kamera am Frontflügel führt dazu, dass Pylonen häufig verdeckt werden, da sich die Kamera auf gleicher Höhe mit den Pylonen befindet.
3. Der Frontflügel ist ein bewegliches Bauteil mit einer hohen Bewegungsamplitude, insbesondere während Kurvenfahrten. Aufgrund der Position der Kamera an dieser Stelle kommt es zu stärkeren Bewegungen der Kamera, was zu unscharfen Aufnahmen führt.

Diese Faktoren sind in Abbildung 3.10 zu sehen.

Zur Analyse der Erkennungsleistung der neuronalen Netze wurde ein Evaluationsskript entwickelt, das die Inferenzen der Modelle mit den zuvor erstellten Ground Truths abgleicht und daraus die im Folgenden zu sehenden Metriken berechnet. Die Entwicklung eines eigenen Skripts war notwendig, da für diesen spezifischen Anwendungsfall keine einsetzbare Evaluationslösung zur Verfügung stand. Zwar stellt das YOLO-Framework eine integrierte Validierungsmethodik bereit, jedoch werden die Bilder dabei, wie auch im Trainingsprozess, quadratisch skaliert, da mit mehreren

Bildern gleichzeitig gearbeitet wird. Für den vorliegenden Anwendungsfall ist diese Vorgehensweise daher ungeeignet. Außerdem existieren weitere Ansätze zur Evaluierung von Objekterkennungsmodellen [24], [25], allerdings ist keiner dieser Methoden ohne zusätzliche Anpassungen einsetzbar.

Die Berechnungen des entwickelten Evaluationsskripts basieren dabei auf der Implementierung der Intersection over Union (IoU) [16]. Zur Zuordnung der Bounding Boxes wurde ein Best-Match-Algorithmus eingesetzt, der eine eindeutige Paarung von Predictions und Ground Truths gewährleistet. Als Bewertungsmetrik kommt die Mean Average Precision (mAP) zum Einsatz, welche als etablierter Standard zur Leistungsmessung von Objekterkennungsmodellen gilt [16].

Im Folgenden wird nun die Leistung der trainierten Modelle auf den zuvor erstellten Ground Truths des Tests evaluiert. In die Berechnung fließen die Klassen der *blue-*, *yellow-* sowie *large orange-cones* mit gleicher Gewichtung ein. Die Klasse der *orange cones* wurde aufgrund ihrer Abwesenheit am Testtag nicht berücksichtigt.
Das Medium-Modell erreicht dabei eine mAP von 0.85 über alle Kameraperspektiven hinweg, während das Nano-Modell mit einer mAP von 0.87 leicht darüber liegt (siehe Abbildung 3.11). Dieses zunächst kontraintuitiv erscheinende Ergebnis lässt sich durch die spezifische Zusammensetzung des Ground-Truth-Datensatzes erklären:

Im Vergleich zum ursprünglichen Trainingsdatensatz ist der für die Evaluation verwendete Ground-Truth-Datensatz deutlich kleiner und weist eine höhere Dichte an Pylonen unter erschwerten Bedingungen auf, etwa bei Bewegungsunschärfe oder durch ruckartige Kamerabewegungen. Diese Faktoren erschweren zwar die Erkennung, spiegeln jedoch realistischere Bedingungen wider, wie sie auch in praktischen Fahrszenarien zu erwarten sind. Ein robustes Modell muss in der Lage sein, auch unter solchen Umständen zuverlässig zu arbeiten.
Dieser Unterschied im Datensatz wirkt sich zugunsten des Nano-Modells aus, da dieses tendenziell eher etwas detektiert als das Medium-Modell. Dies liegt daran, dass das Nano-Modell, wie in Unterabschnitt 3.1.1 gezeigt, weniger Parameter besitzt und dadurch schneller eine Detektion ausgibt, also früher „etwas erkennt“ als komplexere Modelle. Diese Tendenz wird durch die Betrachtung der False-Positives in der Confusion Matrix bestätigt: Während beim Nano-Modell insgesamt 197 False Positives auftreten, liegt dieser Wert beim Medium-Modell lediglich bei 78 (siehe Abbildung 3.12).

Das Medium-Modell ordnet einem Pylon demnach nur dann eine Detektion zu, wenn mit hoher Wahrscheinlichkeit eine korrekte Klassifikation vorliegt. Diese Vorgehensweise führt jedoch dazu, dass im Rahmen der Tests zahlreiche Pylonen nicht erkannt werden, insbesondere in Bildern mit unscharfen Aufnahmen, wie sie im Evaluationsdatensatz auftreten. Dies ist zu sehen an den False-Negatives in der Confusion Matrix (siehe Abbildung 3.12), welche beim Medium-Modell insgesamt 328 mehr betragen.

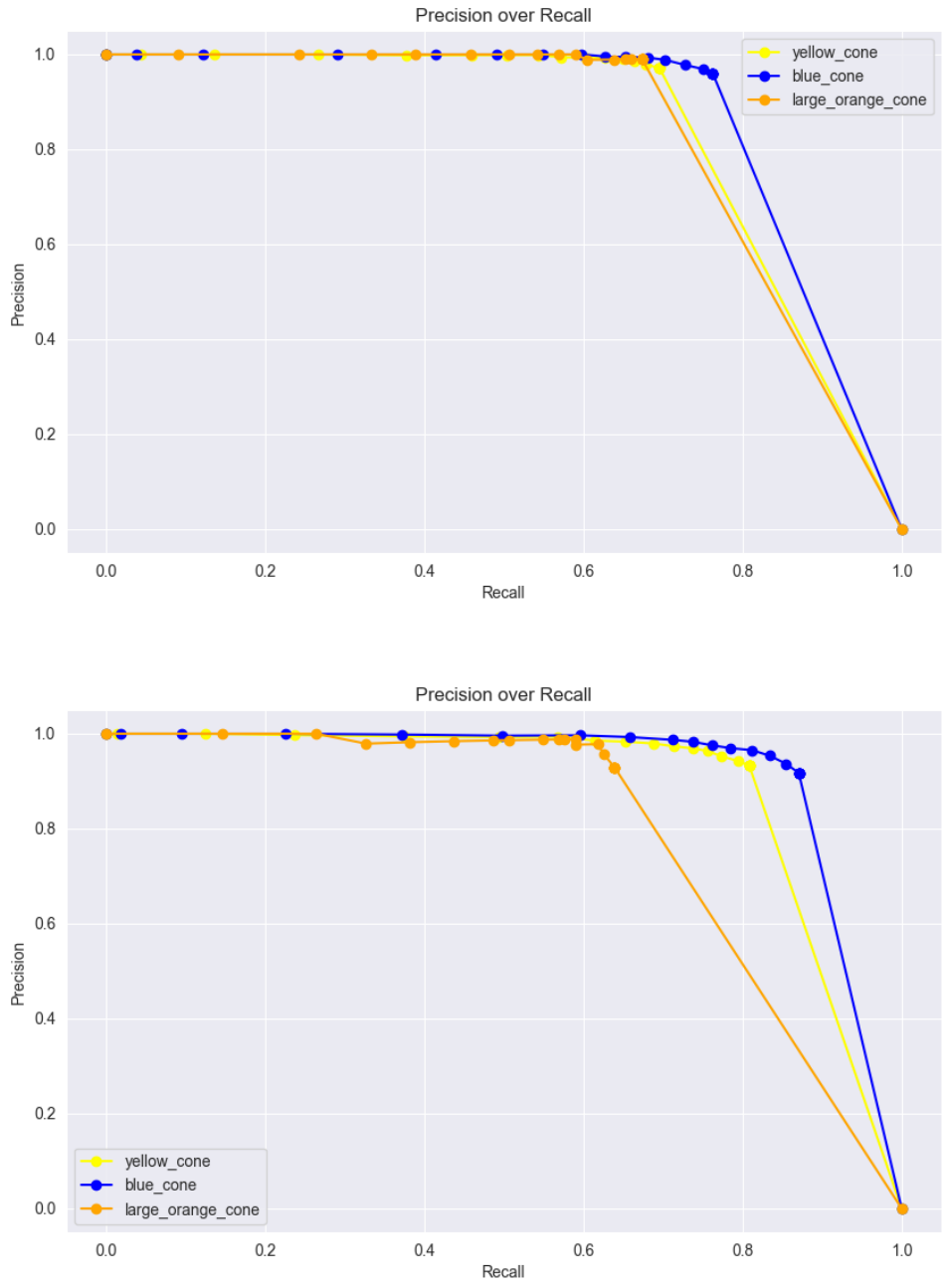

Abbildung 3.11: Vergleich von Precision over Recall des Medium-Modells (oben) mit dem des Nano-Modells (unten) auf den erstellten Ground Truths des Tests. Aus der Fläche unterhalb der Graphen wird die Mean Averag Precision (mAP) errechnet.

Tabelle 3.2: Mean Average Precision (mAP) des Ground Truth Datensatzes, unterteilt pro Kamera.

Kamera	Medium-Modell	Nano-Modell
GoPro Mainhoop	0.92	0.94
GoPro Engine Cover	0.83	0.85
RealSense Frontwing	0.75	0.77

Da derartige Bedingungen auch unter realen Fahrbedingungen zu erwarten sind, kann das Nano-Modell hier von Vorteil sein.

Zusammenfassend lässt sich feststellen, dass das Nano-Modell auf dem vorliegenden Ground-Truth-Datensatz eine insgesamt um 0.02 mAP bessere Leistung zeigt als das Medium-Modell. Interessant ist nun die Frage, ob dieses Ergebnis so bestehen bleibt, wenn man den Datensatz auf die verschiedenen Kameraperspektiven aufteilt. In diesem Fall ergibt sich folgendes Bild (siehe Tabelle 3.2).

Daraus lassen sich zwei zentrale Beobachtungen ableiten:

1. Das Nano-Modell erzielt in jeder betrachteten Kameraperspektive einen durchschnittlich um 0.02 höheren mAP Wert als das Medium-Modell.
2. Die mAP steigt mit zunehmender Höhe der Kameraposition am Fahrzeug. Insgesamt ist ein Anstieg von etwa 0.1 mAP zu verzeichnen, je höher die Kamera montiert wurde.

Diese Zusammenhänge lassen sich auf den veränderten Blickwinkel zurückführen, der mit einer höheren Kameraposition einhergeht. Ein erhöhter Kamerawinkel reduziert potenzielle Verdeckungen von Objekten im Sichtfeld und sorgt zugleich für eine stabilere Bildwahrnehmung. Dies trägt wesentlich zur Verbesserung der Erkennungsleistung bei.
Daraus lässt sich schlussfolgern, dass die Position der Mainhoop-Kamera den optimalen Montagepunkt für die Objekterkennung im Rahmen dieses Anwendungsfalls darstellt. Eine Erkennungsleistung von ca. 0.93 mAP gemittelt über beide Modelle ist dabei sehr gut.

