Technische Hochschule
Brandenburg
University of
Applied Sciences

Vergleich von Methoden zur Echtzeit-Hinderniserkennung in Forward-Looking-Sonardaten für die Kollisionsvermeidung

Bachelorarbeit

zur Erlangung des Grades Bachelor of Science

vorgelegt von:

Marie-Luise Korn

Betreuer: Prof. Dr. Emanuel Kitzelmann

Zweitgutachter: Iván Santibáñez Koref, EvoLogics GmbH

Studiengang Applied Computer Science

des Fachbereichs Informatik und Medien

der Technischen Hochschule Brandenburg

Brandenburg an der Havel, 12. September 2025

Inhalt

1 Einleitung

Der Einsatz autonomer Unterwasserfahrzeuge (Autonomous Underwater Vehicles, AUVs) gewinnt in Anwendungsfeldern wie Umweltmonitoring, Meeresforschung, Offshore-Industrie und Verteidigung zunehmend an Bedeutung. Für den sicheren und effizienten Betrieb solcher Systeme ist die zuverlässige Hinderniserkennung eine zentrale Voraussetzung, da sie die Grundlage für eine effektive Kollisionsvermeidung bildet. Die Fähigkeit, in Echtzeit auf potenzielle Gefahren in der Umgebung zu reagieren, entscheidet maßgeblich über die Einsatzfähigkeit und Betriebssicherheit dieser Fahrzeuge.

Klassische optische Sensoren wie Kameras, die über Wasser durch ihre hohe Auflösung überzeugen, stoßen in der Unterwasserumgebung an deutliche Grenzen. Faktoren wie Trübung, Schwebstoffe, schlechte Lichtverhältnisse sowie Streu- und Reflexionseffekte führen zu erheblichen Qualitätseinbußen bei der Bildaufnahme. Eine robuste Alternative stellt die akustische Umgebungserfassung durch Forward-Looking-Sonare (FLS) dar. Diese Systeme senden Schallimpulse in mehreren Strahlen aus und registrieren deren Reflexionen, wodurch ein zweidimensionales Abbild des Volumens vor dem Sensor entsteht. Die so gewonnenen Sonardaten lassen sich als 2D-Bilder repräsentieren und können mit Verfahren der Bildverarbeitung analysiert werden. Sie bieten damit eine vielversprechende Grundlage für die Hinderniserkennung unter Wasser.

Trotz der vorhandenen Forschungsarbeiten zur Hinderniserkennung mit Sonaren stellt die Übertragbarkeit und Praxistauglichkeit der Verfahren weiterhin eine Herausforderung dar. Viele Ansätze sind stark vom jeweils verwendeten Forward-Looking-Sonar abhängig, so dass ihre Leistungsfähigkeit nicht ohne Weiteres auf andere Sensoren übertragen werden kann. Zudem werden in der Literatur zahlreiche Verfahren in simulationsbasierten Umgebungen evaluiert, ohne eine Validierung unter realen Einsatzbedingungen vorzunehmen. Dies erschwert eine realistische Einschätzung ihrer Eignung für den praktischen Betrieb von AUVs.

Vor diesem Hintergrund leistet die vorliegende Arbeit einen Beitrag, indem ausgewählte Verfahren auf realen FLS-Daten getestet, miteinander verglichen und ihre Anwendbarkeit für die Echtzeit-Hinderniserkennung untersucht werden.

Das Ziel dieser Arbeit ist es, ausgewählte Methoden zu implementieren, zu evaluieren und miteinander zu vergleichen. Die Forschungsfragen lauten dabei insbesondere:

- Welche Verfahren eignen sich für die Hinderniserkennung in FLS-Daten und wie unterscheiden sie sich in ihrer Performance?
- Wie gut lassen sich die Verfahren in Echtzeit auf einem autonomen Fahrzeug einsetzen?

- Welche Stärken und Schwächen zeigen die Ansätze im Praxistest unter realen Bedingungen?

Zur Beantwortung dieser Fragen wurden in Zusammenarbeit mit dem Berliner Unternehmen EvoLogics mit einem ihrer autonomen Fahrzeuge FLS-Daten aufgezeichnet. Basierend auf diesen Daten wurden vier ausgewählte Verfahren zur Hinderniserkennung implementiert und evaluiert. Die Methoden umfassen klassische Bildverarbeitungsansätze, einen probabilistischen Ansatz sowie ein neuronales Netzwerk. Die Evaluation erfolgte zunächst anhand der aufgezeichneten Daten und wurde anschließend durch eine Implementierung auf dem Roboter ergänzt. Dabei konnte die Leistungsfähigkeit der Verfahren in einem Praxistest im See überprüft werden. Obwohl in dieser Arbeit ausschließlich die Hinderniserkennung betrachtet wird, ist die spätere Integration in ein vollständiges System zur Kollisionsvermeidung mitgedacht. Damit leistet die Arbeit einen Beitrag zur Entwicklung zuverlässiger, sonarbasierter Hinderniserkennungsmethoden.

Die Arbeit ist wie folgt aufgebaut: Nach der Einleitung werden in Kapitel 2 die theoretischen Grundlagen und der Stand der Forschung dargestellt. Dabei werden zunächst die zentralen Begriffe Objekterkennung, Hinderniserkennung und Kollisionsvermeidung erläutert und voneinander abgegrenzt. Darauf folgt ein Einblick in das Themengebiet künstliche Intelligenz als Grundlage für Deep Learning basierte Hinderniserkennung.

Kapitel 3 behandelt die Hinderniserkennung dann konkret im Unterwasserkontext. Dazu wird zunächst die Funktionsweise des Forward-Looking Sonars erläutert und anschließend die Vorverarbeitung, Segmentierung und Detektion als typische Verarbeitungsschritte der Hinderniserkennung erläutert. Den Abschluss des Kapitels bildet die Auswahl der vier untersuchten Verfahren.

Kapitel 4 beschreibt die Ausgangslage für die Arbeit. Nach einem kurzen Überblick des Partnerunternehmens EvoLogics und deren eingesetzter Roboter, wird das bestehende ROS2-System vorgestellt. Dazu werden zunächst kurz die Grundlagen von ROS2 vermittelt. Anschließend wird die Datenaufnahme beschrieben, die die Basis für die Entwicklung der Hinderniserkennungsverfahren bildet.

Kapitel 5 stellt eine detaillierte Beschreibung der ausgewählten Verfahren zur Hinderniserkennung sowie deren Umsetzung im Rahmen dieser Arbeit dar. Die Evaluation der Verfahren folgt in Kapitel 6. Dazu werden zunächst Metriken definiert, dann die Ergebnisse auf den Testdaten sowie im Praxistest vorgestellt und abschließend diskutiert. Kapitel 7 fasst die Ergebnisse der Arbeit zusammen und gibt einen Ausblick auf mögliche Weiterentwicklungen.

2 Grundlagen und Stand der Forschung

2.1 Begriffsklärung

Tabelle 1 enthält eine kurze Übersicht der Begriffe, die im Folgenden erläutert werden.

Englischer Begriff	Deutscher Begriff	Definition
Object Classification	Objektklassifikation	Zuweisung eines Bildes oder Bildausschnitts zu einer Objektklasse. Ergebnis: Label
Object Localization	Objektlokalisierung	Lokalisierung eines einzelnen Objekts in einem Bild. Ergebnis: z.B. Bounding Box
Object Detection	Objekterkennung	Erkennung und Lokalisierung aller Objekte in einem Bild inkl. Klassenzuweisung. Ergebnis: z.B. Bounding Boxes + Labels.
Object Recognition	Objekt-wiedererkennung	Erkennen oder Wiedererkennen bekannter Objektklassen. Kann Detektion, Klassifikation und ggf. Identifikation umfassen. Oft mit semantischem Verständnis verbunden.
Obstacle Detection	Hinderniserkennung	Detektion von Objekten oder Strukturen, die die Bewegung eines Systems behindern könnten. Spezialform der Objekterkennung.
Obstacle Avoidance	Hindernisvermeidung	Erkennung von Hindernissen und Einleitung von Kurskorrekturen.
Collision Avoidance	Kollisionsvermeidung	Maßnahmen zur Vermeidung von Zusammenstößen mit beweglichen oder festen Objekten.
Navigation	Navigation	Planung und Durchführung der Fortbewegung in einer Umgebung (Pfadplanung, Positionsbestimmung, Zielverfolgung)

Tabelle 1: Begriffsklärung

Die Aufgabe der Objekterkennung (Object Detection) besteht darin, alle Objekte in einem Satz von Sensordaten (beispielsweise einem Bild oder Videoframe) zu finden und einer oder mehrerer vordefinierter Klassen zuzuordnen. Sie umfasst somit zwei Teilaufgaben: die Objektlokalisierung (Object Localization) und die Objektklassifizierung (Object Classification) (vgl. Zhao et al. 2019, S. 3212). Das Ergebnis besteht typischerweise aus einem Klassenzugehörigkeitslabel sowie einer Lageinformation, welche in Form einer Bounding Box, einer Position mit Maßstab oder auch zusätzlich mit Parametern einer geometrischen Transformation angegeben werden kann (vgl. Amit und Felzenszwalb 2014, S. 537). Die Objekterkennung bildet die Grundlage vieler anderer Aufgaben, wie z.B. dem Object Tracking, der Szenenanalyse oder dem autonomen Fahren.

Ein zentrales Problem in der Objekterkennung ist die hohe Variabilität der Objekte innerhalb einer Klasse. Diese resultiert zum einen aus dem unterschiedlichen Erscheinungsbild der Objekte selbst und zum anderen aus externen Einflussfaktoren bei der Datenaufnahme – etwa variierende Blickwinkel, unterschiedliche Lichtverhältnisse, sowie Unterschiede in Größe und Rotation der Objekte. Zusätzlich wird die Erkennung erschwert, wenn Objekte teilweise verdeckt sind oder in dichter Anordnung auftreten (vgl. Amit und Felzenszwalb 2014, S. 537).

Die Herausforderung besteht darin, Detektionsalgorithmen zu entwickeln, die gegenüber solchen Variationen robust sind und gleichzeitig effizient arbeiten. Die wichtigsten Bewertungsmetriken für Objekterkennungsverfahren sind die Erkennungsgenauigkeit, sowie die Verarbeitungsgeschwindigkeit, insbesondere bei Echtzeitanwendungen.

Historisch werden meist zwei Phasen der Objekterkennung unterschieden:

- Die **Phase der traditionellen Objekterkennung** (bis 2014): Die Algorithmen basieren auf einer manuellen Merkmalsextraktion. Dieses leitet sich aus den Kenntnissen der Entwickler:innen über die Objektklassen ab. Im zweiten Schritt werden traditionelle Klassifikatoren wie Decision Trees oder State Vector Machines angewendet.
- Die **Phase der Deep Learning basierten Objekterkennung** (seit 2014): Neuronale Netze können selbstständig Merkmale extrahieren und die Unterscheidung von Objektklassen erlernen. Insbesondere Convolutional Neural Networks werden für die Objekterkennung eingesetzt.
  (vgl. Zou et al. 2023, S. 2; vgl. Chen et al. 2024, S. 4)

Cao et al. (2023, S. 9199) kritisieren traditionelle Objekterkennungsmethoden vor allem aufgrund ihrer Komplexität und begrenzten Übertragbarkeit. Die klassischen Verfahren bestehen aus mehreren, oft aufwendig aufeinander abgestimmten Verarbeitungsschritten. Zudem hängt ihre Leistungsfähigkeit stark von der manuellen Merkmalsextraktion ab, wodurch diese Ansätze nur eingeschränkt auf neue Anwendungsbereiche oder unbekannte Objekte übertragbar sind.

Wie Kot (2022, S. 8) betont, ist die Generalisierungsfähigkeit auf unbekannte Objektklassen auch in Deep Learning Ansätzen nicht allgemein gegeben. Darüber hinaus erfordert das Training neuronaler Netze in der Regel große, annotierte Datensätze, deren Erstellung sehr zeit- und ressourcenintensiv ist. Dennoch erreichen Deep Learning Modelle in der Praxis meist eine deutlich höhere Genauigkeit und Effizienz bei der Objekterkennung im Vergleich zu klassischen Verfahren. Kapitel 2.2. vermittelt grundlegende Konzepte der künstlichen Intelligenz, insbesondere im Kontext der Objekterkennung.

Die Hinderniserkennung (Obstacle Detection) stellt eine spezielle Anwendung der Objekterkennung dar, die insbesondere in mobilen Systemen (z. B. autonome Fahrzeuge,

Roboter) von zentraler Bedeutung ist. Ziel ist es Objekte zu erkennen, die eine Bewegung behindern oder zu einer Kollision führen könnten.

Die Hinderniserkennung ist ein Systemproblem, das mehre Teilkomponenten umfasst:

1. Sensorik (z. B. Kamera, LiDAR, Sonar), die Rohdaten über die Umgebung liefert,
2. Ein Weltmodell, das diese Sensordaten auf geeignete Weise darstellt,
3. Ein mathematisches Modell, das die Interaktion von Fahrzeug und Umwelt beschreibt, sowie
4. Algorithmen zur Interpretation der Daten und zur Lokalisierung von Hindernissen.

Dabei können Hindernisse über unterschiedliche Eigenschaften wie Form, Größe oder Materialeigenschaften definiert sein. Da Sensordaten oft verrauscht sind, ist eine Vorverarbeitung essenziell (vgl. Matthies 2014, S. 543). Die Ausgabe der Hinderniserkennung kann dann z. B. in Form einer 2D/3D-Bounding-Box, Punktwolke oder sog. Occupancy Grid Map erfolgen. Bei letzterer handelt es sich um eine gitterbasierte Karte, bei der jeder Zelle ein Wahrscheinlichkeitswert zugewiesen wird, der angibt, ob diese Zelle ein Hindernis enthält (vgl. ebd., S. 545).

Die Hinderniserkennung ist eine notwendige Voraussetzung für die Hindernisvermeidung (Obstacle Avoidance). Deren Ziel ist es, ein mobiles System von einem Startpunkt zu einem Zielpunkt zu bewegen und dabei Kollisionen mit Hindernissen zu vermeiden. Nach Wang und Herath besteht der Prozess der Hindernisvermeidung immer aus der Hinderniserkennung und der Kollisionsvermeidung (Collision Avoidance) und wird meist mit Pfadplanung kombiniert. Voraussetzung ist ein gewisses Maß an Umgebungsverständnis, z. B. eine vollständige oder teilweise Karte der Umgebung, eine exakte Lokalisierung des Systems sowie Sensorinformationen über Hindernisse (vgl. Wang und Herath 2022, S. 234).

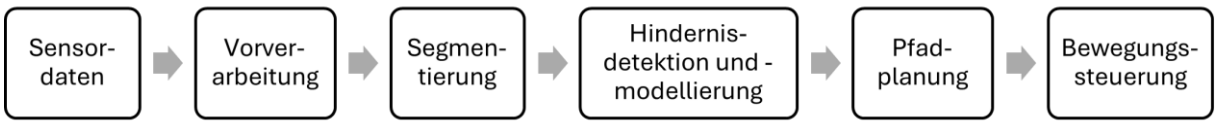


Abbildung 1: Typischer Prozess der Hindernisvermeidung, eigene Darstellung angelehnt an Wilts et al. (2022, S. 2) und Kot (2022, S. 2)

In Abbildung 1 sind die typischen Verarbeitungsschritte zur Hindernisvermeidung in einem mobilen System zusammengefasst. Die Sensorik liefert Rohdaten, die in der Vorverarbeitung gefiltert, normalisiert oder synchronisiert werden. Anschließend erfolgt die Segmentierung der Sensordaten, um relevante Objektbereiche zu isolieren. Die daraus abgeleiteten Hindernisse werden in einem geeigneten Format repräsentiert, was als Basis für die Pfadplanung dient. Die geplante Trajektorie wird anschließend an die Bewegungssteuerung übergeben, um das System sicher und kollisionsfrei zum Ziel zu führen.

Diese Arbeit konzentriert sich ausschließlich auf die Hinderniserkennung als Teilprozess der Hindernisvermeidung. Im Fokus stehen dabei die ersten vier Schritte der typischen Verarbeitungskette. Kapitel 3 beleuchtet diese Schritte speziell im Hinblick auf den Einsatz in einem Unterwasserkontext. Dabei wird ein Forward-Looking Sonar (FLS) als Sensorquelle verwendet, was spezifische Anforderungen an die Rauschunterdrückung, die Segmentierung der Sensordaten sowie die nachfolgenden Analyseverfahren stellt.

2.2 Künstliche Intelligenz in der Objekterkennung

Abbildung 2 verdeutlicht die historische Entwicklung des Fachgebiets Künstlicher Intelligenz sowohl im allgemeinen Kontext als auch speziell in der Unterwasser-Objekterkennung. Frühe Ansätze umfassten symbolische Methoden und Expertensysteme, welche Wissen in Regeln und Symbolen darstellten, so dass Maschinen durch logisches Schließen Probleme bearbeiten konnten. Sie blieben jedoch unflexibel und stießen bei unsicheren oder unstrukturierten Daten wie Bildern schnell an ihre Grenzen (vgl. Russell und Norvig 2021, S. 22ff).

In den 1980er- und 1990er-Jahren fokussierte sich die Forschung daher vermehrt auf Maschinelles Lernen (ML). Dabei lernen die Algorithmen aus großen Datenmengen Muster zu erkennen und Vorhersagen zu treffen (vgl. Russell und Norvig 2021, S. 24ff). Im Teilbereich Deep Learning (DL) werden künstliche neuronale Netze mit vielen Schichten („deep architecture“) eingesetzt. Dieses konnte sich vor allem seit den 2010er-Jahren durchsetzen, insbesondere dank gestiegener Rechenleistung, großer Datenverfügbarkeit und der Entwicklung leistungsfähiger Algorithmen (vgl. Zhao et al. 2019, S. 3213).

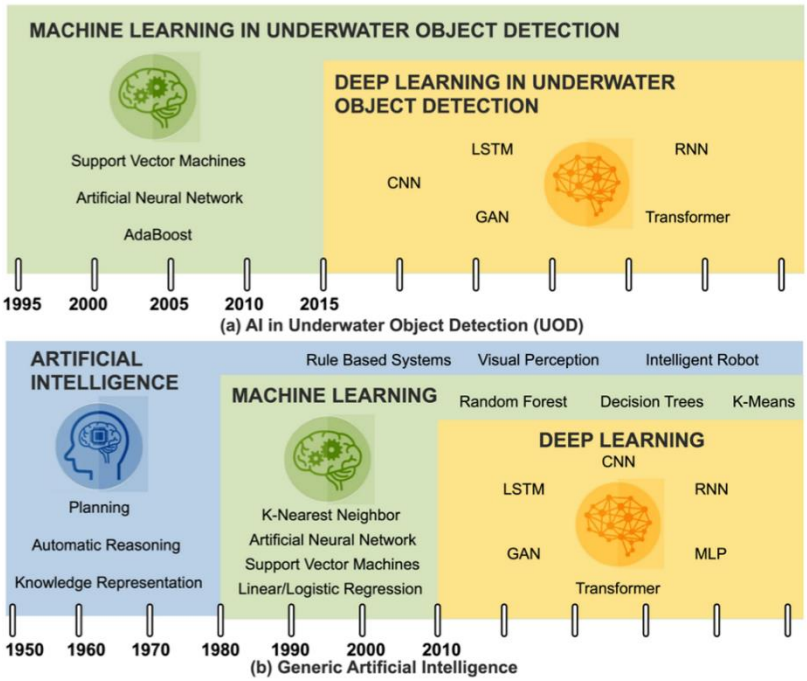


Abbildung 2: Historische Entwicklung des Fachgebiets Künstliche Intelligenz, aus Chen et al. 2024, S. 2

Das Ziel eines neuronalen Netzwerks ist die Approximation einer unbekannten Funktion f^*, die einem Eingabewert x eine Zielausgabe y zuordnet – beispielsweise einem Bild eine Klassenzugehörigkeit. Dazu werden in einem Trainingsprozess lernbare Parameter des Netzwerks (Gewichte und Bias-Werte) so angepasst, dass eine Abbildung $y = f(x, \theta)$ die Zielausgabe f^* möglichst gut trifft (vgl. Goodfellow et al. 2016, S. 168). Grundlage dafür sind Trainingsdaten (x, y), wobei $y = f^*(x)$ die sogenannte Ground Truth darstellt – also die erwartete Ausgabe für einen gegebenen Input. Das Netzwerk ermittelt für jeden Input eine Vorhersage, die mit der Ground Truth verglichen wird. Die Abweichung wird durch eine Verlustfunktion (loss function) quantifiziert (vgl.ebd., S. 169). Typische Beispiele hierfür sind Mean Squared Error (MSE) für Regressionsaufgaben sowie die Cross-Entropy-Loss für Klassifikationsprobleme.

Ein neuronales Netz besteht aus mehreren Schichten zahlreicher Neuronen. Das Netzwerk lernt durch den Backpropagation-Algorithmus, bei dem der Fehler zwischen der aktuellen Netzwerkausgabe und der Ground Truth durch die Netzwerkschichten zurückpropagiert wird. Dabei wird für jede Verbindung berechnet, wie stark sie zum Gesamtfehler beigetragen hat. Mithilfe eines Optimierungsverfahrens werden die Gewichte angepasst, sodass die Vorhersage des Netzwerks im nächsten Schritt näher an der Zielausgabe liegt (vgl. Russell und Norvig 2021, S. 754f). Ein typisches Beispiel ist der Gradientenabstieg, bei dem die Gewichte schrittweise in die Richtung verändert werden, in der der Fehler am stärksten abnimmt. Der Adam-Algorithmus erweitert dieses Prinzip, indem er die Schrittweite für jedes Gewicht automatisch anpasst und frühere Gradienten mit einbezieht.

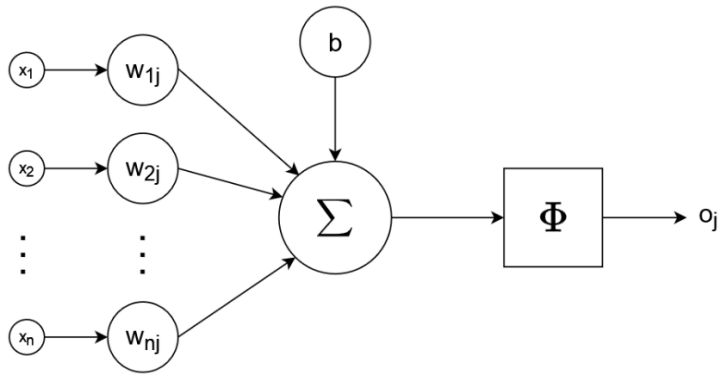


Abbildung 3: Schematische Darstellung eines künstlichen Neurons mit dem Index j.
Eigene Darstellung angelehnt an Burgmer 2005.

Abbildung 3 zeigt die Funktionsweise eines einzelnen künstlichen Neurons mit Index j. Es berechnet aus mehreren Eingabewerten $x_1, \dots, x_n$ eine gewichtete Summe wobei w_i die Gewichtungen und b ein Bias-Wert sind:

$$z = \sum_{i=1}^{n} w_i \cdot x_i + b$$

Auf diese Linearkombination wird eine Aktivierungsfunktion $\Phi(z)$ angewendet, zum Beispiel eine Sigmoid-, Tanh- oder ReLU-Funktion. Diese nichtlineare Transformation entscheidet, ob und in welchem Maße das Neuron aktiviert wird, also ein Signal an die nächste Schicht weitergibt (vgl. Russell und Norvig 2021, S. 751f).

Für die Analyse bildbasierter Daten haben sich insbesondere Convolutional Neural Networks (CNNs) bewährt. Dabei bewegen sich Filter (Kernels) schrittweise über die Eingabe und erzeugen Feature Maps, die charakteristische Merkmale wie Kanten oder Texturen abbilden (vgl. ebd., S. 760). Da durch die Verwendung von Filtern dieselben Gewichte über das gesamte Bild genutzt werden, reduziert sich die Anzahl der zu erlernenden Parameter erheblich. CNNs sind daher besonders effizient und leistungsfähig für datenintensive Aufgaben wie Bild- und Spracherkennung. Sie benötigen deutlich weniger Speicher und Rechenleistung als vollständig verbundene Netze (vgl. Goodfellow et al. 2016, S. 335).

Im Bereich der Objekterkennung werden CNNs in Two-Stage- und One-Stage-Detektoren unterschieden. Two-Stage-Modelle wie Faster R-CNN (Regions with CNN Features) arbeiten in zwei Schritte: Zunächst werden potenzielle Objektregionen vorgeschlagen. Diese werden danach einem CNN übergeben, um Merkmale zu extrahieren, die Objekte zu klassifizieren und die Lokalisierung zu verfeinern (vgl. Zou et al. 2023, S. 3). Diese Methode erreicht in der Regel hohe Genauigkeiten, ist jedoch rechenintensiv (vgl. Chen et al. 2024, S. 4).

Im Gegensatz dazu verarbeiten One-Stage-Detektoren wie YOLO (You Only Look Once) oder SSD (Single Shot Multibox Detector) das gesamte Bild in einem einzigen Vorwärtsdurchlauf. Sie unterteilen das Bild in Regionen und sagen für jede Region potenzielle Objekte mit zugehöriger Klasse und Bounding Box vorher. One-Stage-Detektoren erreichen eine sehr hohe Geschwindigkeit und sind insbesondere für Echtzeitanwendungen geeignet, wenngleich die Genauigkeit in komplexen Szenarien etwas niedriger sein kann (vgl. Zou et al. 2023, S. 4; vgl. Chen et al. 2024, S. 4).

Da die Erstellung großer, annotierter Datensätze oft zeitaufwendig ist, gewinnt das Transfer Learning zunehmend an Bedeutung. Dabei werden die gelernten Gewichte eines vortrainerten Netzwerks auf eine neue, verwandte Aufgabe übertragen (vgl. Russell und Norvig 2021, S. 781). Fuchs et al. (2018, S. 2f) unterscheiden dabei zwei gängige Ansätze:

- **Feature Extractor:** In diesem Szenario werden die letzten Schichten eines vortrainierten CNNs entfernt, sodass nicht die finale Klassifikation, sondern lediglich die extrahierten Merkmale (Features) als Ausgabe dienen. Diese Merkmale werden anschließend von einem separaten maschinellen Lernverfahren (z.B. Decision Trees, Support Vector Machines) für eine neue Klassifikationsaufgabe genutzt.

- **Fine-Tuning:** Hierbei werden die frühen Schichten des CNN – welche allgemeine visuelle Merkmale wie Kanten oder Texturen erkennen – eingefroren, während die späteren Schichten auf die neuen Daten weitertrainiert werden.

Beiden Ansätzen ist gemein, dass die grundlegende Merkmalsextraktion erhalten bleibt, während die Klassifikation an die neue Zielaufgabe angepasst wird. Transfer Learning eignet sich daher besonders in Situationen mit kleinen oder stark abweichenden Datensätzen sowie beim Erlernen neuer Klassen, die im ursprünglichen Trainingsdatensatz nicht enthalten waren (vgl. Fuchs et al. 2018, S. 3).

3 Hinderniserkennung für Unterwasserfahrzeuge

3.1 Forward-Looking Sonar

Für die Objekterkennung über Wasser kommen in der Regel visuelle Kameras zum Einsatz, die hochauflösende Bilder liefern. Unter Wasser stoßen solche optischen Sensoren jedoch schnell an ihre Grenzen: Partikel im Wasser, eingeschränkte Lichtverhältnisse, Trübung sowie Streu- und Reflexionseffekte führen zu deutlichen Qualitätseinbußen bei der Bildgebung (vgl. Chen et al. 2024, S. 1).

