DLR Projektträger

Projektträger für das
Bundesministerium
für Wirtschaft
und Klimaschutz

INDUSTRIELLE GEMEINSCHAFTSFORSCHUNG

Schlussbericht vom 13.05.2025

zum IGF-Vorhaben 01IF22665N

Thema

Kollektives Umgebungsinformationssystem für mobile Roboter

Berichtszeitraum

01.11.2022 bis 28.02.2025

Forschungsvereinigung

Bundesvereinigung Logistik (BVL) e.V.
Schlachte 31
28195 Bremen

Forschungseinrichtung(en)

TUM Lehrstuhl für Fördertechnik Materialfluss Logistik (fml)
Boltzmannstraße 15
85748 Garching

INDUSTRIELLE GEMEINSCHAFTSFORSCHUNG

Inhaltsverzeichnis

Abkürzungsverzeichnis *4*

1 Einleitung *5*

1.1 Wissenschaftliche, technische und wirtschaftliche Problemstellung **5**

1.2 Zielsetzung des Projekts **7**

1.3 Methodik und Lösungsweg **9**

2 Technische Grundlagen *11*

2.1 Simulation von Umgebungssensordaten **11**

2.1.1 Möglichkeiten und Herausforderungen der Simulation von Sensordaten 11

2.1.2 Entwicklungsumgebungen für Sensordatensimulation 12

2.1.3 Beschreibung von Industrieumgebungen 13

2.1.4 Beschreibung Mobiler Roboter 14

2.1.5 Sensorik für Mobile Roboter 15

2.1.6 Datenformat zum Speichern von Datensets 16

2.2 Kartierung und Lokalisierung **17**

2.2.1 Metrische Karten 17

2.2.2 Topologische Karten 18

2.2.3 Hybride Karten 19

2.2.4 Semantische Karten 19

2.3 Semantik **19**

2.3.1 Grundlagen Maschinelles Lernen 20

2.3.2 Objektdetektion mittels Deep-Learning-Algorithmen 20

2.3.3 Semantische Segmentierung mittels Deep-Learning-Algorithmen 22

2.3.4 Metriken zur Bewertung von Deep-Learning-Algorithmen 22

2.3.5 Datensets und Ground-Truth 23

2.4 Datentransfer zwischen mobilen Robotern und zentralem Server **24**

2.4.1 Übertragungstechnologie 24

2.4.2 Sensordatensynchronisierung 26

3 Simulation von Sensordaten *28*

3.1.1 Anforderungen and die Simulation 28

3.1.2 Auswahl der Simulationssoftware 32

3.1.3 Entwicklung der Module der Simulation 34

3.1.4 Datengenerierung 39

4 Kollektives Umgebungsinformationssystem *49*

4.1 Gesamtarchitektur **49**

4.2 Datentransfer **50**

4.2.1 Übertragungstechnologie 50

4.2.2 Sensordatensynchronisierung 51

4.2.3 Queuing 51

4.3 Statischer und dynamischer Layer 52
4.3.1 Grid-basierter Ansatz 52
4.3.2 Kartierung mittels geometrischer Primitive 56

4.4 Semantischer Layer 60
4.4.1 Abbildung von Objekten im semantischen Layer 61
4.4.2 Erkennung des befahrbaren Bereichs 63

4.5 Fusion und Visualisierung 64

5 Fazit und Ausblick 65

5.1 Abgleich von Zielen / Anforderungen mit den Ergebnissen 66

5.2 Ausblick 67

6 Ressourcenverwendung und Ergebnistransfer 68

6.1 Verwendung der Zuwendung 68
6.1.1 Wissenschaftlich-technisches Personal und studentische Hilfskräfte 68
6.1.2 Geräte (Einzelansatz B des Finanzierungsplans) 68
6.1.3 Leistungen Dritter (Einzelansatz C des Finanzierungsplans) 68

6.2 Notwendigkeit und Angemessenheit der geleisteten Arbeit 68

6.3 Darstellung des wissenschaftlich-technischen und wirtschaftlichen Nutzens der erzielten Ergebnisse insbesondere für KMU sowie ihres innovativen Beitrags und ihrer industriellen Anwendungsmöglichkeiten 69

6.4 Wissenstransfer in die Wirtschaft 69

6.5 Geplante spezifische Transfermaßnahmen nach der Projektlaufzeit 71

6.6 Veröffentlichungen 72

6.7 Studienarbeiten 72

7 Literaturverzeichnis 73

A Anhang I

A.1 Fragebogen zur Anforderungsermittlung für Simulationsdaten I

A.2 Bewertungsmatrix Simulationssoftware IV

A.3 Umgebungsobjekte in der Simulationsszenerie VI

A.4 Maße des simulierten Roboters VIII

A.5 Positionierung der Sensoren auf dem Roboter IX

Abkürzungsverzeichnis

AMR ..... *Autonomous Mobile Robot*
FOV ..... *Field of View*
FTF ..... *Fahrerloses Transportfahrzeug*
FTS ..... *Fahrerloses Transportsystem*
GLT ..... *Großladungsträger*
IoU ..... *Intersection over Union*
KLT ..... *Kleinladungsträger*
LiDAR ..... *Light Detection And Ranging*
MAC ..... Medium Access Control
mAP ..... *mean Average Precision*
ROS ..... *Robot Operating System*

1 Einleitung

1.1 Wissenschaftliche, technische und wirtschaftliche Problemstellung

In vielen Ländern der Welt wird der Umbau hin zu einer automatisierten Produktion und Logistik vorangetrieben. Gründe dafür sind unter anderem der nach wie vor anhaltende Fachkräfte-Mangel und die Senkung von Produktionskosten durch Einsparung manueller Arbeit. Um dieses Ziel zu erreichen, sind hohe Investitionen auch im Bereich der Robotik notwendig. Bis 2029 soll daher der Umsatz von Service-Robotern im Logistik-Bereich weltweit auf 3,3 Mrd. US$ anwachsen (s. Abbildung 1). Service-Roboter sind hierbei Roboter, die entworfen wurden, um Menschen bei der Ausführung bestimmter Tätigkeiten zu unterstützen [1].

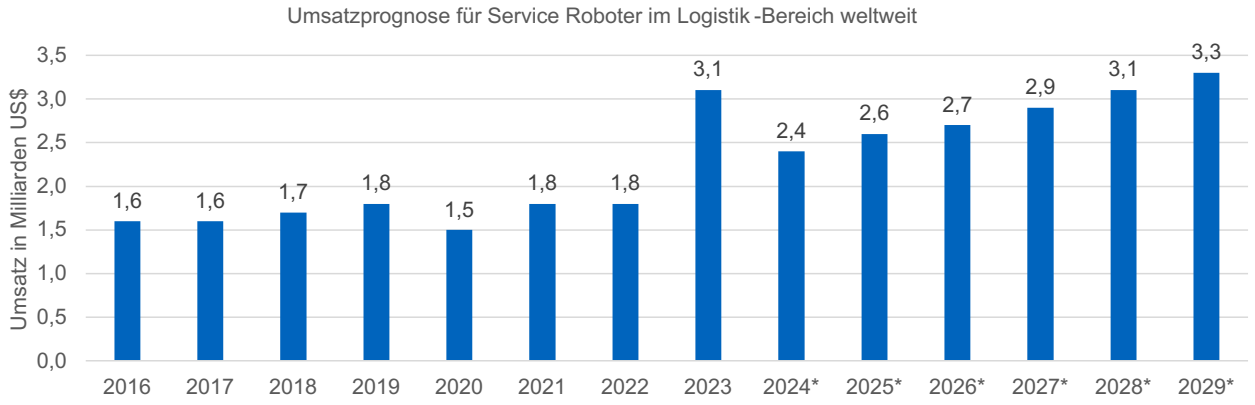


Abbildung 1: Umsatzprognose für Service Roboter im Logistik-Bereich weltweit ab 2024 basierend auf den Werten von 2026 bis 2023 [1]

Neben stationären Robotern, die vor allem in der Fertigung zum Einsatz kommen, sind für die Umsetzung dieses Automatisierungstrends insbesondere Autonome Mobile Roboter (AMR) wichtig. Allein in Europa soll der Markt für AMR bis 2028 auf 3,752 Mrd. US$ anwachsen (s. Abbildung 2).

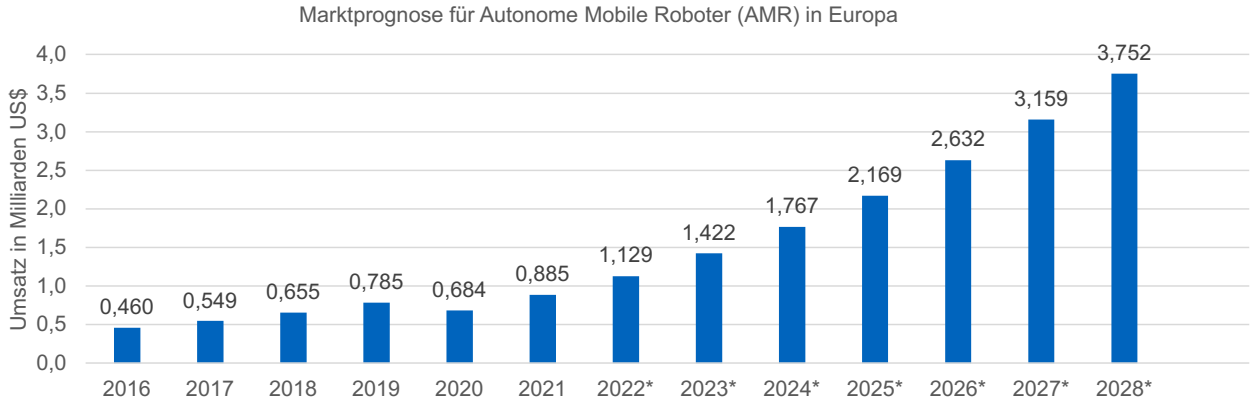


Abbildung 2: Marktprognose für Autonome Mobile Roboter (AMR) in Europa bis 2028 basierend auf den Werten von 2016 bis 2021 [2]

Das unterstreicht deutlich, dass AMR auch zukünftig in der Automatisierungsstrategie vieler Unternehmen eine tragende Rolle darstellen werden.