3.4.2 Laufzeitanalyse

Nachdem im vorangegangenen Kapitel die optimale Kameraposition identifiziert wurde, erfolgt im Folgenden eine Analyse der Laufzeit. Da die Objekterkennung den ersten Verarbeitungsschritt in der Driverless Pipeline bildet, wirkt sich ihre Laufzeit direkt auf die Gesamtleistung des Systems aus. Eine langsame Erkennung

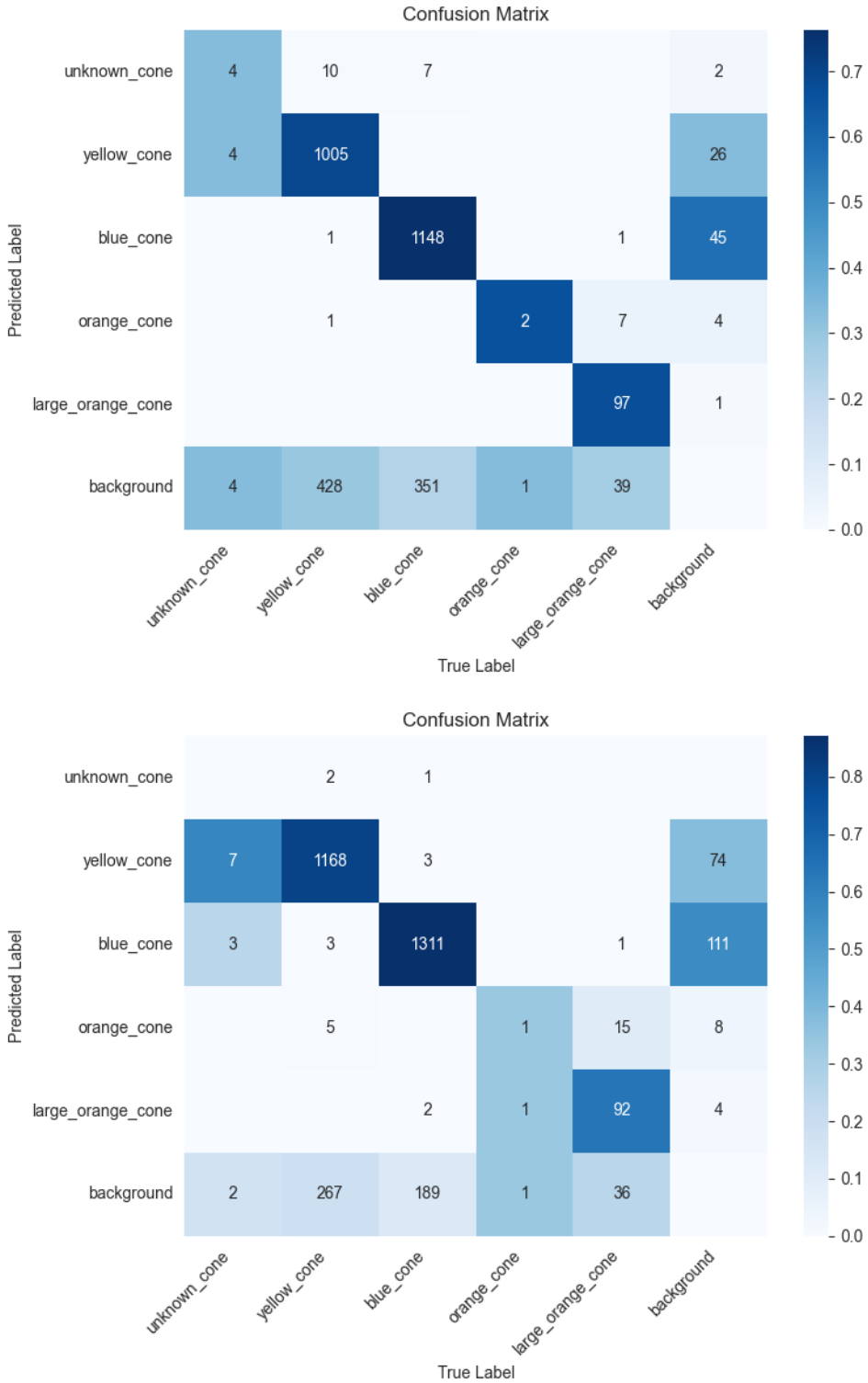

Abbildung 3.12: Vergleich der Confusion Matrix des Medium-Modells (oben) mit dem des Nano-Modells (unten) auf den erstellten Ground Truths. False Positives zeigen sich in der Spalte der „background“-Klasse, während False Negatives in der Zeile der „background“-Klasse erscheinen.

begrenzt zwangsläufig die maximal mögliche Geschwindigkeit und Reaktionsfähigkeit des Systems.

Da im vorangegangenen Kapitel gezeigt wurde, dass die Position der Kamera am Mainhoop die besten Erkennungsraten liefert, wird die Analyse der Laufzeit ausschließlich für diese Position durchgeführt. Im Rahmen der Analyse werden verschiedene Bildauflösungen getestet sowie erneut die beiden Modellvarianten, Medium und Nano, gegenübergestellt. Der Fokus liegt dabei insbesondere auf der zur Inferenz benötigten Zeit, die anschließend in Relation zur erzielten Erkennungsleistung gesetzt wird, um eine fundierte Bewertung der optimalen Auflösung zu ermöglichen.

Das methodische Vorgehen gestaltet sich dabei wie folgt: Für jedes der beiden Modelle wird der Ground-Truth-Datensatz der Mainhoop-Kamera, bestehend aus 100 Bildern, in vier verschiedenen Varianten getestet. Diese Varianten basieren auf die in Unterabschnitt 3.3.2 vorgestellten Ansätzen (siehe Abbildung 3.13):

- Bilder in Originalauflösung 1920x1088 Pixel. (Eigentlich 1920x1080, aber, da wie in den Grundlagen erwähnt, YOLO während der Berechnungen die Auflösung auf 1/32 der Ursprungsauflösung herunterskaliert, muss die Eingabegröße ein Vielfaches von 32 sein, dementsprechend 1088. Selbiges gilt für alle Varianten.)
- Bilder in halbierter Auflösung 960x544 Pixel.
- Horizontal zugeschnittene Bilder 1920x544 Pixel.
- Zugeschnittene Bilder in halbierter Auflösung 960x288 Pixel.

Jede dieser Konfigurationen wurde dreimal vollständig durchlaufen, wobei die Zeit zur Berechnung der Inferenz jeweils gemessen und anschließend der Mittelwert gebildet wurde. Die Erkennungsleistung (gemessen in mAP) wurde dabei analog zum vorherigen Kapitel bestimmt. Es ist zudem zu berücksichtigen, dass der zeitliche Aufwand für das Zuschneiden der Bilder, insofern er für die jeweilige Konfiguration erforderlich war, ebenfalls in die Zeiterfassung mit einbezogen wurde. Diese zusätzliche Vorverarbeitung stellt eine weitere Rechenoperation in der Verarbeitungskette dar und beeinflusst damit direkt die Gesamtlaufzeit.

Die resultierenden Ergebnisse zur Laufzeit sowie zur Erkennungsleistung sind in Tabelle 3.3 und Tabelle 3.4 dargestellt.

Die Ergebnisse zeigen erwartungsgemäß, dass das Nano-Modell über alle Testläufe hinweg eine deutlich geringere Inferenzzeit aufweist als das Medium-Modell. In Originalauflösung ist es dabei acht Sekunden schneller. Bemerkenswert ist jedoch, dass es trotz der geringeren Modellkomplexität eine zum Medium-Modell vergleichbare mAP erzielt.

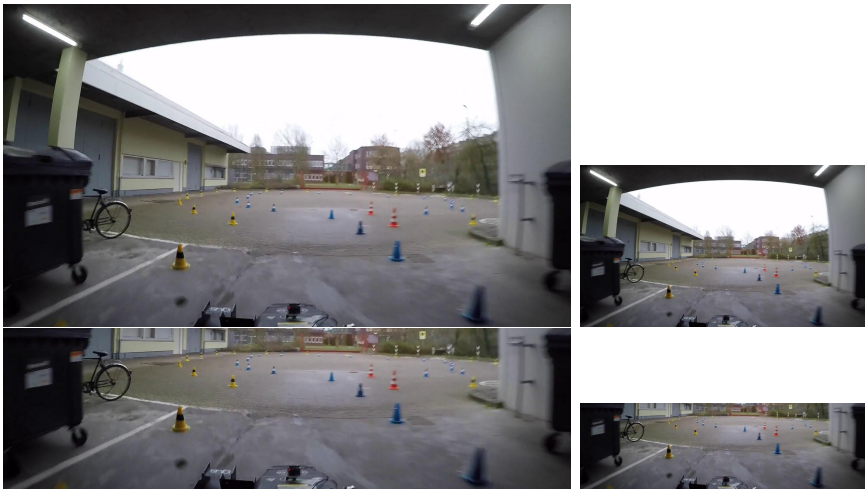

Abbildung 3.13: Die vier Durchläufe der Laufzeitanalyse (maßstabsgetreu skaliert).

Tabelle 3.3: Laufzeitanalyse des Medium Modells.

Auflösung	Gesamtzeit (s)	Pro Bild (ms)	Hertz	mAP
Original	13.13	130	7.69	0.9185
Halbiert	5.34	52.8	18.93	0.9591
Zugeschnitten	9.09	90	11.11	0.902
Zugeschnitten + Halbiert	4.58	45.3	22.08	0.9491

Tabelle 3.4: Laufzeitanalyse des Nano Modells.

Auflösung	Gesamtzeit (s)	Pro Bild (ms)	Hertz	mAP
Original	5.13	50.8	19.70	0.9417
Halbiert	3.92	38.9	25.74	0.9467
Zugeschnitten	4.81	47.6	20.99	0.9441
Zugeschnitten + Halbiert	**3.84**	**38.1**	**26.27**	**0.945**

Darüber hinaus lässt sich beobachten, dass niedrigere Bildauflösungen zu einer verbesserten Laufzeit führen, ohne dabei signifikante Einbußen in der Erkennungsgenauigkeit hinzunehmen. Besonders hervorzuheben ist hierbei die Konfiguration mit zugeschnittenen Bildern in halbierter Auflösung. Das Nano-Modell erreicht in dieser Einstellung eine Frequenz von 26.27Hz bei gleichzeitig hoher Erkennungsleistung mit einer mAP von 0.945. Diese Ergebnisse sind insofern bemerkenswert, da Pylonen nur eine geringe Größe im Bild einnehmen. Dass eine derart hohe Erkennungsleistung auch bei stark reduzierter Auflösung erreicht wird, unterstreicht die Effizienz und Robustheit der YOLO-Architektur.