Aus diesem Grund werden unter Wasser bevorzugt Sonarsysteme eingesetzt, die ihre Umgebung mittels akustischer Signale erfassen. Je nach Bauart unterscheiden sich diese Sensoren – wie etwa Side-Scan Sonare, Echosounder oder Forward-Looking Sonare – in ihrer Strahlengeometrie, Reichweite und Betriebsfrequenz (vgl. Kot 2022, S. 4).

Insbesondere Forward-Looking Sonare (FLS) haben sich für die Hinderniserkennung bewährt, da sie eine vergleichsweise hohe Auflösung bieten. Durch das gleichzeitige Aussenden mehrerer Schallstrahlen erzeugen sie ähnlich einer Kamera ein zweidimensionales akustisches Bild der Szene (vgl. Hurtós Vilarnau 2014, S. 4).

Ein FLS ist ein aktives Multibeam-Sonar, das akustische Signale in mehreren Strahlen in einem fächerförmigen Bereich nach vorne aussendet. Das Sichtfeld (Field of View – FOV) wird in horizontaler Richtung durch den Azimuth-Winkel (θ) und in vertikaler Richtung durch den Elevation-Winkel (ϕ) beschrieben (siehe Abbildung 4). Die Schallwellen breiten sich im Wasser aus und werden von Objekten oder Strukturveränderungen zum Teil absorbiert und zum Teil reflektiert. Die Echos werden werden von einer Reihe von Hydrophonen aufgezeichnet und entsprechend ihrer Strahlrichtung und Entfernung in Polarkoordinaten organisiert (vgl. dos Santos et al. 2017, S. 3).

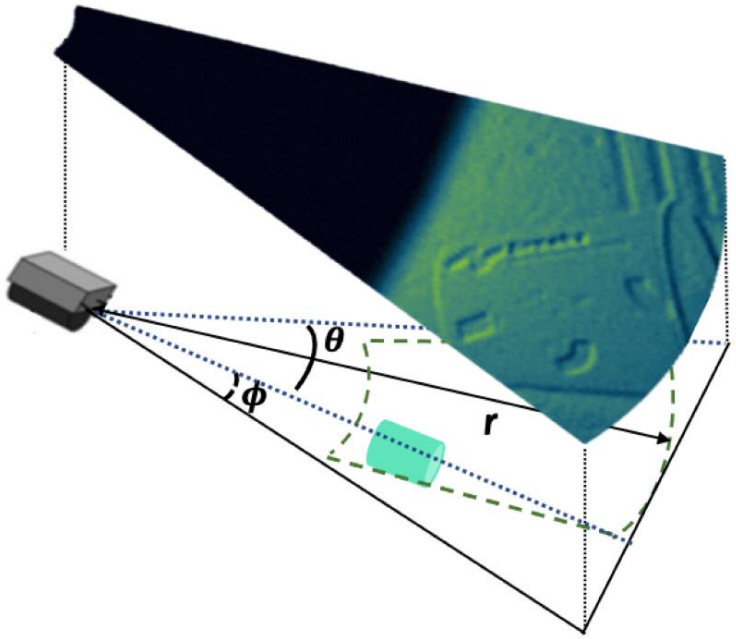


Abbildung 4: Field of View eines FLS, aus Gaspar und Matos 2023, S. 4

Die Richtung entspricht dem Azimut-Winkel θ, in dem der jeweilige Strahl mit dem akustischen Signal ausgesendet wurde. Echos aus derselben Richtung gehören zu demselben Strahl (*Beam*) und werden entlang dieses Strahls in sogenannte *Bins* unterteilt, die jeweils einem bestimmten Zeitintervall entsprechen (siehe Abbildung 5).

Die Berechnung der Entfernung r erfolgt anhand der Laufzeit t zwischen Aussenden des Impulses und Rückkehr des Echos sowie der Schallgeschwindigkeit im Wasser c, die abhängig von Temperatur, Salzgehalt und Druck etwa 1500m/s beträgt:

$$r = \frac{c \cdot t}{2}$$

Um die Daten visuell interpretierbar zu machen, erfolgt häufig eine Transformation in ein kartesisches Koordinatensystem. Dabei wird jeder Punkt mithilfe der folgenden Umrechnungsformel auf eine x-y-Ebene projiziert:

$$p = \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} r \cdot \cos(\theta) \\ r \cdot \sin(\theta) \end{pmatrix}$$

Dabei handelt es sich um eine Approximation, da der tatsächliche Ort des reflektierenden Punkts im dreidimensionalen Raum liegt. Das bedeutet, Objekte, die auf derselben Sichtachse und in gleicher Entfernung liegen, können hinsichtlich ihrer vertikalen Position im Elevationbogen und ihrer Höhe nicht unterschieden werden. Somit handelt es sich bei Sonardaten um eine Projektion der 3D-Daten auf eine 2D-Ebene (vgl. Hurtós Vilarnau 2014, S. 14ff).

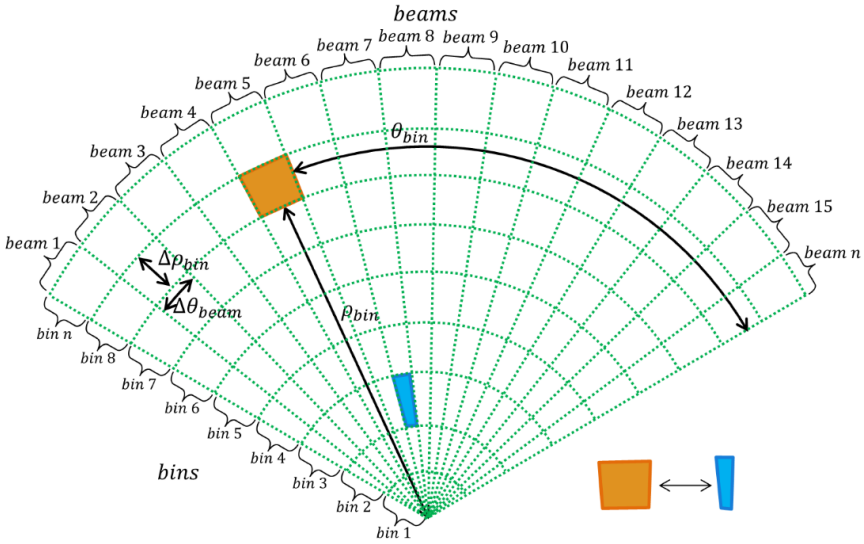


Abbildung 5: Organisation von FLS-Daten, aus dos Santos et al. 2017, S. 3

Wie in Abbildung 5 dargestellt ist, vergrößern sich mit zunehmender Entfernung die Abstände zwischen den Sonarstrahlen bei konstantem Winkelabstand. Verdeutlicht wird dies grafisch durch die orange und blau markierten Flächen, die jeweils einen *Bin* darstellen. Dies führt zu einer inhomogenen Auflösung des Sonarbildes: Ein einzelner Messpunkt in Polarkoordinaten wird auf mehrere Pixel in kartesische Koordinaten abgebildet (vgl. dos Santos et al. 2017, S. 3f).

Die Auflösung und Reichweite eines FLS hängt von mehreren Faktoren ab, unter anderem von der Betriebsfrequenz, der Anzahl und Anordnung der Strahlen sowie den akustischen Eigenschaften des Wassers. Höhere Frequenzen ermöglichen zwar eine bessere Auflösung, sind jedoch aufgrund stärkerer Dämpfung nur für kürzere Distanzen geeignet. Tabelle 2 zeigt die zwei unterschiedlichen Betriebsmodi des für diese Arbeit verwendeten FLS-Geräts.

Betriebsfrequenz	700kHz	400kHz
Max. Aktualisierungsrate	25Hz	25Hz
Max. Tauchtiefe	350m	350m
Max. Reichweite	30m	100m
Min. Reichweite	<0.2m	<0.2m
Entfernungsauflösung	<8mm	<8mm
Horizontaler Sichtwinkel	120°	90°
Vertikaler Sichtwinkel	20°	20°
Anzahl der Beams	256	256
Winkelauflösung	1.5°/2.5°	1.5°/2.5°
Strahlenabstand	0.47°	0.47°

Tabelle 2: Technische Daten des verwendeten FLS-Geräts von BlueLink

3.2 Vorverarbeitung der Sonardaten

Sonarbilder sind mit spezifischen Artefakten und Störeinflüssen behaftet:

- **Multipath-Artefakte:** Wenn dasselbe Objekt mehrere akustische Echos verursacht, erscheint es im Bild mehrfach, obwohl es physisch nur einmal vorhanden ist.
- **Akustische Schatten:** Diese entstehen, wenn ein Objekt die Ausbreitung der akustischen Wellen blockiert. Hinter diesem Objekt entsteht ein Bereich ohne akustisches Feedback, der sich im Bild als dunkle Zone zeigt und zu einer teilweisen Verdeckung (Okklusion) anderer Objekte führen kann.
- **Speckle-Noise:** Sonarsignale weisen generell eine geringe Signal-to-Noise-Ratio auf. Durch gegenseitige Inferenz der reflektierten Schallwellen kann dieses Rauschen zusätzlich verstärkt werden.
  (vgl. dos Santos et al. 2017, S. 4)

Die Ausprägung und Intensität dieser Störeinflüsse hängen von verschiedenen Faktoren ab, darunter die Betriebsfrequenz des Sonars, der Einfallswinkel der Schallwellen, die Beschaffenheit des Untergrunds sowie der gewählte Sichtwinkel (vgl. Fuchs et al. 2018, S. 2).

Ein Beispiel für typische Störeinflüsse im verwendeten FLS ist in Abbildung 6 dargestellt. Auffällig ist die körnige Struktur im gesamten Bildhintergrund, die auf Speckle Noise zurückzuführen ist. In den rot markierten Bereichen sind zudem leicht ausgeprägte Multipath-Artefakte erkennbar, die sich in Form von unsauberen Mehrfachreflexionen zeigen. Direkt im Nahbereich des Sensors treten starke Rückstreuungen auf (gelb markiert), die sich auf die vom Fahrzeug erzeugte Wasserverwirbelung zurückführen lassen.

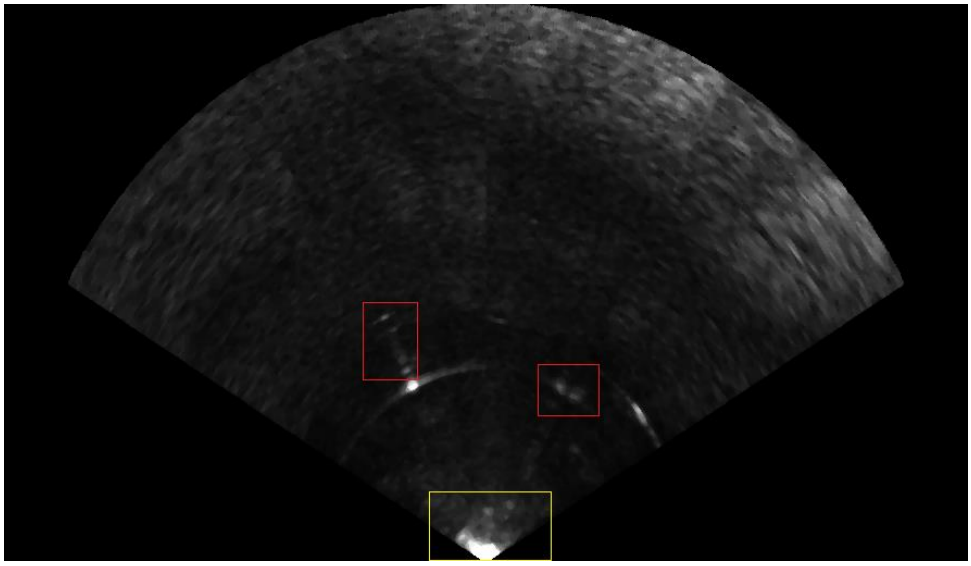


Abbildung 6: Beispiel für Störeffekte in den verwendeten FLS-Bildern

Eine effektive Rauschreduzierung ist die Voraussetzung für eine zuverlässige Hinderniserkennung. Nur durch die gezielte Unterdrückung von Störungen lassen sich falsch-positive Ergebnisse minimieren und die Bildqualität so weit steigern, dass weiterführende Verfahren sinnvoll angewendet werden können (vgl. Wilts et al. 2022, S. 3). Dabei ist die Wahl geeigneter Filtermethoden sowie die Bildauflösung entscheidend, da sie maßgeblich die Genauigkeit und Robustheit der nachfolgenden Verarbeitungsschritte beeinflussen (vgl. Kot 2022, S. 7, 10, 12).

Jin et al. Unterscheiden Verfahren zur Rauschreduktion in zwei Hauptkategorien: Filterungen im Bildraum und Filterungen im Frequenzraum. Filterungen im Bildraum arbeiten direkt auf den Pixelwerten des Bildes. Dabei wird typischerweise die lokale Nachbarschaft eines Pixels analysiert, um dessen Wert auf Basis statistischer oder gewichteter Kriterien neu zu berechnen. Ziel ist die Unterdrückung von impulsartigem Rauschen. Zu den gängigen Verfahren im Bildraum zählen Median-, Gauß-, Lee- oder bilaterale Filter (vgl. Jin et al. 2024, S. 188).

Bei den Operationen im Frequenzraum werden bestimmte Signalanteile gezielt gefiltert. Zu den relevanten Ansätzen zählen unter anderem die diskrete Kosinustransformation (DCT) und die Hauptkomponentenanalyse (PCA) (vgl. ebd.). Ferner gibt es einige Deep Learning Ansätze zur Rauschreduzierung. Neuronale Netze können komplexe Rauschmuster adaptiv lernen und entfernen. Diese sind aufgrund der hohen Rechenintensität jedoch nicht auf Echtzeitszenarien anwendbar (vgl. ebd.).

Neben der Rauschreduzierung gehört auch die Auswahl einer Teilfläche des FLS-Bilds, der sogenannten Region of Interest (ROI), zur Vorverarbeitung. Die folgenden Verarbeitungsschritte beschränken sich damit auf die Teile des Bilds, die für den Anwendungsfall relevant sind. Bei FLS-Daten werden vor allem die Bereiche nahe am Sensor entfernt, da diese besonders von Störeffekten beeinträchtigt werden. Die Auswahl einer ROI reduziert die Datenmenge und damit die Verarbeitungszeit in den weiteren Schritten und trägt somit zur Echtzeitfähigkeit bei (vgl. Kot 2022, S. 9).

3.3 Segmentierung und Detektion

Der nächste Schritt ist die Segmentierung, also die Zuweisung jedes einzelnen Pixels zu einer semantischen Klasse. Kot erwähnt drei Verfahren zur Segmentierung: Thresholding, Clustering und Markov Random Fields (MRF).

Thresholding basiert auf dem Vergleich von Pixelwerten mit einem oder mehreren festgelegten Schwellenwerten. Je nachdem ob der Pixelwert über oder unter einem Schwellenwert liegt, wird der Pixel einer Klasse zugewiesen. Varianten wie Otsu Thresholding wählen Schwellenwerte automatisch anhand der Bildhistogramme aus. Andere Erweiterungen kombinieren Thresholding z.B. mit Verfahren zur Kantenerkennung (vgl. ebd., S. 11).

Clustering Algorithmen gruppieren die Pixel basierend auf ihren Ähnlichkeiten. Beim k-Means Algorithmus werden dazu k zufällige Punkte als Ausgangsbasis für die Gruppen gewählt und die Zentren der Gruppen stetig aktualisiert. Andere Methoden arbeiten nach der Region Growing Technik, bei der einem Ausgangspixel benachbarte Pixel hinzugefügt werden, wenn deren Werte innerhalb eines Toleranzbereichs liegen (vgl. Kot 2022, S. 11).

MRF-Methoden modellieren die Segmentierung probabilistisch, basierend auf den Beziehungen zwischen benachbarten Pixeln. Die Wahrscheinlichkeit, dass ein Pixel zu einer bestimmten Klasse gehört, hängt also auch von der Klassenzugehörigkeit seiner Nachbarn ab (vgl. ebd.).

Segmentierte Bilder weisen häufig Imperfektionen wie kleine Löcher, unvollständige Objektkonturen oder verrauschte Kanten auf. Daher werden anschließend an die Segmentierung morphologische Operationen angewendet. Hierbei kommen Methoden wie Kantenerkennung, Erosion und Dilation zum Einsatz. Bei der Dilation wird die geometrische Struktur eines Objektes basierend auf einem kleinen Fenster (typischerweise 3x3 oder 5x5 Pixel) um einige Pixel erweitert. Im Gegensatz dazu reduziert Erosion die Objektfläche, indem einige Randpixel entfernt werden. Beide Operationen werden häufig nacheinander angewendet, um zunächst Lücken zu schließend und dann anschließend die ursprüngliche Objektkante wiederherzustellen (vgl. ebd.).

Die Segmentierung identifiziert Bildbereiche, die potenziell Objekte enthalten – ohne diese notwendigerweise zu benennen oder zu klassifizieren. Das Ergebnis ist meist eine Menge von Regionen oder eine Maske. Im Anschluss an die Segmentierung folgt die Detektion, die die Objekte lokalisiert und ggf. klassifiziert (vgl. Wilts et al. 2022, S. 2).

Bei einfachen binären Klassifikationsaufgaben wie der Hinderniserkennung verschwimmt die Trennlinie zwischen Segmentierung und Detektion jedoch häufig. Hier liefert bereits die Segmentierung oft eine binäre Maske (z.B. 0 – Hintergrund, 1 – Objekt), bei der die zusammenhängenden Regionen direkt als Objektkandidaten interpretiert werden können. Die Detektion reduziert sich dann auf eine Validierung dieser Objektkandidaten, etwa anhand der Größe, Form oder Lage – anhängig davon, was im jeweiligen Anwendungsfall als Hindernis definiert wird.

Nach Wilts et al. wird der Detektionsschritt in probabilistischen Systemen oft erweitert: Auf Basis segmentierter Bildmerkmale wird ein Wahrscheinlichkeitsmodell über die Umwelt aufgebaut, das Unsicherheiten in der Hinderniserkennung explizit berücksichtigt. Wilts et al. beschreiben diesen Schritt als die Konstruktion eines „belief“ über die Existenz und Lage potenzieller Hindernisse (vgl. ebd.). Eine gängige Repräsentation hierfür ist die Occupancy Grid Map (OGM), in der die Zellbelegung unter Einbezug eines Messmodels, eines Bewegungsmodells und durch Bayessche Filterung berechnet wird. Diese Modellierung erlaubt robustere Entscheidungen – insbesondere in verrauschten oder dynamischen Sonarumgebungen. Eine Erweiterung wäre eine sogenannte Cost

Map, die Merkmale wie die Nähe zum Hindernis als zusätzliche Zellkosten angibt. Das Ausgabeformat der Detektion ist stark davon abhängig, welche Verfahren für die anschließende Pfadplanung verwendet werden.

3.4 Ausgewählte Verfahren

Für die vorliegende Arbeit wurden vier Detektionsverfahren ausgewählt, die unterschiedliche methodische Ansätze repräsentieren. Ziel ist es, ein Spektrum von klassischen bildbasierten Verfahren über probabilistische Ansätze bis hin zu neuronalen Netzen abzudecken.

Als klassische Bildverarbeitungsmethode wurde das Verfahren nach Galceran et al. (2012) ausgewählt. Es ist einer der ersten Ansätze zur Erkennung von menschgemachten Objekten in FLS-Daten. Die Methode basiert auf der grundlegenden Idee, Echo Highlights zu erkennen, die sich anhand ihrer Helligkeit vom Hintergrund abheben. Das Verfahren ist daher im Folgenden als BG-Detektor (Background Substraction) bezeichnet. Der Algorithmus passt sich auf wechselnde Umweltbedingungen an, wobei kein menschlicher Eingriff erforderlich ist. Das Verfahren ist somit auch für den Einsatz auf autonomen Fahrzeugen geeignet und wurde zudem mit Fokus auf Echtzeit-Lauffähigkeit entwickelt.

Das zweite Verfahren nach dos Santos et al. (2017) wurde ursprünglich zur Klassifikation verschiedener Objektarten in FLS-Daten entwickelt. Der Ansatz kann jedoch sinnvoll auf die binäre Hinderniserkennung reduziert werden, indem auf die abschließende Klassifikation verzichtet wird. Das Verfahren ist insbesondere von Interesse, da es direkt auf den Daten der einzelnen Sonarstrahlen arbeitet und darin nach Echo Höhepunkten, sogenannten Peaks sucht. In der Arbeit ist das Verfahren daher als PEAK-Detektor bezeichnet.

Als probabilistisches Verfahren wurde der Ansatz nach Jin et al. (2024) ausgewählt. Er umfasst sowohl die Hinderniserkennung als auch die Kollisionsvermeidung und ist damit exakt für den vorliegenden Anwendungsfall geeignet. Für die Hinderniserkennung werden die Rohdaten des FLS in eine Occupancy Grid Map (OGM) überführt, deren Zellen mithilfe Bayesscher Inferenz eine Belegungswahrscheinlichkeit erhalten. Für die Aktualisierung der OGM wird die Bewegung des Fahrzeugs miteinbezogen. Das Verfahren ist hier als OGM-Detektor bezeichnet.

Das letzte betrachtete Verfahren stammt von Cao et al. (2023) und repräsentiert die Phase der Deep Learning basierten Hinderniserkennung. Es ist ebenfalls für die Hinderniserkennung und Kollisionsvermeidung entwickelt worden. Für die Hinderniserkennung verwenden die Autoren ein YOLOv3-Netzwerk zur Identifikation potenzieller Objektregionen auf die anschließend ein Thresholding Algorithmus angewendet wird, um die exakten Hindernisse zu segmentieren. Das Verfahren wurde mit

Fokus auf Echtzeitfähigkeit und den Einsatz auf AUVs entwickelt, allerdings nicht in der Praxis getestet. Im Folgenden wird es als NN-Detektor benannt.

Kapitel 5 umfasst eine detaillierte Vorstellung der einzelnen Ansätze sowie die Beschreibung der eigenen Implementierung im Rahmen dieser Arbeit.

4 Systemumgebung und Datengrundlage

4.1 Anwendungsszenario

Die vorliegende Arbeit entstand in Zusammenarbeit mit der EvoLogics GmbH. Das Berliner Tech-Unternehmen hat sich auf innovative Unterwasserkommunikation, Positionierung und Robotik auf der Grundlage von Bionik spezialisiert hat. Ziel dieses Forschungsfelds ist es, Erkenntnisse aus der Natur und Tierwelt zu nutzen und diese auf die Technik zu übertragen, um effiziente, robuste, anpassungsfähige oder energieeffiziente Lösungen zu entwickeln. Die von EvoLogics entwickelten Technologien werden in der maritimen Forschung z.B. zur Kartierung und Umweltbeobachtung, sowie in der Offshore Industrie, z. B. zur Inspektion von Unterwasserstrukturen in der Öl-, Gas- und Windenergiebranche eingesetzt. Auch in der Sicherheitstechnik, etwa zur Hafenüberwachung, oder zur Optimierung der modernen Fischerei finden die Produkte Anwendung.

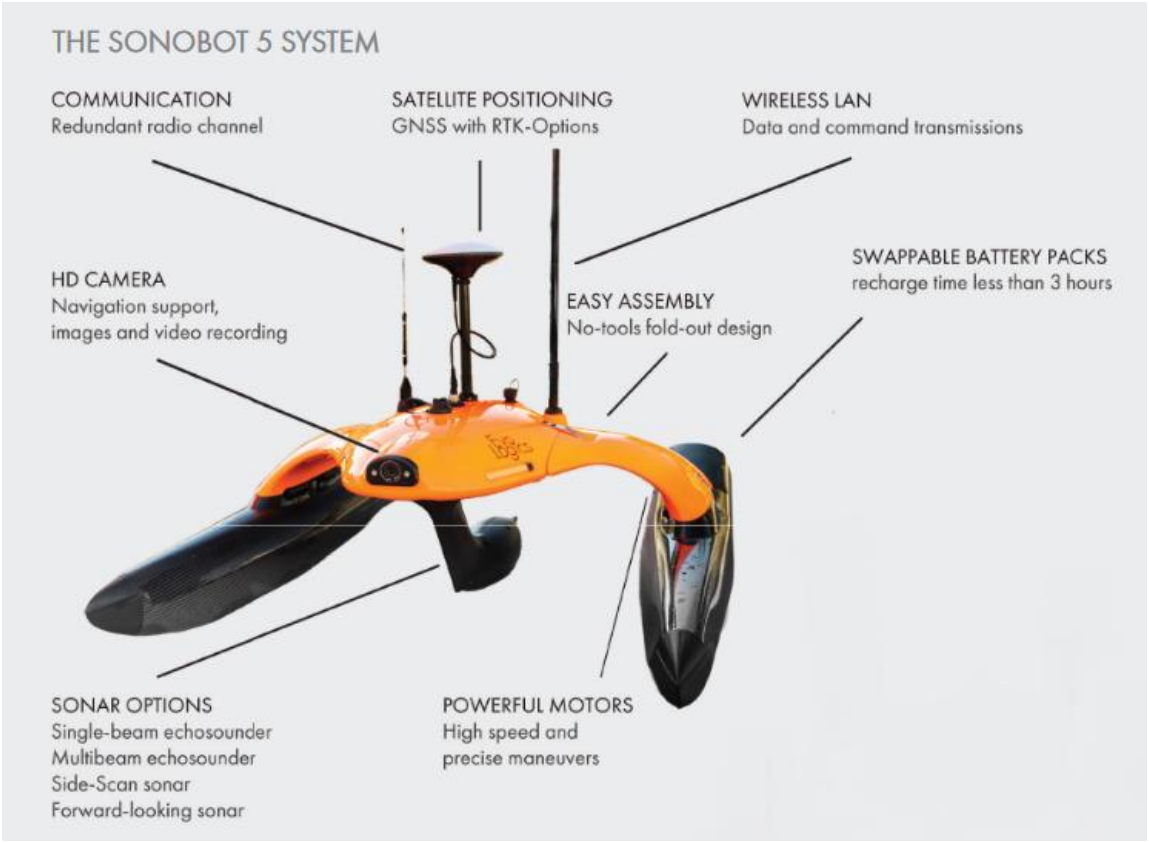


Abbildung 7: Sonobot 5 von EvoLogics

Aushängeschild der Firma sind unter anderem zwei autonome Fahrzeuge für den Einsatz auf und unter Wasser: der Sonobot und der Quadroin (siehe Abbildung 7 und 8). Der Sonobot ist ein unbemanntes Oberflächenfahrzeug (Uncrewed Surface Vehicle - USV), das für präzise hydrografische Messungen in Binnengewässern konzipiert wurde. Der Quadroin ist ein leichtes autonomes Unterwasserfahrzeug (Autonomous Underwater Vehivle - AUV). Besonders hervorzuheben ist seine Fähigkeit zur Schwarmarbeit: Bis zu sechs Quadroins können koordiniert agieren und für weiträumige Unterwasserkommunikation genutzt werden. Der Sonobot dient dabei als Schnittstelle zwischen den unter Wasser agierenden Quadroins und der Überwasser-Infrastruktur.

Beide Fahrzeuge können je nach Einsatzgebiet mit verschiedenen Sensoren ausgestattet werden, darunter Single-Beam Echosounder, Sidescan und Forward-Looking Sonare sowie HD-, Stereo-, Wärmekameras. Die Datenverarbeitung kann vollständig autonom erfolgen und wird auf einem Jetson Board Xavier bzw. Jetson Orin ausgeführt. Zwei KI-basierte Systeme zur Objekterkennung (Automated Target Recognition – ATR) und Kollisionsvermeidung (Obstacle Avoidance System – OAS) werden stetig weiterentwickelt. Die Kollisionsvermeidung beruht derzeit auf Kameras. Insbesondere beim Quadroin ist die Hinderniserkennung mit diesem Sensor aufgrund eingeschränkter Sichtverhältnisse unter Wasser nicht immer ausreichend zuverlässig. Daher soll der Einsatz eines FLS erprobt werden, welches Objekte innerhalb der Wassersäule wahrnehmen kann.