Für den Betrieb von AMR ist neben den Robotern selbst häufig weitergehende Infrastruktur notwendig. In der Richtlinie VDI 4451-7:2005 [3] sind beispielhafte Konfigurationen für Fahrerlose Transportsysteme (FTS) beschrieben, die mit Fahrerlosen Transportfahrzeugen (FTF) betrieben werden. FTF bieten im Vergleich zu AMR aufgrund geringerer Navigationsfähigkeiten weniger flexible Pfadplanungsmöglichkeiten [4]. Da für AMR die Konfiguration jedoch häufig ähnlich ist, wird diese Richtlinie hier zur Veranschaulichung des Gesamtsystems herangezogen (s. Abbildung 3).

Danach könnte ein Transport von Waren beispielsweise wie folgt aussehen: Der Transportauftrag wird durch einen übergeordneten Host-Rechner (üblicherweise durch ein Enterprise-Resource-System) erstellt. Dieser Transportauftrag wird anschließend an ein Leitsystem geschickt, das entscheidet, welcher der Roboter der Flotte den Auftrag ausführen soll (z. B. wegen Leerlaufs oder örtlicher Nähe). Über eine Funkverbindung wird dann der Auftrag mit Start- und Zielkoordinaten oder auch Pfad-Splines (s. VDA 5050 [5]) an den betreffenden Roboter verschickt, der diesen nun ausführt. Hat der Roboter den Transport abgeschlossen, meldet er die erfolgreiche Durchführung an das Leitsystem zurück, das die Information an den Host-Rechner weitergibt. Zur Überwachung und Wartung des Gesamtsystems ist in der Regel ebenfalls eine Visualisierung integriert.

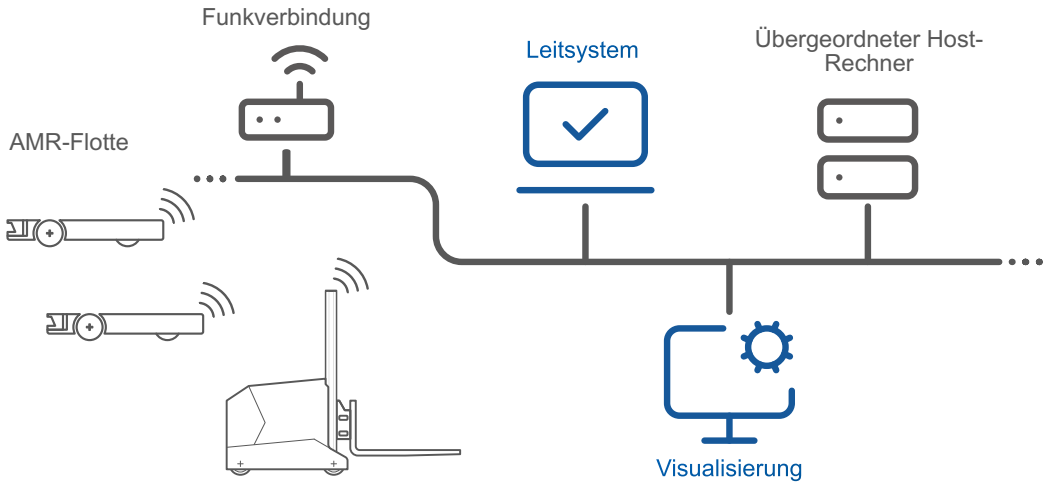


Abbildung 3: Bestandteile eines Fahrerlosen Transportsystems nach VDI 4451-7 [3]

Damit AMR die geforderten Wege korrekt abfahren kann, müssen sie sich in ihrer Umgebung zurechtfinden können. Nur die Start- und Ziel-Koordinaten allein ermöglichen keine vollständige Pfadplanung und auch keine lokale Neuplanung des Pfades bspw. aufgrund von Hindernissen. Für die autonome Fahrfunktion verlassen sich AMR daher, genau wie autonome Kraftfahrzeuge (vgl. [6]), auf fünf wesentliche Bausteine:

Abbildung 4: Fünf Bestandteile für autonome Fahrfunktionen nach [6]; Planung und Steuerung werden nicht im Projekt betrachtet

Durch verschiedene Sensoren wird die Umgebung wahrgenommen. Diese Rohdaten werden anschließend durch eine Lokalisierung in räumlichen Zusammenhang gebracht und optional werden einzelne Merkmale, wie z. B. detektierte Objekte, extrahiert. Die lokalisierten Daten werden dann in Kombination mit etwaigen Merkmalen im Schritt der Abstraktion zu einem Umgebungsmodell fusioniert, das beispielsweise eine zweidimensionale Karte der Umgebung sein kann. Darauf basierend können nun Pfade geplant und wiederum die Aktoren angesteuert werden.

Auch wenn jeder AMR im Regelfall für die Auftragsdistribution mit einem zentralen Server kommuniziert, werden Expertengesprächen zufolge die oben beschriebenen Schritte heutzutage üblicherweise lokal auf jedem Roboter einzeln durchgeführt.

Das führt insbesondere für die ersten drei Schritte zu mehreren Nachteilen im Gesamtsystem:

- Zunächst wird die Erkennungsleistung eines Roboters immer durch die auf diesem verbaute Sensorik limitiert. Nur, was mit den verbauten Sensoren erfasst werden kann, steht auch für die nachfolgenden Schritte zur Verfügung.
- Darüber hinaus ist die Verfolgung von Umgebungsobjekten über größere Strecken allein mit mobil verbauter Sensorik nicht möglich. Durch die lokale Berechnung verliert ein Roboter jegliche lokalisierungsrelevante Informationen über Umgebungsobjekte, sobald diese seinen Sensorbereich verlassen.
- Häufig ist der Zustand anderer Roboter der Flotte nur dem Leitsystem bekannt und wird auch nur eingeschränkt erfasst. Beispielsweise fehlt in einigen Fällen eine genaue Pose des Roboters. Dadurch können Roboter in einer Begegnungssituation unter Umständen nicht optimal ausweichen.

Unter anderem aus diesen Gründen müssen mobile Roboter in vielen Situationen langsamer als möglich fahren. Gerade in kritischen Bereichen mit eingeschränkter Übersicht ist es notwendig regelmäßig die Geschwindigkeit zu reduzieren, um Kollisionen vorzubeugen, auch wenn eine Gefährdung durch einen globalen Blick auf die Szene ausgeschlossen wäre. Insgesamt schmälert das die Produktivität das Gesamtsystems.

1.2 Zielsetzung des Projekts

Diese Limitierung der Produktivität resultiert primär aus der Tatsache, dass kein ausreichender Informationsaustausch über Umgebungsinformationen zwischen den mobilen Robotern (im Folgenden verwendet für FTF und AMR, sofern auf diesen Umgebungssensorik verbaut ist) einer Flotte stattfindet. Im Rahmen des Projekts „Kollektives Umgebungsinformationssystem für mobile Roboter“ (KolUmBot) wurde daher ein System entwickelt werden, das diesen Austausch ermöglicht. Dieses Umgebungsinformationssystem soll sich an zentraler Stelle im Gesamtsystem befinden und Sensordaten der gesamten Flotte mobiler Roboter sammeln und zu einem gemeinsamen Umgebungsmodell fusionieren. Aus diesem Modell sollen für einzelne Roboter relevante Informationen abgeleitet und an diese spezifisch zurückgespielt werden können. Ebenso sollen aus dem Umgebungsmodell relevante Informationen für ein Leitsystem abgeleitet und Daten für Visualisierungs- oder Wartungszwecke bereitgestellt werden können (vgl. Abbildung 3). Denkbar wäre unter anderem eine gezielte Optimierung des gesamten Verkehrsflusses. Außerdem könnten auch zusätzliche Informationen wie Bodenbeschaffenheit oder Verbindungsqualität erfasst werden, um damit die Infrastruktur zu verbessern. Die Umsetzung der Nutzung der Modellinformationen (Planung und Steuerung, vgl. Abbildung 4) liegt jedoch außerhalb des Betrachtungshorizonts dieses Projekts.

Basierend auf diesen Randbedingungen lassen sich die folgenden Teilziele ableiten:

- *Definition Bestandteile*
  Die Bestandteile des Umgebungsinformationssystems müssen klar definiert werden.
- *Karte und Semantik*
  Das Umgebungsmodell soll neben einer Umgebungskarte auch semantische Informationen beinhalten, um darauf aufbauende Funktionen, wie die Pfadplanung, zu optimieren und zusätzliche Funktionen zu ermöglichen.
- *Datentransfer*
  Es soll eine Strategie zur Übertragung der Sensordaten auf den zentralen Server erarbeitet werden (bspw. Übertragungstechnik, …).
- *Zugriffskonzept*
  Es soll ein Konzept für den Schreib- und Lesezugriff auf das Modell entwickelt werden. Hier könnte es sonst aufgrund der Tatsache, dass mehrere Roboter auf ein zentrales Modell zugreifen, zu Kollisionen in der Datenübertragung kommen.

Außerdem wurden im Rahmen der Sitzungen des Projektbegleitenden Ausschusses und durch unstrukturierte Experteninterviews weitere Anforderungen an das Gesamtsystem ermittelt:

- *Größe des Umgebungsmodells*
  Das Modell muss mindestens eine so große Fläche abbilden, dass es den Betriebsbereich der Roboter abdecken kann. Das kann mehrere Hallen oder nur Teile von Hallen beinhalten.
- *Umgang mit verschiedenen Sensormodalitäten*
  Die bei mobilen Robotern verwendeten Sensoren variieren stark zwischen den Modellen verschiedener Hersteller, aber auch innerhalb des Produktportfolios eines Herstellers. Bestrebungen zur Kommunikation zwischen Robotern wie die VDA 5050 verstärken diesen Trend zu Flotten mit inhomogener Sensorauswahl. Daher muss das Umgebungsinformationssystem mit verschiedenen Sensormodalitäten umgehen können.
- *Nachhaltige Betrachtung von Sensoren*
  Aktuell eingesetzte Sensoren zu betrachten, ist nicht ausreichend. Diese haben verglichen mit neueren Entwicklungen oft eine geringere Informationsdichte. Außerdem ist zu erwarten, dass mit Trendthemen wie künstlicher Intelligenz zukünftig der Einsatz anderer oder neuer Sensoren notwendig werden wird. Für ein nachhaltiges Modell müssen daher auch mittel- bis langfristig eingesetzte Sensoren mit dem Umgebungsinformationssystem eingesetzt werden können.
- *Betrachtung relevanter Situationen*
  Das System kann seine Vorteile besonders in bestimmten Situationen ausspielen. Das können beispielsweise Situationen mit toten Winkeln oder verteilten dynamischen Objekten sein. Diese Situationen sollten bei der Entwicklung besonders mitbetrachtet werden.