Es ist an dieser Stelle anzumerken, dass die beschriebenen Experimente auf einem externen Computer mit zuvor aufgezeichnetem Videomaterial durchgeführt wurden. Die zugrunde liegende Hardware bestand aus einer NVIDIA GeForce GTX 1070Ti Grafikkarte sowie einem AMD Ryzen 5 2600X Prozessor. Im finalen System auf dem Fahrzeug wird jedoch, wie in Abschnitt 2.3 erwähnt, ein leistungsfähigeres System verwendet.
Daraus ergibt sich das Potenzial für zusätzliche Optimierungen der Laufzeit. So unterstützt das im Fahrzeug verbaute System die Möglichkeit einer effizienteren Berechnung für Gleitkommazahlen (FP16), was auf der im Experiment eingesetzten Hardware nicht möglich war. Diese Optimierungsmöglichkeiten können zu signifikanten Leistungssteigerungen führen [26].
Ein weiterer Laufzeitvorteil ergibt sich daraus, dass im Live-System die Bilddaten direkt von der Kamera in die Berechnungsschritte geladen werden, ohne zusätzliche Festplattenzugriffe. Auch dies wird die Gesamtlatenz weiter reduzieren und somit die Echtzeitfähigkeit des Systems verbessern.

Insgesamt lässt sich schlussfolgern, dass das Nano-Modell mit zugeschnittenen Bildern in halber Auflösung die besten Ergebnisse erzielt und die Anforderungen von 20Hz mit einer Frequenz von 26.27Hz bei gleichzeitig exzellenter Detektionsqualität mit einer mAP von 0.945 übertrifft. Zum Vergleich, das in [7] entwickelte neuronale Netz zur Pylonenerkennung, welches ebenfalls auf einem Monokamera-System basiert, erreicht im Schnitt eine mAP von 0.77 auf deren Datensatz.

3.4.3 Erkennungsleistung in Abhängigkeit zur Distanz

Da in der Literatur vereinzelt berichtet wird, dass YOLO Schwierigkeiten bei der Erkennung kleiner Objekte hat [16], [2], und Pylonen durchaus als solche eingestuft werden können, wurde zusätzlich eine Evaluation der Erkennungsleistung über verschiedene Distanzen hinweg durchgeführt. Für diese Analyse wurde die zuvor als optimal identifizierte Konfiguration verwendet: die Mainhoop-Kamera in der zugeschnittenen Variante bei halber Auflösung, ausgewertet mit dem Nano-Modell. Die Bilder wurden dabei in drei Distanzbereiche unterteilt:

Abbildung 3.14: Distanzgruppen der Evaluation am Beispiel der Mainhoop Kamera.

Tabelle 3.5: Mean Average Precision unterteilt pro Distanzgruppe.

Distanzgruppe	mAP
Weit	0.91
Mittel	0.99
Nah	0.98

- Nah (0-4 Meter)
- Mittel (4-7 Meter)
- Weit (7 Meter oder mehr)

Diese Einteilung wurde gewählt, da sich damit das Bild in drei gleichmäßige Teile unterteilen lässt (siehe Abbildung 3.14). Die Distanzen wurden dabei mithilfe der in Kapitel 4 beschriebenen Kamerakalibrierung geschätzt und sind relativ zur Hinterachse des Fahrzeuges.

Auf diesen Distanzgruppen wurde anschließend die in Unterabschnitt 3.4.1 beschriebene Evaluationsmethode zur Bestimmung der mAP angewandt. Die Ergebnisse sind in Tabelle 3.5 zu sehen.

Es zeigt sich ein klarer Zusammenhang zwischen der Erkennungsleistung des Netzes und der Entfernung der Objekte: In der Gruppe der weit entfernten Pylonen ist die Mean Average Precision um 0.07 niedriger als bei nahen und mittelweit entfernten Objekten. Somit lässt sich festhalten, dass die Erkennungsleistung bei kleineren Objekten tatsächlich abnimmt, wobei eine mAP von 0.91 weiterhin als sehr gutes Ergebnis bewertet werden kann.

4 Entwicklung einer Kamerakalibrierung zur Distanzberechnung

Dieses Kapitel beschreibt, wie die Kamera zur Distanzberechnung der Pylonen genutzt werden kann. Grundlegendes Wissen über Kameras und Bildverarbeitung wird dabei vorausgesetzt und kann hier [27] nachgelesen werden.

4.1 Grundlagen

Zur Bestimmung der Pylonenabstände relativ zum Fahrzeug wird das Konzept der Homographie genutzt. Dieses ist deutlich ressourcenschonender als das in [7] verwendete keypoint regression System mit anschließender Perspective-n-Point Berechnung, da es sich um eine 2D-lineare Transformation handelt, welche Punkte einer Ebene in eine andere Ebene überführt [28]. In [7] werden dahingegen zusätzliche neuronale Netze für die keypoint regression verwendet sowie eine aufwendige 3D-Position der Objekte berechnet.

Die Homographie lässt sich in diesem Anwendungsfall einsetzen, da alle Pylonen auf derselben Bodenebene liegen. Die Homographie-Matrix beschreibt dabei die perspektivische Abbildung zwischen der Bildebene der Kamera und der realen Bodenebene. Durch diese Transformation lassen sich die Pixelkoordinaten detektierter Pylonen in Koordinaten der Bodenebene überführen, woraus sich der Abstand zum Fahrzeug berechnen lässt. Der Vorteil dieser Methode liegt darin, dass die dreidimensionale Kameraperspektive effektiv auf eine zweidimensionale Ebene reduziert wird, was die Komplexität im Vergleich zu den in [7] beschriebenen Tiefenrekonstruktionsverfahren deutlich verringert. Sie ist jedoch nur gültig, solange die Bodenfläche als planar angenommen werden kann. Diese Voraussetzung ist im Kontext der Formula Student Driverless gegeben.

Mathematisch gesehen wird eine Homographie durch eine 3×3-Matrix dargestellt, die die Beziehung zwischen den homogenen Koordinaten korrespondierender Punkte in zwei Bildern beschreibt. Diese Matrix kann berechnet werden, wenn mindestens vier Punktkorrespondenzen zwischen den Bildern bekannt sind [29].

Die praktische Umsetzung erfolgt mit der Open-Source-Bibliothek für Computer Vision (OpenCV), welche optimierte Algorithmen für Homographie-Berechnungen

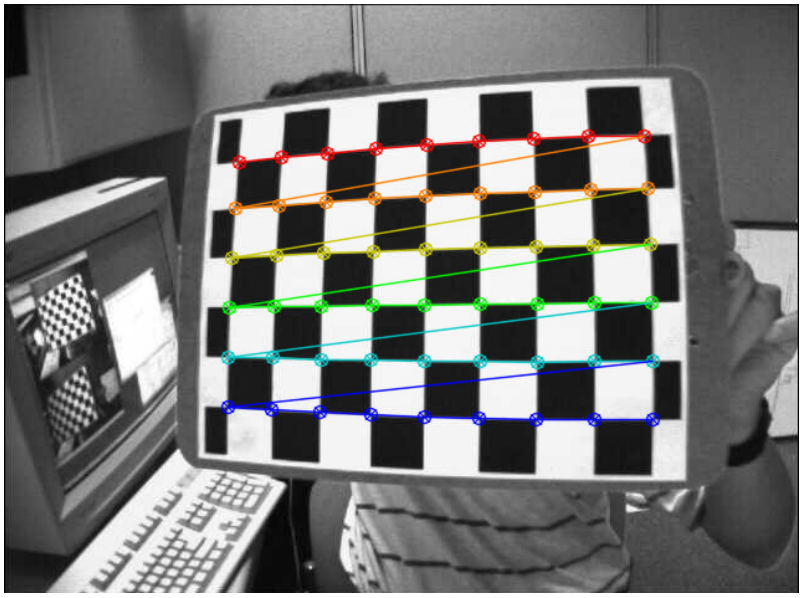

Abbildung 4.1: Beispiel einer Schachbrettmusterplatte, welche zur Kamerakalibrierung genutzt wird (aus [30]).

bereitstellt. OpenCV ist eine weit verbreitete Bibliothek und bietet unter anderem effiziente Implementierungen für Bildverarbeitung, Objekterkennung und geometrische Transformationen.

4.2 Konzept

Das Ziel in diesem Teil ist es, den Abstand der erkannten Pylonen zum Fahrzeug über die Kamera zu errechnen. Hierfür wird zunächst eine Kalibrierung durchgeführt, um eine projektive Abbildung zwischen der 3D-Umgebung und der 2D-Bildebene der Kamera zu modellieren. Als Kalibrierungsobjekt dient eine Schachbrettmuster-Platte (siehe Abbildung 4.1), deren bekannte 3D-Punktkoordinaten (in der realen Welt) und detektierte 2D-Bildkoordinaten genutzt werden. Mit diesen Punkten wird dann über die OpenCV-Funktion `cv2.findHomography()` die Homographie-Matrix H berechnet, die die Transformation zwischen der Bodenebene und der Bildebene beschreibt.

Nach der Kalibrierung können die Pixelkoordinaten detektierter Pylonen in weiteren Bildern mittels der inversen Homographie-Matrix H^{-1} in die Referenzebene des Kalibrierungsbildes transformiert werden. Aus den resultierenden Koordinaten lässt sich nun der horizontale Abstand zum Fahrzeug berechnen (siehe Abbildung 4.2).

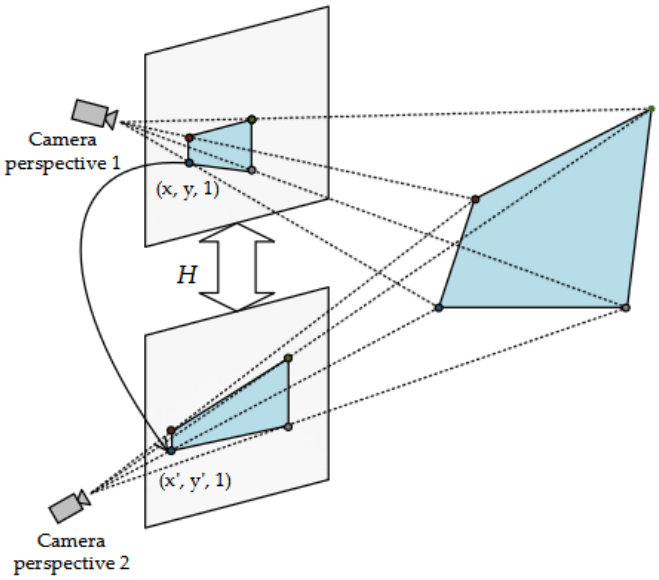

Abbildung 4.2: Schematisches Diagramm einer Homographie Transformation (aus [28]). Zu sehen ist, wie mithilfe einer Homographie die Bildpunkte zweier Bilder auf die selben Punkte der Umgebung transformiert werden.

Für zuverlässige Messungen ist dabei die Konstanz der Kameraposition entscheidend, da bereits geringe Veränderungen der Kamerahöhe oder des Neigungswinkels zu starken Abweichungen führen kann. Aus diesem Grund muss die Kamera fest am Fahrzeug montiert werden.