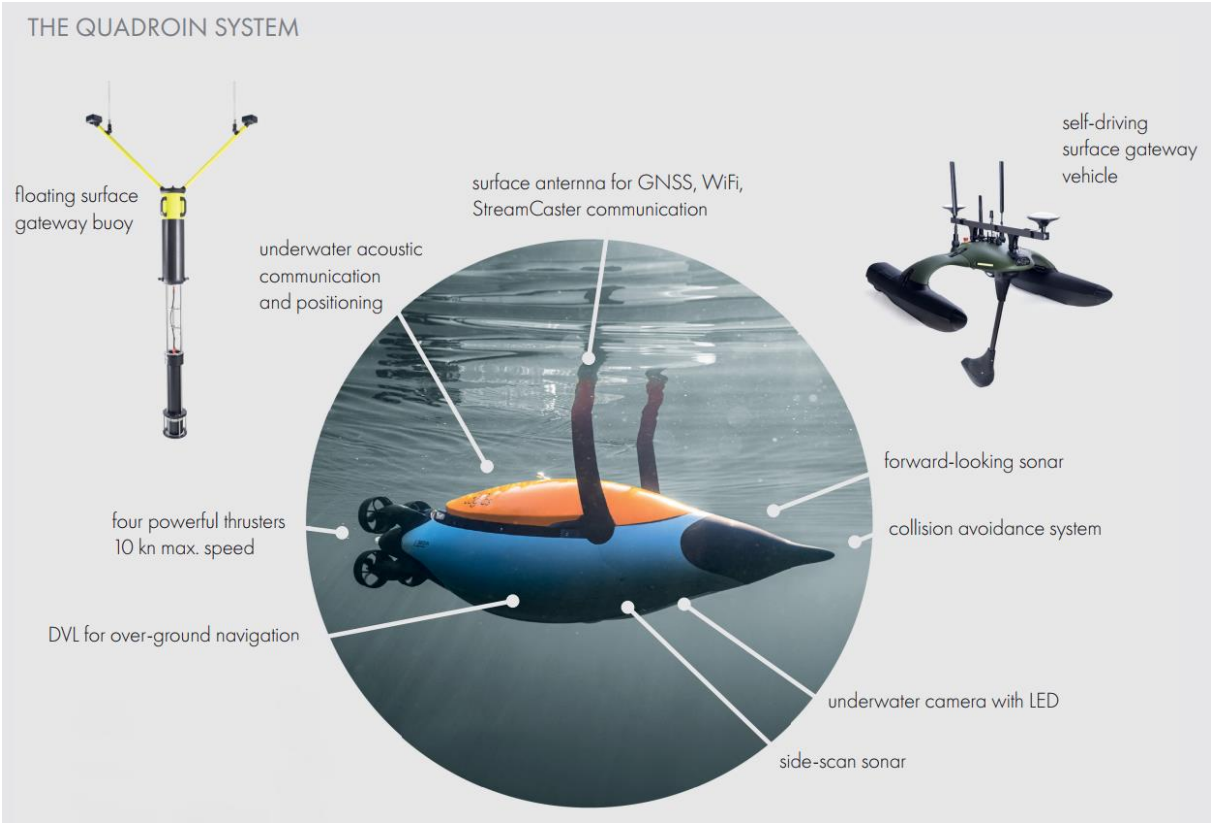


Abbildung 8: Quadroin von EvoLogics

Momentan kann nur der Sonobot mit einem FLS-Gerät ausgestattet werden. Da sich das Fahrzeug auf dem Wasser bewegt, stellen Objekte, die in der Wassersäule liegen, in der Regel kein Hindernis für das Fahrzeug dar. Es besteht jedoch die Möglichkeit, dass der Sonobot die Ergebnisse aus der Hinderniserkennung mit dem FLS an die Quadroins weitergibt, die direkt von Hindernissen in der Wassersäule betroffen sind. Auch die Quadroins können perspektivisch mit einem Forward-Looking Sonar ausgerüstet werden.

Für die Entwicklung einer Hinderniserkennung stand ein Sonobot zur Aufzeichnung von Daten (Kapitel 4.3) und für einen kurzen Praxistest (Kapitel 6.3) zur Verfügung. In einer vorigen Arbeit wurde bereits ein ROS2-Paket für den Sonobot entwickelt, mit welchem die FLS-Daten als rosbags aufgezeichnet werden können. Im Folgenden werden die Grundlagen von ROS2 und das bestehende ROS2-Paket erläutert.

4.2 Bestehendes ROS2-System

4.2.1 Grundlagen von ROS2

Das Robot Operating System 2 (ROS 2) ist – entgegen seinem Namen – kein Betriebssystem, sondern ein Framework, das aus einer Vielzahl von Softwarebibliotheken und Tools besteht. Es dient der Entwicklung modularer Robotersoftware und stellt die grundlegende Kommunikations- und Steuerungsinfrastruktur bereit, die für komplexe Robotikanwendungen erforderlich ist. Daneben bietet ROS2 spezialisierte Tools wie rviz2 zur Echtzeit-Visualisierung von Sensordaten und Positionsinformationen sowie Gazebo, eine umfassende Simulationsumgebung für realitätsnahe Tests (vgl. Renard 2024, S. 6).

Die zentrale Idee von ROS2 ist die Zerlegung eines Robotersystems in einzelne, voneinander unabhängige Programme, sogenannte Nodes. Jeder Node übernimmt eine bestimmte Aufgabe – zum Beispiel das Auslesen eines Sensors, das Verarbeiten von Daten oder das Ansteuern eines Aktors. Diese Nodes können unabhängig voneinander gestartet, gestoppt, ausgetauscht oder erweitert werden, was die Entwicklung und Wartung großer Systeme deutlich vereinfacht (vgl. ebd., S. 65ff).

Zur Konfiguration einzelner Nodes oder ganzer Systeme werden Parameter verwendet, die zur Laufzeit gesetzt oder über Konfigurationsdateien eingebunden werden können (vgl. ebd., S. 187f). Für komplexere Anwendungen kommen Launch Files zum Einsatz. Diese ermöglichen es, mehrere Nodes gleichzeitig zu starten, Abhängigkeiten zu definieren und Parameter zentral zu verwalten (vgl. ebd., S. 211f).

Es gibt drei Arten der Kommunikation zwischen Nodes: Topics, Services und Actions. Alle drei Arten der Kommunikation sind definiert durch einen Namen und ein Interface, welches den Datentyp der Nachricht definiert (vgl. ebd., S. 40).

Bei Topics handelt es sich um benannte Datenkanäle, auf denen Informationen ausgetauscht werden. Sie folgen dem Publisher/Subscriber-Prinzip: Ein Node kann Daten

auf einem Topic publizieren und andere Nodes, die diese Daten benötigen, abonnieren das entsprechende Topic. Es können auch mehrere Nodes auf demselben Topic publizieren und jeder Node kann beliebig viele Topics publizieren und abonnieren. Die Datenübertragung erfolgt asynchron. Publisher und Subscriber müssen demnach nicht gleichzeitig aktiv sein, was das System robuster gegen Ausfälle macht (vgl. Renard 2024, S. 84ff).

Eine synchrone Kommunikation erfolgt über Services und folgt dem Client/Server Prinzip. Dabei sendet ein Node eine Anfrage (Request) an einen Service, der von einem anderen Node bereitgestellt wird. Dieser verarbeitet die Anfrage und gibt eine Antwort (Response) zurück. Services eignen sich für gezielte Befehle oder Anfragen, etwa das Starten einer Messung oder das Abfragen eines Systemstatus (vgl. ebd., S. 120ff). Für länger ablaufende Prozesse gibt es Actions, die zusätzlich Feedback während der Ausführung liefern und auch abgebrochen werden können (vgl. ebd., S. 154ff).

ROS2 bietet mit den sogenannten rosbags ein eigenes Aufzeichnungsformat, das während der Laufzeit beliebige Topics aufnimmt und in einer Datenbank speichert. Die Nachrichten können später in Echtzeit abgespielt werden. Diese Funktion ist besonders hilfreich, wenn der Roboter nicht permanent zur Verfügung steht oder dessen Einsatz an bestimmte Umweltbedingungen geknüpft ist, die nicht ohne weiteres reproduziert werden können. Darüber hinaus eignen sich rosbags auch zur Aufnahme von Trainingsdaten (vgl. ebd., S. 206f).

Die Entwicklung in ROS2 erfolgt modular innerhalb sogenannter Packages, die jeweils eine in sich geschlossene Funktionseinheit darstellen – beispielsweise zur Sensoranbindung, Datenverarbeitung oder Aktorensteuerung (vgl. ebd., S. 61ff). Ein Package kann beliebig viele Nodes, Konfigurationsdateien, Launch-Skripte und Abhängigkeiten enthalten. ROS2 unterstützt dabei verschiedene Programmiersprachen, hautpsächlich C++ und Python, vereinzelt auch Lisp.

4.2.2 Bestehendes ROS2-Paket

Das bestehende ROS2-Paket ist in Abbildung 9 dargestellt. Es ist über den Parameter *run_rosbag* in zwei Modi verwendbar: Wird der Parameter auf *true* gesetzt, muss parallel ein bereits aufgezeichnetes rosbag abgespielt werden. Ist der Parameter auf *false* gesetzt, wird der *fls_driver_node* gestartet und liest live FLS-Daten vom Sonobot ein.

Abbildung 9: Systemübersicht des ROS2-Pakets

Die Einstellungen des FLS-Geräts können anhand von ROS2-Parametern des *fls_driver_node* über eine API vorgenommen werden. Dieser Node baut eine Netzwerkverbindung zum FLS-Gerät auf und liest die Daten über eine TCIP/IP-Verbindung ein. Neben den Sonarbildern werden auch die Konfigurationsdaten des FLS-Geräts übertragen. Der *fls_driver_node* veröffentlicht die Daten im Topic */fls_data* in Form einer FLSFrameData Nachricht, dessen Struktur in Abbildung 10 dargestellt ist. Sie besteht aus einem Header mit Metainformationen, der FLSConfig und dem FLSFrame.

```
root@mlk-ThinkPad-P14s-Gen-5:/# ros2 interface show communication/msg/FLSFrameData
std_msgs/Header header
        builtin_interfaces/Time stamp
                int32 sec
                uint32 nanosec
        string frame_id
float64 estimated_state_timestamp
communication/FLSFrame frame
        std_msgs/Header header
                builtin_interfaces/Time stamp
                        int32 sec
                        uint32 nanosec
                string frame_id
        float64 time_stamp
        float32 range
        float32 duration
        uint16[]  frame
communication/FLSConfig config
        uint32 t0
        float32 snd_velocity
        float32 sample_rate
        uint32 n
        uint32 m
        uint32 m_orig
        float32[] beam_angles
        float32 range
        float32[] distances
```

Abbildung 10: ROS-Nachrichtenformate für FLS-Daten und Konfiguration

FLSConfig besteht aus:

- *t0:* Nummer der ersten Probe im Datensatz
- *snd_velocity:* Schallgeschwindigkeit im Wasser in m/s
- *sample_rate:* Abtastrate in Hz
- *n:* Anzahl der Beams
- *m:* Anzahl der verwendeten Samples pro Beam
- *m_orig:* Anzahl der Samples pro Beam
- *Beam_angles:* Liste aller Beams
- *range:* maximale erfasste Entfernung
- *distances:* Liste aller Entfernungen entlang eines Beams

Dabei können die meisten Daten direkt aus den FLS-Daten übernommen werden. Range und distances lassen sich aus der Schallgeschwindigkeit, der Abtastrate und der Anzahl der Samples berechnen.

FLSFrame besteht aus:

- *header:* enthält den Zeitstempel des ROS2-Systems, sowie eine frame_id
- *timestamp:* Zeistempel des FLS-Geräts
- *range:* maximale erfasste Entfernung
- *duration:* Verarbeitungsdauer zwischen Ping-Auslösung und Datenausgabe
- *frame*: Intensitätswerte

Für den OGM-Detektor wurde das FLSFrame um einen *estimated_state_timestamp* ergänzt. Dieser dient zur Synchronisation der FLS-Daten mit der Position des Sonobots, welche als ROS2-Topic unter dem Namen */EstimatedState* veröffentlicht wird.

Die Aufteilung der Daten in FLSConfig und FLSFrame hat funktionale Gründe: Die FLSFrame-Nachrichten enthalten kontinuierlich eingehende Sensordaten, die vom Gerät regelmäßig gesendet und vom System laufend verarbeitet werden müssen. Daher erfolgt ihre Übertragung über ein Topic, das sich ideal für asynchrone, wiederkehrende Datenströme eignet. Die Konfigurationsdaten hingegen ändern sich selten und werden nur bei Bedarf abgefragt oder gesetzt. Aus diesem Grund werden sie über einen Service bereitgestellt, der gezielte Anfragen mit einer direkten Antwort ermöglicht – ohne den Kommunikationskanal dauerhaft zu belasten.

Da bei der Aufnahme von rosbags der Nachrichtenverkehr von Services nicht aufgezeichnet werden kann, werden die Informationen zunächst als Topic-Nachrichten (FLSFrameData) gesendet. Die Aufteilung in Service und Topic erfolgt anschließend im separaten *fls_message_node*. So wird sichergestellt, dass alle benötigten Daten vollständig aufgezeichnet und später korrekt verarbeitet werden können.

Die Frame- und Konfigurationsdaten können von verschiedenen Nodes weiterverwendet werden. Der *fls_visualisation_node* dient der Visualisierung der Sonardaten in einem kartesischen Koordinatensystem. Der *fls_detector_node* steht exemplarisch für die verschiedenen Hinderniserkennungsmethoden, die im Rahmen dieser Arbeit implementiert wurden. Der *fls_detection_visualisation_node* stellt die Detektionsergebnisse ebenfalls im kartesischen Koordinatensystem dar und kann verwendet werden, um die Detektion mit dem Originalbild und der Ground Truth Segmentierung zu vergleichen.

Die gesamte Anwendung kann über ein Launch File gestartet werden. Dabei kann über die Launch Parameter gesteuert werden, welche Nodes laufen sollen:

- *run_rosbag:* entscheidet zwischen Live-Betrieb mit *fls_driver_node* oder Abspielen des rosbags
- *visualize:* startet *fls_visualisation_node* und *fls_detection_visualisation_node* und sorgt für die Visualisierung der Detektionsschritte der Detektoren
- *detector:* wählt den Detektor aus, 0 – BG, 1 – PEAK, 2 – OGM, 3 - NN

Bis auf den *fls_driver_node* ist das Starten aller Nodes sowohl im Live-Betrieb mit dem FLS-Gerät als auch während des Abspielens eines rosbags möglich.

4.3 Datengrundlage

4.3.1 Allgemein

Für die Entwicklung und den Vergleich unterschiedlicher Verfahren zur Hinderniserkennung und Kollisionsvermeidung in Unterwasserumgebungen ist die Wahl einer geeigneten Datengrundlage zentraler Bedeutung. Insbesondere datengetriebene Ansätze wie Convolutional Neural Networks benötigen typischerweise große Mengen annotiertet Daten.

Grundsätzlich stehen hierfür zwei Quellen zur Verfügung: synthetische Daten und reale Datensätze. Synthetische Daten lassen sich mit Hilfe von Simulationsumgebungen, Ray-tracing-Ansätzen oder generativen Verfahren erzeugen. Sie haben den Vorteil, dass sie in großen Mengen generierbar sind und exakte Ground-Truth-Annotationen erlauben. Gleichwohl zeigen Studien, dass viele Verfahren in simulierten Szenarien evaluiert werden, ohne dass eine Überprüfung unter realen Bedingungen erfolgt (vgl. Kot 2022, S. 1). Hinzu kommt, dass die Bildcharakteristik von der verwendeten Sensortechnologie abhängt, sodass die Generalisierbarkeit synthetischer Daten eingeschränkt ist.

Im Bereich realer Daten existieren einige frei verfügbare Datensätze wie ARCATI, UATD, URPC, RUOD, oder DUO. Der UATD-Datensatz (Underwater Accoustic Target Detection Dataset) umfasst 9200 Bilder in zehn Objektkategorien. Davon wurden 2900 Bilder mit einer Betriebsfrequenz von 720kHz und 6300 Bilder mit 1200kHz aufgenommen (vgl. Xie et al. 2022, S. 4). Der ARCATI Datensatz wurde mit einem FLS von BlueView mit einer Betriebsfrequenz von 900kHz erstellt. Er enthält 257 akustische Bilder mit insgesamt 531 manuell annotierten Segmenten, die in fünf Klassen eingeteilt wurden: Pole, Boat Hull, Stone, Fish und Swimmer (vgl. dos Santos et al. 2017, S. 8f).

Obwohl solche Datensätze eine wertvolle Ressource darstellen, sind sie für die vorliegende Arbeit nur bedingt geeignet. Zum einen unterscheiden sich die Sensoreigenschaften: Das eingesetzte FLS arbeitet mit einer Frequenz von 700kHz und liefert Bilddaten mit anderen Auflösungs- und Rauschcharakteristika. Zum anderen enthalten die genannten Datensätze häufig Einzelbilder, während für die hier untersuchten Methoden teilweise Sequenzen erforderlich sind, um zeitliche Informationen berücksichtigen zu können. Darüber hinaus beinalten viele Datensätze Bildinhalte wie Fische oder Bodenobjekte, die zwar für Klassifikationsaufgaben interessant sind, jedoch keine unmittelbare Relevanz für die Hinderniserkennung im Sinne der Kollisionsvermeidung besitzen.

Aus diesen Gründen wurde in dieser Arbeit bewusst darauf verzichtet, synthetische Daten oder externe Referenzdatensätze zu verwenden. Stattdessen wurde ein eigener Datensatz mit dem Sonobot aufgenommen, so dass dieser die Sensorspezifik des tatsächlich verwendeten FLS widerspiegelt. Der Datensatz gewährleistet sowohl die

Anwendbarkeit klassischer Bildverarbeitungsverfahren als auch die Möglichkeit, ein neuronales Netz mit praxisrelevanten Daten zu trainieren.

4.3.2 Eigene Daten

Die Datenaufnahme wurde mithilfe des ersten grundlegenden ROS2-Pakets durchgeführt, welches den *fls_driver_node*, *fls_message_node* sowie die *fls_visualisation_node* umfasst. Die Aufnahme fand am 11.06.2025 am Werbellinsee im Norden von Berlin statt. Der See wurde gewählt, da sich die Tiefe von 15-20m im Aufnahmebereich gut für FLS-Daten eignet. In flacheren Seen sind die Daten stärker mit Bodenreflexionen behaftet.

Es wurden Objekte mit unterschiedlicher Beschaffenheit und Form ausgewählt (siehe Abbildung 11) und in verschiedenen Tiefen unter Wasser ausgelegt. Der Sonobot umfuhr diese in einen Reihenmanöver, wie in Abbildung 12 dargestellt. Dabei wurden die Objekte mehrmals aus verschiedenen Winkeln vom FLS erfasst, so dass eine höhere Varianz in den Aufnahmen erhalten wird. Jede Mission wurde sowohl mit einer Reichweite von 15m als auch mit einer Reichweite von 30m aufgezeichnet. Die Aufnahmen mit geringerer Reichweite bilden vor allem die Wassersäule ab, während die Aufnahmen mit höherer Reichweite viele Bodenreflexionen enthalten. Insbesondere beim Zufahren auf das Ufer sind weite Teile des Sonarbilds mit Bodenreflexionen behaftet, die die Hinderniserkennung erschweren können.

Abbildung 11: Objekte für die Datenaufnahme

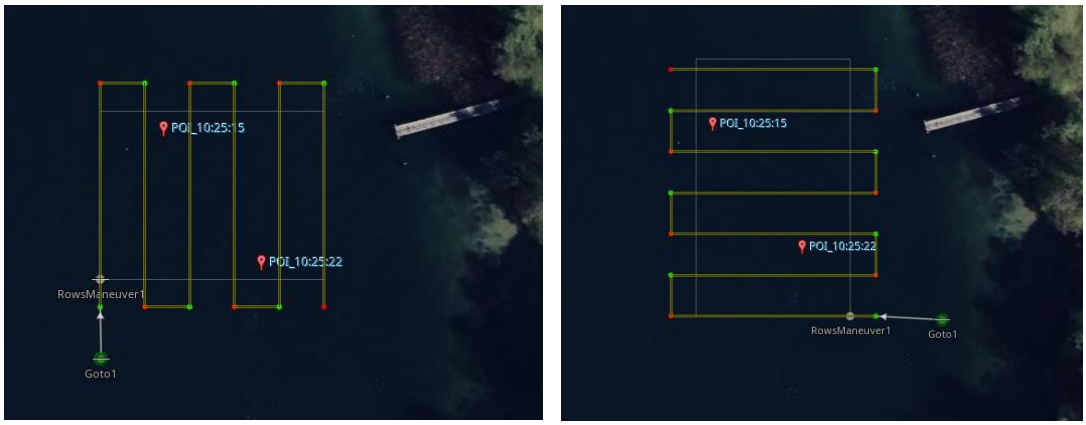


Abbildung 12: Reihenmanöver

Die aufgenommen Daten wurden in Trainings- und Testdaten aufgeteilt. Die Trainingsdaten wurden bei der Implementierung der Objekterkennungsmethoden verwendet, um die idealen Parameter der verschiedenen Objekterkennungsmethoden zu ermitteln. Hierfür wurden die rosbags abgespielt und die verschiedenen Detektionsschritte visuell dargestellt, so dass die Parameteränderungen direkt erkennbar waren. Das Testdatenset wurde für den Vergleich der Methoden verwendet. Diese wurden quantitativ anhand der in Kapitel 6.1 beschriebenen Metriken beurteilt.

Dazu wurden einzelne Bilder aus den Testsequenzen ausgewählt. Die Daten wurden zuvor mit dem *fls_message_node* aufbereitet. Der Node führt eine Normalisierung der Intensitäten durch und passt die Bildwerte mithilfe von Parametern für Helligkeit, Kontrast und Gamma an, um Objekte deutlicher hervorzuheben. Bereiche in unmittelbarer Nähe des Sensors werden verworfen, da sie hauptsächlich Störungen enthalten. Die so vorverarbeiteten Daten dienen als einheitliche Eingabe für alle Detektoren und sind für die Annotation exportiert worden. Diese wurde mit dem Onlinetool CVAT vorgenommen und umfasste einerseits die Annotation mit Bounding Boxes für das YOLOv3 Netzwerk (Kapitel 5.2) und die Erzeugung einer binären Maske für die Evaluation aller Detektoren.

Abbildungen 13 zeigt die FLS-Daten nach der beschriebenen Vorverarbeitung. Die rot markierten Bereiche stellen die abgeschnittenen Bereiche vor dem Sensor dar. Für die Annotation wurden die Daten in kartesische Koordinaten transformiert, wie in Abbildung 14 dargestellt. Dieses Format wird auch von einzelnen Verarbeitungsschritten einiger Detektoren verwendet.

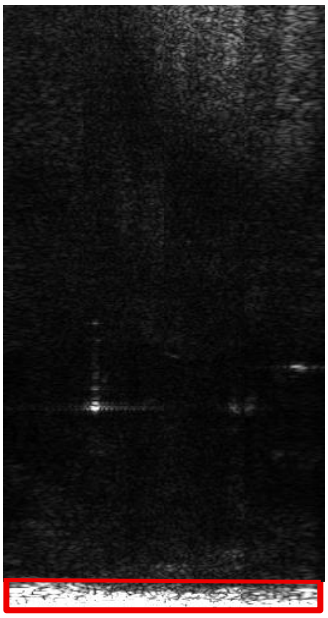


Abbildung 143: FLS-Bild in Polarkoordinaten

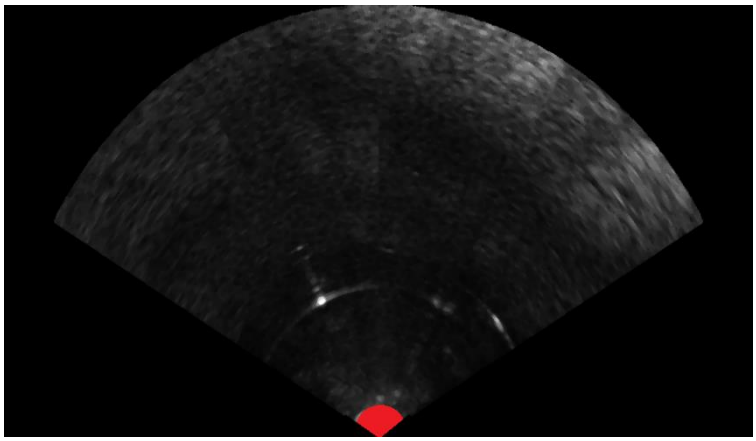


Abbildung 134: FLS-Bild in kartesischen Koordinaten

5 Implementierung ausgewählter Methoden

Im Folgenden werden die ausgewählten Verfahren zur Hinderniserkennung erläutert sowie deren konkrete Umsetzung im Rahmen der Arbeit beschrieben. Jeder Detektor wurde als *fls_detector_node* innerhalb des bestehenden ROS2-Pakets implementiert. Die vollständige Implementierung einschließlich der Testdaten sowie der Aufzeichnungen des Praxistests in Form von rosbags ist der Arbeit beigefügt und kann in einer Dockerumgebung reproduzierbar ausgeführt werden. Eine Anleitung zur Einrichtung und Nutzung ist im README.md dokumentiert.

5.1 BG-Detektor

5.1.1 Beschreibung

Die grundliegende Idee des Algorithmus von Galceran et al. (2012) ist es, Echo Highlights zu erkennen, die sich anhand ihrer Helligkeit vom Hintergrund abheben. Dabei werden alle Schritte nicht auf dem gesamten Sonarbild, sondern auf einer rechteckigen Region of Interest (ROI) ausgeführt. Dadurch können einerseits Bereich vermieden werden, in denen die Sonarbilder eine schlechte Qualität aufweisen, und andererseits wird die Rechenintensität verringert, da nicht auf dem gesamten Bild gearbeitet wird (vgl. Galceran et al. 2012, S. 2).

Um die Laufzeit des Algorithmus weiter zu beschleunigen, wird zunächst ein Integralbild erstellt, das für die folgenden Verarbeitungsschritte verwendet werden kann. Ein Integralbild (auch bekannt als Summed-Area Table) dient der schnellen Berechnung von Pixelsummen innerhalb rechteckiger Ausschnitte von Bildern. Jeder Punkt (x, y) des

Integralbilds I gibt die Summe aller Pixel innerhalb des Rechtecks zwischen Ursprung und (x, y) des Originalbildes A an:

$$I(x,y) = \sum_{i=0}^{i \leq x} \sum_{j=0}^{j \leq y} A(i,j)$$

Das Integralbild lässt sich innerhalb eines Durchlaufs schnell durch die folgende rekursive Formel berechnen:

$$I(x,y) = I(x-1,y) + z(x,y)$$

$$z(x,y) = z(x,y-1) + A(x,y)$$

Wobei $z(x, y)$ die Summe der Pixel in einer Zeile des Originalbilds darstellt.