1.3 Methodik und Lösungsweg

Grundlegend besteht das Umgebungsinformationssystem aus drei Bestandteilen: Zuerst erfasst die Flotte von mobilen Robotern mit ihrer Sensorik die Umgebung. Diese Umgebungsdaten werden dann über eine Schnittstelle an einen zentralen Server übertragen, auf dem schlussendlich das Umgebungsmodell gespeichert ist und fortlaufend aktualisiert wird. Im Gegenzug erhalten die mobilen Roboter Informationen aus dem Umgebungsmodell. In Abbildung 5 sind die einzelnen Bestandteile dargestellt.

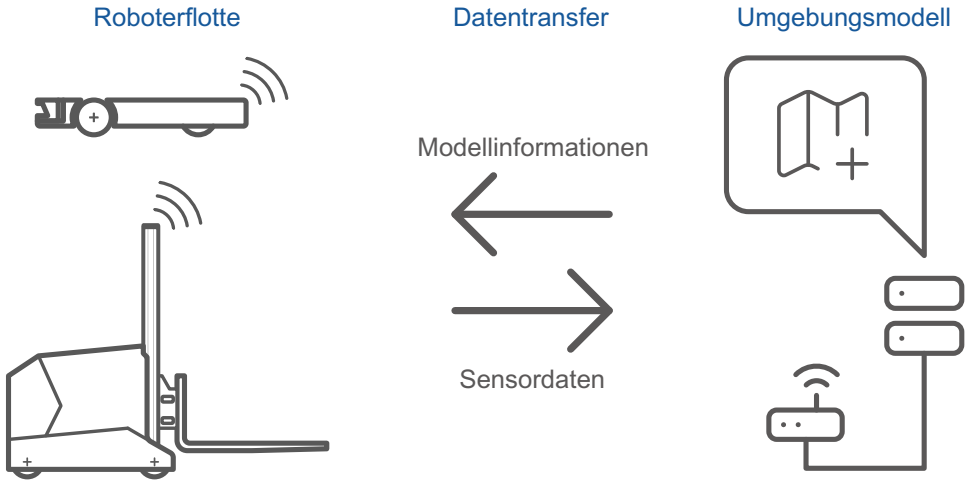


Abbildung 5: Bestandteile des Umgebungsinformationssystems: Roboterflotte, Datentransfer, Umgebungsmodell

Die Entwicklung des Umgebungsinformationssystems wurde in sieben Teilschritte gegliedert (s. Abbildung 6). Nach einer Anforderungsanalyse und einer eingehenden Betrachtung des Standes der Technik und Wissenschaft (Schritt 1) wurde zunächst mit der Entwicklung einer Simulationsumgebung begonnen (Schritt 2). Dieser Schritt war im ursprünglichen nicht vorgesehen. Während der Beantragung wurde angenommen, dass Realdaten für die Entwicklung zur Verfügung stünden. Das war jedoch aufgrund verschiedener Umstände (s. Abschnitt 3) schlussendlich nicht der Fall, weshalb die Simulation die einzige Möglichkeit für die Durchführung der nachfolgenden Arbeitspakete darstellte. In diesem Schritt wurde damit die Datengrundlage für die Entwicklung des Gesamtsystems geschaffen. Anschließend wurde basierend auf den simulierten Sensordaten aus dem vorherigen Schritt parallel in iterativer Art und Weise das Gesamtsystem konzeptioniert (Schritt 3), ein Datenmodell für die Zusammenführung der Umgebungsdaten (Schritt 4) und eine Strategie für das Management der Ein- und Ausgangsdaten (Schritt 5) erarbeitet. Diese Parallelität in der Entwicklung war notwendig, da alle Bereiche stark voneinander abhängen und eine Änderung in einem der drei Bereich direkte Auswirkungen auf die anderen Bereiche hat. In Schritt 6 wurde das Umgebungsinformationssystem demonstratorisch implementiert und nachfolgend in Schritt 7 simulativ evaluiert.

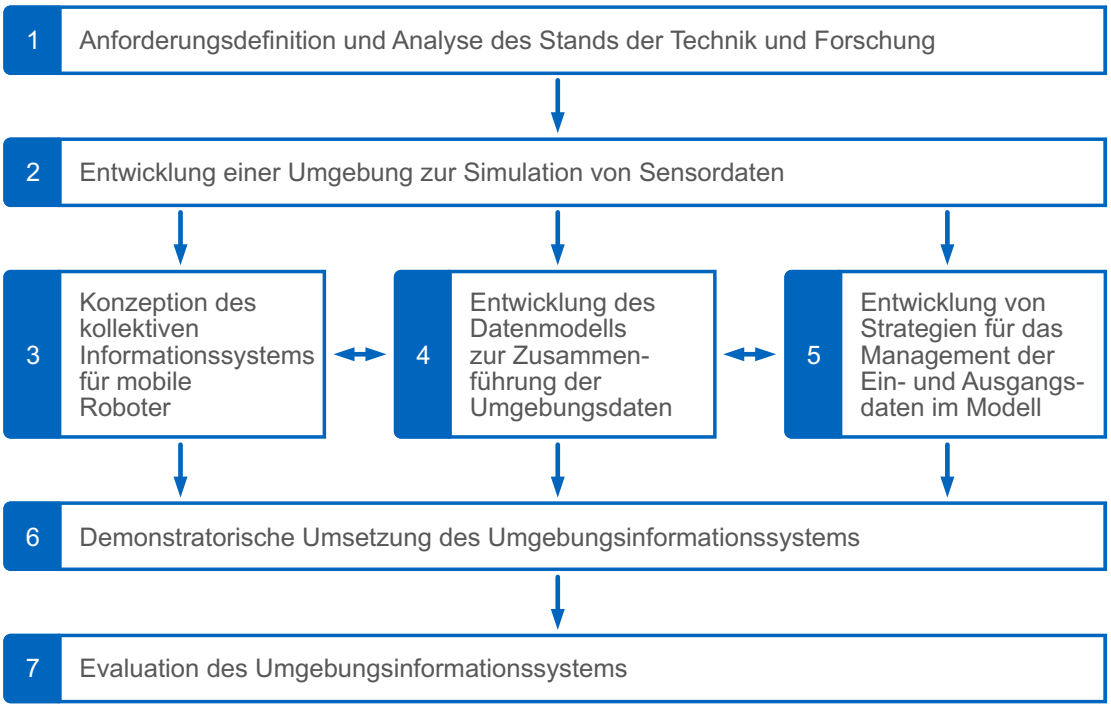


Abbildung 6: Vorgehensweise zur Entwicklung des Umgebungsinformationssystems bestehend aus sieben Schritten

2 Technische Grundlagen

Im nachfolgenden Abschnitt werden die Grundlagen beschrieben, die zum Verständnis des Gesamtansatzes notwendig sind. Beginnend bei den Grundlagen zur Simulation von Sensordaten, werden anschließend Ansätze zur Kartierung beschrieben. Um bestehende Karten mit semantischen Informationen anreichern zu können, werden verschiedene Ansätze aus dem Bereich Deep Learning eingesetzt. Deren Basiskonzepte werden danach eingeführt. Zuletzt wird ein Blick auf aktuelle Möglichkeiten zum Datentransfer zwischen Roboter und Server geworfen.

2.1 Simulation von Umgebungssensordaten

Als Simulation wird häufig der Prozess beschrieben, mithilfe von Computern Wissen über physikalische Systeme zu erlangen [7]. Sie ist eine ökonomische Alternative, komplexe System auch bei beschränkten Ressourcen zu validieren und ermöglicht es, Systeme zu entwickeln, bevor diese in Hardware umgesetzt werden [8]. Durch die Abbildung komplexer physikalischer Zusammenhänge und die Darstellung mittels realistischer Visualisierung, können Anwendungen in verschiedensten Sparten (Robotik, Automotive, Bau, Biomedizin, …) umgesetzt werden [8]. Für die Erstellung der Simulation sollte ein Simulationsziel definiert werden, aus dem sich die Randbedingungen der Simulation ableiten. Im Rahmen dieses Projekts ist das Simulationsziel die Generierung von Umgebungsdaten für mobile Roboter im industriellen Umfeld.

Nachfolgend wird daher auf verschiedene Aspekte der Simulation in den Bereichen mobiler Robotik und Sensorik eingegangen.

2.1.1 Möglichkeiten und Herausforderungen der Simulation von Sensordaten

Choi et al. beschreiben in [9] verschiedenste Potenziale, die sich aus der Simulation in Robotik-Anwendungen ergeben. Für die Simulation von Sensordaten sind vor allem die folgenden Punke relevant:

- *Schnellere Entwicklung*
  Da keine Hardware gefertigt werden muss, können Design und Regelungsstrategien eines Roboters schneller auf Tauglichkeit geprüft werden. Dadurch findet eine Beschleunigung des Entwicklungszyklus bei gleichzeitiger Reduzierung der Kosten statt.
- *Trainingsdaten für maschinelles Lernen*
  Durch Simulation können große Mengen an Trainingsdaten schnell und kostengünstig zur Anwendung mit maschinellem Lernen generiert werden. Das ermöglicht eine schnelle Entwicklung von Regelungs- und Umgebungserkennungsansätzen.
- *Vereinfachter Versuch*
  Verschiedenen Szenarien und gezielte Angriffsmethoden abzutesten ist in der Regel zeitaufwändig, teuer und zum Teil für Mensch und Roboter risikobehaftet. Durch die vollständig kontrollierte Umgebung einer Simulation, können Versuch und Verifikation unter sicheren Bedingungen stattfinden und dadurch beschleunigt werden. Zusätzlich können unter diesen Umständen insbesondere Multi-Roboter-Systeme bei geringerem Risiko betrachtet werden.