4.3 Umsetzung

Der erste Schritt zur Vorbereitung des Systems besteht in der Kalibrierung der Kamera. Zu diesem Zweck wird die zuvor erwähnte Schachbrettplatte so vor dem Fahrzeug positioniert, dass möglichst viele Quadrate sichtbar sind. Dies ermöglicht die Definition von Referenzpunkten im Bildraum, die später zur Berechnung der Homographiematrix herangezogen werden. Zusätzlich können weitere Referenzpunkte, beispielsweise in Form von Pylonen, ergänzt werden. Diese erweitern die geometrische Verteilung der Punkte insbesondere in Randbereichen des Bildes und verbessern dadurch die Genauigkeit der Homographie. Idealerweise sind die Referenzpunkte gleichmäßig über das gesamte Bild verteilt, um eine möglichst präzise Transformation zu gewährleisten. In den durchgeführten Tests konnte dies lediglich für die Kamera am Frontwing realisiert werden, da diese über den engsten Bildwinkel verfügt. Aus Gründen der Vereinfachung wurden die Pylonen für die Kalibrierung der

Abbildung 4.3: Vergleich der Referenzpunktverteilung zwischen der Kamera im Frontwing (links) und der Kamera am Engine Cover (rechts).

übrigen Kameras an denselben Positionen belassen, wodurch die Referenzpunktverteilung dort weniger optimal ausfiel (siehe Abbildung 4.3).

Für die Berechnung der Abstände der Pylonen wurde die Mitte der Hinterachse des Fahrzeuges als Bezugspunkt gewählt, da sie die Bewegungsbasis des Fahrzeugs darstellt. Um diese Berechnung umzusetzen, wird zunächst ein Koordinatensystem im Raum definiert mit Ursprung in der Mitte der Schachbrettplatte. Dieses Koordinatensystem wird im Folgenden mit S_w bezeichnet. Anschließend wird das Fahrzeug so positioniert, dass eine der Achsen von S_w exakt entlang der Mittellinie der Hinterachse verläuft (siehe Abbildung 4.4) Diese Ausrichtung muss mit höchster Präzision erfolgen, da bereits geringe Abweichungen signifikante Fehler in der späteren Abstandsschätzung verursachen können.

Für eine korrekte Homographieberechnung ist es essenziell, die Koordinaten aller Referenzpunkte sowie die Kamerapositionen im Koordinatensystem S_w exakt zu dokumentieren, am besten in Metern. Sobald diese Voraussetzungen erfüllt sind, wird für jede Kamera ein Kalibrierungsbild aufgenommen, auf dessen Basis anschließend die jeweilige Homographiematrix berechnet werden kann.

Für die Berechnung der Homographie-Matrix müssen alle verfügbaren Referenzpunkte eingelesen werden. Dabei werden die Bildkoordinaten der Referenzpunkte ihren jeweiligen Weltkoordinaten im Koordinatensystem S_w zugeordnet. Dieser Matching-Prozess erfolgt mithilfe eines Python-Skripts, welches eine manuelle Zuordnung ermöglicht. Durch Anklicken der Referenzpunkte im Kalibrierungsbild werden deren Bildpixelkoordinaten erfasst und mit den bekannten Weltkoordinaten verknüpft. Die so erzeugten Zuordnungen werden anschließend in einer Datei gespeichert und für die Berechnung der Homographie-Matrix verwendet (siehe Abbildung 4.5).

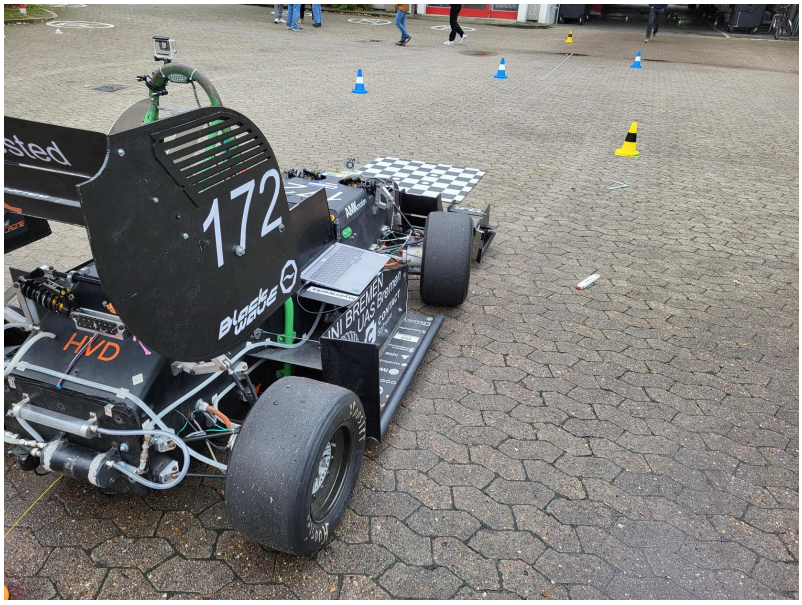

Abbildung 4.4: Ausrichtung des Fahrzeugs und der Referenzpunkte während der Kamerakalibrierung.

Abbildung 4.5: Script zum Einlesen der Kalibrierungsdaten, hier am Beispiel der Engine Cover-Kamera. Durch Klicken setzt man ein grünes Kreuz. Jedes grüne Kreuz entspricht einem Bild-Referenzpunkt, für welchen Realwelt-Koordinaten vorliegen.

Nachdem jedem Referenzpunkt erfolgreich ein entsprechender Bildpixelpunkt zugeordnet wurde, können diese Punktpaare an die OpenCV-Bibliothek übergeben werden. OpenCV berechnet daraus die Homographie-Matrix, die im Folgenden als $H_{S \leftarrow B}$ bezeichnet wird. Diese Matrix transformiert Bildkoordinaten in das Weltkoordinatensystem S_w, welches auf der Schachbrettplatte definiert ist. Da jedoch die Pylonenpositionen relativ zur Hinterachse des Fahrzeuges bestimmt werden sollen, muss ein weiteres Koordinatensystem mit Ursprung an dieser Stelle erstellt werden. Dieses wird im Weiteren mit F_w bezeichnet. Nun wird eine Transformationsmatrix $T_{F \leftarrow S}$ berechnet, welche die Koordinaten aus dem System S_w in das fahrzeugbezogene Koordinatensystem F_w überführt. Die resultierende Homographiematrix $H_{F \leftarrow B}$, welche Bildpunkte direkt in Fahrzeugkoordinaten transformiert, ergibt sich durch folgende Matrixmultiplikation:

$$H_{F \leftarrow B} = T_{F \leftarrow S} \cdot H_{S \leftarrow B}$$

Da mit homogenen Koordinaten gearbeitet wird, besitzt die Transformationsmatrix $T_{F \leftarrow S}$ zunächst die allgemeine Form einer 4×4-Matrix:

$$\begin{pmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Da sich alle relevanten Objekte auf einer gemeinsamen Ebene, dem Boden, befinden, entfällt die Abhängigkeit von der z-Koordinate. Die Transformation kann somit auf eine 3×3-Matrix reduziert werden:

$$\begin{pmatrix} 1 & 0 & x \\ 0 & 1 & y \\ 0 & 0 & 1 \end{pmatrix}$$

Die spezielle Form dieser Matrix ergibt sich daraus, dass das Koordinatensystem F_w so gewählt wurde, dass seine Achsen in dieselbe Richtung zeigen, wie die Achsen von S_w. Das Koordinatensystem F_w ist somit lediglich gegenüber S_w translatiert, ohne zusätzliche Rotation oder Skalierung. Weiterhin wurde durch die zuvor beschriebene präzise Ausrichtung des Fahrzeugs sichergestellt, dass der Ursprung von F_w auf einer Achse von S_w liegt, wodurch die Verschiebung entlang dieser Achse, hier die y-Achse, auf Null gesetzt werden kann. Die finale Transformationsmatrix $T_{F \leftarrow S}$ vereinfacht sich damit zu:

$$T_{F \leftarrow S} = \begin{pmatrix} 1 & 0 & x \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

‚wobei x die zu Beginn notierte Distanz des Ursprungs von F_w zum Ursprung von S_w beschreibt. Somit ergibt sich die finale Homographiematrix $H_{F \leftarrow B}$ durch:

$$H_{F \leftarrow B} = \begin{pmatrix} 1 & 0 & x \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot H_{S \leftarrow B}$$

Mit dieser Matrix können nun beliebige Bildpunkte in Realwelt-Koordinaten in Metern (x, y) relativ zur Hinterachse des Fahrzeugs berechnet werden.

4.4 Evaluation

Für die Evaluation wird erneut auf die in Abschnitt 3.4 beschriebenen Aufnahmen des Testlaufs Bezug genommen. Ziel ist es, die verschiedenen Kamerapositionen erneut miteinander zu vergleichen, um zu ermitteln, welche Position die höchste Präzision bei der Abstandsmessung ermöglicht. Das Vorgehen bei der Analyse ist wie folgt:

Zunächst werden die Bilder in drei Distanzgruppen unterteilt (jeweils relativer Abstand zur Hinterachse des Fahrzeuges):

- Nah (0-4 Meter)
- Mittel (4-7 Meter)
- Weit (7 Meter oder mehr)

Diese Einteilung geschieht analog zu Unterabschnitt 3.4.3 und wurde auch für diese Evaluation gewählt, da die Mainhoop Kamera den größten Winkel zum Boden hat und damit die Distanzen am gleichmäßigsten unterteilen kann (siehe Abbildung 3.14).

Wie bereits in Abschnitt 3.4 erläutert, wurde die Teststrecke gemäß den Regularien der Formula Student aufgebaut [9]. Dabei wurde zugunsten der Evaluation darauf geachtet, dass der Abstand zwischen blauen und gelben Pylonen nicht wie laut Regelwerk minimum, sondern konstant 3 Meter beträgt. Basierend auf den Testaufnahmen werden nun Paare aus blauen und gelben Pylonen ausgewählt, zwischen denen eine Abstandsmessung durchgeführt wird.
Zur Bestimmung des Abstands werden zunächst die jeweils fußnächsten Punkte der beiden Pylonen per Hand extrahiert. Diese Referenzpunkte werden anschließend mithilfe der zuvor berechneten Homographie-Matrix in das Fahrzeugkoordinatensystem transformiert. Aus den transformierten Koordinaten wird schließlich der Betrag

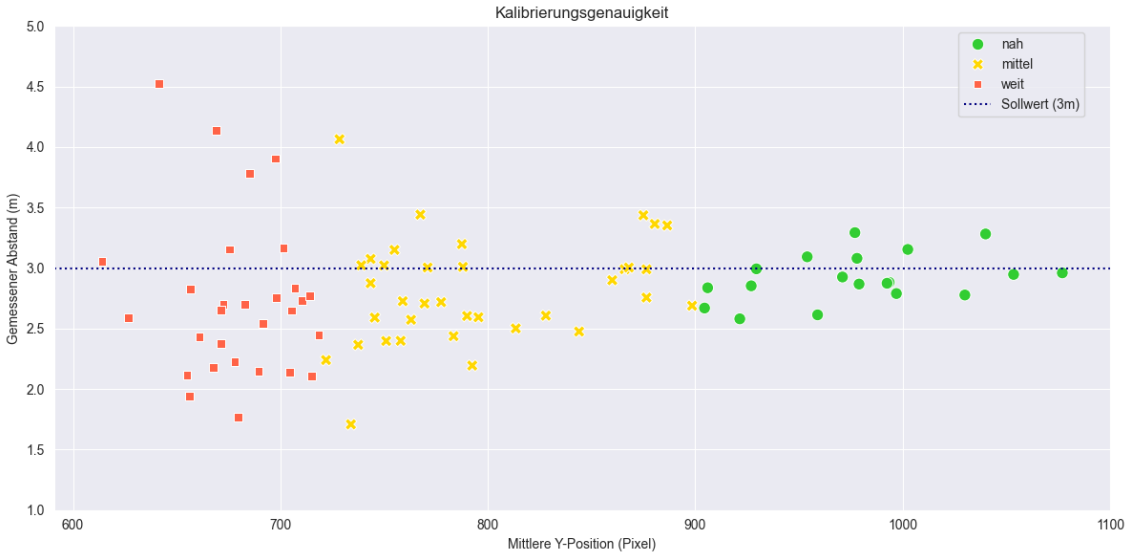

Abbildung 4.6: Scatterplot der Abstandsmessung auf der Mainhoop Kamera.

des Verbindungsvektors berechnet, wodurch der Abstand zwischen den Pylonen bestimmt werden kann. Zudem wird das Pylonen-Paar, basierend auf dem Mittelwert der y-Bildkoordinaten beider Referenzpunkte, jeweils einer Distanzgruppe zugeordnet. Insgesamt werden so ca. 25 Pylonen-paare pro Distanzgruppe selektiert.