Das Integralbild lässt sich verwenden, um die Pixelsumme eines beliebigen Rechtecks innerhalb des Originalbildes zu berechnen, wobei nur auf 4 Punkte im Integralbild zugegriffen werden muss. Beispielsweise lässt sich die Pixelsumme der roten Fläche in Abbildung 15 aus den vier Punkten A, B, C, D des Integralbilds wie folgt berechnen:

$$I(rot) = I(D) - I(C) - I(B) + I(A)$$

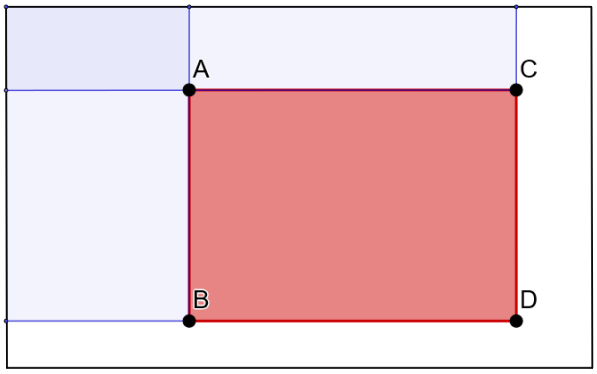


Abbildung 15: Veranschaulichung der Flächenberechnung mit Integralbild

Das Integralbild wird von Galceran et al. verwendet, um eine Hintergrundmaske B und eine Echo Maske E zu erstellen.

Für die lokale Hintergrundschätzung werden zwei Fenster unterschiedlicher Größen verwendet. Für jeden Pixel innerhalb der ROI wird der Mittelwert der Nachbarpixel gebildet, die innerhalb größeren Fensters W_1, aber nicht im kleineren Fenster W_2 liegen. Das kleinere Fenster wird ignoriert, um den Einfluss von Objekten bei der Hintergrundschätzung gering zu halten. Galceran et al. verwenden das Integralbild um die Pixelsumme beider Fenster zu berechnen und dann die des kleineren Fensters vom größeren abzuziehen:

$$B(x,y) = \frac{I(W_1) - I(W_2)}{n}$$

Wobei $n = n_{W_1} - n_{W_2}$ der Anzahl von Pixeln entspricht, die zum Hintergrund beitragen.

Allerdings haben Galceran et al. in ihrem Feldtest festgestellt, dass sich dadurch nicht die erwartete Hintergrundschätzung ergeben hat. Stattdessen empfehlen sie, die Berechnung entweder auf dem Originalbild durchzuführen oder das Integralbild zu verwenden, allerdings nur mit den vier Rechtecken, die zur Hintergrundschätzung beitragen sollen:

$$B(x,y) = \frac{I(W_1 - W_2)}{n}$$

Wobei $W_1 - W_2$ in vier Rechtecke zerlegt wird.

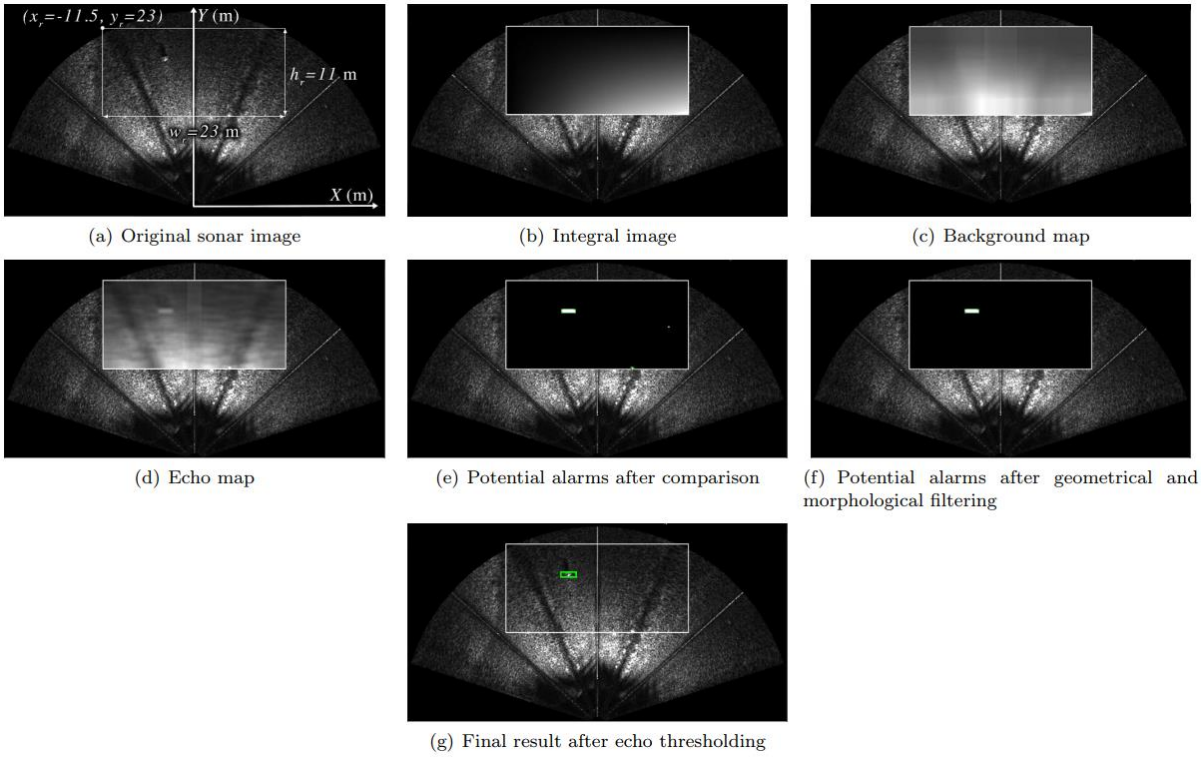
Für die Echomaske E wird wiederum nur ein Fenster verwendet und mithilfe des Integralbilds der Mittelwert der darin enthaltenen Pixel berechnet.

Auf Basis der Hintergrundmaske B und Echomaske E erfolgt eine Segmentierung mithilfe eines lokalen adaptiven Thresholding, bei dem für jedes Pixel die Intensität des Echos mit dem geschätzten Hintergrundwert verglichen wird. Gilt $E(x,y) > \beta \cdot B(x,y)$, wird der Pixel als Teil ein potenzielles Objekt markiert.

Anschließend werden alle potenziellen Objekte anhand geometrischer und morphologischer Merkmale gefiltert. Je nach Art der zu detektierenden Objekte können dabei verschiedene Merkmale verwendet werden. Galceran et al. überprüfen lediglich, ob die potenzielle Objektregion bestimmten Minimal- und Maximalmaßen entspricht.

Im finalen Schritt wird für jeden Alarm i ein Echoscore s_i als Mittelwert aller Pixel innerhalb der potenziellen Objektregion berechnet. Die Berechnung erfolgt auf dem Originalbild, da die Objekte in der Regel keiner rechteckigen Form entsprechen, was die Verwendung des Integralbilds ausschließt. Alle potenziellen Objektregionen mit einem Echoscore über einem bestimmten Schwellenwert t werden als finale Objektdetektionen markiert.

Abbildung 16 zeigt die verschiedenen Stufen des Algorithmus:



(a) Original sonar image (b) Integral image (c) Background map

(d) Echo map (e) Potential alarms after comparison (f) Potential alarms after geometrical and morphological filtering

(g) Final result after echo thresholding

Abbildung 16: Verarbeitungsschritte des Verfahrens von Galceran et al. 2012, S. 5

5.1.2 Eigene Umsetzung

Aufgrund der fächerartigen Form des Sonarbilds erfordert die Festlegung einer rechteckigen ROI besondere Sorgfalt. Einerseits soll vermieden werden, dass relevante Bildbereiche abgeschnitten werden: andererseits dürfen nicht zu viele Bereiche ohne Sonardaten einbezogen werden, da diese die Rechenkomplexität unnötig erhöhen.

Das von Galceran et al. vorgestellte Verfahren ist primär auf die Detektion von Objekten am Seeboden ausgerichtet. Entsprechend wählen die Autoren eine ROI, die in größerer Entfernung vom Sensor liegt und einen großen Teil des Seebodens abdeckt, während Bereiche ohne Messdaten vermieden werden.

Im hier beschriebenen Anwendungsfall steht jedoch die Erkennung von Hindernissen in der Wassersäule im Vordergrund. Daher sind auch die Bildbereiche in unmittelbarer Nähe des Sensors relevant. Die stark störbehafteten Regionen direkt vor dem Sonar werden bereits im vorgelagerten Verarbeitungsschritt durch den *fls_message_node* entfernt. Die ROI wurde deshalb so gewählt, dass sie möglichst nah am Sensor beginnt, wobei ein gewisses Maß an Bereichen ohne Messdaten bewusst in Kauf genommen wird.

Breite und Höhe der ROI wurden anschließend so angepasst, dass keine zusätzlichen Bereiche ohne Sonardaten enthalten sind. Dadurch werden die seitlichen Bildbereiche sowie die am weitesten entfernten Daten abgeschnitten. Für die Hinderniserkennung ist

dies jedoch unkritisch: Objekte an den Seiten stellen keine unmittelbare Blockade der Fahrbahn dar und weiter entfernte Objekte stellen noch keine akute Gefahr dar. Sobald sie in die ROI eintreten, verbleibt ausreichend Zeit für ihre Detektion und das Einleiten von Ausweichmanövern. Die ROI ist in Abbildung 17 dargestellt.

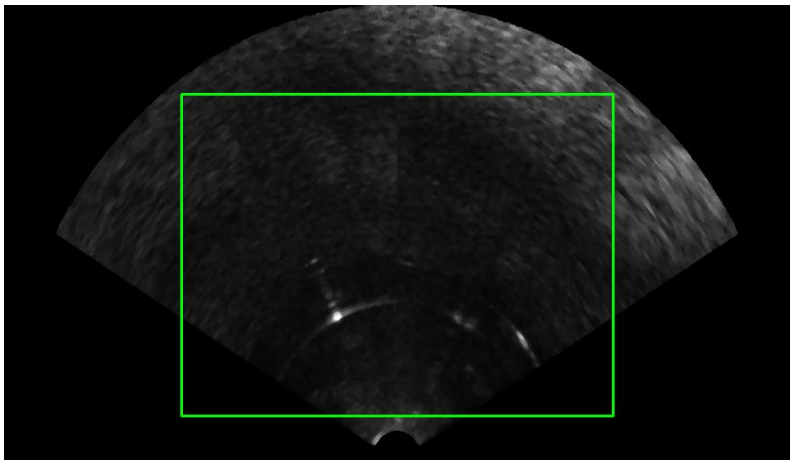


Abbildung 17: ROI des BG-Detektors im Verhältnis zum Gesamtbild

Für die Berechnung des Integralbilds wurde die Funktion der OpenCV Bibliothek genutzt. Diese liefert ein Integralbild, das eine Zeile und eine Spalte größer ist als das Originalbild, um die Berechnung von Randsummen zu erleichtern. Bei Zugriffen muss daher stets ein +1-Offset berücksichtigt werden.

Zur Berechnung des Hintergrunds wurden beide von Galceran et al. beschriebenen Methoden verwendet: Einerseits die ursprüngliche Variante, bei der das kleine Fenster vom größeren Fenster abgezogen wird. Andererseits die Empfehlung, statt der Differenz direkt nur die relevanten Bereiche zu verwenden. Da sich kein wesentlicher Unterschied zwischen den beiden Verfahren zeigte, wurde die ursprüngliche, rechensparende Variante gewählt.

Um Randartefakte zu vermeiden, werden Pixel am Bildrand nach der Berechnung auf den Maximalwert der Karte gesetzt. Die Berechnung der Echokarte erfolgte analog, wobei die Pixel am Bildrand auf 0 gesetzt wurden. Dadurch wird verhindert, dass unvollständig definierte Fensterbereiche zu falschen Alarmen führen.

Zur morphologischen Filterung der Alarmmaske werden weitere Funktionen der OpenCV Bibliothek verwendet. Zunächst werden zusammenhängende weiße Regionen („Blobs“) in der binären Alarmmaske identifiziert (*cv::findContours*). Jede dieser Regionen wird als Kontur, d. h. eine Punktliste entlang ihrer Begrenzung, gespeichert. Für jede gefundene Kontur, die mindestens fünf Punkte enthält, wird eine Ellipse gefittet (*cv::fitEllipse*). Die Ellipse approximiert die Form des Blobs und liefert dabei zwei Hauptachsen. Nur Regionen, deren Major- und Minor-Achsen innerhalb parametrierbarer Grenzen liegen, werden weiter berücksichtigt. Auf diese Weise werden sehr kleine, rauschbedingte Regionen sowie unrealistisch große Strukturen verworfen.

Für die verbliebenen Regionen wird ein Echo-Score berechnet, indem der Mittelwert der Intensitätswerte im Originalbild innerhalb der jeweiligen Kontur bestimmt wird. Übersteigt dieser Wert einen parametrierbaren Schwellwert, wird die Region als gültige Detektion übernommen. Aus den so ausgewählten Konturen wird eine finale Detektionsmaske erzeugt.

Die einstellbaren Parameter wurden als ROS Node Parameter implementiert, um sie während der Ausführung anpassen und verschiedene Konfigurationen vergleichen zu können. Die optimalen Werte wurden anhand der Trainingsdaten experimentell ermittelt und für die Tests wie folgt festgelegt:

- ***outer_window_size = 30, inner_window_size = 7:*** Fenstergröße für die adaptive Hintergrundberechnung
- ***echo_window_size = 20:*** Fenstergröße für die Berechnung der Echomaske
- ***beta = 1.5:*** Thresholdfaktor für den Vergleich von Hintergrund und Echomaske
- ***mmin = 3, mmax = 40:*** Minimum und Maximum der Minor-Achse eines Segments
- ***Mmin = 5, Mmax = 80:*** Minimum und Maximum der Major-Achse eines Segments
- ***echo_threshold = 20:*** Intensitätsschwelle für die finale Segmentselektion

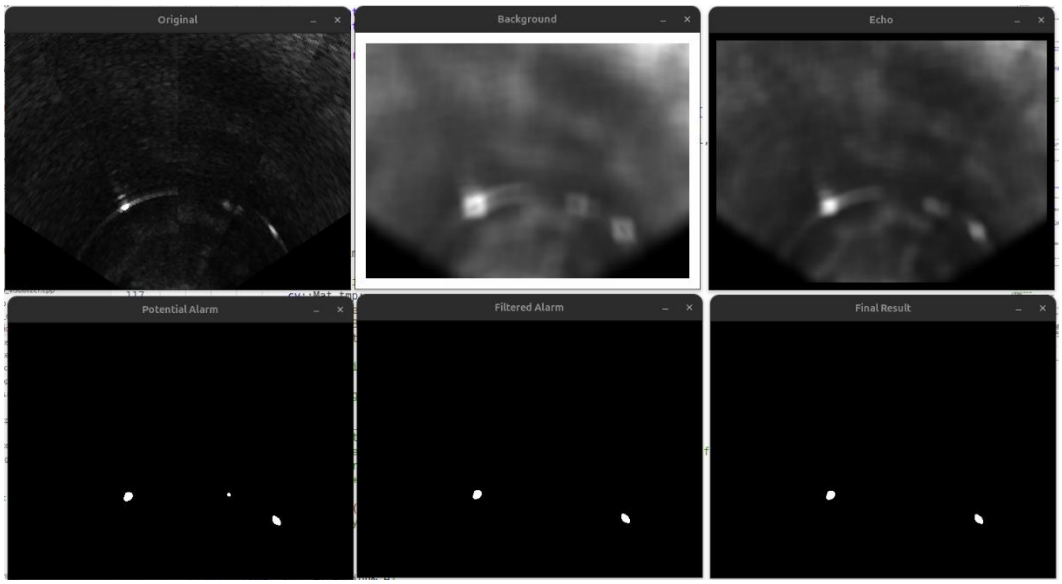


Abbildung 18: Verarbeitungsprozess des BG-Detektors

Abbildung 18 zeigt die Verarbeitungsschritte des implementierten BG-Detektors. Links oben ist das Originalbild dargestellt, welches in kartesische Koordinaten transformiert und auf die verwendete ROI zugeschnitten wurde. Daneben sind die berechnete Hindergrundmaske und Echomaske zu sehen. In der unteren Reihe ist die erzeugte Alarmmaske gezeigt, die durch geometrische Filterung der Segmente bereinigt wird und nach Anwendung des Thresholds die finale Detektion erzeugt.

5.2 PEAK-Detektor

5.2.1 Beschreibung

Der Ansatz nach dos Santos et al. (2017) umfasst wie in Abbildung 19 dargestellt vier Schritte: die Vorverarbeitung des Sonarbilds, die Segmentierung von potenziellen Objekten, die Merkmalbeschreibung der Objekte und die abschließende Klassifizierung.

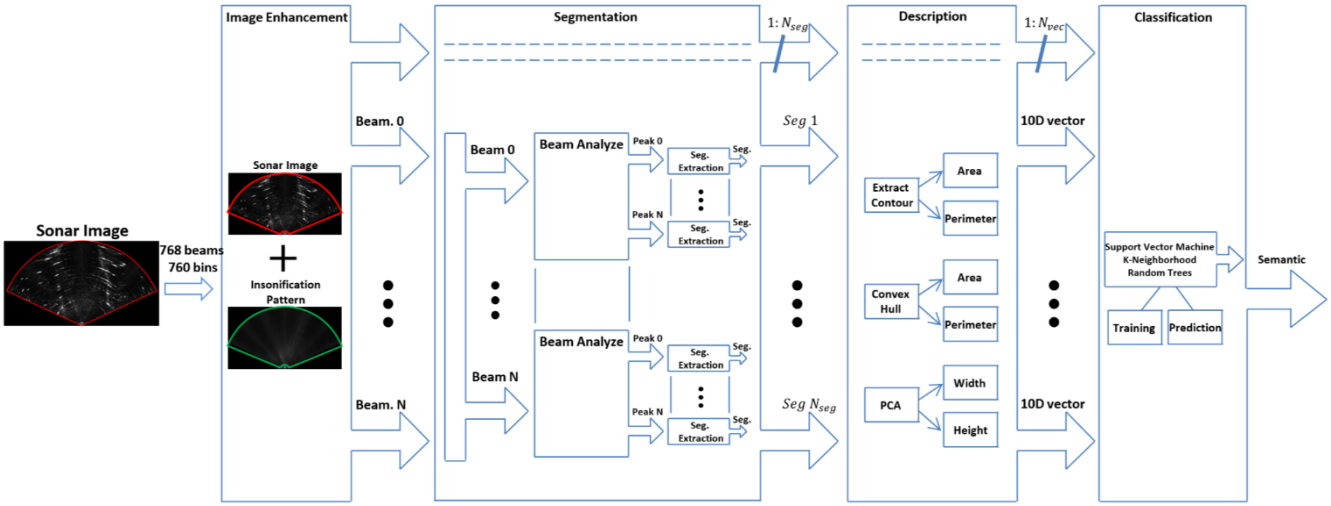


Abbildung 19: Ablauf des Verfahrens von dos Santos et al. 2017, S. 4

Zur Vorbereitung der Sonarbilder entwickelten die Autoren ein Insonifikationsmuster, das typische Reflexionseigenschaften abbildet. Dieses Modell basiert auf einer bekannten Referenzmenge von Sonarbildern und beschreibt die zu erwartende Intensitätsverteilung im Bild. Durch Anwendung dieses Musters auf neue Aufnahmen werden Helligkeitsunterscheide, die durch die uneinheitliche Beschallung (Insonifikation) entstehen, ausgeglichen und das Bild hinsichtlich seiner Intensitätsverteilung normiert.

Anschließend erfolgt die Segmentierung in Objekte und Hintergrund. Da die Objekte sich durch höhere Echointensitäten auszeichnen, wird ein Algorithmus angewendet, der entlang der Beams nach diesen Peaks sucht. Dafür wird zunächst für jeden Bin b innerhalb eines Beams B die durchschnittliche Helligkeit $I_{mean}(b, B)$ berechnet:

$$I_{mean}(b, B) = \frac{1}{win} \sum_{i=b-win}^{b} I(i, B)$$

Dabei steht win für die Fenstergröße bzw. Anzahl der Bins, die in die Berechnung des Durchschnitts einfließen und $I(i, B)$ für die Intensität des i-ten Bins im B-ten Beam. Aus $I_{mean}(b, B)$ wird das offset $I_{peak}(b, B)$ wie folgt berechnet:

$$I_{peak}(b, B) = I_{mean}(b, B) + h_{peak}$$

Dabei beschreibt die Konstante h_{peak} die minimale Intensitätsabweichung von der lokalen Umgebung, die ein Bin aufweisen muss, um als Teil eines Peaks betrachtet zu werden.

Eine Sequenz von Bins, für die $I(b,B) > I_{peak}(b,B)$ gilt, stellt einen Peak dar und fließt nicht in die beschriebene Mittelwertbildung ein. Innerhalb einer solchen Sequenz bezeichnet b_{peak} den Bin mit der höchsten Intensität. Für jeden Peak wird die Position (x,y) von b_{peak}, dessen Intensität $I(b_{peak},B)$ sowie der Mittelwert $I_{mean}(b_{peak},B)$ gespeichert.

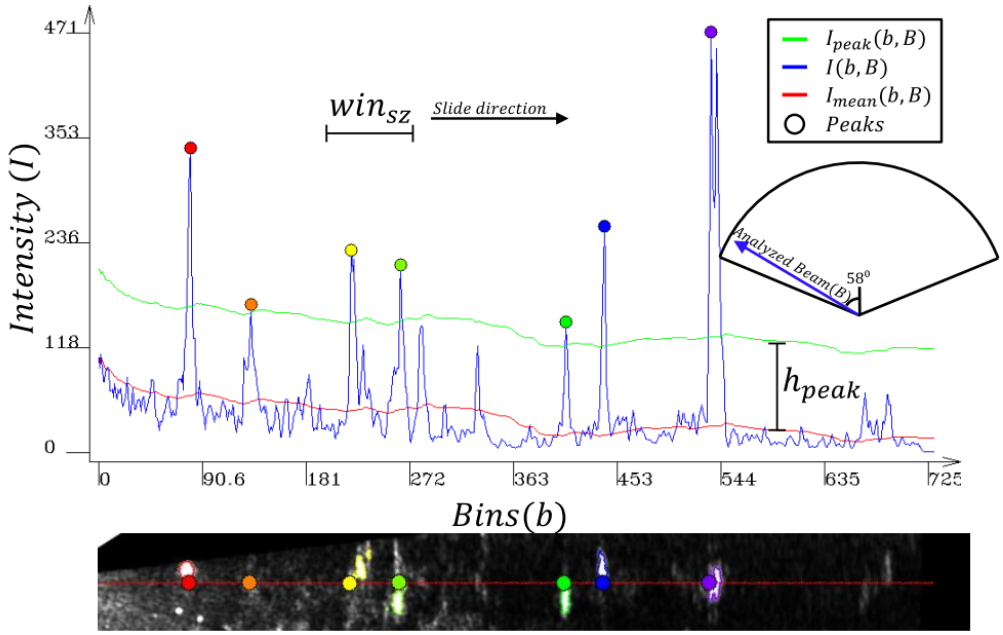


Abbildung 20: Peakerkennung auf einem Beam, aus dos Santos et al. 2017, S. 6

Abbildung 20 zeigt das Verfahren zur Ermittlung der Peaks für einen Beam. Nachdem die Peaks aller Beams des Sonarbilds ermittelt wurden, wird eine Region Growing Technik angewendet, um für jeden Peak die zusammengehörenden Pixel zu finden. Dos Santos et al. wenden dafür eine Breitensuche (Breadth-First Search, BFS) mit dem folgenden Kriterium an:

Ein Nachbarpixel b_{vis} wird dem aktuellen Segment hinzugefügt, wenn

- seine Intensität über dem lokalen Mittelwert des Peaks liegt:

$$I(b_{vis},B) > I_{mean}(b_{peak},B)$$

- oder seine Entfernung zu einem bereits hinzugefügten Segmentrand kleiner als ein Schwellwert D_{seg} ist.

Als Nachbarpixel gelten dabei alle acht angrenzenden Pixel – horizontal, vertikal und diagonal. Die Suche beginnt bei dem b_{peak} mit der niedrigsten Intensität $I(b_{peak},B)$.

Die Einführung des Toleranzabstands D_{seg} erlaubt es, Zwischenbereiche zu überbrücken – etwa bei Schattenwurf oder Rauscheffekten – um zu verhindern, dass zusammenhängende Objekte fälschlicherweise in mehrere Segmente zerfallen.

Nach der Segmentierung werden die erkannten Objekte durch eine 10-dimensionalen Merkmalsvektor beschrieben, um anschließend eine Klassifikation mit Support Vector Machine (SVM), Random Trees (RT) und K-Nearest Neighbor (KNN) durchzuführen. Der Klassifikationsschritt ist jedoch für die Hinderniserkennung nicht notwendig und wird daher nicht weiter ausgeführt.

5.2.2 Eigene Umsetzung

Für die vorliegende Arbeit wurden die ersten beiden Schritte – Vorverarbeitung und Segmentierung – des Ansatzes nach dos Santos et al. implementiert. Auf eine abschließende Merkmalsextraktion und klassifizierende Bewertung der Objektkandidaten wird verzichtet, da im hier beschriebenen Anwendungsfall ausschließlich die Entscheidung zwischen *Hindernis vorhanden* und *kein Hindernis* getroffen werden soll. Die Detektion ergibt sich somit direkt aus der Segmentierung.

Dos Santos et al. verwenden in ihrem Verfahren ein Insonifikationsmodell, um das Bild vor der Segmentierung zu normalisieren. Dieses Modell beschreibt die typische Intensitätsverteilung entlang eines Sonarstrahls in Abhängigkeit von der Entfernung und berücksichtigt dabei, dass nahelegende Bereiche tendenziell stärker insonifiziert werden als weiter entfernte. Ziel dieser Modellierung ist es, systematische Helligkeitsverläufe zu kompensieren und so die Detektion der Peaks robuster zu machen.

Im vorliegenden Anwendungsfalls ist jedoch vorgesehen, mit variabler Reichweite des Sonars zu arbeiten. Das erschwert die Anwendung eines festen Insonifikationsprofils erheblich, da nicht ohne Weiteres für jede Reichweite anwendbar ist. Hinzu kommt, dass die tatsächliche Insonifikation stark von den Umgebungsbedingungen abhängt – insbesondere bei wechselndem Boden, Wassertrübung oder wenn das Fahrzeug sich beispielsweise auf ein Ufer zubewegt.

Stattdessen wird in der hier vorgestellten Umsetzung auf ein explizites Insonifikationsmodell verzichtet. Zur Vorverarbeitung wird ein strahlenweiser Medianfilter verwendet, der jeweils entlang der einzelnen Beams wirkt. Diese Filterung erfolgt somit entlang der Richtung, in der das Sonarsignal tatsächlich ausgesendet und reflektiert wird. Dadurch wird das Bild entlang der natürlichen Geometrie des Sensors geglättet.

Da sowohl der entwickelte Medianfilter als auch die anschließende Peakerkennung in Richtung der Beams arbeiten, wird für diese Schritte das ursprüngliche Sonarbild in Polarkoordinaten verwendet. Diese Darstellung entspricht der natürlichen Geometrie der Sonardaten und ermöglicht eine effiziente und präzise Analyse entlang der

Sonarstrahlen. Die Positionen der erkannten Peaks können anschließend mithilfe projektiver Umrechnungen von Polar- in Bildkoordinaten transformiert werden.