Dem entgegen stehen einige Herausforderungen, die die Umsetzung in einer Simulation be- oder sogar verhindern können [9]:

- *Unvollständige Abbildung von Unsicherheiten*
  Simulationen sollen die Realität bestmöglich abbilden können. Allerdings sind besonders die Unsicherheiten und Ungenauigkeiten, denen ein reales System unterworfen ist, schwierig abzubilden. Häufig entstehen daher zu stark idealisierte Daten.
- *Aufwendige Kalibrierung*
  Hinter jedem Simulationsmodell stehen verschiedene Parameter, die Verhalten und Interaktion der einzelnen Bestandteile bestimmen (bspw. Reibung zwischen Roboter und Boden). Diese Parameter bestimmen maßgeblich die Qualität des Modells. Sie korrekt zu ermitteln und so in das Modell zu übertragen, dass die Realität bestmöglich abgebildet wird, ist zeitaufwendig.
- *Unausgereifte Modellzusammenstellung*
  Komplexe Simulationen werden üblicherweise aus verschiedenen Einzelbausteinen zusammengesetzt. Das bezieht sich nicht nur auf hierarchisch aufgebaut 3D-Modelle, sondern auch auf Funktionsbausteine verschiedener anderer Disziplinen (z. B. Regelungstechnik). Diese Aufteilung kann insbesondere bei paralleler Berechnung zu temporalen und örtlichen Inkohärenzen führen.
- *Reality Gap [10]*
  Komplexe Modelle, die hochgradig akkurate Ergebnisse erzeugen, erfordern häufig eine große Vielzahl an Parametern und lange Rechenzeiten. Daher müssen in der Regel an verschiedenen Stellen Annahmen zur Vereinfachung getroffen werden. Die Diskrepanz zwischen dem vereinfachten Simulationsmodell und dem realen Vorbild wird häufig als „Reality Gap“ bezeichnet und ist verantwortlich für die oft schlechte Übertragbarkeit von simulierten Ergebnissen auf reale Anwendungen.

2.1.2 Entwicklungsumgebungen für Sensordatensimulation

Der Markt hält verschiedene Entwicklungsumgebungen für Simulationen vor. Da die Realität sehr komplex ist und simulativ nicht vollständig abgebildet werden kann, müssen Simulationsumgebungen stark abstrahieren. In der Regel sind sie darüber hinaus auf einzelne Teilbereiche des Gesamtproblems beschränkt. Nach Art ihrer Anwendung können sie verschiedenen Typen zugeordnet werden:

- *Physik-Engines*
  Dieser Art Simulation wird nur für die Berechnung grundlegender physikalischer Effekte eingesetzt. In der Regel gibt es kein grafisches Interface und es können keine Modelle erzeugt werden. Physik Engines werden meist als Bestandteil der anderen Simulationstypen eingesetzt. Beispiele sind MuJoCo [11] oder Pybullet [12].
- *Spezialisierte Simulationsumgebungen*
  Diese Art Simulationsumgebung liefert spezialisierte Ergebnisse für einen bestimmten Anwendungsfall. Sim-Grasp [13] beispielsweise ist darauf beschränkt Greifszenarien für Greifroboter abzubilden. Blender [14] kann hochrealistische Bilder für Kameraanwendungen erstellen, ist jedoch nicht für die Berechnung anderer Sensorik ausgelegt.
- *Komplettlösungen*
  Daneben gibt es Simulationsumgebungen, die einen ganzheitlichen Ansatz verfolgen und alle mit der Simulation von Sensordaten erforderlichen Teilbereiche abdecken. Ein sehr bekanntes Beispiel ist Gazebo [15], das eng mit dem Robot Operating System (ROS) verknüpft ist. Es erlaubt die

Simulation mobiler, wie stationärer Roboter und verschiedener Sensoren. Nvidia hat mit Isaac Sim [16] ein ähnliches Produkt auf dem Markt etabliert, was sich insbesondere durch detailgetreue Darstellung und die starke Ausrichtung auf Sensordatensimulation für Machine-Learning-Anwendungen abhebt.

- *Game-Engines*
  Konzipiert für die Entwicklung von Videospielen, finden sind auch Game Engines immer öfter bei der Entwicklung von Robotersimulationen anzutreffen. Die Simulationen profitieren von guter optischer Darstellung bei oft ausreichender physikalischer Genauigkeit und hoher Flexibilität in der Entwicklung. Bekannte Beispiele sind Unity [17] und Unreal [18].

2.1.3 Beschreibung von Industrieumgebungen

Für die Simulation mobiler Roboter muss der Umgebungskontext festgelegt werden, in dem sich die Roboter bewegen. Häufige Anwendungen sind beispielsweise die Modellierung von Unterwasserfahrzeugen, Flugrobotern oder medizinischen Robotern [19]. Je nach Umgebung und Simulationsziel ergibt sich ein entsprechender Kontext wie z. B. Korallenriffe oder Krankenhausflure. Das Simulationsziel ergibt sich aus dem angestrebten Ergebnis. Das kann z. B. die Abbildung physikalischer Zusammenhänge sein oder im Fall von Sensordatensimulation die Generierung von Sensordaten im Simulationskontext. Für eine aussagekräftige Simulation ist es von essenzieller Bedeutung, diesen Kontext genau zu kennen und in der simulierten Umgebung entsprechend den aufgestellten Randbedingungen ausreichend genau abzubilden. Da die hier betrachteten mobilen Roboter überwiegend im Bereich der Industrieanwendung eingesetzt werden, wird im Folgenden näher auf die für diese Umgebung spezifischen Rahmenbedingungen eingegangen.

Nach Hompel et al. [20, S. 3] sind in der Logistik folgende Gegenstände vertreten:

- **Transformierte Objekte**
  - *Güter*
  - *Informationen*
  - *Energie*
- **Transformierende Objekte**
  - *Produktionsmittel*
  - *Materialflussmittel*
  - *Informationsflussmittel*
  - *Infrastruktur (Gebäude, Flächen, Wege)*
- **Personen**
  als Verkehrsteilnehmer

Transformierte Objekte sind Objekte, die „im Laufe Ihres Aufenthalts im logistischen System einen Transformationsprozess durchlaufen“ [20, S. 3], während Transformierende Objekte „als Arbeitsmittel (Materialflussmittel, Produktionsmittel, Informationsflussmittel) zusammen mit der notwendigen Infrastruktur (Gebäude, Flächen, Wege) die Änderung der Objekte in Systemen bewirken“ [20, S. 3]. Diese Einteilung kann aufgrund ihrer allgemeingültigen Formulierung problemlos vom Logistikbereich verallgemeinernd auf generelle Industrieanwendungen angewendet werden.

Für die Simulation von Sensordaten sind insbesondere die Objekte relevant, die durch Sensorik erfasst werden können. Für Umgebungsdaten werden überwiegend optisch und akustisch arbeitende Sensoren, wie Kamera, Light Detection And Ranging (LiDAR) oder Ultraschall-Sensoren eingesetzt (vgl. Abschnitt 2.1.5). Daher

sind besonders schalharte und reflektierende Objekte relevant. Das betrifft Güter, Produktionsmittel, Materialflussmittel, Infrastruktur und Personen. In Tabelle 1 sind für diese Kategorien Beispiele gegeben.

Tabelle 1: Beispiele für Objekte in den für Simulation relevanten Kategorien

Güter	**Produktionsmittel**	**Materialfluss-mittel**	**Infrastruktur**	**Personen**
• Materialien o Bleche o Dichtungen o … • Produkte o Kotflügel o Tür o …	• Stanzmaschine • Biegemaschine • Schweißanlage • …	• Gütertrans-portmittel o Gabel-stapler o AMR o Band o … • Personen-transport-mittel o Roller o Fahrräder o …	• Gebäude • Flächen • Wege • Regale • Säulen • …	• Mitarbeitende Band • Mitarbeitende Entwicklung • Besucher/ innen • …

Die Ausgestaltung der einzelnen Kategorien ist abhängig vom übergeordneten Simulationsziel. Es ist beispielsweise nicht in jedem Fall notwendig, alle Kategorien in der Simulation abzubilden. Außerdem sollte sich bei der Anordnung der Objekte ebenfalls am Simulationszweck und darüber hinaus an realen Vorbildern orientiert werden, um ggf. eine Übertragbarkeit auf Realprobleme sicherzustellen [21].

2.1.4 Beschreibung Mobiler Roboter

Ullrich et al. unterscheiden in [22] zehn verschiedene Bauformen von FTF:

Gabelhub-FTF als Serien- und Sondergerät, Huckepack-FTF, Schlepper, Unterfahr-FTF, Montage-FTF, Schwerlast-FTF, Mini-FTF, Outdoor-FTF und Sonder-FTF

Jede dieser Bauformen ist für typische Lasten und Anwendungsszenarien optimiert, wie z. B. Transport von Regalen für die Produktionsversorgung. Da sich FTF und AMR in der Regel nur durch unterschiedliche Sensorik und Software unterscheiden, lassen sich beinahe alle Bauformen auch auf AMR übertragen. Besonders häufig sind Unterfahr- und Gabelhub-AMR anzutreffen.

Mobile Roboter bestehen aus verschiedenen Komponenten, die je nach Anwendung unterschiedlich ausgestaltet werden. In Bezug auf die Hardware sind Chassis, Manipulatoren, Antriebsstrang, Energieversorgung, eine zentrale Recheneinheit, Not-Aus-Einrichtungen und Sensoren notwendig [23].

Für einen Unterfahr-Roboter muss das Chassis beispielsweise eher niedrig sein, um die Lasten unterfahren und mit einer Hebeeinheit als Manipulator anheben zu können. Häufig wird ein Differentialantrieb für gute Manövrierbarkeit eingesetzt. Gabelhub-Roboter manipulieren Lasten hingegen mithilfe der Gabeln und benötigen ein erheblich größeres Chassis, um hohe Stellen erreichen zu können und Gegengewichte unterzubringen.