Die Auswertung der gemessenen Abstände zeigt deutlich, dass die Präzision der Abstandsmessung mit zunehmender Distanz abnimmt (siehe Abbildung 4.6). Die Abweichungen reichen dabei von wenigen Zentimetern im Nahbereich bis hin zu etwa einem Meter bei Pylonen, die weiter als 7 Meter vom Fahrzeug entfernt sind (siehe Abbildung 4.7). Dieses Verhalten ist erwartungsgemäß, da weiter entfernte Objekte im Bild durch eine geringere Anzahl an Bildpixeln dargestellt werden. Beispielsweise liegen zwischen 8m und 9m weniger Bildpixel als zwischen 3m und 4m, was zu einer abnehmenden Genauigkeit bei der Distanzberechnung führt. Trotz dieser Einschränkungen zeigt die Methode insgesamt eine solide Leistung: Für Pylonen mit einer Entfernung von bis zu 7m liegt der Median des Messfehlers bei lediglich -0.26m (siehe Abbildung 4.7). An dieser Stelle und im Folgenden wurde der Median als Evaluationsmetrik gewählt, da nicht alle Gruppen der Evaluation eine Normalverteilung aufweisen (siehe Abbildung 4.8). In solchen Fällen ist der Median gegenüber Ausreißern robuster und stellt somit eine geeignetere Wahl dar.

Vergleicht man die Ergebnisse der Mainhoop-Kamera mit denen der anderen Kamerapositionen, so zeigen diese eine deutlich schlechtere Präzision bei der Abstandsmessung. Bei der Engine Cover-Kamera beträgt der Median des Messfehlers für weit entfernte Pylonen rund -0.4m (siehe Abbildung 4.9). Noch gravierender ist die Abweichung bei der Frontwing-Kamera: Hier liegt der Median des Messfehlers bei

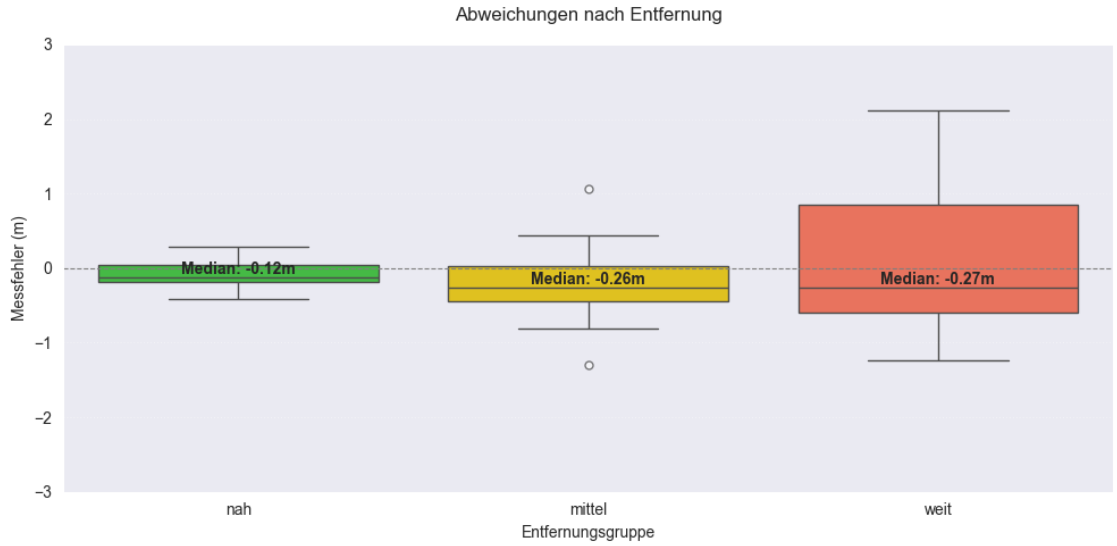

Abbildung 4.7: Boxplot des Messfehlers bei der Abstandsmessung auf der Mainhoop Kamera.

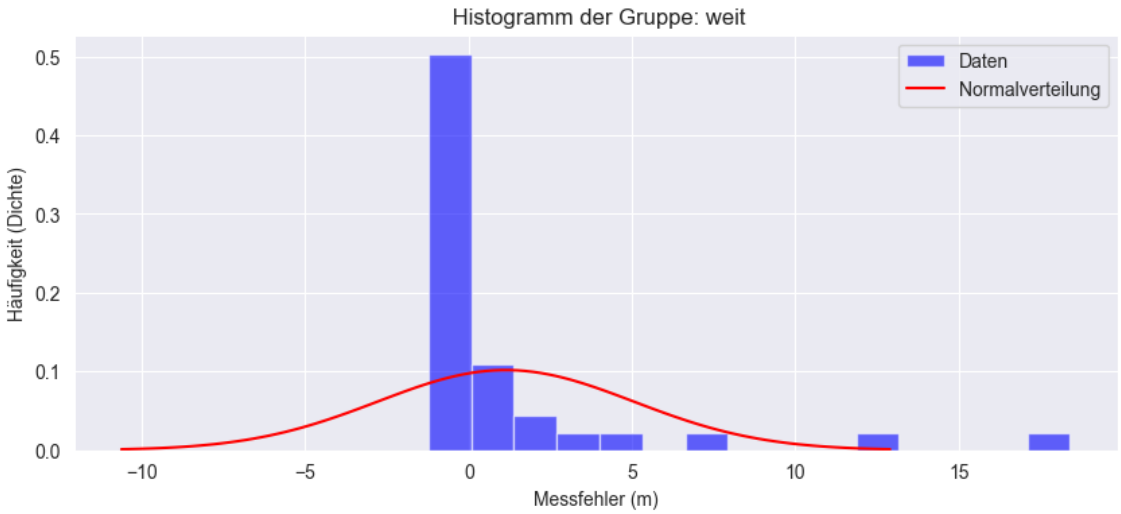

Abbildung 4.8: Histogramm des Messfehlers bei der Abstandsmessung auf der Mainhoop Kamera für die Gruppe „weit“. Diese Daten sind nicht normalverteilt.

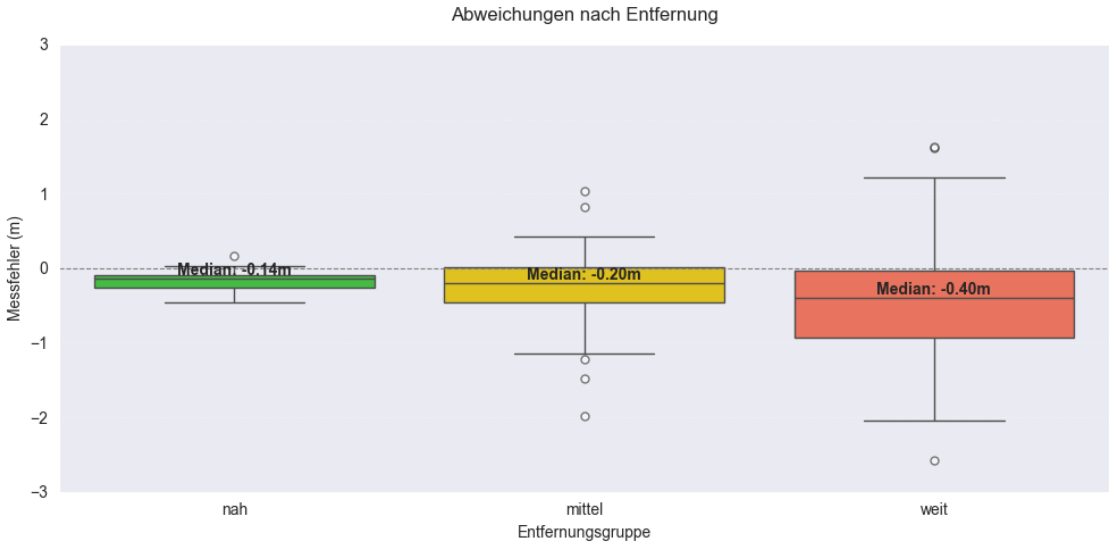

Abbildung 4.9: Boxplot des Messfehlers bei der Abstandsmessung auf der Engine Cover Kamera.

weit entfernten Pylonen bei 0.72m, wobei einige Messungen um mehrere Meter vom tatsächlichen Abstand abweichen (siehe Abbildung 4.10). Diese geringere Genauigkeit lässt sich auf den deutlich kleineren Winkel zwischen Kameraachse und Boden zurückführen, der bei tiefer montierten Kameras entsteht. Ein niedriger Blickwinkel führt nicht nur zu häufigeren Verdeckungen, wie bereits in Abschnitt 3.4 festgestellt, sondern verringert auch die vertikale Bildausdehnung von entfernten Objekten. Dadurch werden Distanzunterschiede auf weniger Bildpixel projiziert, was die Genauigkeit der Homographie-basierten Abstandsmessung stark reduziert. Dies zeigt sich insbesondere in der Analyse der Distanzverteilung (siehe Abbildung 4.11). Deshalb existieren auch für die Frontwing Kamera im Testlauf keinerlei Aufnahmen, bei denen sich ein Pylonenpaar im Bereich von 0–4 Metern Entfernung befindet.

Damit steht fest, dass auch für die Kamerakalibrierung die Position der Mainhoop-Kamera die besten Ergebnisse liefert. Eine genauere Analyse dieser Kamera offenbart zudem einen Zusammenhang zwischen der x-Position der Pylonen im Bild und dem jeweiligen Messfehler (siehe Abbildung 4.12). Konkret bedeutet das, je näher ein Pylonenpaar am Bildrand liegt, desto größer fällt der Messfehler aus. Für dieses Verhalten lasse sich zwei Hauptursachen identifizieren:

1. Wie Abschnitt 4.3 erläutert, verlief die Verteilung der Referenzpunkte zur Kalibrierung nicht optimal auf dieser Kamera, insbesondere im Randbereich des Bildes. Dadurch entstehen dort größere Umrechnungsfehler bei der Anwendung der Homographie.