Für die Segmentierung wurden zwei Varianten implementiert und verglichen: Zum einen die Durchführung direkt im Polarbild, zum anderen im zuvor in kartesische Koordinaten transformierten Sonarbild. Die erste Methode hat den Vorteil einer geringeren Rechenlast, da auf eine Koordinatentransformationen verzichtet werden kann. Allerdings basiert die Segmentierung auf der Analyse lokaler Nachbarschaften, was im kartesischen Bild genauer möglich ist, da dort die Abstände zwischen benachbarten Pixeln geometrisch konsistent sind. Entsprechend wurde bei dieser Variante eine robustere Segmentbildung erwartet, die jedoch nicht bestätigt werden konnte. Daher wurden alle Verarbeitungsschritte auf dem Polarbild durchgeführt.

Wie bei dos Santos et al. wurde für die Segmentierung eine 8er-Nachbarschaft als Kriterium für die Breitensuch verwendet. In empirischen Tests mit den Trainingsdaten zeigte sich, dass der Parameter D_{seg} am besten auf 0 gesetzt wird. Da die untersuchten Hindernisse in der Regel keine nennenswerten Lücken enthalten, wirkt sich das Distanzkriterium nachteilig aus: Es führt häufiger dazu, dass auch isolierte Rauschpixel zum Segment hinzugefügt werden.

Die folgenden ROS-Parameter wurden abhängig von der gewählten Sonarreichweite (15m / 30m) angepasst:

- ***win_size= 3:*** Fenstergröße für lokale Mittelwertberechnung bei der Peakerkennung
- ***h_peak=10000.0/5000.0***: Threshold für Intensitätsabweichung eines Peaks
- ***d_seg=0***: Toleranzabstand für Distanzkriterium
- ***min_seg_size=50/15:*** minimale Anzahl an Pixeln in einem Segment
- ***max_seg_size=500/150:*** maximale Anzahl an Pixeln in einem Segment

Abbildung 21 zeigt den Verarbeitungsprozess des PEAK-Detektors. Links ist das Originalbild in Polarkoordinaten dargestellt, wobei jede Spalte einem Sonarbeam entspricht. Entlang dieser Beams folgt die Peakerkennung, deren Ergebnisse im mittleren Bild sichtbar sind. Das rechte Bild zeigt die finale Detektion nach Anwendung der Region-Growing-Methode und anschließendem Thresholding.

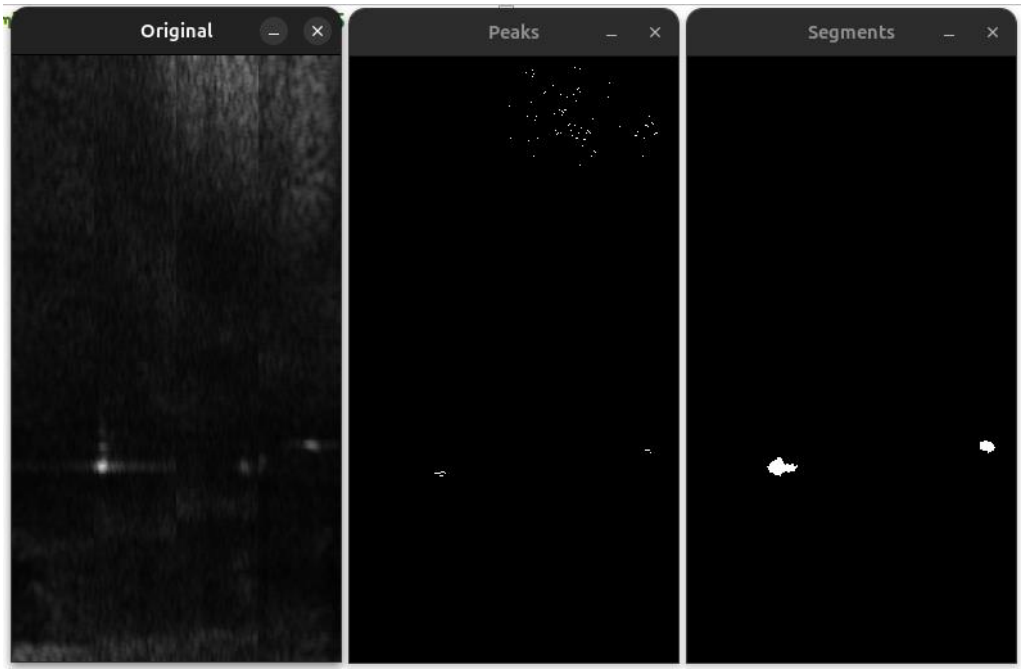


Abbildung 21: Verarbeitungsprozess des PEAK-Detektors

5.3 OGM-Detektor

5.3.1 Beschreibung

Im Verfahren von Jin et al., dargestellt in Abbildung 22, werden die Rohdaten des Forward Lookin Sonars (FLS) zunächst in eine Occupancy Grid Map (OGM) überführt. Das in der Studie verwendete FLS verfügt über 512 Beams und 1506 Bins bei einer Reichweite von 50m. Daraus ergibt sich eine Entfernungsauflösung von etwa 0,033m pro Bin. Die seitliche Abdeckung jedes Beams nimmt mit der Entfernung zu und beträgt in 50m Entfernung rund 0,218m (vgl. ebd., S. 191f).

Für die Erzeugung der OGM reduzieren die Autoren ausschließlich die Anzahl der Bins, sodass sich entlang der Entfernung eine Rasterauflösung von 1m ergibt. Diese Wahl orientiert sich am Wendekreis des eingesetzten AUVs, der bei 10m liegt. Da das Fahrzeug für die sichere Navigation keine feinere Entfernungsauflösung benötigt, geht die exakte Form einzelner Hindernisse in der OGM bewusst verloren. Die ursprüngliche Anzahl der Beams (512) bleibt hingegen erhalten, wodurch die Winkelauflösung des Sonars vollständig beibehalten wird (vgl. ebd., S. 192).

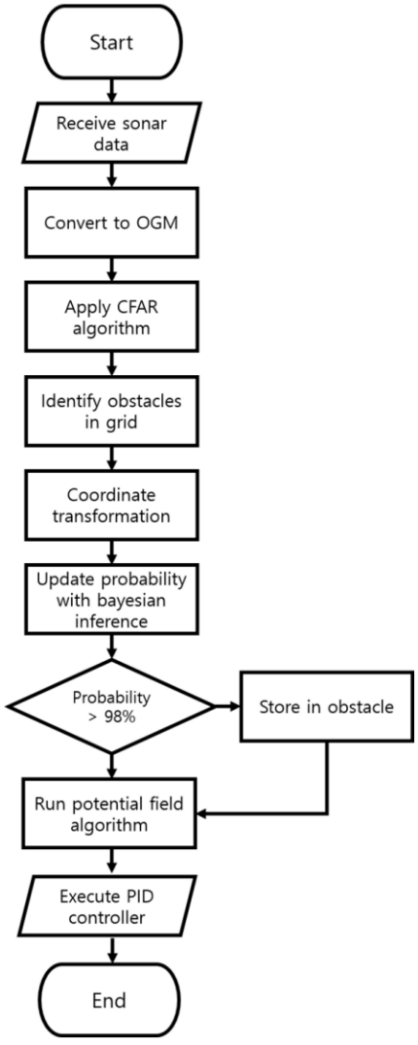


Abbildung 22: Ablauf des Verfahrens von Jin et al. 2024, S. 191

Die Reduktion der Auflösung erfolgt mithilfe der Area Interpolation der OpenCV-Bibliothek. Dabei wird beim Verkleinern eines Bildes der Mittelwert der Intensitäten in jedem Bereich berechnet, der auf eine Zelle der Zielauflösung abgebildet wird. Im Gegensatz zu einfacheren Verfahren wie der nearest-neighbor-Methode, bei der nur der Wert eines repräsentativen Pixels übernommen wird, berücksichtigt die Area Interpolation die Flächenanteile aller beteiligten Quellpixel. Dadurch werden Informationen gleichmäßig auf die Zielzellen verteilt und die Gesamtenergie des Signals erhalten, was eine verlässlichere Grundlage für die weitere Verarbeitung bietet.

Nach der Reduktion der Rohdaten in eine OGM erfolgt eine Rauschunterdrückung und Detektion potenzieller Hindernisse mittels des Greatest-Of Constant False Alarm Rate (GO-CFAR) Algorithmus. Der CFAR-Ansatz stammt ursprünglich aus der Radarsignalverarbeitung und zielt darauf ab, Objekte in verrauschten Messungen zuverlässig von Hintergrundrauschen zu trennen (Jin et al. 2024, S. 192). Die Grundidee besteht darin, dass die Entscheidungsschwelle für das Vorliegen eines Ziels nicht fest vorgegeben, sondern dynamisch an das lokale Rauschniveau angepasst wird. Dadurch bleibt die False-Alarm-Rate trotz stark schwankender Umgebungsbedingungen weitgehend konstant.

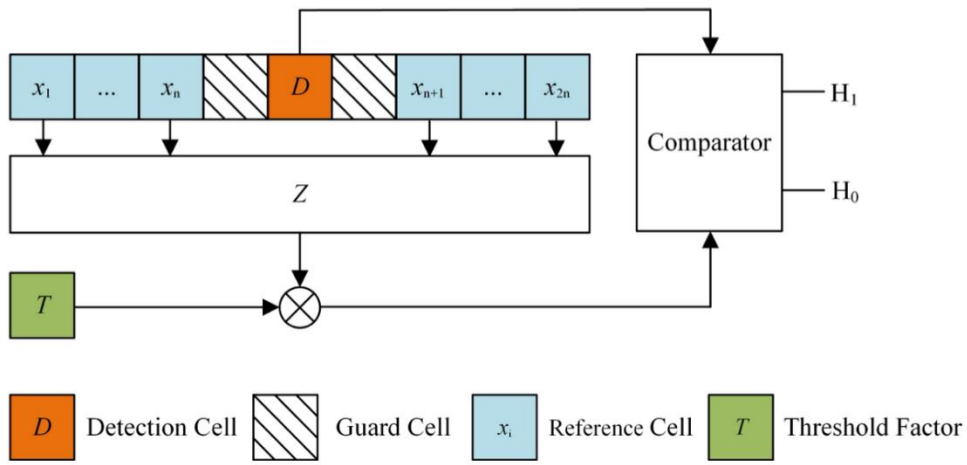


Abbildung 23: Funktionsweise des CFAR-Algorithmus, aus Na et al. 2025, S. 4

Die Funktionsweise des Verfahrens ist in Abbildung 23 dargestellt. Um die betrachtete Zelle D wird ein Fenster definiert, das in Guardzellen und Referenzzellen unterteilt ist. Die Guardzellen befinden sich direkt neben der betrachteten Zelle und sollen verhindern, dass das Signal des potenziellen Ziels die Rauschschätzung verfälscht. Die Referenzzellen dienen dazu, das lokale Stör- bzw. Hintergrundniveau Z abzuschätzen. Während klassische CFAR-Varianten mit dem Mittelwert oder Median der Referenzzellen arbeiten, nutzt GO-CFAR stattdessen das Maximum der Referenzzellen:

$$Z = \max(x_1, \dots, x_{2n}),$$

Die Detektionsschwelle TZ ergibt sich durch Multiplikation des Maximums mit einem Schwellenfaktor T. Jin et al. setzen diesen auf 0,8. Der Entscheidungsprozess folgt dann der Regel:

$$H_1: \quad D \geq TZ \quad (Hindernis\ vorhanden)$$

$$H_0: \quad D < TZ \quad (kein\ Hindernis\ vorhanden)$$

Das Ergebnis dieser Stufe ist ein binäres Maskenbild der OGM, in dem nur diejenigen Zellen markiert sind, die vermutlich ein Hindernis enthalten könnten. Dieses gefilterte Bild bildet die Grundlage für die anschließende probabilistische Bewertung über Bayes'sche Inferenz.

Zunächst werden dafür die Koordinaten unter Nutzung der Positionsinformationen des Fahrzeugs auf eine globale Karte abgebildet. Jeder belegte Zellwert der lokalen OGM wird dabei in das Weltkoordinatensystem projiziert und an der korrekten globalen Position in einer fortlaufenden Karte abgelegt. So entsteht im Verlauf der Fahrt eine wachsende Karte der gesamten Umgebung.

Die Projektion der Pixelkoordinaten (x_r, y_r, z_r) in das Referenzsystem erfolgt nach folgender Gleichung:

$$\begin{bmatrix} x_r \\ y_r \\ z_r \end{bmatrix} = \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} + C_b^r \begin{bmatrix} r \cdot \cos(\theta_b) \\ r \cdot \sin(\theta_b) \\ 0 \end{bmatrix}$$

Dabei beschreibt (x_c, y_c, z_c) die Fahrzeugprosition im Referenzsystem, r die gemessene Entfernung im Sonarstrahl (Bin) und θ_b den jeweiligen Strahlenwinkel (Beam). Die Transformationsmatrix C_b^r berücksichtigt die Lage des Fahrzeugs im Raum mit Roll (ϕ), Pitch (θ) und Yaw (ψ). Sie ist wie folgt definiert:

$$C_b^r = \begin{bmatrix} \mathrm{c}(\theta) \cdot \mathrm{c}(\psi) & -\mathrm{c}(\phi) \cdot s(\psi) + \mathrm{s}(\phi) \cdot s(\theta) \cdot c(\psi) & \mathrm{s}(\phi) \cdot s(\psi) + \mathrm{c}(\phi) \cdot s(\theta) \cdot c(\psi) \\ \mathrm{c}(\theta) \cdot \mathrm{s}(\psi) & \mathrm{c}(\phi) \cdot c(\psi) + \mathrm{s}(\phi) \cdot s(\theta) \cdot s(\psi) & -\mathrm{s}(\phi) \cdot s(\psi) + \mathrm{c}(\phi) \cdot s(\theta) \cdot c(\psi) \\ -\mathrm{s}(\theta) & \mathrm{s}(\phi) \cdot \mathrm{c}(\theta) & \mathrm{c}(\phi) \cdot \mathrm{c}(\theta) \end{bmatrix}$$

Hierbei stehen c und s für den Kosinus und Sinus der jeweiligen Winkel.

Die Bayessche Inferenz arbeitet auf der globalen OGM. Grundidee ist es, die a-priori Wahrscheinlichkeit P(B) – also den bisherigen Glauben, dass eine bestimmte Zelle ein Hindernis enthält – mit den neuen Beobachtungen zu kombinieren. Dazu wird jeder Pixel betrachtet, der laut der binären Maske ein Hindernis enthält. Übersteigt der Intensitätswert I dieses Pixels im Originalbild einen Schwellwert T, gilt die Detektion als bestätigt. Die Wahrscheinlichkeit wird daraufhin nach folgender Formel aktualisiert:

$$I \geq T: \qquad P(B \mid D) = \frac{P(D \mid B) \cdot P(B)}{P(D \mid B) \cdot P(B) + P(D \mid \neg B) \cdot (1 - P(B))}$$

$$I < T: \qquad P(B \mid \neg D) = \frac{P(D \mid \neg B) \cdot P(B)}{P(D \mid \neg B) \cdot P(B) + P(D \mid B) \cdot (1 - P(B))}$$

Dabei gilt:

- $P(B)$: a-priori Wahrscheinlichkeit, dass die Zelle belegt ist,
- $P(D \mid B)$: Wahrscheinlichkeit, dass ein Hindernis erkannt wird, wenn tatsächlich eines vorhanden ist (True Positive Rate),
- $P(D \mid \neg B)$: Wahrscheinlichkeit, dass ein Hindernis fälschlicherweise erkannt wird, obwohl keines vorhanden ist (False Positive Rate),
- $P(B \mid D)bzw.P(B \mid \neg D)$: a-posteriori Wahrscheinlichkeit, dass die Zelle belegt ist gegeben die aktuelle Detektion.

Um eine endgültige Entscheidung zu treffen, legen die Autoren einen Schwellenwert von 98% fest. Nur wenn der Belief einer Zelle diesen Wert überschreitet, wird sie als tatsächliches Hindernis in der Karte gespeichert. Auf diese Weise entsteht eine robuste Hinderniskarte, in der Störungen und Ausreißer unterdrückt werden, während reale Objekte zuverlässig hervorgehoben sind.

5.3.2 Eigene Umsetzung

Das verwendete Forward Looking Sonar (FLS) verfügt über 256 Beams. Die Anzahl der Bins variiert in Abhängigkeit von der eingestellten Reichweite: bei einer Reichweite von 30m werden 675 Bins erfasst, bei 15m Reichweite 506 Bins. Dies entspricht einer Entfernungsauflösung von 0,044m bzw. 0,029m pro Bin. Die seitliche Abdeckung eines einzelnen Beams lässt sich aus der Winkelauflösung des Sonars bestimmen: bei einer Reichweite von 30m und einem Strahlenabstand 0,47° ergibt sich eine seitliche Ausdehnun von rund 0,246m bei 15m Reichweite entsprechend etwa 0,123m.

Für die Überführung in eine Occupancy Grid Map wurde die Anzahl der Bins so reduziert, dass eine Rastergröße von 0,7m entlang der Entfernung entsteht. Diese Auflösung ist im Hinblick auf die sichere Navigation ausreichend, da feinere Strukturen für die Hindernisvermeidung nicht erforderlich sind. Im Gegensatz zu Jin et al. wurde auch die Zahle der Beams reduziert. Diese trägt zu einer Glättung in Azimutrichtung bei. Zufällige Artefakte einzelner Beams werden abgeschwächt, während echte Hindernisse, die sich über mehrere Beams hinweg erstrecken, erhalten bleiben.

Für die eigentliche Reskalierung wurde nicht die von Jin et al. verwendete Area Interpolation eingesetzt, da sich zeigte, dass schwache, aber relevante Signale durch die Mittelwertbildung abgeschwächt oder ganz unterdrückt wurden. Stattdessen kam eine Max-Pooling-Verfahren zur Anwendung. Dabei wird für jede Zielzelle nicht der Mittelwert

aller beteiligten Quellpixel, sondern deren Maximum übernommen. Dieses Vorgehen stellt sicher, dass hochintensive Reflexionen erhalten bleiben.

Im nächsten Verarbeitungsschritt erfolgte eine Rauschunterdrückung mithilfe des GO-CFAR Algorithmus. Jin et al. benennen in ihrer Arbeit nicht explizit, in welcher Form sie den Algorithmus anwenden. Aus der Beschreibung lässt sich jedoch ableiten, dass das Verfahren eindimensional entlang der Beams ausgeführt wurde. Diese Vorgehensweise ist angesichts der reduzierten Anzahl an Bins sinnvoll. Zusätzlich wurde eine Variante getestet, bei der nicht nur die seitlichen Referenzzellen entlang eines Beams berücksichtigt wurden, sondern auch benachbarte Bins in vertikaler Richtung in die Rauschabschätzung einbezogen wurden. Diese Erweiterung führte jedoch, wie erwartet, nicht zu einer Verbesserung der Ergebnisse.

Während Jin et al. die durch das Sonar erzeugte lokale OGM mithilfe präziser Navigationsdaten in ein globales Referenzsystem transformieren, wurde in der hier entwickelten Implementierung ein anderes Vorgehen gewählt. Anstelle der Projektion in Weltkoordinaten kommt eine fahrzeugzentrierte Rolling-Map zum Einsatz. Das Fahrzeug bleibt dabei stets an einer festen Position innerhalb der Karte (unten mittig), während sich die belief_map entsprechend der relativen Bewegung von Frame zu Frame verschiebt. Neue Messungen werden unmittelbar in die verschobene Karte integriert.

Technisch wird die Verschiebung in zwei Schritten umgesetzt. Zunächst erfolgt ein globaler Shift in Reichweitenrichtung (vertikal), wobei eine konstante Auflösung genutzt wird, die sich aus der gewählten Zellgröße in Metern ergibt. Anschließend wird jede Zeile der Karte separat in Azimutrichtung verschoben. Hierbei wird berücksichtigt, dass die laterale Auflösung nicht konstant ist, sondern mit wachsender Entfernung zunimmt. Für jede Zeile wird daher eine individuelle Zellbreite $\Delta y(r) = r \cdot \Delta\theta$ berechnet, wobei r der Abstand in Metern und $\Delta\theta$ der effektive Winkelbereich einer OGM-Spalte ist. Auf diese Weise entspricht der horizontale Shift in Pixeln genau der physikalischen Verschiebung in Metern, auch wenn die Auflösung mit der Entfernung variiert.

Mit dieser Rolling-Map-Darstellung wird das lokale Umfeld des Fahrzeugs zuverlässig aktualisiert. Für die Aufgabe der Hindernisvermeidung, bei der vor allem die unmittelbare Umgebung entscheidend ist, stellt dieses Verfahren eine robuste und praxistaugliche Alternative zur globalen Kartierung dar.

Die Bayessche Inferenz zur Aktualisierung der Belief Map entspricht dem von Jin et al. beschriebenen Ansatz: Für jede Zelle, die durch den CFAR-Schritt als potenzielles Hindernis markiert ist, wird geprüft, ob die Intensität im Originalbild einen vordefinierten Schwellwert überschreitet. In Abhängigkeit davon wird der Belief gemäß der Bayes'schen Formel angepasst. Zellen ohne aktuelle CFAR-Detektion erfahren zusätzlich einen leichten linearen Abbau der Belegungswahrscheinlichkeit. Auf diese Weise klingen

kurzzeitige Störungen ab, während echte Hindernisse durch wiederholte Detektion stabilisiert werden.

Daneben wurde eine Erweiterung implementiert, die auch benachbarte Zellen in die Berechnung der Wahrscheinlichkeiten einbezieht. Hierbei wird für jede Zelle ein lokaler Kontext im Umkreis von 3x3 Nachbarn berücksichtigt. Diese Maßnahmen zeigte insbesondere bei höheren Fahrgeschwindigkeiten Vorteile: Wenn sich das Fahrzeug schnell bewegte, könnte es vorkommen, dass ein Objekt nicht in derselben Zelle wir zuvor detektiert wurde, sondern in einer benachbarten. Dadurch stieg die Wahrscheinlichkeit einer Zelle zwar kurzzeitig an, fiel jedoch im nächsten Frame wieder ab, weil die Detektion auf eine Nachbarzelle überging. Durch den Einbezug der Nachbarschaft konnte dieses Verhalten kompensiert werden, da hohe Beliefs benachbarter Zellen den Prior der betrachteten Zelle stützen und damit eine konsistentere Hindernisdarstellung gewährleisteten.

Die für die Umsetzung gewählten ROS2-Parameter und ihre finalen Werte sind im Folgenden aufgeführt:

- ***ogm_resolution = 0.7:*** räumliche Auflösung der OGM in Bezug auf Entfernung
- ***threshold_scale = 2.0:*** Skalierungsfaktor der GO-CFAR-Schwelle
- ***guard_cells = 5, reference_cells = 3:*** Fenster für GO-CFAR
- ***belief_decay = 0.01:*** Abbaugeschwindigkeit des Beliefs wenn keine Detektion vorliegt
- ***TP = 0.9, FP = 0.05:*** Wahrscheinlichkeiten für wahre und falsche Detektionen im Bayes-Update
- ***detection_threshold =150.0:*** Intensitätsschwelle für gültige Detektionen
- ***belief_threshold = 0.4:*** Schwellwert für die endgültige Hindernisdetektion auf Grundlage der Zellenwahrscheinlichkeit

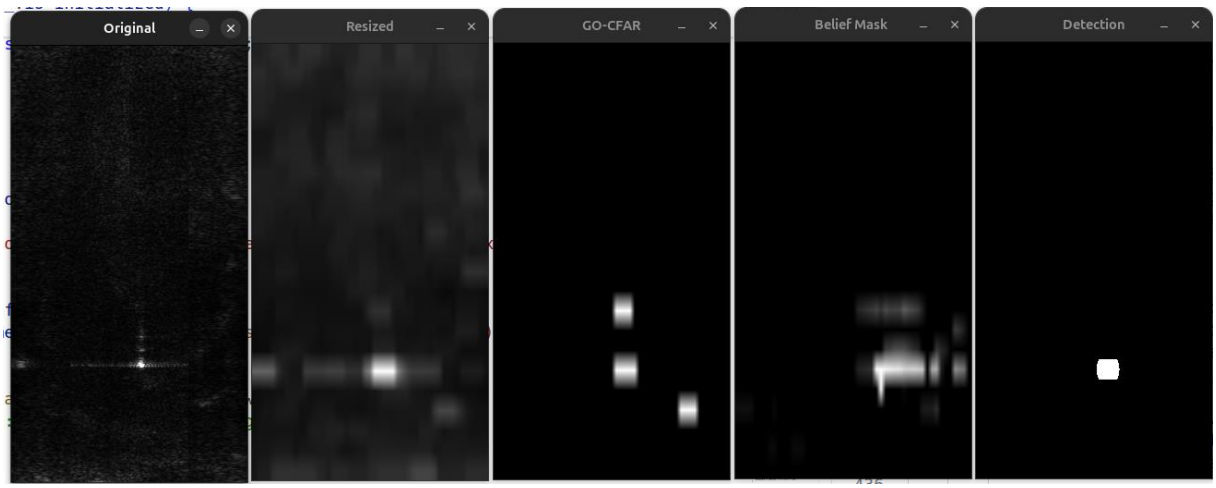


Abbildung 24: Verarbeitungsprozess des OGM-Detektors

Abbildung 24 zeigt exemplarisch die einzelnen Verarbeitungsschritte des Verfahrens. Ausgehend vom Originalbild (links) werden die Sonardaten zunächst in die reduzierte

OGM-Auflösung überführt („Resized“) und anschließend durch den GO-CFAR-Algorithmus gefiltert. In der Belief Map akkumulieren sich die Wahrscheinlichkeiten über mehrere Frames, wodurch Hindernisse deutlicher hervortreten. Die finale Detektionsmaske enthält schließlich nur noch die Zellen, deren Belief den gewählten Schwellwert überschreitet.

5.4 NN-Detektor

5.4.1 Beschreibung

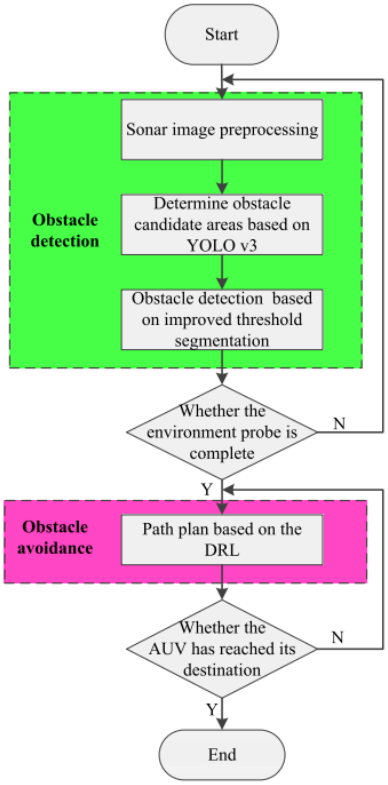


Abbildung 25: Ablauf des Verfahrens von Cao et al. 2023, S. 9200

Das Verfahren zur Hinderniserkennung und -vermeidung nach Cao et al. (2023) gliedert sich in drei Hauptkomponenten: eine Vorverarbeitung der Sonarbilder, die Detektion potenzieller Hindernisregionen mit Hilfe eines YOLOv3-Netzwerks sowie eine anschließende pixelgenaue Segmentierung durch ein adaptives Schwellwertverfahren. Die Hindernisvermeidung erfolgt in einem separaten Schritt mittels Deep Reinforcement Learning zur Pfadplanung (siehe Abbildung 25).

Die Vorverarbeitung ist minimal gehalten. Um impulsartigen Rauschen zu unterdrücken, ohne die Kanteninformationen der Objekte zu verfälschen, wird ein Medianfilter mit einer Fenstergröße von 5x5 Pixeln eingesetzt (vgl. Cao et al. 2023, S. 9200).