Um den Roboter zu steuern, wird darüber hinaus Software benötigt, die auf der zentralen Recheneinheit läuft. Elementar ist hier die Pfad- und Bewegungsplanung. Darüber werden Pfade auf globaler Ebene in Abhängigkeit der eingesteuerten Aufträge geplant. Mithilfe der verbauten Sensorik können dann lokal Bewegungen um Objekte herum geplant werden, die in der globalen Karte nicht verzeichnet sind. Dieser Prozess erfordert eine vorab aufgenommene Karte. Soll diese während des Betriebs aufgezeichnet werden, kommt ein sogenannter Simultaneous-Localization-And-Mapping Ansatz zum Einsatz. Hier lokalisiert sich der Roboter in der Karte, während er gleichzeitig diese Karte aufnimmt [23].

Je nach Ziel und angestrebter Komplexität der Simulation sind für die Abbildung mobiler Roboter in der Simulation verschiedene Komponenten unterschiedlich relevant. Bei der Simulation von Sensordaten ist vor allem das Erscheinungsbild und die Art und Positionierung der Sensoren relevant. Der Antrieb hingegen kann durch fest einprogrammierte Bewegungen abgebildet werden und muss nicht detailliert ausgestaltet werden. Sollen Interaktionen mit Lasten betrachtetet werden, müssen Manipulatoren in der Simulation inkl. korrektem physikalischen Verhalten abgebildet werden.

2.1.5 Sensorik für Mobile Roboter

Für die Simulation von Sensordaten ist insbesondere eine korrekte Implementierung von Sensoren in der Simulation essenziell. Nach Haun [24] sind Sensoren „Vorrichtungen, die die physikalischen Eigenschaften der Umgebung messen können“. Sensoren, die die Umwelt des Roboters z. B. zur Kollisionsvermeidung erfassen, werden als exterozeptive Sensoren bezeichnet [25, S. 22]. Typische Vertreter im Bereich der mobilen Robotik sind [23]:

- 2D und 3D LiDAR
- 3D-Tiefen-Kamera
- Ultraschall
- Infrarot

Der Sensorerfassungsbereich wird als Field Of View (FOV) bezeichnet [25, S. 31]. Bei 2D-LiDAR-Sensoren ist das beispielsweise der von den Laser-Strahlen überstrichenen Bereich in Grad, bei Kameras der durch den Sensor erfasste Ausschnitt der Umgebung.

Die Wahl der Sensoren ist abhängig von der angestrebten Anwendung einerseits, aber auch von der vorgeschriebenen Sicherheitseinstufung andererseits. In DIN EN ISO 13849 [26] sind dafür sogenannte Performance Level festgelegt. Diese bestimmen, welche Sensoren für eine bestimmte Anwendung eingesetzt werden müssen, um anhand der vorgeschriebenen Ausfallwahrscheinlichkeit den sicheren Betrieb der Fahrzeuge gewährleisten zu können.

DIN EN ISO 3691-4 [27] macht darüber hinaus Vorgaben, welche Funktion eines Flurförderzeugs welches Performance Level voraussetzt. Da die Umgebungssensorik insbesondere für den Schutz von Menschen zuständig ist, muss diese Funktion besonders stark abgesichert sein. Das überträgt sich auch auf den dafür eingesetzten Sensor. Aktuell erfüllen die gestellten Anforderungen fast ausschließlich LiDAR-Sensoren. Da sich mit LiDAR-Sensoren auch Lokalisierung und Kartierung gut umsetzen lassen, ist bei den heutigen mobilen Robotern immer mindestens ein LiDAR verbaut (vgl. 3.1.1).

Die Positionierung des LiDARs muss dabei so erfolgen, dass Menschen sicher erkannt werden können. Die DIN EN ISO 3694-4 stellt dies durch zwei verschiedene Tests sicher. Insbesondere Test A limitiert dabei die mögliche Montagehöhe. Hier liegt ein Zylinder mit 200 mm Durchmesser und 600 mm Länge auf dem Boden

und muss durch die Sensoren zuverlässig erkannt werden. Aufgrund von Toleranzen und Montagewinkeln bedeutet das in der Praxis eine maximale Höhe von ca. 180 mm. Die minimale Montagehöhe ist meist durch die steigende Anzahl an unbeabsichtigten Bodenreflektionen begrenzt.

3D-Tiefenkameras werden meist nur für spezifische Anwendungen, wie z. B. Feinpositionierung, eingesetzt [23]. Ihre Montageposition kann daher in der Regel unabhängig von Sicherheitsaspekten gewählt werden.

2.1.6 Datenformat zum Speichern von Datensets

Wird eine Datenset erstellt, muss eine geeignete Struktur festgelegt werden, in der dies Daten abgespeichert werden. Das erleichtert die spätere Nutzung der Daten. Wie in Abschnitt 3.1.2 beschrieben, wurde für dieses Projekt die Game Engine Unity für die Generierung synthetischer Sensordaten gewählt. Dafür stellt Unity das sogenannten „Perception Package“ [28] bereit. Es beinhaltet verschiedene Funktionen zur Erzeugung von Ground-Truth-Daten (s. Abschnitt 2.3.5) für Kamera-Sensoren. Darüber hinaus bietet es die Möglichkeit, selbst erstellte Sensoren in das Framework einzubinden. Die Struktur, in der die erzeugten Datensets gespeichert werden, wird als SOLO (Synthetic Optimized Labeled Objects) bezeichnet [29]. Es ist wie folgt aufgebaut:

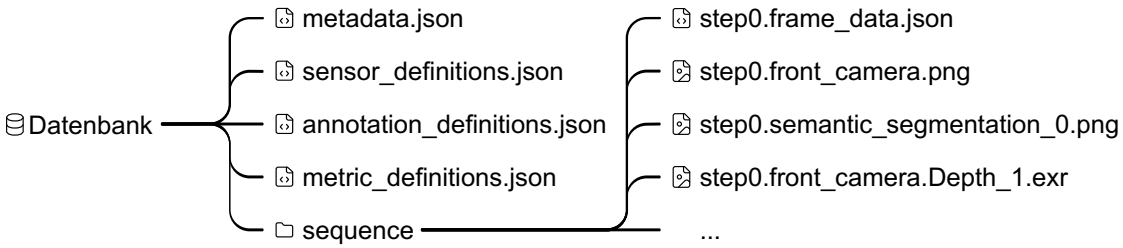


Abbildung 7: SOLO-Datenstruktur nach [30]; Hauptebene links, Sequenz-Ebene rechts

In der obersten Ebene liegen Dateien für Metadaten zum Datenset, Sensordefinitionen, Annotationsdefinitionen und weitere definierte Metriken. Daneben gibt es einen Ordner für eine Sequenz (sequence), was einem Simulationsdurchlauf entspricht. In diesem Ordner gibt es für jeden Simulationsschritt (frame) sog. Frame-Daten, die die Annotationen, wie z. B. Bounding-Boxen, und Pfade für die zugehörigen Bilder enthalten. Aus dieser Datei können alle Zusatzinformationen zum entsprechenden Zeitschritt rekonstruiert werden. Außerdem liegen dort, je nach Konfiguration des Sensors, das 2D-RGB-Bild, die zugehörige Segmentierungsmaske und das Tiefen-Bild ab.

2.2 Kartierung und Lokalisierung

Als robotische Kartierung wird der Vorgang bezeichnet, mithilfe verbauter Sensorik ein räumliches Modell der Umgebung des Roboters zu erstellen, das häufig für die Navigation eingesetzt wird [31].

In der Literatur werden vier Arten von Kartierungsmodellen nach Art der Umgebungsrepräsentation unterschieden [32]:

- Metrische Karten
- Topologische Karten
- Hybride Karten
- Semantische Karten

Während metrische Karten die Umgebung durch ein geometrisches Abbild repräsentieren, abstrahieren topologische Karten die Umgebung in Graphen ohne direkte geometrische Darstellung. Die Eigenschaften beider Karten-Typen lassen sich in hybriden Karten verbinden. Werden zusätzlich weitere deskriptive Eigenschaften zur Umgebung gespeichert, spricht man von semantischen Karten.

Im Folgenden werden die Unterschiede zwischen den einzelnen Arten erläutert und beispielhaft Ansätze vorgestellt.

2.2.1 Metrische Karten

Wie zuvor erwähnt, beschreiben metrische Karten die Umgebung durch geometrische Information. Sie werden oft in Form einer sogenannten Grid-Map umgesetzt. Bei diesen wird die Umgebung in uniforme Zellen diskretisiert, wobei jede davon ein zwei- oder drei-dimensionales Stück der Umgebung beschreibt [31].

In diesem Zusammenhang besonders erwähnenswert ist die Occupancy-Grid-Map. Sie wurde ursprünglich von Moravec und Elfes [33] entworfen und hat seitdem in zahllosen Ausprägungen verschiedenste Verbesserungen erfahren. Grundsätzlich wird jeder Zelle ein Wert zugeordnet, der die Belegungswahrscheinlichkeit dieser Zelle beschreibt. Mithilfe der Sensormesswerte werden die Wahrscheinlichkeitswerte in jedem Zeitschritt aktualisiert. Wird ein festgelegter Grenzwert überschritten, wird die Zelle als belegt und damit nicht befahrbar markiert. Auf diese Weise kann nach und nach ein Abbild der Umgebung erzeugt werden [31].

Abbildung 8 zeigt, wie eine Occupancy-Grid-Map aussehen kann:

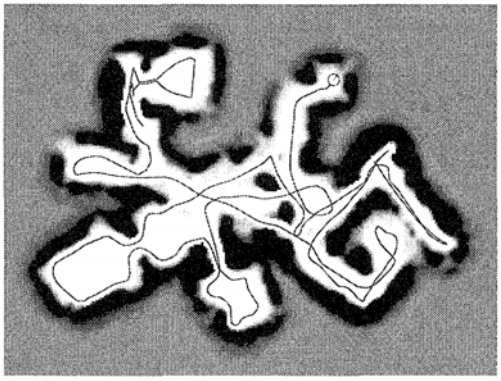


Abbildung 8: Beispiel Occupancy-Grid-Map [34]; dunkle Bereiche: Hohe Belegungswahrscheinlichkeit, helle Bereiche: geringe Belegungswahrscheinlichkeit, graue Bereiche: unbekannte Belegungswahrscheinlichkeit

In der Regel repräsentieren dunkle Bereiche hohe Belegungswahrscheinlichkeiten, während helle Bereiche geringe Belegungswahrscheinlichkeiten repräsentieren. Ist ein Bereich grau, so kann keine genaue Aussage über dessen Belegung getroffen werden, da er durch den Sensor noch nicht erfasst wurde.