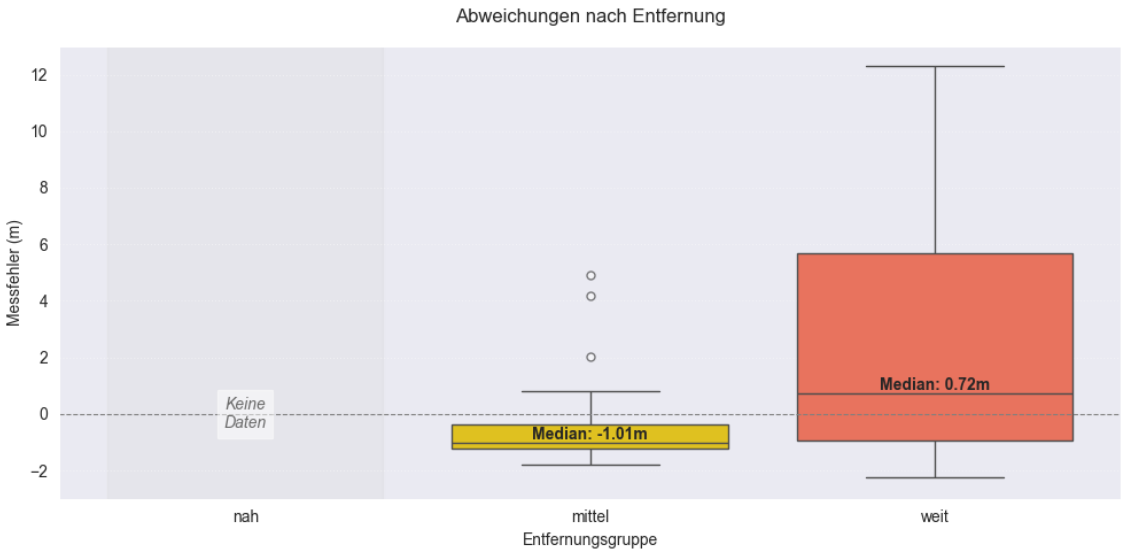

Abbildung 4.10: Boxplot des Messfehlers bei der Abstandsmessung auf der Frontwing Kamera.

Abbildung 4.11: Distanzgruppen der Evaluation am Beispiel der Frontwing Kamera.

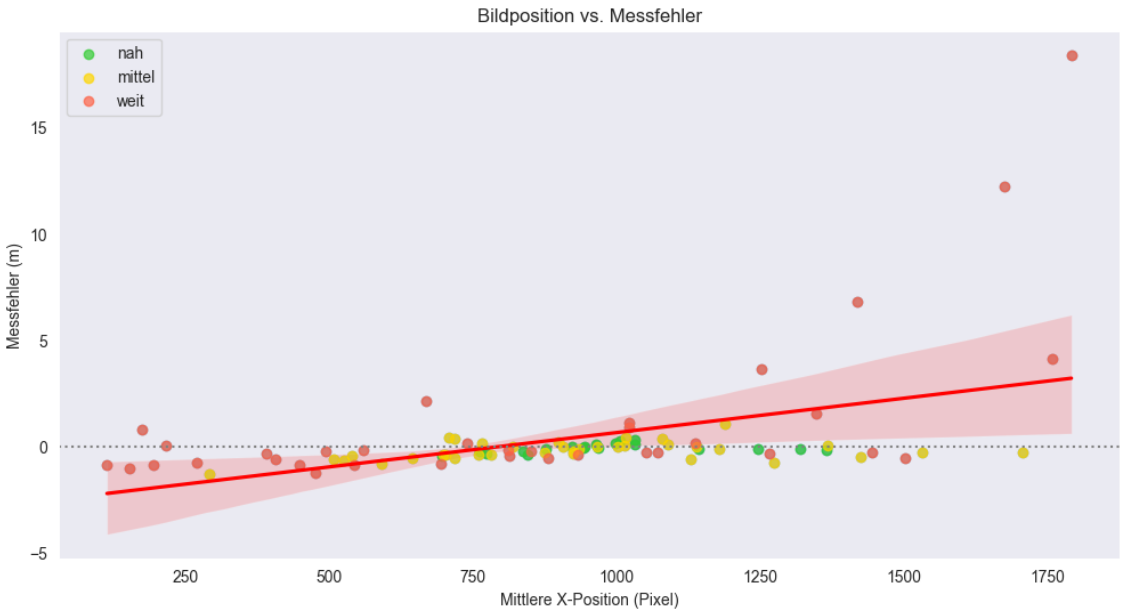

Abbildung 4.12: Verhältnis zwischen der X-Position im Bild und des Messfehlers bei der Abstandsmessung auf der Mainhoop Kamera.

2. Pylonenpaare am Bildrand befinden sich häufig in Kurven und stehen daher eher hinter- statt nebeneinander. In solchen Fällen liegt ein größerer Anteil des Abstands in y-Richtung vor, wodurch, wie bereits beschrieben, die Genauigkeit der Umrechnung abnimmt.

Insgesamt lässt sich schlussfolgern, dass ein Median des Messfehlers von -0.27m bei Pylonen über 7 Meter entfernt vom Fahrzeug, wie er auf der Mainhoop Kamera beobachtet wurde, ein solides Ergebnis darstellt. Die Position der Kamera am Mainhoop liefert auch für dieses Kapitel die besten Ergebnisse.

5 Implementierung des Gesamtsystems

Dieses Kapitel beschreibt, wie die entwickelten Systeme zusammengeführt und für den Einsatz auf dem Fahrzeug vorbereitet wurden.

5.1 Grundlagen

Die Software der Driverless Pipeline wird typischerweise in Middleware-Systemen realisiert, da diese eine effiziente Kommunikation zwischen den einzelnen Komponenten ermöglichen. Ein in der Formula Student weit verbreitetes System ist das Robot Operating System (ROS). ROS ist eine Open-Source-Softwareplattform zur Entwicklung von Robotersystemen, die unter anderem Werkzeuge für Nachrichtenübertragung, Hardware-Abstraktion und Prozessmanagement bereitstellt. Aufgrund seiner Flexibilität und umfangreichen Bibliotheken findet ROS in vielen Bereichen Anwendung, darunter in der Robotikforschung, industriellen Automatisierung oder autonomen Fahrzeugen [31]. Auch in der Formula Student wird es von vielen Teams eingesetzt [3], [5], [6], so auch von Bremergy. Dabei verwendet Bremergy jedoch nicht die ursprüngliche Version, sondern das modernere Nachfolgesystem ROS2, welches unter anderem bessere Implementierungen für Echtzeitanwendungen und Sicherheit aufweist [31].

Grundlegend funktioniert ROS2 so, dass komplexe Systeme in einzelne sogennante Nodes unterteilt werden. Jede Node übernimmt dabei eine klar definierte Aufgabe innerhalb des Gesamtsystems, beispielsweise das Einlesen von Sensordaten oder die Steuerung von Aktuatoren. Die Kommunikation zwischen diesen Nodes erfolgt über sogenannte Topics. Nodes können dabei entweder Informationen auf ein Topic veröffentlichen (Publisher) oder Informationen von einem Topic abonnieren (Subscriber). Dieses Publish-Subscribe-Prinzip bildet die zentrale Kommunikationsstruktur von ROS2 und ermöglicht eine entkoppelte, modulare Systemarchitektur. Weitere Details lassen sich der offiziellen ROS2-Dokumentation entnehmen [32].

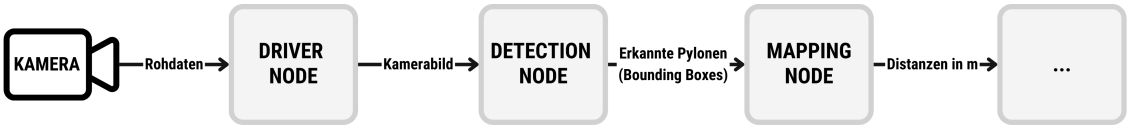

Abbildung 5.1: Konzept des Gesamtsystems realisiert in ROS2.

5.2 Konzept

Um das entwickelte System in die ROS2-Architektur zu überführen, werden Nodes aus den jeweiligen Funktionseinheiten der Anwendung konstruiert (siehe Abbildung 5.1). Die Kamera ist dabei direkt mit dem System verbunden und sendet zunächst Rohdaten an eine Driver Node. Diese verarbeitet die Daten und wandelt sie in ein gängiges Format um, in diesem Fall YUYV. Das resultierende Kamerabild wird anschließend an eine Detection Node übergeben, welche die Erkennung der Pylonen mithilfe des in Kapitel 3 beschriebenen neuronalen Netzes übernimmt. Die Bounding Boxes der erkannten Pylonen werden schließlich an eine Mapping Node weitergeleitet, welche mithilfe der in Kapitel 4 berechneten Homographie-Matrix die Distanzen relativ zur Hinterachse des Fahrzeugs in Metern bestimmt. Diese Werte können im weiteren Verlauf des Systems verwendet werden.

5.3 Umsetzung

Im ersten Schritt geht es darum, die Driver Node zu erstellen. Viele Sensorhersteller stellen bereits vorgefertigte ROS2-kompatible Treiber bereit und so ist auch der Fall für die RealSense-Kamera. Die entsprechende Node ist unter [33] verfügbar und nach einer initialen Einrichtung sofort einsatzbereit.

Bei der Detection Node wird zunächst das Kamerabild von der Driver Node empfangen und in das BGR-Format umgewandelt. Dies ist notwendig, da YOLO nicht mit dem YUYV-Format umgehen kann. Anschließend wird das Bild auf die in Unterabschnitt 3.4.2 ermittelte optimale Auflösung zugeschnitten und daraufhin erfolgt die Pylonenerkennung mithilfe des in Kapitel 3 trainierten Nano-Modells. Für jedes Bild werden alle erkannten Bounding Boxes gesammelt und anschließend auf ein Topic veröffentlicht, sodass sie von der Mapping Node empfangen und weiterverarbeitet werden können.

Die Mapping Node nimmt schlussendlich die Bounding Boxes und führt mithilfe der in Kapitel 4 errechneten Homographie-Matrix die Distanzberechnung durch. Als Referenzpunkte werden dabei jeweils die unteren Ecken der Bounding Boxes gewählt, die sich näher zum Fahrzeug befinden. Für Pylonen der Klasse *Blue Cone* ist dies die

untere rechte Ecke, da diese Pylonen im Streckenverlauf stets auf der linken Fahrzeugseite platziert sind. Analog dazu wird für Pylonen der Klasse *Yellow Cone* die untere linke Ecke herangezogen, da diese auf der rechten Fahrzeugseite positioniert sind. Bei Pylonen der Klasse *Large Orange Cone* und *Orange Cone* wird die Mitte der unteren Kante der Bounding Box als Referenzpunkt gewählt, da diese Pylonen sowohl auf der linken als auch auf der rechten Seite stehen können. Die berechneten Distanzen werden abschließend auf einem separaten Topic veröffentlicht, sodass sie vom restlichen System weiterverwendet werden können.