Anschließend werden die vorverarbeiten Bilder in das YOLOv3-Netzwerk eingespeist, um rechteckige Bereiche zu identifizieren, die potenziell Hindernisse enthalten. Die Architektur von YOLOv3 ist in Abbildung 26 dargestellt. Sie basiert im Wesentlichen auf zwei Komponenten: einem Netzwerk zur Merkmalsextraktion und mehreren Detektionsköpfen. Das Netzwerk zur Merkmalsextraktion (Backbone) ist bei YOLOv3 das sogenannte Darknet-53. Es handelt sich dabei um ein CNN mit 53 Schichten, das speziell für die effiziente Verarbeitung großer Bilddaten entwickelt wurde. Die Besonderheit von Darknet-53 liegt in der Kombination von klassischen Faltungsschichten mit sogenannten Residual-Verbindungen. Diese Verbindungen erlauben es, Informationen auch in sehr tiefen Netzen verlustfrei weiterzugeben und mindern Probleme wie den „Vanishing Gradient Effekt“. Dieser tritt beim Training tiefer neuronaler Netze auf, wenn die Gradienten während der Backpropagation immer kleiner werden und in den oberen Schichten nahezu verschwinden. Dadurch lernen diese Schichten kaum noch, was das Training extrem verlangsamt oder ganz verhindert. Mithilfe der Residual-Verbindungen kann Darknet-53 besonders robuste und hierarchische Merkmale (Features) extrahieren, die später für die Objekterkennung genutzt werden (vgl. ebd., S. 9201).

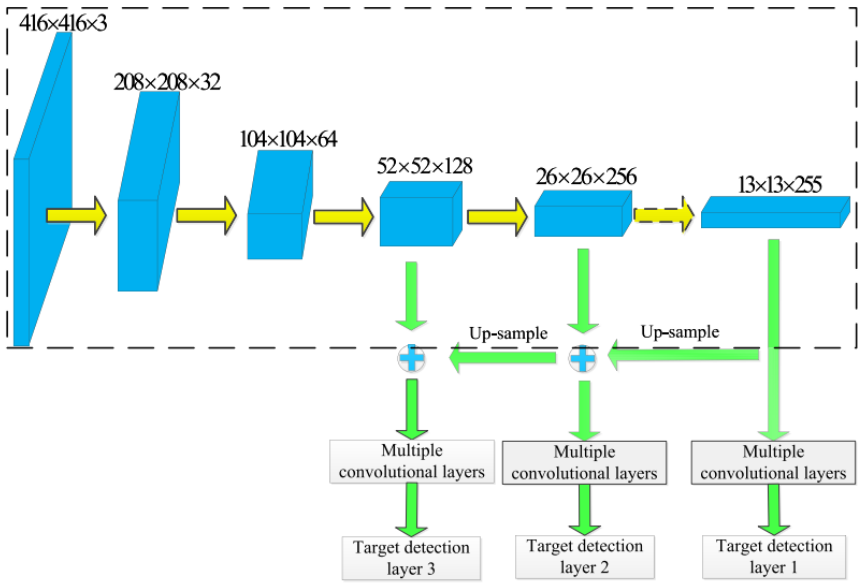


Abbildung 26: Architektur des YOLOv3 Netzwerks, aus Cao et al. 2023, S. 9201

An dieses Backbone schließen sich die Detektionsköpfe an. YOLOv3 setzt hier auf eine mehrskalige Detektionsstrategie, bei der Objekte auf drei verschiedenen Auflösungsstufen gleichzeitig erkannt werden: eine grobe Ebene für große Objekte, eine mittlere für mittelgroße und eine feine für sehr kleine Objekte. Dies wird durch eine pyramidenartige Struktur erreicht, die tiefere (semantisch reichhaltige, aber grobe) Merkmalskarten mit früheren (räumlich genaueren, aber weniger abstrakten) Ebenen kombiniert. Diese Fusion von Informationen ermöglicht es, Objekte unabhängig von ihrer Größe zuverlässiger zu erkennen (Cao et al. 2023, S. 9201).

Jeder Detektionskopf nutzt sogenannte Anchor-Boxen, also vordefinierte rechteckige Referenzrahmen mit unterschiedlichen Seitenverhältnissen und Größen. Diese dienen als Ausgangspunkte für die Objekterkennung: Das Netzwerk berechnet nicht die endgültige Box direkt, sondern nur Korrekturen (Offsets) zu einem passenden Anker.

Die Verlustfunktion von YOLOv3 setzt sich aus fünf Komponenten zusammen, die jeweils unterschiedliche Aspekte der Vorhersage bewerten. Jede Vorhersagebox wird dabei anhand ihrer Position, Größe, Klassenzuweisung und Objektwahrscheinlichkeit beurteilt. Die Gesamtverlustfunktion ergibt sich als Summe der folgenden Teilverluste:

- **Positionsverlust der Boxzentren (loss1):**
  Dieser Term misst die Abweichung zwischen den vorhergesagten und den tatsächlichen Mittelpunktkoordinaten einer Box. Er sorgt dafür, dass die Boxen an der richtigen Stelle im Bild zentriert sind.
- **Größenverlust der Boxen (loss2):**
  Um Unterschiede in Breite und Höhe zwischen Vorhersage und Ground-Truth-Box zu berücksichtigen, wird nicht direkt der Abstand der Werte verglichen, sondern die Differenz der Quadratwurzeln. Dies verhindert, dass große Objekte den Verlust dominieren, und stabilisiert das Training bei sehr unterschiedlich großen Boxen.

- **Klassifikationsverlust (loss3):**
  Für jede Zelle des Gitters wird berechnet, wie stark die vorhergesagte Klassenzugehörigkeit von der tatsächlichen Klasse abweicht. Dieser Term stellt sicher, dass die Boxen nicht nur an der richtigen Position liegen, sondern auch die richtige Objektklasse repräsentieren.
- **Objektkonfidenzverlust (loss5):**
  Der vierte Term bewertet die Genauigkeit der Vorhersage, ob in einer Box tatsächlich ein Objekt vorhanden ist. Dazu wird der vorhergesagte Confidence-Wert mit dem Sollwert verglichen. Nur Boxen, die tatsächlich ein Objekt enthalten, gehen in diesen Term ein.
- **Kein-Objekt-Konfidenzverlust (loss5):**
  Da in einem Bild sehr viele Boxen leer bleiben (kein Objekt enthalten), gibt es einen zusätzlichen Term für diese Fälle. Dieser wird schwächer gewichtet, um zu verhindern, dass das Modell nur lernt, „kein Objekt“ vorherzusagen.
  (vgl. Cao et al. 2023, S. 9201f).

Cao et al. verwendeten für das Training des YOLOv3 Netzwerks 1000 Sonarbilder, die aus verschiedenen Sonarsequenzen ausgewählt wurden. Der Datensatz wurde in 800 Trainingsdaten und 200 Testdaten aufgeteilt. Die Sonarbilder zeigen Stahlrohre in einem Experimentierpool sowie Äste in einem Fluss.

Da Cao et al. keine detaillierten Angaben zum Training machen, wird ergänzend die Arbeit von Zhang et al. (2022) herangezogen. Diese verwenden ein erweitertes YOLOv5 Network zur Objekterkennung in FLS-Daten. Grundlage ist ein Transfer Learning Ansatz, wie in Abbildung 27 dargestellt: Das verwendete YOLOv5 Model ist auf einem COCO-Datensatz vortrainiert und wird anschließend mit einem kleinen FLS-Datensatz von 3240 Trainingsbildern mit acht Zielklassen feinjustiert (vgl. Zhang et al. 2022, S. 18027). Um die Abweichungen der Objektgrößen und Seitenverhältnisse zwischen Sonar- und optischen Bildern zu berücksichtigen, wurden die Anker-Boxen durch ein angepasstes k-means-Clustering neu bestimmt. Dabei setzten die Autoren die Intersection over Union (IoU) als Distanzmaß. Diese Anpassung erhöhte sowohl die Regressionsgenauigkeit als auch die Detektionsgeschwindigkeit vgl. ebd., S. 18028) Das Training wurde mit PyTorch durchgeführt und konvergierte stabil nach rund 100 Epochen (vgl. ebd., S. 18030).

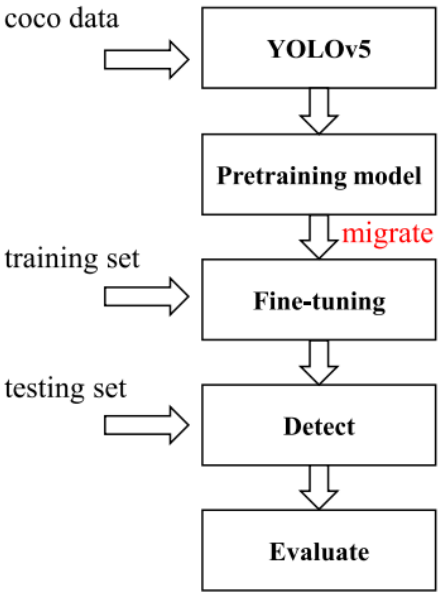


Abbildung 27: Trainingsablauf des Transfer Learning Ansatzes von Zhang et al. 2022, S. 18027

Nach der initialen Detektion potenzieller Hindernisbereiche durch YOLOv3 wird in dem von Cao et al. vorgeschlagenen Verfahren ein Schwellwert-Segmentierungsverfahren angewendet, um die genauen Objektflächen zu identifizieren. Im Gegensatz zu klassischen globalen Thresholding-Ansätzen, die für jedes Bild einen festen Schwellwert annehmen, basiert dieser Algorithmus auf einer dynamischen Anpassung des optimalen Schwellwerts T. Das Verfahren umfasst zwei Kernelemente: eine iterative Bestimmung des optimalen Grauwert-Schwellenwerts basierend auf der Maximierung der Interklassenvarianz, und eine adaptive Aktualisierung dieses Schwellenwerts, die kleine, rauschbedingte Regionen gezielt unterdrückt (Small Area Supression).

Zunächst wird das Grauwert-Histogramm des Sonarbildes gebildet. Dabei werden alle Pixelwerte in L Graustufenebenen $\{1,2,\dots,L-1\}$ eingeteilt, und die Häufigkeit n_i jedes Grauwerts i bestimmt. Durch Normierung ergibt sich die Wahrscheinlichkeitsverteilung:

$$p_i = \frac{n_i}{\sum_{i=0}^{L-1} n_i} \quad , p_i \geq 0$$

Auf dieser Basis werden drei Grauwertgrenzen k_1, k_2, k_3 definiert, mit denen das Histogramm in vier Intervalle unterteilt wird. Dabei ist k_2 der gewichtete Mittelwert des gesamten Histogramms, während k_2 und k_3 die Mittelwerte der linken und rechten Histogrammhälften darstellen:

$$k_2 = \sum_{i=0}^{L-1} i \cdot p_i, \quad k_1 = \frac{\sum_{i=0}^{k_2} i \cdot p_i}{\sum_{i=0}^{k_2} p_i}, \qquad k_3 = \frac{\sum_{k_2+1}^{L-1} i \cdot p_i}{\sum_{k_2+1}^{L-1} p_i}$$

Für jede dieser Intervallgrenzen wird die Interklassenvarianz σ_B^2 berechnet:

$$\sigma_B^2 = w_0(u_0 - u_T)^2 + w_1(u_1 - u_T)^2$$

Wobei w_0, w_1 die Wahrscheinlichkeiten des rechten und linken Intervalls, u_0, u_1 die Mittelwerte des rechten und linken Intervalls und u_T den Gesamtmittelwert darstellen. Diese berechnen sich für k_2 wie folgt:

$$w_0 = \sum_{i=k_1}^{k_2} p_i\,, \qquad u_0 = \sum_{i=k_1}^{k_2} \frac{i \cdot p_i}{w_0}$$

$$w_1 = \sum_{i=k_2}^{k_3} p_i\,, \qquad u_1 = \sum_{i=k_2}^{k_3} \frac{i \cdot p_i}{w_1}$$

$$u_T = w_0 \cdot u_0 + w_1 \cdot u_1$$

Die Berechnung für k_1 und k_3 folgt analog. Wenn die Interklassenvarianz an k_2 größer ist als die an k_1 und k_3, wurde mit $T = k_2$ der optimale Schwellwert gefunden. Liegt das

Maximum der drei Interklassenvarianzen stattdessen bei k_1 bzw. k_3 wird der Prozess im linken bzw. rechten Teilintervall wiederholt.

Dieses Vorgehen reduziert die Rechenkomplexität: Anstatt das gesamte Histogramm mit allen möglichen Schwellenwerten zu durchsuchen, beschränkt der Algorithmus die Suche schrittweise auf immer kleinere Intervalle.

Trotz dieser Optimierung treten in Sonarbildern jedoch häufig viele kleine, isolierte Segmente auf, die durch Sedimente oder Fischschwärme entstehen. Diese Rauschbereiche führen zu einer Vielzahl von falsch erkannten Objekten. Um dem entgegenzuwirken, ergänzen Cao et al. den Algorithmus um ein Verfahren zu Unterdrückung kleiner Flächen (Small Area Suppression). Dazu wird für jede zusammenhängende Region die Fläche A_i bestimmt. Regionen mit weniger als 50 Pixeln werden als Rauschen eingestuft und entfernt.

Weiterhin wird die Zahl solcher kleinen Regionen N_{50} gezählt. Überschreitet sie einen Schwellwert $\tau = 20$, wird angenommen, dass das Segmentierungsergebnis noch immer stark verrauscht ist. In diesem Fall wird der Schwellwert T erneut aktualisiert, indem die Berechnung der Interklassenvarianz mit den reduzierten Regionen wiederholt wird. Dieser iterative Prozess wird solange fortgesetzt, bis $N_{50} < \tau$ erreicht ist.

5.4.2 Eigene Umsetzung

Wie bei Cao et al. wurde aus den aufgenommenen rosbags (siehe Kapitel 4.3) ein Datensatz von 1000 Sonarbildern ausgewählt. Dabei wurden Redundanzen aufgrund ähnlicher aufeinanderfolgender Bilder vermieden und die „Verhältnisse“ der Originaldaten beibehalten, indem der Anteil der unterschiedlichen Reihenmanöver, Objekte und Sonarreichweiten im selben Verhältnis steht. Die Daten unterteilen sich in einen Trainingsdatensatz mit 800 Bildern und einen Testdatensatz von 200 Bildern.

Die Bilder wurden durch den *fls_message_node* in Bezug auf Kontrast, Helligkeit, etc. vorverarbeitet. Anschließend wurden die Sonardaten in kartesische Koordinaten transformiert, um die reale Form der Objekte zu zeigen. Wie im Vorgehen von Cao et al. wurde auf die Sonarbilder ein Medianfilter mit einer Größe von 5x5 Pixeln angewendet. Da das YOLOv3 Netzwerk einen quadratischen Input erwartet, wurden die Sonarbilder auf 416x416 Pixel skaliert. Da die Breite deutlich größer als die Höhe ist und die Seitenverhältnisse erhalten bleiben sollten, wurde der obere und untere Bildrand mit schwarzen Pixeln gepaded.

Für das Training wurden ausgewählten Sonarbilder zunächst mit zugehörigen Annotationen in Form von Bounding Boxen versehen, die potenzielle Objektbereiche angeben. Diese Koordinaten der Bounding Boxen werden typischerweise normalisiert, d.h. sie werden nicht in Pixelwerten, sondern relativ zur Bildgröße gespeichert. Die Annotation erfolgte mithilfe des Online Tool CVAT und ist in Abbildung 28 dargestellt.

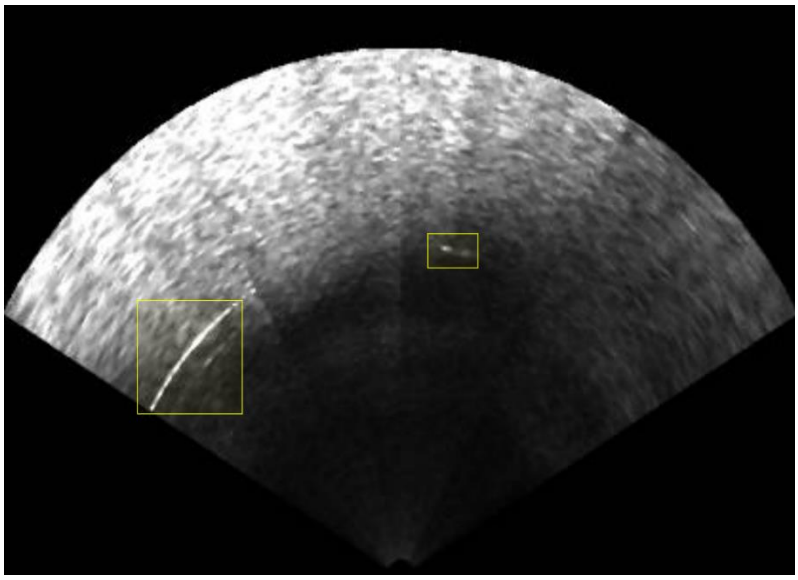


Abbildung 28: Annotation vorhandener Hindernisse mit Bounding Boxen

Das Training wurde mit der Softwareumgebung Darknet durchgeführt, die speziell für das Training von YOLO-Netzwerken optimiert ist. Darknet ist ein in C und CUDA implementiertes Framework, das eine effiziente Nutzung von GPU-Ressourcen erlaubt und dadurch ein besonders schnelles Training ermöglicht. Zur Durchführung des Trainings wurde ein Google Colab Notebook verwendet, in dem die Darknet-Umgebung installiert und kompiliert wurde.

Als Ausgangspunkt diente das vortrainierte Gewichtungsset *yolov3.weights*, das ursprünglich auf dem COCO-Datensatz trainiert wurde. Um eine stabile Feinabstimmung auf die vergleichsweise kleine Sonar-Datenbasis zu ermöglichen, wurden die Schichten des Darknet-Backbones eingefroren. Dies erfolgte durch Setzen des Parameters *learning_rate=0* für die entsprechenden Layer in der Konfigurationsdatei, sodass nur die nachgelagerten Detektionsköpfe weitertrainiert wurden.

Vor dem eigentlichen Training wurde zunächst ein Satz geeigneter Anchor-Boxen berechnet, um die Detektionsarchitektur an die charakteristischen Objektgrößen des vorliegenden Datensatzes anzupassen. Dazu wurde das in Darknet integrierte k-Mean Clustering-Verfahren eingesetzt, das typische Boxdimensionen aus den annotierten Ground-Truth-Bounding-Boxes extrahiert. Diese Anchor-Boxen dienen im Training als Ausgangspunkte für die Vorhersage von Objektgrenzen und stellen damit sicher, dass das Netzwerk bereits zu Beginn über plausible Hypothesen für die verschiedenen Objektgrößen verfügt.

Die zentralen Trainingsparameter wurden in der Darknet-Konfigurationsdatei festgelegt. Dazu zählten unter anderem die Batchgröße (64 Bilder), die in kleinere Subdivisions (16) unterteilt wurde, um die GPU effizient zu nutzen, sowie die Eingabebildgröße (416x416 Pixel). Die Lernrate wurde auf 0.0015 festgelegt und über eine sogenannte Burn-in-Phase von 600 Iterationen schrittweise auf diesen Wert erhöht, um eine stabile Anfangsphase des Trainings zu gewährleisten. Anschließend blieb die Lernrate konstant und wurde bei

80% und 90% der maximalen Iterationen jeweils um den Faktor 0.1 abgesenkt, sodass eine feinere Anpassung des Modells an die Daten in der Endphase möglich wurde.

Die Konfiguration enthält zudem Parameter für die Datenaugmentation. Diese steuern, in welchem Maße die Eingabebilder während des Trainings zufällig variiert werden. Mit dem Wert angle=0 wurden Rotationen vollständig deaktiviert, sodass die Bilder ohne Drehungen in das Netz eingespeist wurden. Der Parameter *saturation*=1.0 legt fest, dass keine Veränderung der Farbsättigung vorgenommen wird, was insbesondere bei Graustufen-Sonarbildern sinnvoll ist. Mit *exposure*=1.1 wurde eine leichte Variation der Belichtung zugelassen, wodurch die Pixelintensitäten um etwa zehn Prozent schwanken konnten. Dies erhöht die Robustheit des Modells gegenüber Unterschieden in Helligkeit und Kontrast, wie sie bei Sonardaten häufig auftreten. Schließlich bestimmt der Parameter *hue*=0.0, dass keine Veränderungen des Farbtons vorgenommen, werden: auch dies ist für graustufige Eingabedaten angemessen. Insgesamt sorgen diese Einstellungen dafür, dass die grundlegende Struktur der Trainingsbilder erhalten bleibt, während zugleich eine gewisse Variabilität in der Helligkeit eingebracht wird, um eine bessere Generalisierungsfähigkeit des Netzes zu fördern.

Cao et al. schreiben, dass das Training innerhalb von 2000 training steps schnell konvergierte und eine weitere Erhöhung der Iterationszahl keinen signifikanten Leistungszuwachs mehr brachte. Dies stelle sich auch in diesem Training als optimaler Wert heraus. Während des Trainings wurde die Performance des Modells regelmäßig anhand der separaten Testdaten validiert. Als zentrale Gütemaßzahl diente die *mean Average Precision (mAP)* bei einem Intersection-over-Union-Schwellenwert von 0.5 (vgl. Kapitel 6.1). Der Trainingsverlauf ist in Abbildung 29 grafisch dargestellt.

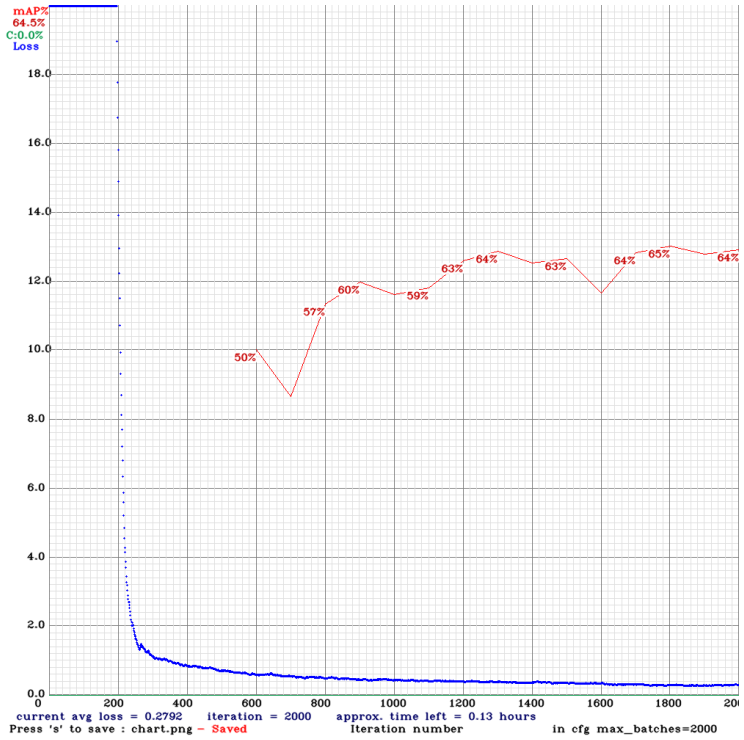


Abbildung 29: Trainingsverlauf des YOLOv3 Netzwerks

Für die Inferenz wurde das trainierte Netzwerk über das DNN-Modul von OpenCV in die Anwendung eingebunden. Dieses Modul stellt eine generische Schnittstelle für verschiedene neuronale Netze dar und erlaubt es, Modelle aus externen Frameworks (z. B. Darknet, TensorFlow, Caffe oder ONNX) zu laden und auf Bildern auszuführen, ohne dass die ursprüngliche Trainingsumgebung benötigt wird. Damit entfällt die Abhängigkeit von Darknet während der Laufzeit, was die Integration in C++-Anwendungen erheblich erleichtert und zugleich die Portabilität des Codes erhöht.

Die Sonarbilder werden wie eingangs beschrieben in kartesische Koordinaten transformiert, mit einem Medianfilter vorverarbeitet und auf eine Größe von 416x416 Pixel skaliert. Anschließend werden sie mit der Funktion *cv::dnn::blobFromImage* normalisiert und in ein Tensorformat („Blob“) umgewandelt, das als Eingabe für das Netz dient. Nach dem Forward-Pass werden die drei Ausgabetensoren der YOLO-Heads verarbeitet: Jede Zeile beschreibt einen Kandidatenbox durch Zentrum, Breite, Höhe und Objektwahrscheinlichkeit. Kandidaten unterhalb eines parametrierbaren Konfidenzschwellwerts werden verworfen.

Da ein neuronales Detektionsnetz wie YOLOv3 für ein einzelnes Objekt in der Regel eine Vielzahl an überlappenden Bounding Boxes mit unterschiedlichen Konfidenzwerten erzeugt, wird im Anschluss eine Non-Maximum Supression (NMS) durchgeführt. Dieses Verfahren dient der Reduktion redundanter Boxen und sorgt dafür, dass jedes Objekt im Bild durch genau eine Bounding Box repräsentiert wird. Dabei werden zunächst alle Kandidaten nach ihrem Konfidenzwert sortiert. Die Box mit der höchsten Konfidenz wird beibehalten, während alle anderen Boxen, die mit ihr einen hohen Überschneidungsgrad aufweisen, verworfen werden. Die verbleibenden Boxen werden zurück in die Koordinaten des Originalbildes transformiert und mit Konfidenzwert und Klassen-ID ausgegeben.

Durch die Transformation der Bounding Boxen in Koordinaten des Originalbilds kann die anschließende Objektsegmentierung auf der vollen Auflösung der Sonarbilder erfolgen. Die Segmentierung wurde entsprechend der Darstellung von Cao et al. implementiert. Für die effiziente Berechnung von Klassengewichten und -mitteln werden die die kumulativen Wahrscheinlichkeiten und Intensitätssummen abgespeichert. Diese können dann für jeden potenziellen Schwellenwert k in konstanter Zeit abgerufen werden, anstatt sie für jede Iteration erneut durch Summation über das gesamte Histogramm bestimmen zu müssen. Auf diese Weise wird die Schwellwertsuche signifikant beschleunigt, was insbesondere bei der Verarbeitung vieler Bildausschnitte in Echtzeit relevant ist.

Zur Unterdrückung kleiner Segmente wird die OpenCV-Funktion *cv::connectedComponentsWithStats* verwendet. Diese Funktion analysiert eine binäre Maske und weist jedem Pixel ein Label zu, das angibt, zu welcher zusammenhängenden Komponente es gehört. Dabei wird in der vorliegenden Implementierung die 8-Nachbarschaft genutzt, da sie eine vollständige Erfassung diagonal verbundener Strukturen erlaubt. Zusätzlich liefert die Funktion die Fläche jeder Komponente in Pixeln

sowie deren Begrenzungsrechtecke und Schwerpunkte. Auf Grundlage der ermittelten Flächen wird – analog zu Cao et al. – der Schwellwert iterativ angepasst. Darüber hinaus werden alle Komponenten, deren Fläche unterhalb eines definierten Grenzwerts liegt, konsequent aus der Maske entfernt, sodass kleinere Störsegmente nicht mehr in die weitere Verarbeitung eingehen.