Vorteile von metrischen Karten sind deren einfache Generierung, Darstellung und Aktualisierung. Außerdem ist die Wiedererkennung von Orten immer eindeutig und blickwinkelunabhängig. Eine Berechnung des kürzesten Pfads innerhalb der Karte ist verglichen mit anderen Kartierungsmethoden einfach möglich [34].

Demgegenüber steht eine ineffiziente Nutzung von Speicherplatz, da die Auflösung unabhängig von der Komplexität der Umgebung ist. Die Auflösung wird daher immer durch die kleinste darzustellende Struktur bestimmt, was für große Strukturen in einfachen Umgebungen möglicherweise zu unnötig hohem Speicherplatzbedarf führen kann. Außerdem muss für eine hohe Genauigkeit der Karte die Roboterposition sehr genau erfasst werden können. Darüber hinaus bieten metrische Karten eine schlechte Schnittstelle für die meisten symbolischen Problemlöser [34]. Da bei jeder neuen Messung alle durch den Sensor tangierten Zellen der Karten aktualisiert werden müssen, kann die Performanz bei Sensoren mit hoher Auflösung und/oder Wiederholrate sinken.

2.2.2 Topologische Karten

Im Gegensatz zu metrischen Karten abstrahieren topologische Karten die Umgebung in einen Graph. Der Graph setzt sich aus Knoten und Kanten zusammen. Knoten repräsentieren charakteristische Landmarken (z. B. Reflektoren), die durch Kanten (Wege) miteinander verbunden sind. Diesen Wegen können zusätzliche Eigenschaften zugewiesen werden (z. B. Distanz, Fahrzeit) [34].

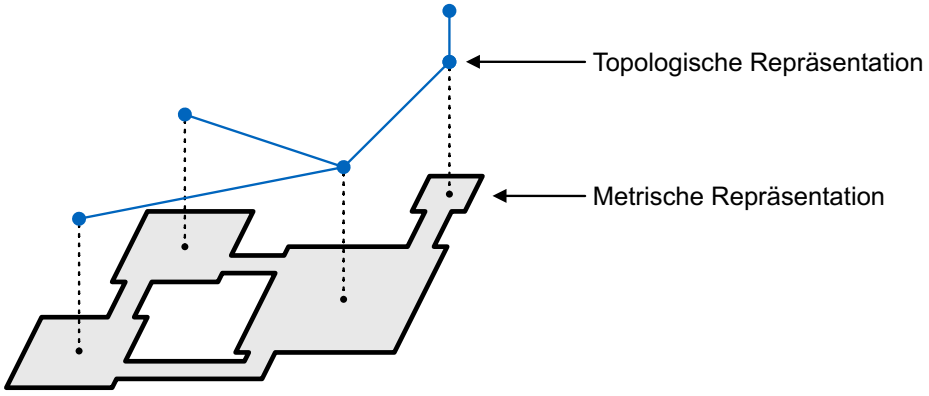


Abbildung 9: Beispiel einer topologischen Karte nach [35]; metrische Repräsentation unten und abstrahierte topologische Repräsentation oben; Knoten werden verbunden durch Kanten

Abbildung 9 zeigt das Beispiel einer topologischen Karte. Unten ist die Umgebung in Form einer metrischen Repräsentation abgebildet. Darüber ist die topologische Karte durch mit Kanten verbundenen Knoten (Zahlen entsprechen Raumnummern) dargestellt. Knoten haben immer Landmarken in der Umgebung des Roboters als Entsprechung.

Da sich der Roboter nur relativ zu Landmarken lokalisieren muss, reicht eine gröbere Lokalisierung aus. Darüber hinaus ist die Darstellung vor allem in größeren Umgebungen erheblich ressourceneffizienter verglichen zu metrischen Ansätzen und bietet eine einfache Schnittstelle zu symbolischen Planern, Problemlösern und Schnittstellen für natürliche Sprache [34].

Gerade in großen Umgebungen sind topologische Karten allerdings schwieriger zu erzeugen und zu aktualisieren. Das resultiert auch aus der Tatsache, dass die Erkennung von Orten basierend auf Landmarken oft uneindeutig und abhängig vom Blickwinkel ist. Die Pfadplanung liefert darüber hinaus möglicherweise nicht optimale Ergebnisse [34].

2.2.3 Hybride Karten

Die Vorteile metrischer und topologischer Karten lassen sich in sogenannten hybriden Karten verbinden. Vor allem die bessere Speicherskalierung und globale Pfadplanung topologischer Karten in großen Umgebungen und die bessere lokale Navigation und Lokalisierung bei metrischen Karten können durch Kombination der Ansätze gleichzeitig genutzt werden [34]. So schlagen beispielsweise Gomez et al. [35] eine globale topologische Karte vor, die durch 3D-Grid-Maps für die lokale Navigation ergänzt wird.

2.2.4 Semantische Karten

Semantische Karten erweitern die zuvor genannten Kartentypen um weitere Informationen zur Umgebung. Das können Beschreibungen der Räume oder Klassifizierungen von Objekten sein [32].

Crespo et al. [36] beschreiben semantische Navigation als Paradigma, abstrakte Konzepte wie Objekte oder Orte in den Navigations-Stack (u. A. die Karte) zu integrieren. Außerdem werden Beziehungen zwischen diesen Konzepten genutzt, um beispielsweise zu entscheiden, an welchen Orten welche Objekte zu erwarten sind oder welche Aktionen auf welche Objekte angewendet werden können.

Daraus ergeben sich folgende Möglichkeiten:

- *Modelle nachvollziehbar für Menschen* [36], [32]
  Die Roboter erfassen die Umgebung nach denselben Konzepten wie Menschen. Die Modelle sind daher auch leichter für Menschen verständlich.
- *Autonomie* [36]
  Der Roboter kann selbstständig Schlüsse ziehen, zu welchem Raum er fahren muss.
- *Robustheit* [36]
  Dem Roboter ist es möglich, fehlende Informationen zu kompensieren.
- *Verbesserte Lokalisierung* [36]
  Durch Informationen über die Umgebung ist es dem Roboter möglich, einen Abgleich mit Wissen über seine Umgebung durchzuführen. Dadurch kann die Lokalisierung überprüft und verbessert werden.
- *Zusätzliche Funktionen*
  Semantische Informationen ermöglichen zusätzliche Funktionen wie das Verfolgen von Objekten über größere Distanzen (bessere Schätzung durch Kenntnis der Objekteigenschaften) oder die Umsetzung komplexerer Pfadplanungsalgorithmen

2.3 Semantik

Sollen, wie im vorherigen Abschnitt angesprochen, Karten mit semantischen Informationen implementiert werden, müssen diese Informationen zuerst erzeugt werden. Nachfolgend werden daher nach einer allgemeinen Einführung in das Gebiet des Maschinellen Lernens Möglichkeiten zur Objektdetektion und

semantischen Segmentierung vorgestellt. Danach wird noch einmal gesondert auf die Bedeutung der zur Entwicklung dieser Algorithmen notwendigen Daten eingegangen.

2.3.1 Grundlagen Maschinelles Lernen

Algorithmen des maschinellen Lernens basieren auf dem Prinzip von bestehenden Daten zu lernen. Dieser Lernvorgang wird auch als Training bezeichnet.

Nach der Art des Trainings können vier Grundtypen unterschieden werden [37]:

- *Supervised Learning*
  Hier ist ein sogenanntes gelabeltes Datenset erforderlich (s. Abschnitt 2.3.5). Zu jedem Datenpunkt gibt es die korrekte Antwort (Label) auf eine zu beantwortende Frage (z. B. Welche Objektklasse ist das?). Während des Trainings wird die Antwort des Algorithmus mit der korrekten Antwort des Datensets verglichen. Durch eine entsprechende Anpassung interner Parameter lernt der Algorithmus Schritt für Schritt wesentliche Eigenschaften der Daten. Eine typische Anwendung ist die Klassifikation.
- *Unsupervised Learning*
  Im Gegensatz zum Supervised Learning müssen Datensets für das Unsupervised Learning keine korrekten Antworten enthalten. Ziel ist, aus den Daten Trends oder Ähnlichkeiten abzuleiten. Das Zusammenfassen gleicher Datenpunkte (Clustering) ist ein typischer Anwendungsfall von Unsupervised learning.
- *Semi-Supervised Learning*
  Dieser Ansatz liegt zwischen Un- und Supervised Learning. Anhand weniger gelabelter Datenpunkte in einem sehr viel größeren Datenset werden Rückschlüsse auf die Eigenschaften der übrigen ungelabelten Datenpunkte gezogen. Das erspart viel Aufwand beim Labeln, ist jedoch immer noch Gegenstand aktueller Forschung.
- *Reinforcement Learning*
  Hier soll der Algorithmus auf Basis seines eigenen Zustands und des Zustands seiner Umgebung Entscheidungen treffen, sodass eine vorgegebene Zielgröße maximiert wird. Dieser Ansatz eignet sich besonders für Probleme mit sich ständig ändernden Umgebungen und großen Zustandsräumen.

Ein wesentliches Feld von Machine Learning ist Computer Vision. Nach Szeliski [38, S. 3] ist es der Versuch, die auf Bildern dargestellte Umwelt zu rekonstruieren und ihre Eigenschaften zu ermitteln. Anwendungen sind z. B. die Erkennung von Schrift, Inspektion von Maschinen oder verschiedene Bereiche in der Intralogistik, wie autonome Paketzustellung oder Picking-Applikationen [38, S. 3–4]. Computer Vision ist damit eine Möglichkeit, auf Basis von Umgebungssensordaten semantische Informationen über die Umwelt zu erlangen. Da diese Informationen für die Modellbildung benötigt werden, wird im Folgenden auf die Teilbereiche Objektdetektion und semantische Segmentierung näher eingegangen.