Ein Anwendungsbeispiel des Gesamtsystems ist in Abbildung 5.2 dargestellt, wobei die finalen Distanzen in einem Punktdiagramm visualisiert werden.

5.4 Evaluation

Zur Evaluation des Gesamtsystems wird erneut eine Abstandsmessung durchgeführt. Das Vorgehen orientiert sich stark an Abschnitt 4.4, mit dem Unterschied, dass die Bilder nun zunächst vom neuronalen Netz verarbeitet werden. Als Referenzpunkte für die Abstandsmessung dienen dabei Punkte auf dem Rahmen der vorhergesagten Bounding Boxes. Die Auswahl dieser Punkte erfolgt wie zuvor beschrieben: Für blaue Pylonen wird die untere rechte Ecke verwendet, für gelbe Pylonen die untere linke Ecke und für orangene Pylonen die Mitte der unteren Kante. Die Analyse wird ausschließlich auf der in den vorherigen Kapiteln identifizierten besten Konstellation durchgeführt: Die Mainhoop Kamera mit zugeschnittenem Bild in halbierter Auflösung, ausgewertet mit dem Nano-Modell.

In den Ergebnissen zeigt sich, dass das Gesamtsystem um ca. 0,13m unpräziser ist als die manuelle Referenzpunktwahl aus Abschnitt 4.4, gemittelt über alle Klassen hinweg (siehe Abbildung 5.3, Abbildung 4.7). Während der gemittelte absolute Median des Messfehlers bei der manuellen Bestimmung der Pylonenfußpunkte bei 0,22m lag, beträgt er im Gesamtsystem 0,35m. Diese Abweichung lässt sich darauf zurückführen, dass die ermittelten Bounding Boxes nicht exakt mit dem tatsächlichen Fußpunkt der Pylonen übereinstimmen, wodurch sich ein systematischer Fehler ergibt.

Bemerkenswert ist jedoch, dass der Median des Messfehler selbst bei über 7 Meter entfernten Pylonen bei lediglich 0.43m liegt. Zum Vergleich: Das Team AMZ berichtet in seiner Softwarelösung über Messfehler von mehr als 0.5m bei Pylonen in einer solchen Entfernung [5]. In [7] werden ebenfalls Messfehler um die 0.5m in solch einer Entfernung berichtet, dabei basieren beide Lösungen auf deutlich komplexeren Systemen.

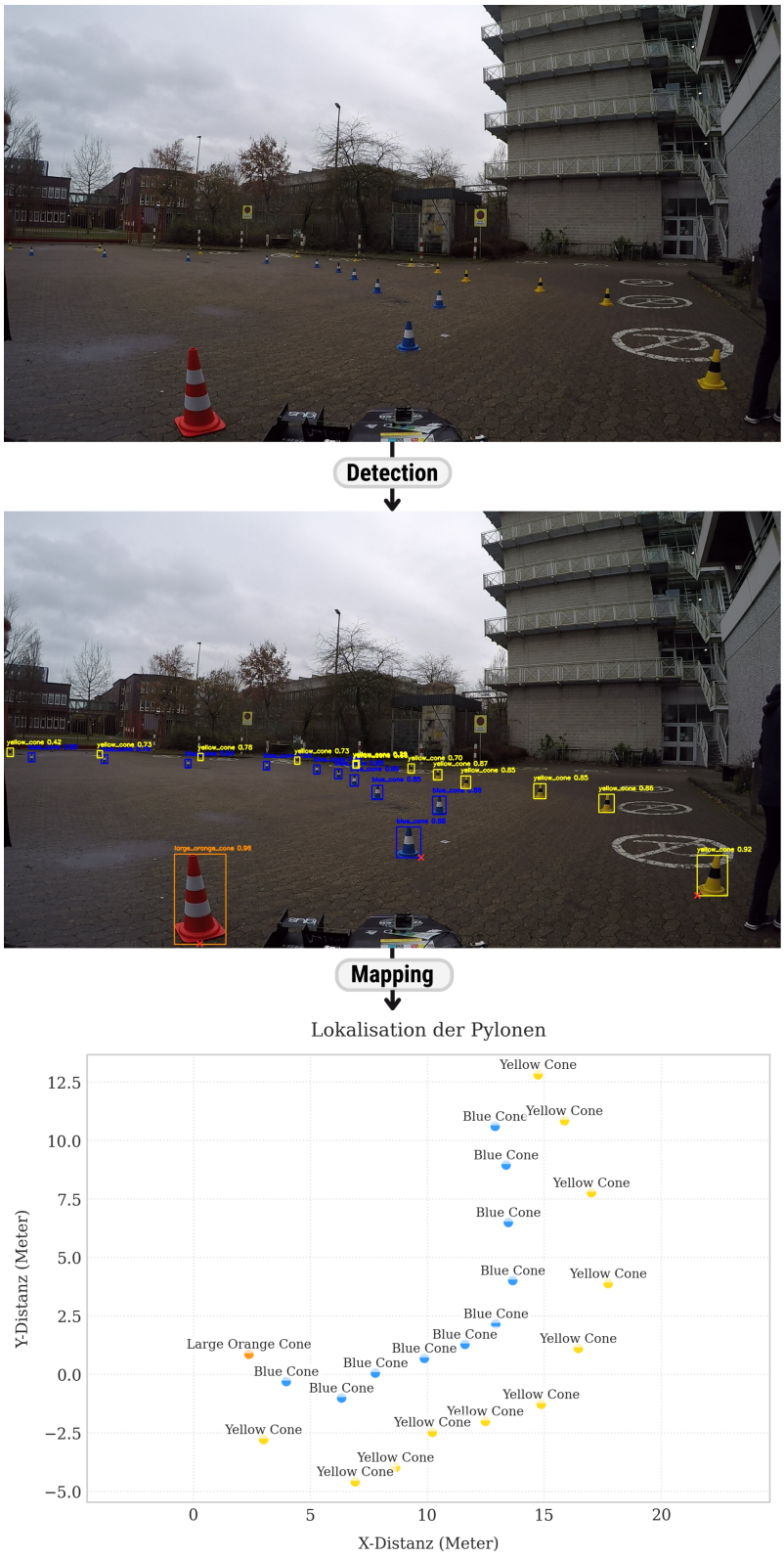

Abbildung 5.2: Anwendungsbeispiel der Gesamtsystems. Beispiele der Selektion der Referenzpunkte sind mit roten kreuzen dargestellt. Die Hinterachse des Fahrzeuges bildet den Ursprung des Punktdiagramms.

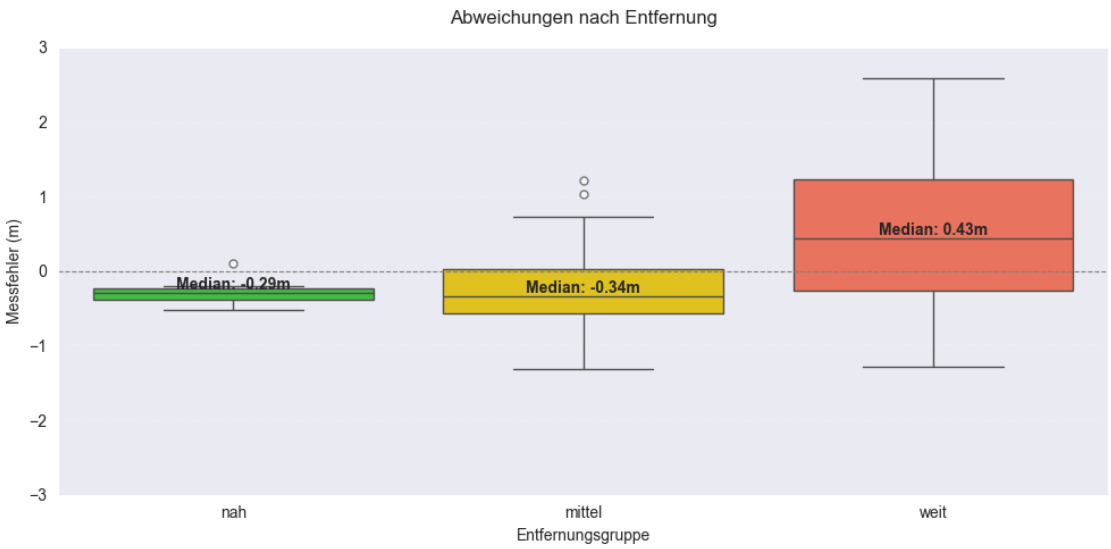

Abbildung 5.3: Boxplot des Messfehlers bei der Abstandsmessung auf dem Gesamtsystem.

Es ist jedoch zu beachten, dass sich die dort angegebenen Fehler auf die Distanz zwischen Fahrzeug und Pylon bezieht, während in dieser Arbeit der Abstand zwischen Pylonenpaaren bewertet wurde. Würde der Fehler zum Fahrzeug hin analog zu [5], [7] bestimmt werden, wäre er vermutlich ebenfalls größer, da dieser Abstand primär entlang der y-Bildachse gemessen wird, welche, wie zuvor beschrieben, anfälliger für Ungenauigkeiten ist als die x-Achse.

Trotz dieser Einschränkungen lässt sich festhalten, dass ein absoluter mittlerer Messfehler von 0,35m über alle Distanzgruppen hinweg ein sehr solides Ergebnis darstellt, insbesondere angesichts der minimalistischen Systemarchitektur mit lediglich einer Kamera. Diese Genauigkeit erscheint ausreichend, um eine autonome Fahrzeugführung zu ermöglichen. Das ist jedoch nur eine Einschätzung und bedarf einer experimentell Validierung, sobald der Bremo25 einsatzbereit ist.

6 Zusammenfassung und Ausblick

In dieser Arbeit wurde ein minimalistisches Monokamera Perzeptionssystem für Bremergy zur Teilnahme am Driverless Cup in der Formula Student entwickelt. Basierend auf YOLOv11 wurde ein neuronales Netz zur Pylonenerkennung trainiert und eine Kamerakalibrierung mittels Homographie zur Distanzberechnung implementiert. Das Gesamtsystem wurde anschließend erfolgreich in die bestehende Architektur von Bremergy integriert, wodurch es auf dem Fahrzeug eingesetzt werden kann. Dies ist ein fundamentaler Baustein im System von Bremergy und stellt damit einen bedeutender Schritt für die Wettbewerbsfähigkeit des Teams dar.