Die folgenden ROS-Parameter steuern die Verarbeitung durch das YOLOv3 Netzwerk sowie die Segmentierung:

- ***conf_threshold = 0.8:*** Mindestkonfidenz, ab der eine Bounding Box weiter betrachtet wird
- ***nms_threshold =0.4:*** IoU-Schwelle zur Unterdrückung mehrfacher Boxen
- ***area_threshold = 20:*** Flächengrenzwert, unter dem Segmente verworfen werden
- ***tau = 20:*** maximal erlaubte Anzahl kleiner Segmente (Kriterium für Schwellwertanpassung)
- ***max_updates = 20:*** Begrenzung der Iteration bei Schwellwertanpassung

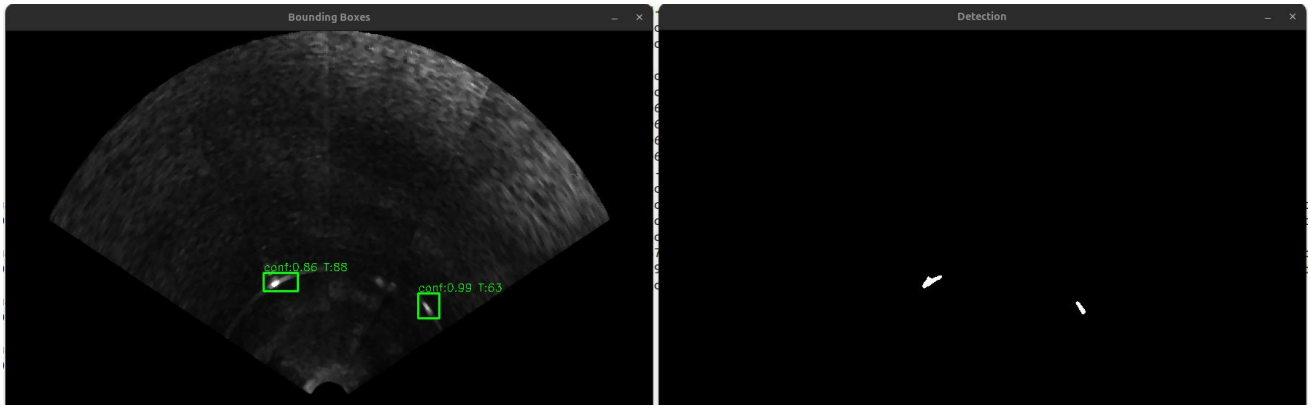


Abbildung 30: Verarbeitungsschritte des NN-Detektors

Abbildung 30 zeigt links die durch YOLOv3 erkannten potenziellen Objektregionen, die mit Bounding Boxen gekennzeichnet sind. Zu jeder Box sind die jeweilige Konfidenz und der ermittelte Schwellwert T angegeben. Rechts ist die finale Detektion nach Anwendung der Segmentierung dargestellt.

6 Evaluation

In diesem Abschnitt werden die ausgewählten Detektionsverfahren anhand der zuvor definierten Metriken quantitativ bewertet. Ziel ist es, eine objektive Vergleichsbasis zu schaffen, bevor in Kapitel 7 eine übergreifende Diskussion der Ergebnisse erfolgt. Die Metriken umfassen pixel- und instanzbasierte Genauigkeitsmaße (IoU, Precision, Recall, F1) sowie Effizienzmaße (Laufzeit, FPS). Während hier die reinen Resultate vorgestellt werden, folgt die Interpretation in Hinblick auf Ursachen und Implikationen im nachfolgenden Diskussionsteil.

6.1 Bewertungskriterien

Ziel der Hinderniserkennung zur Kollisionsvermeidung ist die Ermöglichung einer sicheren Navigation. Dazu müssen Hindernisse zuverlässig entdeckt werden und ihre räumliche Lage hinreichend genau geschätzt sein, damit Ausweichmanöver rechtzeitig eingeleitet werden können. Zudem sollten Falschalarme möglichst niedrig bleiben, um unnötige Kursänderungen oder Stopps zu vermeiden und die Missionsfähigkeit des Fahrzeugs nicht einzuschränken. Für die Bewertung der eingesetzten Detektionsverfahren wird daher eine Kombination verschiedener Metriken notwendig, die sowohl die Flächenübereinstimmung der Segmente, die klassische Erkennungsleistung als auch die Objektzahl und -trennung abbilden.

Grundlage fast aller dieser Kennzahlen bilden die folgenden Konzepte:

- **True Positive (TP):** korrekt als Hinder bzw. Teil eines Hindernisses erkannt
- **False Positive (FP):** fälschlicherweise als Hindernis markiert, obwohl in der Ground Truth kein Objekt vorliegt
- **False Negative (FN):** fälschlicherweise nicht als Hindernis erkannt, obwohl in der Ground Truth ein Objekt vorliegt
- **True Negative (TN):** korrekt als Hintergrund erkannt (kein Hindernis)

Als Standardmaß für die Bewertung der Segmentierung ist in der Literatur die Intersection over Union (IoU) etabliert. Diese misst das Verhältnis der Schnittmengen zur Vereinigungsmenge zweier Masken:

$$IoU(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{TP}{TP + FP + FN}$$

Ein IoU-Wert von 1 bedeutet perfekte Überdeckung, ein Wert von 0 völlige Fehlklassifikation (vgl. Cao et al. 2023, S. 9203f).

Weitere klassische Messwerte sind Precision und Recall. Recall beschreibt den Anteil der tatsächlich vorhandenen Hindernisse, die korrekt erkannt wurden. Ein hoher Recall bedeutet, dass kaum Hindernisse, die korrekt erkannt wurden. Ein hoher Recall bedeutet, dass kaum Hindernisse übersehen werden, was für sicherheitskritische Anwendungen wie die Kollisionsvermeidung essenziell ist. Precision hingegen misst den Anteil der markierten Hindernisse, die tatsächlich korrekt sind. Eine hohe Precision signalisiert also, dass wenige Fehlalarme entstehen. Die Werte werden wie folgt berechnet (vgl. Hou et al. , S. 492; Goodfellow et al. 2016, S. 423; Russell & Norvig, S. 900):

$$Recall = \frac{TP}{TP + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

Da sich Recall und Precision in der Praxis oft gegenläufig verhalten, wird häufig der *F-Score* herangezogen, der als harmonisches Mittel beider Kennzahlen berechnet wird:

$$F_\beta = \frac{(1+\beta^2) \cdot Precision \cdot Recall}{\beta^2 \cdot Precision + Recall}$$

Über den Parameter β kann die Gewichtung von Precision bzw. Recall angepasst werden. Recall wird β -mal so stark gewichtet wie Precision. Die gängigste Form ist jedoch der F1-Score, der beide Werte gleich gewichtet. Der F-Score balanciert zwischen Vollständigkeit und Genauigkeit der Detektion. Das ist besonders nützlich, wenn ein Verfahren z. B. sehr konservativ (hohe Precision, niedriger Recall) oder sehr großzügig (umgekehrt) segmentiert.

Alle diese Kennzahlen können entweder pixelbasiert oder instanzbasiert berechnet werden. In der pixelbasierten Variante wird jeder Bildpunkt einzeln betrachtet, sodass die Kennzahlen, die Gesamtmenge aller korrekt oder falsch klassifizierten Pixel widerspiegeln. In der instanzbasierten Variante werden zunächst zusammenhängende Objektregionen („Instanzen“) identifiziert und anschließend mit den Ground-Truth-Instanzen verglichen. Hierbei wird ein IoU-Schwellenwert verwendet, um zu entscheiden, ob eine vorhergesagte Instanz als korrekt erkannt gilt. Instanzbasiert lassen sich so insbesondere Fehler wie *Fragmentierung* (ein Objekt wird in mehrere Vorhersagen zerlegt) oder *Merging* (mehrere Objekte werden zu einem verschmolzen) sichtbar machen.

Währen Precision, Recall und der F-Score typischerweise für eine Entscheidungsschwelle berechnet werden, ist es oft sinnvoll, die Modellleistung über eine Variation der Schwelle hinweg zu betrachten. Ein etabliertes Maß hierfür ist *die mean Average Precision* (mAP). Grundlage ist die *Precision-Recall-Kurve* $p(r)$, welche die Abhängigkeit der Precision vom Recall darstellt. Die Average Precision (AP) wird dabei als Fläche unter dieser Kurve bestimmt:

$$AP@\alpha = \int_0^1 p(r)dr$$

Wobei α den zugrunde gelegten IoU-Schwellwert bezeichnet. Wird AP über mehrere Objektklassen hinweg gemittelt, ergibt sich die *mean Average Precision* für eine feste IoU-Schwelle

$$mAP@\alpha = \frac{1}{n}\sum_{i=1}^{n} AP_{i,\alpha}$$

Wobei n die Anzahl der Klassen und $AP_{i,\alpha}$ die AP der Klasse i bei Schwelle α bezeichnet.

Im COCO-Benchmark wird die mAP zusätzlich über mehrere IoU-Schwellen gemittelt. Dabei werden zehn Schwellenwerte zwischen 0,50 und 0,95 in Schritten von 0,05 berücksichtigt:

$$mAP = \frac{1}{10} \sum_{\alpha \epsilon \{0.5, 0.55, \ldots, 0.95\}} mAP@\alpha$$

Die mAP belohnt Methoden, deren Leistung über eine Bandbreite an Genauigkeitsanforderungen hinweg konsistent bleibt und stellt damit ein umfassendes Maß für die Leistungsfähigkeit eines Detektions- oder Segmentierungsverfahrens dar (vgl. Hou et al. , S. 492; Cao et al. 2023, S. 9204).

Neben den bisher aufgeführten Metriken, die Auskunft über die Genauigkeit der Detektionsmethoden geben, ist für die geplante Echtzeitanwendung auch deren Effizienz von entscheidender Bedeutung. Ein Verfahren, das zwar hohe Werte bei IoU oder mAP erzielt, jedoch nicht in der Lage ist, Datenströme in Echtzeit zu verarbeiten, ist für die praktische Hinderniserkennung im Forward-Looking-Sonar nur eingeschränkt nutzbar.

Zur Beurteilung der Effizienz werden daher insbesondere die Inferenzzeit pro Bild sowie die daraus resultierende Bildfrequenz (*Frames per Second, FPS*) betrachtet. Während die Inferenzzeit Auskunft über die mittlere Laufzeit des Modells für ein einziges Eingabebild gibt, beschreibt die FPS den Gesamtdurchsatz des Systems und berücksichtigt auch, ob mehrere Bilder gleichzeitig oder parallel verarbeitet werden. Zusätzlich kann die CPU-Auslastung während der Ausführung ein wichtiger Faktor sein, da autonome Unterwasserfahrzeuge typischerweise über begrenzte Rechen- und Energieressourcen verfügen.

Damit ergibt sich ein umfassender Bewertungsrahmen, der Genauigkeit und Effizienz gleichermaßen berücksichtigt und so eine fundierte Entscheidung über die Praxistauglichkeit der getesteten Methoden ermöglicht.

6.2 Vergleich an Testdaten

Für den Vergleich wurden die rosbags der vier Test-Sequenzen abgespielt und die Detektionsergebnisse für diejenigen Bilder exportiert, die im Testdatensatz enthalten sind. Dadurch konnten alle Detektoren auf derselben Bildgrundlage ausgewertet werden, ohne dass die Ergebnisse der OGM-Methode verfälscht würden, die auf einer sequentiellen Verarbeitung der Frames beruht.

Die Ergebnisse der pixelbasierten Evaluation der einzelnen Testsequenzen sind in Tabelle 3a – 3d dargestellt. Die Werte für TP, FP und FN wurden über alle Frames des jeweiligen Testdatensatzes akkumuliert. IoU, Recall, Precision und F1 Score wurden pro Frame berechnet und anschließend gemittelt. Tabelle 3e stellt die Detektoren gegenüber und fasst dazu die Ergebnisse über alle Testsequenzen hinweg zusammen. Hierbei wurden TP, FP und FN jeweils aufsummiert, während die übrigen Kennzahlen auf Basis dieser Summen berechnet wurden. Der IoU-Wert ergibt sich als Mittelwert der IoUs der einzelnen Sequenzen.

3a BG	pixelbasiert						
Testsequenz	**IoU**	**TP**	**FP**	**FN**	**Recall**	**Precision**	**F1**
O_obj1_30m	0,54	481	137	2099	0,06	0,13	0,08
P_obj1_15m	**0,58**	5616	2077	3029	**0,72**	**0,80**	**0,70**
P_obj2_15m	0,29	162	6	1504	0,08	0,16	0,10
P_obj2_30m	0,39	16	0	3794	0,00	0,02	0,00

3b PEAK	pixelbasiert						
Testsequenz	**IoU**	**TP**	**FP**	**FN**	**Recall**	**Precision**	**F1**
O_obj1_30m	**0,63**	932	163	1648	**0,17**	0,24	**0,19**
P_obj1_15m	0,45	6745	9221	1900	0,68	0,36	0,46
P_obj2_15m	0,24	147	18	1519	0,03	0,04	0,03
P_obj2_30m	0,32	331	496	3479	0,05	0,09	0,06

3c OGM	pixelbasiert						
Testsequenz	**IoU**	**TP**	**FP**	**FN**	**Recall**	**Precision**	**F1**
O_obj1_30m	0,47	872	8737	1708	0,14	0,04	0,06
P_obj1_15m	0,21	6105	46891	2540	0,62	0,09	0,16
P_obj2_15m	0,19	137	584	1529	0,06	0,02	0,03
P_obj2_30m	0,37	156	453	3654	0,02	0,02	**0,19**

3d NN	pixelbasiert						
Testsequenz	**IoU**	**TP**	**FP**	**FN**	**Recall**	**Precision**	**F1**
O_obj1_30m	0,62	825	27	1755	0,14	**0,30**	0,18
P_obj1_15m	0,56	5064	1448	3581	0,52	0,66	0,55
P_obj2_15m	**0,40**	476	195	1190	**0,22**	**0,44**	**0,27**
P_obj2_30m	**0,45**	374	7	3436	**0,07**	**0,25**	0,10

3e	pixelbasiert						
Detektor	**IoU**	**TP**	**FP**	**FN**	**Recall**	**Precision**	**F1**
BG	0,45	6275	2220	10426	0,38	0,74	0,50
PEAK	0,41	8155	9898	8546	**0,49**	0,45	0,47
OGM	0,31	7270	56665	9431	0,44	0,11	0,18
NN	**0,51**	6739	1677	9962	0,40	**0,80**	**0,54**

Tabelle 3: pixelbasierte Evaluation anhand der Testdaten

Die Ergebnisse zeigen deutliche Unterschiede zwischen den Detektoren. Während BG und NN eine höhere Precision zeigen, weisen PEAK und NN eher eine höheren Recall auf. Auffällig sind zudem Unterschiede zwischen den einzelnen Testsequenzen: So erzielt der BG-Detektor in der Sequenz *P_obj1_15m* mit einem F1-Score von 0,70 das mit Abstand beste Resultat. In den Sequenzen *P_obj2_15m* und *P_obj2_30m* hingegen gelingt keinem Verfahren eine gute Hinderniserkennung; hier erreicht der NN-Detektor mit F1-Scores von 0,27 bzw. 0,10 noch die vergleichsweise besten Werte.

Im Gesamtvergleich zeigt der NN-Detektor die stärksten Ergebnisse: Er erreicht mit 0,51 den höchsten IoU-Mittelwert, mit 0,80 die höchste Precision sowie mit 0,54 den höchsten

F1-Score. Der BG-Detektor liegt mit einem F1-Score von 0,50 und einer Precision von 0,74 ebenfalls im oberen Bereich. Der PEAK-Detektor erzielt den höchsten Recall (0,49) und insgesamt einen F1-Score von 0,47. Der OGM-Detektor bleibt insgesamt hinter den anderen Verfahren zurück; er erreicht nur eine mittlere IoU von 0,31, eine Precision von 0,11 und einen F1-Score von 0,16.

Für die instanzbasierte Evaluation wurden zusammenhängende Pixelbereiche in der Ground Truth Maske als Instanzen definiert und anhand ihrer Position mit Instanzen in der Detektionsmaske abgeglichen. Das Matching erfolgte mit einem Greedy-Matching-Verfahren: Hierbei wird zunächst eine IoU-Matrix zwischen allen Ground-Truth- und Detektionsinstanzen berechnet. Anschließend werden die Paare mit den höchsten IoU-Werten sukzessive ausgewählt. Nach einem erfolgreichen Match werden die betreffenden Instanzen aus der weiteren Betrachtung entfernt, sodass jede Instanz höchstens einmal zugeordnet wird. Als korrekt erkannt (TP) gilt ein Match, wenn die IoU über dem gewählten Schwellwert von 0,5 liegt. Liegt ein Match darunter, zählen die Ground-Truth-Instanzen als FN und Detektionsinstanzen als FP. Für jedes Match wurde unabhängig von Schwellwert zusätzlich die IoU erfasst, sodass sich ein mittlerer IoU-Wert pro Testsequenz berechnen ließ.

Die Tabellen 4a-d zeigen die Metriken jedes Detektors für die verschiedenen Testsequenzen. TP, FP und FN sind über alle Frames einer Testsequenz aufsummiert. Recall, Precision und F1-Score wurden über die gesamte Testsequenz berechnet. Tabelle 4e stellt die Detektoren wie gehabt gegenüber, indem IoU gemittelt sowie TP, FP und FN aufsummiert und Recall, Precision und F1-Score daraus berechnet wurden.

Der BG-Detektor zeigt weiterhin gute Ergebnisse in der Sequenz *P_obj1_15m*, wo er mit einer IoU von 0,63 den höchsten Wert erzielt. In allen anderen Sequenzen wird er jedoch vom PEAK-Detektor übertroffen, der auch insgesamt die höchste Übereinstimmung mit der Ground Truth aufweist (IoU = 0,63). In den 30-m-Sequenzen erreicht PEAK zudem die besten Werte für Recall, Precision und F1-Score. In den 15-m-Sequenzen liefert dagegen der NN-Detektor die besten Resultate mit F1-Scores von 0,72 (*P_obj1_15m*) bzw. 0,32 (*P_obj2_15m*). Im Gesamtergebnis erreicht NN mit einem Recall von 0,37, einer Precision von 0,63 und einem F1-Score von 0,47 die beste Balance aller Verfahren. Der OGM-Detektor konnte beim gewählten IoU-Schwellenwert keine Hindernisse zuverlässig erkennen, was sich auch in seiner niedrigen mittleren IoU von 0,20 widerspiegelt.

4a BG		instanzbasiert mit IoU > 0.5					
Testsequenz	**IoU**	**TP**	**FP**	**FN**	**Recall**	**Precision**	**F1**
O_obj1_30m	0,55	5	3	28	0,15	0,63	0,24
P_obj1_15m	**0,63**	47	30	33	0,59	0,61	0,60
P_obj2_15m	0,47	2	2	19	0,10	0,50	0,16
P_obj2_30m	0,16	0	1	43	0,00	0,00	0,00

4b PEAK		instanzbasiert mit IoU > 0.5					
Testsequenz	**IoU**	**TP**	**FP**	**FN**	**Recall**	**Precision**	**F1**
O_obj1_30m	**0,66**	12	2	21	**0,36**	**0,86**	**0,51**
P_obj1_15m	0,46	22	34	58	0,28	0,39	0,32
P_obj2_15m	**0,81**	1	0	20	0,05	**1,00**	0,09
P_obj2_30m	**0,61**	5	6	38	**0,12**	**0,45**	**0,19**

4c OGM		instanzbasiert mit IoU > 0.5					
Testsequenz	**IoU**	**TP**	**FP**	**FN**	**Recall**	**Precision**	**F1**
O_obj1_30m	0,20	0	17	33	0	0	0
P_obj1_15m	0,12	0	52	80	0	0	0
P_obj2_15m	0,23	0	3	21	0	0	0
P_obj2_30m	0,22	0	4	43	0	0	0

4d NN		instanzbasiert mit IoU > 0.5					
Testsequenz	**IoU**	**TP**	**FP**	**FN**	**Recall**	**Precision**	**F1**
O_obj1_30m	0,62	9	10	24	0,27	0,47	0,35
P_obj1_15m	0,62	50	8	30	**0,63**	**0,86**	**0,72**
P_obj2_15m	0,47	6	10	15	**0,29**	0,38	**0,32**
P_obj2_30m	0,37	1	11	42	0,02	0,08	0,04

4e		instanzbasiert mit IoU > 0.5					
Detektor	**IoU**	**TP**	**FP**	**FN**	**Recall**	**Precision**	**F1**
BG	0,45	54	36	123	0,31	0,60	0,40
PEAK	**0,63**	40	42	137	0,23	0,49	0,31
OGM	0,19	0	76	177	0	0	0
NN	0,52	66	39	111	**0,37**	**0,63**	**0,47**

Tabelle 4: Instanzbasierte Evaluation anhand der Testdaten mit IoU-Threshold 0.5

Da für die Kollisionsvermeidung nicht die exakte Form der Hindernisse, sondern vielmehr deren zuverlässige Erkennung entscheidend ist, wurden die Detektoren zusätzlich ohne IoU-Schwellenwert evaluiert. Das Matching der Instanzen wurde dabei manuell nachgeprüft, um Fehlzuordnungen auszuschließen. Die Metriken berechnen sich analog zu den vorangegangenen Auswertungen; die Ergebnisse sind in den Tabellen 5a-5d für die einzelnen Testsequenzen sowie in Tabelle 5e für den Gesamtvergleich dargestellt.

5a BG	instanzbasiert manuell, ohne IoU					
Testsequenz	**TP**	**FP**	**FN**	**Recall**	**Precision**	**F1**
O_obj1_30m	8	0	25	0,24	**1,00**	0,38
P_obj1_15m	60	17	15	**0,80**	0,77	0,78
P_obj2_15m	4	0	17	0,19	**1,00**	0,32
P_obj2_30m	1	0	42	0,02	**1,00**	0,05

5b PEAK	instanzbasiert manuell, ohne IoU					
Testsequenz	**TP**	**FP**	**FN**	**Recall**	**Precision**	**F1**
O_obj1_30m	14	0	19	0,42	**1,00**	0,59
P_obj1_15m	55	1	20	0,73	**0,98**	0,83
P_obj2_15m	1	0	20	0,05	**1,00**	0,09
P_obj2_30m	5	6	38	0,12	0,45	0,19

5c OGM	instanzbasiert manuell, ohne IoU					
Testsequenz	**TP**	**FP**	**FN**	**Recall**	**Precision**	**F1**
O_obj1_30m	13	4	20	0,39	0,76	0,51
P_obj1_15m	51	1	25	0,67	**0,98**	0,79
P_obj2_15m	2	1	19	0,10	0,67	0,17
P_obj2_30m	3	0	40	0,07	**1,00**	0,13

5d NN	instanzbasiert manuell, ohne IoU					
Testsequenz	**TP**	**FP**	**FN**	**Recall**	**Precision**	**F1**
O_obj1_30m	19	0	14	**0,58**	**1,00**	**0,73**
P_obj1_15m	57	1	18	0,76	**0,98**	**0,85**
P_obj2_15m	13	3	8	**0,62**	0,81	**0,70**
P_obj2_30m	12	0	31	**0,28**	**1,00**	**0,44**

5e	instanzbasiert manuell, ohne IoU					
Detektor	**TP**	**FP**	**FN**	**Recall**	**Precision**	**F1**
BG	73	17	99	0,42	0,81	0,56
PEAK	73	6	99	0,42	0,92	0,58
OGM	69	6	104	0,40	0,92	0,56
NN	101	4	71	**0,59**	**0,96**	**0,73**

Tabelle 5: Instanzbasierte Evaluation anhand der Testdaten ohne IoU-Threshold

In dieser Auswertung zeigen alle Detektoren hohe Precision-Werte, die zwischen 0,81 (BG) und 0,96 (NN) liegen. Dies verdeutlich, dass Fehlalarme nur in sehr geringem Umfang auftreten. Deutlich schwächer fallen hingegen die Recall-Werte aus, die weiterhin niedrig bis mittel ausfallen. Während der OGM- PEAK- und BG-Detektor im Gesamtergebnis bei 0,40 bzw. 0,42 liegen, erreicht der NN-Detektor mit 0,59 den höchsten Recall. Damit erkennt er als einziger mehr als die Hälfte der vorhandenen Hindernisse.

Die F1-Scores spiegeln diese Unterschiede wider: BG und OGM liegen mit 0,56 leicht unter dem PEAK-Detektor (0,58), während der NN-Detektor mit 0,73 den mit Abstand höchsten Wert erzielt. Diese Unterschiede sind auch über die verschiedenen Testsequenzen konstant.

Neben der Erkennungsleistung wurde auch die Rechenzeit pro Bild betrachtet. Die Laufzeitmessungen wurden in einem Docker-Container mit Ubuntu 20.04 LTS auf einem Hostsystem mit Intel Core Ultra 7 115H Prozessor durchgeführt. Für den NN-Detektor kam zusätzlich die integrierte NVIDIA RTX 500 Ada Generation GPU zum Einsatz. Erfasst wurde die reine Verarbeitungszeit in Millisekunden pro Frame, angegeben als Mittelwert über alle Frames der jeweiligen Testsequenz.

Testsequenz	BG	PEAK	OGM	NN
O_obj1_30m	27,25	27,52	**1,64**	30,28
P_obj1_15m	24,13	9,68	**1,03**	22,63
P_obj2_15m	24,64	8,98	**1,20**	23,30
P_obj2_30m	27,93	25,28	**1,85**	26,53
Average	25,99	17,87	**1,43**	25,69

Tabelle 6: Evaluation der Laufzeiten anahnd der Testdaten

Die BG- und NN-Detektoren benötigen im Mittel etwa 26ms pro Frame, wobei sich die Verarbeitungszeit je nach Reichweite des Sonars nur um ein paar Millisekunden unterscheidet. Deutlichere Unterschiede zeigen sich stattdessen im PEAK-Detektor, bei dem die Daten bei 15m Reichweite in 9ms verarbeitet werden können, während der Algorithmus bei einer Reichweite von 30m etwa 26ms benötigt. Die besten Ergebnisse zeigt der OGM-Detektor, bei dem die Verarbeitungszeit in allen Fällen unter 2ms liegt.

6.3 Praxistest

6.3.1 Durchführung

Neben der Auswertung anhand der Testsequenzen wurde ein Praxistest durchgeführt, um die Detektoren unter realen Einsatzbedingungen zu bewerten. Während Testdaten eine kontrollierte Vergleichbarkeit ermöglichen, bilden sie die Dynamik einer realen Mission nur eingeschränkt ab. Der Praxistest dient daher dazu, die Verfahren in einer realistischen Fahrsituation zu prüfen und ihre Eignung für die Kollisionsvermeidung unter Wasser zu bewerten.

Die Durchführung des Praxistests mit dem Sonobot erfolgte am 21.08.2025 im Werbellinsee. Im Gegensatz zu den zuvor genutzten Testaufnahmen wurde der Aufnahmebereich diesmal näher am Ufer gewählt. Aufgrund der geringeren Wassertiefe wurde die Reichweite des Sonars auf 12 m eingestellt. Als Hindernis diente eine Glaskugel (Abbildung 31), die bewusst gewählt wurde, da sie nicht im Trainingsdatensatz enthalten war und somit die Generalisierungsfähigkeit der Detektoren testen sollte. Die Mission umfasste ein zusammengesetztes Reihenmanöver um das Hindernis: zuerst

parallel zum Ufer und danach quer dazu mit Annäherung und Entfernung (siehe Abbildung 32). Die Mission wurde mit jedem der vier Detektoren einmal abgefahren und die Daten und Detektionsergebnisse in einem rosbag aufgezeichnet.