2.3.2 Objektdetektion mittels Deep-Learning-Algorithmen

Torralba et al. [39] beschreiben Objektdetektion als Funktion, die einem Bild eine Liste von Bounding-Boxen mit zugehöriger Klasse zuordnet. Bounding-Boxen werden definiert als Repräsentation für Objektlokalisierung in Bildern, die jeder Instanz einer Klasse Bild-Koordinaten einer sie eng umfassenden Box zuordnet.

Abbildung 10 zeigt beispielhaft Objektdetektion in einer simulierten Industrieumgebung. Jede eingezeichnete Box umrandet die Instanz einer bestimmten Klasse. Jede braune Box zeigt hier beispielsweise die Position einer Palette in Bild-Koordinaten, während rosa Boxen die Position von Kartons anzeigen.

Abbildung 10: Beispiel für Objektdetektion; jede Box umrandet die Instanz einer Klasse (Bsp. braune Box → Palette)

Die Veröffentlichung des COCO-Datensatzes 2014 [40] fiel in etwa zusammen mit einer verstärkten Orientierung weg von klassischen Ansätzen hin zu Deep-Learning-Algorithmen. Diese konnten in den nachfolgenden Jahren die Genauigkeit der Objekterkennung erheblich steigern [38, S. 302]. Vor allem Ansätze, die dem Supervised Learning angehören und auf großen gelabelten Datensätzen beruhen, konnten seit dem Durchbruch von „AlexNet“ 2012 [38, S. 17] große Erfolge erzielen.

Einer der bekanntesten Algorithmen ist YOLO (You Only Look Once), ursprünglich veröffentlicht im Jahr 2015 von Redmon et al. [41]. Seitdem wurde er bis zur Version 11 aus dem Jahr 2024 weiterentwickelt [42].

Bis zur Veröffentlichung von YOLO waren hauptsächlich sogenannte Two-Stage-Detektoren weit verbreitet. Diese lokalisieren zuerst mögliche Objekte in einem sogenannten Region-Proposal-Schritt. Anschließend klassifizieren sie die möglichen Objekte [43], [44]. Mit der Einführung von YOLO wurden beide Schritte kombiniert, was die Geschwindigkeit der Detektion erheblich erhöhte. YOLO wird demnach auch als Single-Stage-Detektor bezeichnet und ist vor allem für Echtzeit-Anwendung von Interesse [45].

Für das Detektieren von Objekten durch YOLO wird das Bild in ein $S \times S$ großes Gitter zerlegt (s. Abbildung 11). Jede dieser Gitterzellen macht eine Vorhersage für B Bounding-Boxen und einen Confidence-Score $conf$, der angibt, wie sicher sich der Algorithmus ist, dass die Gitterzelle ein Objekt enthält und wie präzise dieses getroffen wurde. Zu jeder Bounding-Box gehören vier Werte: x, y, w, h. Diese Werte bezeichnen den Mittelpunkt (x, y), Höhe und Breite (w, h). Dazu kommen C Werte für die einzelnen bedingten Wahrscheinlichkeiten für jede Klasse $(c_1, \dots, c_n)$. Um die Confidence-Scores für einzelne Klassen zu erhalten, wird der Confidence-Score der Zelle $(conf)$ mit der jeweiligen bedingten Wahrscheinlichkeit $c_1, \dots, c_n$ multipliziert. Damit ergibt sich der Ergebnis-Tensor insgesamt zu $S \times S \times (B * 5 + C)$ [41].

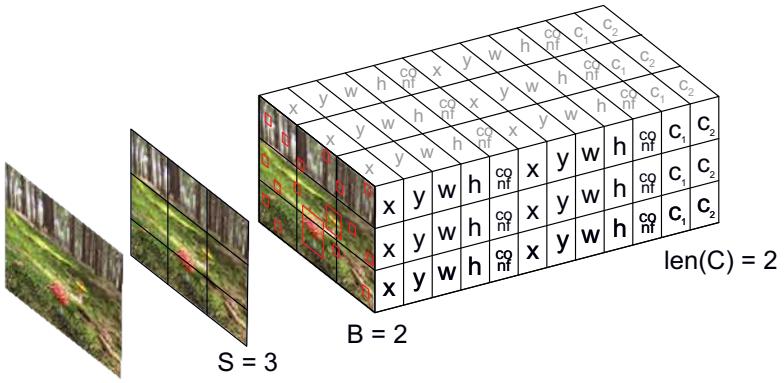


Abbildung 11: Entstehung des Ergebnis-Vektors in der Dimension 3x3x12 bei einer Gittergröße von S=3, einer Bounding-Box Anzahl von B=2 und einer Länge des Klassenvektors von C=2; Eingangsbild (links), Bild mit Gitter (mittig), Bild mit Bounding-Boxen (rot) und Ausgangsvektor (rechts)

Wird YOLO auf ein Bild angewendet, können so Klassen und Bounding-Boxen in einem Schritt erzeugt werden.

2.3.3 Semantische Segmentierung mittels Deep-Learning-Algorithmen

Werden bei der Objektdetektion einzelne Objekte durch Bounding-Boxen voneinander getrennt, findet bei der semantischen Segmentierung eine Trennung auf Pixel-Ebene statt. Das bedeutet, jedem Pixel wird eine Klasse zugeordnet, wobei einzelne Instanzen einer Klasse nicht unterschieden werden können. Üblicherweise werden diese Algorithmen durch Deep-Learning-Ansätze mit Encoder-Decoder-Struktur umgesetzt [39].

Dadurch wird das Bild zunächst im Encoder-Teil durch die Extraktion hierarchischer Merkmale (Kanten, Texturen, Objektteile) in eine stark abstrahierte Darstellung überführt. Anschließend wird diese Darstellung durch den Decoder-Teil in den Pixel-Bereich projiziert, um eine dichte, pixelweise Klassifizierung zu erhalten [46].

Einer der bekanntesten Algorithmen für semantische Segmentierung ist DeepLabv3+ [47] [48]. Dieser kann aufgrund seiner Architektur, Merkmale skalierungsunabhängig erkennen. So kann die gleiche Klasse in einem Bild trotz unterschiedlicher Entfernungen oder Größen dennoch erkannt werden [47].

2.3.4 Metriken zur Bewertung von Deep-Learning-Algorithmen

Um insbesondere Deep-Learning-Algorithmen bewerten und vergleichen zu können, haben sich verschiedene Metriken etabliert:

- *Intersection over Union (IoU) [38, S. 302 f.]*
  Intersection over Union beschreibt die Überlappung zwischen prädizierter Lösung des Algorithmus und dem Ground-Truth-Label des Datensets im Verhältnis zur Kombination beider Flächen. Für Bounding-Boxen bedeutet das die überlappende Fläche zwischen den Boxen, für semantische Segmentierung die Anzahl überlappender Pixel.
- *Precision [38, S. 302 f.]*
  Als Precision wird das Verhältnis von True Positive - TP (also korrekten Prädiktionen) zu allen gemachten Prädiktionen (True Positive und False Positive - FP) bezeichnet:
  $$Precision = \frac{TP}{TP + FP}$$
  Sie beantwortet also die Frage, wie viele der gemachten Prädiktionen korrekt sind.

- *Recall [38, S. 302 f.]*
  Recall hingegen bezeichnet die Anzahl korrekter Prädiktionen (TP) im Verhältnis zu allen auftretenden Positiven (True Positive und False Negative - FN).
  $$Recall = \frac{TP}{TP + FN}$$
  Somit wird die Frage beantwortet, ob der Algorithmus alle Instanzen der Positiv-Klasse findet.
- *Mean Average Precision (mAP) [38, S. 303]*
  Mean Average Precision wird vornehmlich zur Bewertung von Objektdetektions-Algorithmen eingesetzt. Zur Bestimmung der mAP sind alle oben beschriebenen Werte notwendig. Die Entscheidung, ob eine Prädiktion korrekt ist, wird in der Regel über die IoU bestimmt (ab einem festgelegten Grenzwert wird eine Detektion als TP oder FP gewertet). Die Average Precision beschriebt dann die Fläche unter der Precision-Recall-Kurve für die jeweilige Klasse. Wird dieser Wert über alle zu detektierenden Klassen gemittelt, spricht man von der mean Average Precision oder mAP.
  Der mAP-Wert ist damit abhängig von der Wahl des IoU-Grenzwerts. Daher wird der gewählte Grenzwert häufig als Index mit angegeben. Wird ein Bereich angegeben, wird der mAP in festen Schritten über diesen Bereich berechnet und anschließend gemittelt.
  Nachdem, diese Metrik sowohl Precision als auch Recall berücksichtigt, erlaubt sie eine gute globale Vergleichbarkeit verschiedener Algorithmen.

2.3.5 Datensets und Ground-Truth

Vor allem für das Training von Supervised Learning Algorithmen sind umfassende Datensets erforderlich. Hinter jedem Datenset stehen eine oder mehrere Fragen, die mit diesem Datenset beantwortet werden sollen. Bilddatensätze, wie z. B. MS COCO [40], haben beispielsweise häufig zum Ziel, Objekte verschiedener Klassen auf Bildern zu lokalisieren. Daher bestehen Datensätze für Supervised Learning immer aus den eigentlichen Daten (z. B. Bilder) und der dazugehörigen sogenannten Ground-Truth (alle korrekten Antworten, z. B. Klassen und Bounding-Boxen). Die korrekte Antwort auf die Frage wird als Label bezeichnet [49].

Für das Training von Supervised Learning Algorithmen wird das gesamte Datenset in zwei oder drei Teile zerlegt.

- *Training-Set [50, S. 108]*
  Durch die wiederholte Beobachtung des Training-Sets durch den Algorithmus werden die internen Parameter (Gewichte) des Algorithmus so angepasst, dass der auf dem Training-Set gemachte Fehler reduziert wird.
- *Validation-Set [50, S. 121]*
  Neben den internen Parametern gibt es auch sogenannte Hyper-Parameter, die die generelle Kapazität des Algorithmus bestimmen und vor dem Training durch den Entwickler festgelegt werden. Durch Nutzung eines Validation-Sets können neben den Gewichten auch Hyperparameter während des Trainings optimiert werden.
- *Test-Set [50, S. 104]*
  Mithilfe des Test-Sets wird nach dem Training des Algorithmus dessen Performanz gemessen. Dazu wird der Algorithmus auf das Test-Set angewendet und das Ergebnis anhand verschiedener Metriken (s. o.) ausgewertet.