Zur Evaluation wurde ein automatisiertes Skript entwickelt, das Modellvorhersagen mit Ground-Truth-Daten vergleicht. Zudem wurde die optimale Kameraposition am Fahrzeug für ein Monokamera-System ermittelt. Die Ergebnisse der Evaluation des entwickelten Systems zeigen:

1. Hohe Erkennungsleistung: 0.945 mAP (Mean Averag Precision)
2. Präzise Distanzberechnung: Median der Abweichung von 0.43m für Pylonen mit einer Entfernung von sieben Metern oder mehr
3. Echtzeitfähigkeit: Laufzeit von 26.27Hz für die Detektion

Diese Ergebnisse übertreffen die gestellten Anforderungen und leiten dazu, dass sich die Forschungsfrage *„Ermöglicht ein minimalistisches Perzeptionssystem mit nur einer Kamera eine zuverlässige Pylonenerkennung und eine hinreichend präzise Fahrbahnberechnung, um autonomes Fahren in der Formula Student zu realisieren?“* bestätigen lässt.

Das System beweist, dass auch kostengünstige, minimale Ansätze in der Formula Student Driverless konkurrenzfähig sein können, ein wichtiger Beitrag besonders für Teams mit begrenztem Budget. Darüber hinaus liefert die Arbeit Erkenntnisse für minimalistische Perzeptionssystem im autonomen Fahren allgemein.

Das in dieser Arbeit entwickelte Perzeptionssystem bietet bereits einen guten Lösungsweg, dennoch bestehen mehrere vielversprechende Ansätze zur Weiterentwicklung. Zunächst könnten direkte Optimierungen des bestehenden Systems die Leistung weiter steigern: Durch effizienteres Training oder leistungsfähigere Hardware

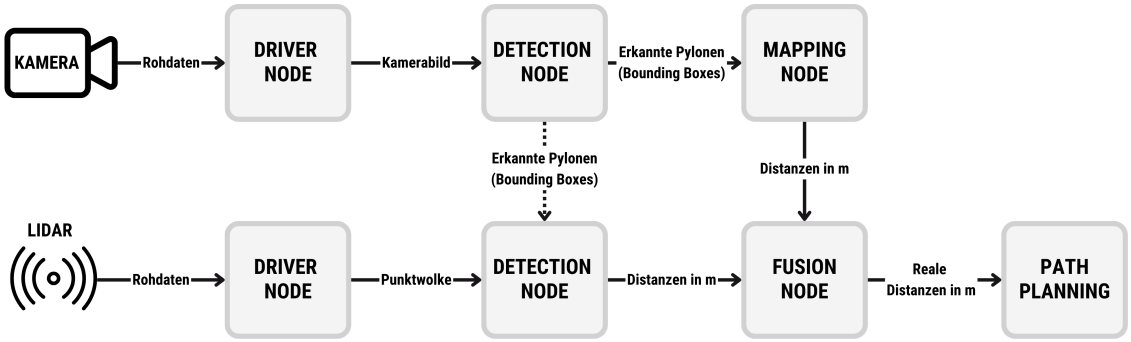

Abbildung 6.1: Mögliche Weiterentwicklung des Systems.

ließe sich die Erkennungsleistung und Laufzeit zusätzlich verbessern, während eine breitere Verteilung der Referenzpunkte bei der Homographie-Kalibrierung die Genauigkeit der Distanzberechnung erhöhen könnte.

Obwohl das System gut mit nur einer Kamera auskommt, sollte die Integration zusätzlicher Sensoren in Betracht gezogen werden, um die Robustheit des Systems zu erhöhen. Besonders naheliegend für Bremergy ist die Einbindung des bereits vorhandenen LiDAR-Sensors, welcher als zweite unabhängige Datenquelle dienen könnte. Durch eine Sensorfusion ließen sich nicht nur die Kameraergebnisse validieren, sondern auch Ausfälle kompensieren. Dies würde die Zuverlässigkeit im Wettbewerb erheblich erhöhen. Ein Konzept dieser möglichen Erweiterung ist in Abbildung 6.1 zu sehen.

Langfristig könnten auch alternative Netzarchitekturen untersucht werden, die bei ähnlicher Genauigkeit weniger Rechenleistung benötigen.

Abbildungsverzeichnis

Tabellenverzeichnis

Quelltextverzeichnis

Literaturverzeichnis

[1] S. Atakishiyev, M. Salameh, H. Yao, and R. Goebel, "Explainable artificial intelligence for autonomous driving: A comprehensive overview and field guide for future research directions," *IEEE Access*, 2024.

[2] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, "A survey of autonomous driving: Common practices and emerging technologies," *IEEE access*, vol. 8, pp. 58 443–58 469, 2020.

[3] A. Alvarez, N. Denner, Z. Feng, D. Fischer, Y. Gao, L. Harsch, S. Herz, N. L. Large, B. Nguyen, C. Rosero *et al.*, "The software stack that won the formula student driverless competition," *arXiv preprint arXiv:2210.10933*, 2022.

[4] J. Janai, F. Güney, A. Behl, A. Geiger *et al.*, "Computer vision for autonomous vehicles: Problems, datasets and state of the art," *Foundations and trends® in computer graphics and vision*, vol. 12, no. 1–3, pp. 1–308, 2020.

[5] J. Kabzan, M. I. Valls, V. J. Reijgwart, H. F. Hendrikx, C. Ehmke, M. Prajapat, A. Bühler, N. Gosala, M. Gupta, R. Sivanesan *et al.*, "Amz driverless: The full autonomous racing system," *Journal of Field Robotics*, vol. 37, no. 7, pp. 1267–1294, 2020.

[6] N. Jun, H. Jibin *et al.*, "Autonomous driving system design for formula student driverless racecar," in *2018 IEEE intelligent vehicles symposium (IV)*. IEEE, 2018, pp. 1–6.

[7] A. Dhall, D. Dai, and L. Van Gool, "Real-time 3d traffic cone detection for autonomous driving," in *2019 IEEE intelligent vehicles symposium (IV)*. IEEE, 2019, pp. 494–501.

[8] S. Hemer and S. Seewaldt, "Startklar für die formula student driverless," *Sonderprojekte ATZ/MTZ*, vol. 21, pp. 12–17, 2016. [Online]. Available: https://link.springer.com/article/10.1007/s41491-016-0585-0

[9] Formula Student, *Formula Student - Rules 2025*, 2025, zugriff am 8. März 2025. [Online]. Available: https://doc.fs-quiz.eu/FS-Rules_2025_v1.1.pdf

[10] Ernst, “Formula student germany image: 20240816_18-21-41_0033_ernst,” Formula Student Germany Media Database, 2024, fotograf: Ernst. Zugriff am 26. März 2025. [Online]. Available: https://media.formulastudent.de/keyword/DV/i-MKmbqFw

[11] BREMOTION - Bremergy Racing. (2024) Technische daten des fahrzeugs. Zugriff am 8. März 2025. [Online]. Available: https://www.bremergy.de/garage/242

[12] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*. MIT press Cambridge, 2016, vol. 1, no. 2.

[13] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[14] R. Khanam and M. Hussain, “Yolov11: An overview of the key architectural enhancements,” *arXiv preprint arXiv:2410.17725*, 2024.

[15] Ultralytics, “Yolov11 - performance metrics,” 2024, zugriff am 24. März 2025. [Online]. Available: https://docs.ultralytics.com/models/yolo11/#performance-metrics

[16] J. Terven, D.-M. Córdova-Esparza, and J.-A. Romero-González, “A comprehensive review of yolo architectures in computer vision: From yolov1 to yolov8 and yolo-nas,” *Machine Learning and Knowledge Extraction*, vol. 5, no. 4, pp. 1680–1716, 2023. [Online]. Available: https://www.mdpi.com/2504-4990/5/4/83

[17] U. Frese, “Vorlesung über yolo im kurs d3bv,” Universität Bremen, Fachbereich Mathematik und Informatik, 2024, zugriff am 26. März 2025. [Online]. Available: https://nc.uni-bremen.de/index.php/s/ezNTSBGGWPwgCjJ?dir=undefined&path=%2F2024&openfile=43408690

[18] N. Vödisch, D. Dodel, and M. Schötz, “Fsoco: The formula student objects in context dataset,” *SAE International Journal of Connected and Automated Vehicles*, vol. 5, no. 12-05-01-0003, 2022.

[19] F. Team, “Fsoco dataset - formula student object detection dataset,” 2025, zugriff am 24. März 2025. [Online]. Available: https://www.fsoco-dataset.com

[20] Ultralytics, “Train - augmentation settings and hyperparameters,” https://docs.ultralytics.com/modes/train/#augmentation-settings-and-hyperparameters, zugriff am 26. März 2025.

[21] F. S. Germany, *FSG Competition Handbook v1.0*, 2025, zugriff am 26. März 2025. [Online]. Available: https://www.formulastudent.de/fileadmin/user_upload/all/2025/important_docs/FSG25_Competition_Handbook_v1.0.pdf

[22] S. Developers, “Shotcut - open source cross-platform video editor,” 2025, software. [Online]. Available: https://www.shotcut.org/

[23] B. Dwyer, J. Nelson, T. Hansen *et al.*, “Roboflow (version 1.0),” 2024, software. [Online]. Available: https://roboflow.com

[24] Roboflow. (2024) Calculate mean average precision (map) from yolo-nas detections. Zugriff am 26. März 2025. [Online]. Available: https://roboflow.com/metrics/calculate-mean-average-precision-from-yolo-nas-detections

[25] ZFTurbo, “Mean average precision for boxes,” https://github.com/ZFTurbo/Mean-Average-Precision-for-Boxes, 2024, gitHub repository. Zugriff am 26. März 2025.

[26] A. Haidar, S. Tomov, J. Dongarra, and N. J. Higham, “Harnessing gpu tensor cores for fast fp16 arithmetic to speed up mixed-precision iterative refinement solvers,” in *SC18: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 2018, pp. 603–613.

[27] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[28] Y. Luo, X. Wang, Y. Liao, Q. Fu, C. Shu, Y. Wu, and Y. He, “A review of homography estimation: Advances and challenges,” *Electronics*, vol. 12, no. 24, 2023. [Online]. Available: https://www.mdpi.com/2079-9292/12/24/4977

[29] Z. Zhang, “A flexible new technique for camera calibration,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, pp. 1330–1334, December 2000, mSR-TR-98-71, Updated March 25, 1999. [Online]. Available: https://www.microsoft.com/en-us/research/publication/a-flexible-new-technique-for-camera-calibration/

[30] O. Team. (2024) Homography — opencv 4.x documentation. Zugriff am 26. März 2025. [Online]. Available: https://docs.opencv.org/4.x/d9/dab/tutorial_homography.html

[31] A. Bonci, F. Gaudeni, M. C. Giannini, and S. Longhi, “Robot operating system 2 (ros2)-based frameworks for increasing robot autonomy: A survey,” *applied sciences*, vol. 13, no. 23, p. 12796, 2023.

[32] Open Robotics. (2024) Ros 2 documentation: Foxy fitzroy. Zugriff am 26. März 2025. [Online]. Available: https://docs.ros.org/en/foxy/index.html

[33] I. RealSense, “realsense-ros: Ros wrapper for intel realsense devices,” https://github.com/IntelRealSense/realsense-ros, 2024, gitHub repository. Zugriff am 26. März 2025.