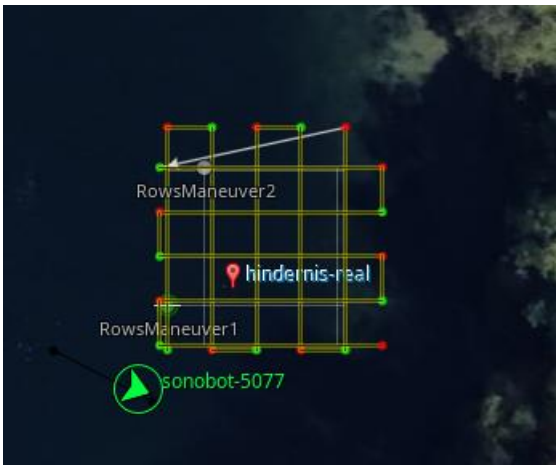


Abbildung 32: Missionsverlauf des Praxistests

Abbildung 31: Glaskugel als Hindernis im Praxist[illegible] Missionen jene Abschnitte herausgefiltert, in denen das Hindernis sichtbar war. Zwar kam es zwischen den Fahrten der verschiedenen Detektoren zu geringfügen Abweichungen durch Umweltbedingungen (z. B. Fische, Wind oder Wellen), dennoch konnten in allen Runs dieselben Abschnitte identifiziert werden. Die Sonardaten wichen dabei nur minimal voneinander ab.

Analog zu den Testsequenzen wurden auch im Praxistest Einzelbilder ausgewählt und manuell annotiert, um die Detektoren anhand der bekannten Metriken (IoU, Precision, Recall, F1) vergleichbar auszuwerten. Für jeden Detektor wurde eine separate Ground-Truth-Maske erstellt, basierend auf den Originaldaten der jeweiligen Mission. Dabei wurde darauf geachtet, dass die ausgewählten Bilder jeweils denselben Missionsabschnitt repräsentieren. Neben den Bildern, in denen das Hindernis enthalten ist, wurden zusätzlich 10 Bilder ohne Hindernisse ausgewählt.

Ergänzend zur Einzelbildanalyse erfolgte im Praxistest eine Bewertung auf Sequenzebene, um weitere für die Kollisionsvermeidung relevante Aspekte abzubilden. In der bildbasierten Auswertung wird ein Hindernis, das über mehrere Frames hinweg sichtbar ist, jeweils als neue Instanz betrachtet. Für die Navigation ist jedoch entscheiden, dass ein Hindernis rechtzeitig erkannt und die Erkennung anschließend über eine ausreichend lange Phase stabil aufrechterhalten wird.

Zur Bewertung werden daher ergänzende Metriken eingeführt:

- **Detektionslatenz:** Zeitabstand vom ersten Auftreten des Hindernisses im Sichtfeld bis zur ersten erfolgreichen Detektion; beschreibt die Vorwarnzeit für das Fahrzeug.

- **Erkennungsquote:** Anteil der Frames einer Hindernissequenz, in denen eine Detektion vorlag; zeigt die Stabilität der Erkennung über die Sequenz hinweg.

Auf Grundlage der Ergebnisse aus den Testsequenzen lassen sich folgende Erwartungen an den Praxistest formulieren:

Der **NN-Detektor** dürfte die größte Genauigkeit in Bezug auf IoU, Precision und Recall liefern. Ob die Erkennungen stabil sein werden, lässt sich schwer voraussagen, da die Merkmalsextraktion des NN nicht vorhersehbar ist.

Da sie in den Testdaten vor allem bei der Sonarreichweite von 15m gute Ergebnisse erreichten, sind für den **BG- und PEAK-Detektor** im Praxistest bei einer Reichweite von 12m erneut mittlere bis hohe Werte für IoU, Precision und Recall zu erwarten. Vermutlich sind die Detektionen jedoch insbesondere bei Kurven des Fahrzeugs oder strandnahen Aufnahmen instabil, da dort mehr Störeffekte und Rauschen vorhanden sind.

Der **OGM-Detektor** wird dagegen stabilere Ergebnisse liefern, da die Wahrscheinlichkeit für die Zellen nur leicht abgesenkt wird und einzelne Frames ohne Detektion weniger Auswirkungen haben. Dafür werden die Hindernisse vermutlich erst spät erkannt, da die Wahrscheinlichkeit der Zellen erst erhöht werden muss.

Mit dieser Versuchsplanung und den neuen Metriken wird im Folgenden die Praxistauglichkeit der Detektoren evaluiert.

6.3.2 Auswertung

Die pixelbasierte Auswertung (Tabelle 7) zeigt deutliche Unterschiede zwischen den Verfahren. Der PEAK- und der OGM-Detektor erreichen mit Recall-Werten von 0,74 bzw, 0,77 die höchste Abdeckung der Hindernisse, während der BG- und der NN-Detektor mit 0,39 bzw. 0,42 deutlich dahinter zurückliegen. Betrachtet man hingegen die Precision, kehrt sich das Bild um: Hier erzielen PEAK und OGM lediglich 0,48 bzw. 0,28, während BG und NN mit 0,77 bzw. 0,79 dominieren. Insgesamt erreicht der NN-Detektor sowohl den höchsten IoU-Wert (0,33) als auch den höchsten F1-Score (0,42).

Diese Ergebnisse verdeutlichen: Unter pixelbasierter Betrachtung liefern PEAK und OGM die größte Abdeckung, allerdings auf Kosten zahlreicher Fehlalarme. BG und NN erkennen Hindernisse präziser, dafür jedoch weniger vollständig.

Detektor	**pixelbasiert**						
	IoU	**TP**	**FP**	**FN**	**Recall**	**Precision**	**F1**
BG	0,31	3409	2041	12513	0,39	0,77	0,39
PEAK	0,31	8042	28525	3047	0,74	0,48	0,40
OGM	0,14	10158	144438	3863	**0,77**	0,28	0,19
NN	**0,33**	5671	3097	13862	0,42	**0,79**	**0,42**

Tabelle 7: Pixelbasierte Evaluation im Praxistest

In der instanzbasierten Analyse mit einem IoU-Schwellenwert von 0,5 (Tabelle 8) zeigen sich ähnliche Ergebnisse wie bei der Auswertung der Testdaten, allerdings weisen alle Detektoren etwas schwächere Ergebnisse auf. Der NN-Detektor erzielt mit einer durchschnittlichen IoU von 0,50 die besten Ergebnisse und erreicht mit Recall = 0,31, Precision = 0,42 und F1 = 0,36 die insgesamt ausgewogensten Werte. Der BG-Detektor folgt mit IoU 0 0,39 und einem F1-Score von 0,25, liegt damit aber spürbar darunter. Der PEAK-Detektor bleibt mit IoU 0 0,31 und F1 = 0,14 klar zurück. Der OGM-Detektor konnte beim gewählten Schwellwert keine Hindernisse erfassen (IoU = 0,07, F1 = 0).

Detektor		**instanzbasiert mit IoU > 0.5**					
	IoU	**TP**	**FP**	**FN**	**Recall**	**Precision**	**F1**
BG	0,39	21	50	78	0,21	0,30	0,25
PEAK	0,31	11	63	77	0,13	0,15	0,14
OGM	0,07	0	83	100	0	0	0
NN	**0,50**	38	53	83	**0,31**	**0,42**	**0,36**

Tabelle 8: Instanzbasierte Evaluation im Praxistest mit IoU Threshold 0.5

In der instanzbasierten Auswertung ohne IoU-Threshold (Tabelle 9) zeigen alle Detektoren hohe Precision-Werte zwischen 0,80 (NN) und 0,84 (PEAK), was auf eine geringe Anzahl von Fehldetektionen hinweist. Die Unterschiede liegen im Recall: Der OGM-Detektor erzielt mit 0,80 den höchsten Recall und damit auch den besten F1-Score von 0,80. Der PEAK-Detektor folgt mit Recall=0,73 und F1 = 0,78. Der NN- und BG-Detektor liegen dicht beieinander mit einem Recall von 0,63 bzw. 0,62, einer Precision von 0,80 bzw. 0,82 und einem F1-Score von 0,71 bzw. 0,70.

Detektor	**instanzbasiert manuell, ohne IoU**					
	TP	**FP**	**FN**	**Recall**	**Precision**	**F1**
BG	56	12	33	0,62	0,82	0,70
PEAK	61	11	22	0,73	**0,84**	0,78
OGM	73	15	18	**0,80**	0,82	**0,80**
NN	66	16	38	0,63	0,80	0,71

Tabelle 9: Instanzbasierte Evaluation im Praxistest ohne IoU Threshold

Die in Tabelle 10 dargestellten Ergebnisse zeigen die Laufzeitmessungen, die während des Praxistest bei der Ausführung der Detektoren auf dem Sonobot aufgezeichnet wurden. Leider war es nicht möglich für die Inferenzen des YOLO Netzwerks als Teil des NN-Detektors die GPU des Jetson Boards zu nutzen, da der Sonobot nur für einen kurzen Test zur Verfügung stand und die notwendigen Bibliotheken nicht kurzfristig konfiguriert werden konnten. Im Anschluss an den Test wurde deshalb ein Jetson Orin Board entsprechend eingerichtet und die Laufzeit des Ansatzes anhand des beim Praxistest aufgezeichneten rosbags getestet.

Mit durchschnittlichen Laufzeiten unter 10ms pro Frame arbeiten die Detektoren BG, PEAK und OGM effizient und echtzeitfähig. Der OGM-Detektor ist mit weniger als 1ms pro Frame besonders schnell. Der NN-Detektor erreicht mit GPU-Nutzung auf dem Jetson Orin eine Laufzeit von 20ms. Dies muss jedoch in der Praxis getestet werden, da die GPU auch von anderen Systemen des Sonobots verwendet wird. Die Bildrate der Detektoren folgt der Bildrate der FLS-Daten und beträgt daher bei allen Detektoren 15,6FPS.

Detektor	Laufzeit in ms	Bildrate in FPS
BG	7,93	15,6
PEAK	8,52	15,6
OGM	**0,92**	15,6
NN-cpu	2362,17	0,1
NN-gpu	20,45	15,6

Tabelle 10: Laufzeitmessung im Praxistest

In Tabelle 11 sind die sequenzbasierten Metriken dargestellt. Die Spalte „erste Detektion“ gibt die Verzögerung an, also nach wie vielen Frames ein Hindernis erstmals erkannt wurde. Dieser Wert wurde anhand der Bildrate und Laufzeit der Detektoren auch in Millisekunden umgerechnet. Ein niedriger Wert bedeutet eine frühere Vorwarnung. Die Spalte „letzte Detektion“ beschreibt den Zeitpunkt, an dem ein Hindernis zuletzt erkannt wurde, bevor die Erkennung endete, obwohl es noch im Sichtfeld war. Ein hoher Wert bedeutet daher, dass die Erkennung früher abbrach und die letzten Frames mit Hindernis nicht mehr erfasst wurden. Die Erkennungsquote gibt ergänzend den Anteil der Frames mit erfolgreicher Detektion über die gesamte Hindernis Passage an.

Für den NN-Detektor wurde das rosbag des Praxistest in der Docker-Umgebung auf dem Jetson Orin mit GPU erneut abgespielt, da aufgrund der langen Laufzeit während des Praxistest nicht für alle Frames Detektionen ermittelt werden konnten. Dies verändert jedoch weder die Detektionsqualität noch die Verzögerung in Frames, sondern nur die Angaben in Millisekunden.

	erste Detektion		letzte Detektion		Erkennungs-
Detektor	**in Frames**	**in ms**	**in Frames**	**in ms**	**quote**
BG	22,89	1.449,93	4,33	280,93	64,89%
PEAK	**5,40**	**348,72**	4,90	317,22	80,70%
OGM	8,78	553,92	4,44	**280,92**	**81,00%**
NN-cpu	8,40	86.362,17	**3,70**	39.362,17	53,10%
NN-gpu	8,40	549,65	**3,70**	**253,55**	53,10%

Tabelle 11: Sequenzbasierte Evaluation im Praxistest

Der PEAK-Detektor erkennt Hindernisse mit durchschnittlich 5,4 Frames (348ms) Verzögerung am frühesten. OGM und NN folgen mit 8-9 Frames Verzögerung (ca. 550ms), während der BG-Detektor mit 22,8 Frames (1,44s) erheblich später reagiert. Beim Zeitpunkt der letzten Detektion liegt der Bereich für BG, PEAK und OGM bei etwa 270-310ms vor Ende der Passage, während der NN-Detektor die Hindernisse noch länger erkennt. Bei der Erkennungsquote liefert OGM mit 81% die stabilste Abdeckung der

Hindernisse, dicht gefolgt vom PEAK-Detektor mit 80,70%. Dagegen zeigt der NN-Detektor mit 53% die geringste Abdeckung.

6.4 Diskussion

Die pixelbasierte Evaluation hat sich als wenig aussagekräftig für die Beurteilung der Praxistauglichkeit erwiesen. Erst die instanz- und sequenzbasierte Analyse bildet die Anforderungen an eine Hinderniserkennung für die Kollisionsvermeidung realistisch ab. Diese zusätzliche Betrachtung relativierte die eher mittelmäßigen Werte der bildbasierten Auswertung und zeigt, dass eine Bewertung auf Sequenzebene notwendig ist, um Stabilität und Verlässlichkeit der Verfahren einschätzen zu können. Grundsätzlich haben alle Detektoren das ausgelegte Hindernis in allen Passagen erkannt, in denen es sichtbar war. Allerdings zeigen sich große Unterschiede in der frühzeitigen und stabilen Erkennung.

Für die Kollisionsvermeidung steht die sichere Navigation im Vordergrund. Oberste Priorität hat daher, Nicht-Erkennungen oder eine zu späte Reaktion zu vermeiden. Lagegenauigkeit und eine geringe Anzahl an Falschmeldungen sind zwar nicht sicherheitskritisch, tragen aber wesentlich zur Verlässlichkeit und Effizienz der Navigation bei. Entsprechend ist die Erkennungsquote (Recall) die zentrale Metrik, da verpasste Hindernisse (False Negatives) das größte Sicherheitsrisiko darstellen. Ebenso wichtig sind die Detektionslatenz, also die Zeitspanne zwischen Sichtbarkeit und erster Erkennung, sowie die Laufzeit des Detektionsverfahrens, da nur eine rechtzeitige Ausgabe die Einleitung von Brems- oder Ausweichmanövern ermöglicht. Die Precision ist im Vergleich zum Recall weniger sicherheitsrelevant, trägt jedoch entscheiden zur Aufrechterhaltung der Missionsfähigkeit bei, da Fehlalarme vermieden und damit unnötige Kursänderungen oder Stopps reduziert werden. Die räumliche Genauigkeit, messbar über die IoU, ist hingegen stark vom eingesetzten Fahrzeug abhängig und für sicherheitskritische Aspekte weniger bedeutend als die zuvor genannten Metriken.

Der **BG-Detektor** erzielte in nahezu allen Metriken nur mittlere Ergebnisse. Ein wesentlicher Nachteil ist die späte Erkennung von Hindernissen, die teilweise auf die gewählte Region-of-Interest-Zuschnittsmethode zurückzuführen ist, welche Hindernisbereiche an den Bildrändern abschneidet. In Kombination mit den eher durchschnittlichen Werten für IoU, Recall und Precision zeigt sich, dass dieser Ansatz für eine robuste Hinderniserkennung nur eingeschränkt geeignet ist.

Der **PEAK-Detektor** erzielte hohe Werte bei IoU, Recall und Precision und detektiert Hindernisse vergleichsweise früh, was für die Praxis ein klarer Vorteil ist. Gleichzeitig hängt seine Leistung stark von der eingestellten Sonarreichweite und den gewählten Parametern ab, was ihn empfindlich gegenüber Umgebungsbedingungen macht. Dennoch bietet PEAK ein hohes Potenzial für ein praxisnahe Implementierung, sofern er dynamisch an die Reichweite angepasst wird.

Der **OGM-Detektor** profitierte von seiner probalistischen Belief-Map, durch die einzelne Frames ohne Detektion weniger ins Gewicht fielen. Dadurch zeigte er eine hohe Konsistenz und erzielte über die Sequenzen hinweg gute Werte bei Recall und Precision. Nachteilig ist jedoch die geringe IoU, die aus der Rasterisierung resultiert. Hindernisse werden zwar zuverlässig, aber nur grob lokalisiert. Dieses Defizit könnte durch höhere Rasterauflösung oder die zusätzliche Segmentierung detektierter Zellen verbessert werden. Da die Laufzeit selbst bei erhöhter Auflösung unkritisch bleibt, ist dieser Ansatz insbesondere für stabile Echtzeitdetektion interessant.

Der **NN-Detektor** erreichte zwar auf den Testsequenzen die höchsten IoU-. Precision- und Recall-Werte, zeigte jedoch im Praxistest deutliche Schwächen. Das mag damit zusammenhängen, dass das NN auf Sonardaten mit einer Range von 15m bzw. 30m trainiert wurde und sich nur schlecht auf die Reichweite von 12m und veränderte Umweltbedingungen anpassen konnte. Es traten viele Falschpositive auf, die sich jedoch potenziell über Größenfilterung reduzieren lassen können (da dies keine komplett falschen Detektionen sind, sondern einfach kleinere Objekte, die nicht zwingend ein Hindernis darstellen). Schwerer wiegt die beobachtete Inkonsistenz: Hindernisse wurden nicht stabil über die gesamte Sequenz erkannt, sondern zeigten ein starkes „Flackern“. Für die Anwendung in einer Kollisionsvermeidung stellt dies ein erhebliches Problem dar, da eine verlässliche Hindernisverfolgung nicht gewährleistet ist.

7 Fazit

Im Rahmen dieser Arbeit konnten vier ausgewählte Detektionsverfahren für Vorwärtsblicksonardaten erfolgreich implementiert und umfassend evaluiert werden. Die Ergebnisse zeigen, dass sich die Verfahren hinsichtlich Genauigkeit, Stabilität und Laufzeit deutlich unterscheiden und damit unterschiedliche Stärken für den praktischen Einsatz aufweisen.

Das Deep Learning basierte Verfahren (NN-Detektor) erzielte zwar die höchste Genauigkeit bei der Lagebestimmung, offenbarte im Praxistest jedoch Schwächen in der Stabilität der Erkennungen sowie einen hohen Laufzeitbedarf. Für Aufgaben der Klassifikation und Objektdifferenzierung erscheinen neuronale Netze daher besonders geeignet, während für die reine Hinderniserkennung eher bildbasierte oder probalistische Verfahren Vorteile bieten. Damit liefert die Arbeit nicht nur einen methodischen Vergleich, sondern auch eine Entscheidungsgrundlage für die Auswahl geeigneter Ansätze in konkreten Anwendungen: Welche Eigenschaften – Präzision, Robustheit oder Geschwindigkeit – im Vordergrund stehen sollten, hängt letztlich von den jeweiligen Einsatzbedingungen und Prioritäten ab.

Auf Grundlage der Ergebnisse des Praxistests bieten sich vor allem der PEAK- und der OGM-Detektor für den Einsatz in der Echtzeit-Hinderniserkennung an. PEAK überzeugt

durch frühe Erkennung, während OGM eine besonders konsistente und stabile Detektion ermöglicht.

Daran anknüpfend ergeben sich mehrere Ansatzpunkte für weiterführende Arbeiten:

- **Untersuchung unterschiedlicher Sonarreichweiten**, um die Robustheit der Verfahren gegenüber Reichweitenänderungen systematisch zu bewerten.
- **Integrationstests mit einem Kollisionsvermeidungsmodul**, um die tatsächliche Wirksamkeit der Detektoren in der autonomen Navigation zu überprüfen.
- **Analyse strandnaher Bereiche**, da diese in den Tests wiederholt zu Fehldetektionen führten. Eine offene Frage ist dabei, ob solche Strukturen grundsätzlich als Hindernisse behandelt werden sollten.
- **Tests mit dynamischen Hindernissen**, da in dieser Arbeit ausschließlich ortsfeste Objekte betrachtet wurden. Dies ist entscheiden, um die Leistungsfähigkeit der Detektoren in realistischen Szenarien mit bewegten Zielen wie großen Fischen oder anderen Fahrzeugen zu bewerten.

8 Quellenverzeichnisse

8.1 Literaturverzeichnis

Amit, Y. & Felzenszwalb, P. F. (2014): Object Detection. In K. Ikeuchi (Ed.), *Computer Vision: A Reference Guide* (S. 537–542). Springer, Boston, MA. https://doi.org/10.1007/978-0-387-31439-6_660

Burgmer, C. (2005): Schema eines künstlichen Neurons. Verfügbar unter: https://commons.wikimedia.org/wiki/File:ArtificialNeuronModel_deutsch.png (Letzter Zugriff 12.09.2025)

BlueLink, LLC. (o. J.). *Sonoptix ECHO – Forward Looking Sonar Datasheet*. San Diego, CA: BlueLink. Verfügbar unter: https://sonoptix.com/specifications (Letzter Zugriff 16.06.2025)

Cao, X., Ren, L. & Sun, C. (2023): Research on Obstacle Detection and Avoidance of Autonomous Underwater Vehicle Based on Forward-Looking Sonar. *IEEE Transactions on Neural Networks and Learning Systems,* vol. 34, no. 11. https://doi.org/10.1109/TNNLS.2022.3156907

Chen, L., Huang, Y., Dong, J., Xu, Q., Kwong, S., Lu, H., Lu, H., & Li, C. (2024): *Underwater Object Detection in the Era of Artificial Intelligence: Current, Challenge, and Future*. ArXiv, https://doi.org/10.48550/arXiv.2410.05577

Dos Santos, M., Ribeiro, P. O., Núñez, P., Drews-Jr, P., & Botelho, S. (2017): Object Classification in Semi Structured Environment Using Forward-Looking Sonar. In *Sensors*, *17* (10), 2235. https://doi.org/10.3390/s17102235

EvoLogics Quadroin: https://www.evologics.com/web/content/259390?unique=67a041daad4b7dd54c28df1e30ab75bc2fd92324&download=true (Letzter Zugriff 12.09.2025)

EvoLogics Sonobot: https://www.evologics.com/web/content/217103?unique=9fe7777f4b82882c09dbc41e70241835ad78154b (Letzter Zugriff 12.09.2025)

Fuchs, L. R., Gällström, A. & Folkesson, J. (2018): Object Recognition in Forward Looking Sonar Images using Transfer Learning. *2018 IEEE/OES Autonomous Underwater Vehicle Workshop (AUV)*, Porto, Portugal, 2018, pp. 1-6, https://doi.org/10.1109/AUV.2018.8729686

Galceran, E., Djapic, V., Carreras, M. & Williams, D. P. (2012). A real-time underwater object detection algorithm for multi-beam forward looking sonar. *IFAC Proceedings Volumes, 45*(5). https://doi.org/10.3182/20120410-3-PT-4028.00051

Gaspar, A. R., & Matos, A. (2023): Feature-Based Place Recognition Using Forward-Looking Sonar. *Journal of Marine Science and Engineering*, *11*(11), 2198. https://doi.org/10.3390/jmse11112198

Goodfellow, I., Bengio, Y. & Courville, A. (2016): *Deep Learning*. MIT Press. https://www.deeplearningbook.org/

Hurtós Vilarnau, N. (2014): *Forward-Looking Sonar Mosaicing for Underwater Environments*. (Doktorarbeit). Universitat de Girona. http://hdl.handle.net/10803/285086

Jin, H.-S., Kang, H., Kim, M.-G., Lee, M.-J. & Li, J.-H. (2024): OGM-Based Real-Time Obstacle Detection and Avoidance Using a Multi-Beam Forward Looking Sonar. *Journal of Ocean Engineering and Technology, 38*(4). https://doi.org/10.26748/KSOE.2024.057

Kot, R. (2022): Review of Obstacle Detection Systems for Collision Avoidance of Autonomous Underwater Vehicles Tested in a Real Environment. *Electronics, 11*(21), 3615. https://doi.org/10.3390/electronics11213615

Matthies, L. (2014): Obstacle Detection. In K. Ikeuchi (Ed.), *Computer Vision: A Reference Guide* (S. 543–548). Springer, Boston, MA. https://doi.org/10.1007/978-0-387-31439-6_52

Na, W., Li, H., Wang, J., Wen, J., Xing, T., & Hou, Y. (2025): *A Constant False Alarm Rate Detection Method for Sonar Imagery Targets Based on Segmented Ordered Weighting*. Journal of Marine Science and Engineering, 13(4), 819. https://doi.org/10.3390/jmse13040819

Renard, E. (2024): *ROS 2 from Scratch: Get Started with ROS 2 and Create Robotics Applications with Python and C++*. Birmingham: Packt Publishing.

Russell, S. & Norvig, P. (2021): *Artificial Intelligence: A Modern Approach*. 4th ed. Pearson Education. ISBN 978-0-13-461099-3

Wang, J. & Herath, D. (2022): How to Move? Control, Navigation and Path Planning for Mobile Robots. In Herath, D. & St-Onge, D. (Ed.), *Foundations of Robotics. A Multidisciplinary Approach with Python and ROS*. Springer. https://doi.org/10.1007/978-981-19-1983-1

Wilts, T., Boenecke, L. & Badri-Hoeher, S. (2022): Overview and Practical Evaluation of Sonar-Based Collision Avoidance Approaches for AUVs. *OCEANS 2022*, Hampton Roads, VA, USA. https://doi.org/10.1109/OCEANS47191.2022.9977167

Xie, K., Yang, J., & Qiu, K. (2022) : *A Dataset with Multibeam Forward-Looking Sonar for Underwater Object Detection*. ArXiv, https://doi.org/10.48550/arXiv.2212.00352

Zhang H., Tian M., Shao G., Cheng J. & Liu J. (2022): Target Detection of Forward-Looking Sonar Image Based on Improved YOLOv5. *IEEE Access*, vol. 10. https://doi.org/10.1109/ACCESS.2022.3150339

Zhao, Z.-Q., Zheng, P., Xu, S. -T. & Wu, X. (2019): Object Detection With Deep Learning: A Review. *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 11, S. 3212-3232, https://doi.org/10.1109/TNNLS.2018.2876865

Zou, Z., Chen, K., Shi, Z., Guo, Y. & Ye, J. (2023): *Object Detection in 20 Years: A Survey.* ArXiv, https://doi.org/10.48550/arXiv.1905.05055

KI-Hilfsmittel:

ChatGPT-5, OpenAI: openai.com/chat

- Vorschläge für Formulierungen einzelner Textabschnitte sowie sprachliche Überarbeitung und Kürzung eigener Texte
- Zusammenfassung und Übersetzung von Literatur
- Generierung von Pythonskripten für einfache Aufgaben wie Datenmanagement, Darstellung der Segmentierungsergebnisse zum Vergleich der Detektoren, etc.
- Formatierung des Literaturverzeichnisses gemäß gewähltem Zitierstil

8.2 Abbildungsverzeichnis

8.3 Tabellenverzeichnis

9 Ehrenwörtliche Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit in allen Teilen selbstständig angefertigt und keine anderen als die in der Arbeit angegebenen Quellen und zur Angabe verpflichteten Hilfsmittel benutzt habe, und dass die Arbeit in gleicher oder ähnlicher Form in noch keiner anderen Prüfung vorgelegen hat. Sämtliche wörtlichen oder sinngemäßen Übernahmen und Zitate sind kenntlich gemacht und nachgewiesen.

Angaben zur Verwendung KI-basierter Hilfsmittel

Ich versichere, dass ich keine KI-basierten Tools verwendet habe, deren Nutzung der Prüfer / die Prüferin explizit schriftlich ausgeschlossen hat. Ich bin mir bewusst, dass die Verwendung von Texten oder anderen Inhalten und Produkten, die durch KI-basierte Tools generiert wurden, keine Garantie für deren Qualität darstellt. Ich verantworte die Übernahme jeglicher von mir verwendeter maschinell generierter Passagen vollumfänglich selbst und trage die Verantwortung für eventuell durch die KI generierte fehlerhafte oder verzerrte Inhalte, fehlerhafte Referenzen, Verstöße gegen das Datenschutz- und Urheberrecht oder Plagiate. Ich versichere zudem, dass in der vorliegenden Arbeit mein gestalterischer Einfluss überwiegt.

Groß-Rietz, 12.09.2025

Ort, Datum

Marie-Luise

Unterschrift