Wichtig ist, dass das Test-Set nicht Teil des Training-Sets ist. So wird verhindert, dass sich der Algorithmus richtige Antworten merkt und somit dessen Generalisierung und Anwendbarkeit auf andere Probleme verloren geht (Overfitting). Das Validation-Set wird aus dem Training-Set üblicherweise mit einem Anteil von etwa 20% abgeleitet [50, S. 121].

2.4 Datentransfer zwischen mobilen Robotern und zentralem Server

Bei der Übertragung von Sensordaten von einem Roboter auf einen zentralen Server müssen verschiedene Punkte beachtet werden. Vor allem die Übertragungstechnologie, die Synchronisierung und das Zwischenspeichern der Daten sind entscheidende Herausforderungen, die näher betrachtet werden sollen. In den folgenden Abschnitten werden daher grundlegende Begriffe und Auslegungskriterien erklärt.

2.4.1 Übertragungstechnologie

Nach Seferagić et al. [51] können Übertragungstechnologien nach sieben Kriterien bewertet werden:

Reichweite — Die Reichweite gibt an, über welche maximale Strecke zwischen Sender und Empfänger eine Datenübertragung stattfinden kann. Sie hängt im Wesentlichen von der Sendeleistung, den Funk- und Ausbreitungseigenschaften, der Kodierung und Modulation und der Topologie ab. Höhere Sendeleistung und geringere Komplexität in der Kodierung erhöhen die Reichweite. Niedrige Datenraten bei niedrigen Frequenzen und Multi-Hop-Topologien verlängern die Reichweite, aber erhöhen auch die Latenzzeit [51].

Datenrate/ Transferrate — Bei der Übertragung einer bestimmten Datenmenge innerhalb eines festen Zeitintervalls zwischen zwei Punkten spricht man von Datenrate [52]. Eine Erhöhung der verfügbaren Bandbreite innerhalb eines Frequenzspektrums ermöglicht in der Regel eine höhere Datenrate. Das kann außerdem mithilfe von Modulation und Kodierung durch dichtere Anordnung der Daten im Signal erreicht werden [51].

Latenz — Latenz im Sinne der Datenübertragung beschreibt die Zeit, die zwischen der Anforderung von Daten und dem Moment vergeht, in dem die tatsächliche Datenübertragung startet [52]. Sie kann durch eine Erhöhung der Datenrate und einfachere Topologien reduziert werden. Außerdem wird sie erheblich durch die Zugriffskontrolle beeinflusst (MAC - Medium Access Control) [51].

Skalierbarkeit — Die Fähigkeit eines Systems, steigende Arbeitslasten eines quasi-stationären Systems zu bewältigen, wird als Skalierbarkeit bezeichnet. Das kann beispielsweise über die Nutzung zusätzlicher Ressourcen oder eine Anpassung der Konfiguration geschehen [53, S. 349]. Im Bereich der Übertragungstechnologie wird Skalierbarkeit primär durch die MAC bestimmt [51].

Energieverbrauch — Abgesehen vom Hardware-Design hängt der Energieverbrauch der Übertragung auch von Datenrate, Netzwerktopologie und der MAC ab. Geringe Datenraten, Multi-Hop-Topologien und komplexe Kodierung erhöhen den Energieverbrauch [51].

Zuverlässigkeit — Die Zuverlässigkeit beschreibt, wie lange die Übertragung unter festgelegten Bedingungen für einen bestimmten Zeitraum ohne Ausfälle funktioniert [52]. Sie wird direkt durch die Netzwerktopologie, die Zugriffskontrolle und Kodierung und Modulation

beeinflusst. Außerdem können in gemeinsam genutzten Frequenzspektren Kollisionen auftreten, was sich negativ auf die Zuverlässigkeit auswirkt. Fehlererkennungsmechanismen in MAC und Kodierung und Redundanzen in der Topologie können die Zuverlässigkeit erhöhen [51].

Regulierung Neben den technischen Aspekten muss auch die Regulierung von Frequenzbändern berücksichtigt werden. Nicht alle Frequenzbänder sind frei zugänglich und lokale Regelungen müssen im Betrieb beachtet werden [51].

Für den industriellen Einsatz gibt es verschiedene mögliche Übertragungsstandards. Seferagić at al. zählen unter anderem LoRa, Wi-Fi HaLow (IEEE 802.11ah), Narrowband-IoT (NB-IoT), WirelessHART, ISA100.11a oder Bluetooth Low Energy (BLE) auf. Aufgrund hoher Datenraten in Kombination mit verhältnismäßig hohen Reichweiten (bis zu mehrere hundert Meter) kommen für die Übertragung von großen Mengen an Sensordaten allerdings vor allem 5G und Wi-Fi in Frage.

Tabelle 2 vergleicht 3GPP 5G mit dem aktuell verbreiteten Standard WiFi 6 (IEEE 802.11ax) und dem Nachfolgestandard Wi-Fi 7 (IEEE 802.11be):

Tabelle 2: Gegenüberstellung von 5G und WiFi 6 (IEEE 802.11ax) & 7 (IEEE 802.11be) nach Oughton et al. [54] und John et al. [55]

	Variable	**3GPP 5G** [54]	**Wi-Fi 6 / Wi-Fi 6E (IEEE 802.11ax)** [54]	**Wi-Fi 7 (IEEE 802.11be) [55]**
Technisch	Spitzen-Datenrate	2 Gbps	9,6 Gbps 8x8	36 Gbps (trilink MLO)
	Spatial Streams	128x128	8x8	Bis zu 16x16 basierend auf Marktrecherche
	Reichweite	100-300 m bei kleinen Zellen, bis zu Dutzende km für große Zellen	< 50 m indoor bis zu 300 m outdoor	Geringer als Wi-Fi 6 aufgrund höherer Modulations- und Trägerfrequenz
Spektrum	Frequenzen	Low-band (<1 GHz) mid-band (1-7 GHz) and high-band (~24-29 GHz)	2.4 GHz, 5 GHz, 6 GHz	2.4 GHz, 5 GHz, 6 GHz
	Kanalbandbreite	20, 40, 80, 100 MHz	20, 40, 80, 160 MHz	20, 40, 80, 160, 320 MHz
Geschäftsmodell und	Bezahlmodell	Pre- or post-pay	As a Service, 'umsonst', WLAN ohne Verbindung nach außen	Vergleichbar Wi-Fi 6
	Kosten Ausrüstung Nutzer	Hoch	Niedrig	Niedriger als 5G

	Öffentlich / Privat	Üblicherweise öffentlich, Slicing oder privat	Üblicherweise privat	Vergleichbar Wi-Fi 6
	Kosten Chip / Modem	Hoch	Niedrig	Niedriger als 5G
Kompetenz	Anforderungen Installation	Hoch	Niedrig	Vergleichbar Wi-Fi 6
	Anforderungen Entwicklung	Hoch	Niedrig	Vergleichbar Wi-Fi 6

Während Wi-Fi 6 (IEEE 802.11ax) und 7 (IEEE 802.11be) vor allem durch niedrigere Kosten und hohe Datenraten charakterisiert wird, kann 3GPP 5G größerer Reichweite vorweisen. Verglichen mit Wi-Fi 6 (IEEE 802.11ax), kann Wi-Fi 7 (IEEE 802.11be) die Datenrate durch verschiedene neu implementierte Mechanismen fast verdreifachen, was jedoch mit leicht reduzierter Reichweite aufgrund der höheren Träger- und Modulationsfrequenzen einhergeht. Da Geräte erst seit Januar 2024 offiziell Wi-Fi 7 (IEEE 802.11be) zertifiziert werden können [56], kann deren Verbreitung im Verhältnis zu Wi-Fi 6 (IEEE 802.11ax) als gering angenommen werden. Offizielle Zahlen liegen zu diesem Zeitpunkt noch nicht vor. Über die Performanz der eingesetzten Übertragungstechnologie entscheidet am Ende dennoch meist die konkrete Implementierung. Nur durch sorgfältige Auslegung der Netzabdeckung, der Access-Points und Endgeräte kann die Technologie die von ihr geforderte Leistung kostenoptimal erreichen.

2.4.2 Sensordatensynchronisierung

Werden Daten aus verschiedenen Quellen fusioniert, müssen diese in Bezug gesetzt werden, um Inkonsistenzen im Ergebnis zu vermeiden (z. B. zeitliche Sprünge aufgrund falscher Reihenfolge). Beginnend von den unterschiedlichen Sensoren auf einem Roboter über die Übertragung der Sensordaten mehrerer Roboter auf einen gemeinsamen Server bis zur eigentlichen Fusion muss immer sichergestellt sein, dass die Relation der Daten untereinander erhalten bleibt. Dieses Problem beinhaltet unterschiedliche Dimensionen [57]:

- *Zeitlicher Bezug*
  Die Sensordaten werden zu unterschiedlichen Zeitpunkten aufgenommen. Um Inkonsistenzen bei der Fusion zu verhindern, muss diese Reihenfolge bewahrt werden. Dafür wird zu jeder Messung ein Zeitstempel angegeben, der den Zeitpunkt der Aufnahme angibt. So können bei Daten aus mehreren Datenquellen oder bei Verlust der Reihenfolge im Zuge der Übertragung, in einem nachträglichen Prozess die Daten entsprechend ihrer Aufnahmezeit korrekt sortiert werden.
- *Räumlicher Bezug*
  Die Sensoren sind an unterschiedlichen Stellen am Fahrzeug montiert, wobei sich das Fahrzeug zusätzlich durch seine Umgebung bewegt. Sollen die Daten daher relativ zu einer gemeinsamen Referenz fusioniert werden, muss dieser räumliche Bezug berücksichtigt werden. Dazu ist zu jeder Messung zusätzlich die lokale Position des Sensors auf dem Roboter bekannt. Außerdem findet über verschiedene Systeme eine Lokalisierung des Roboters relativ zu einem globalen Koordinatensystem statt. Das erlaubt schlussendlich eine Koordinatentransformation der Sensordaten vom Sensor über den Roboter in ein gemeinsames globales Koordinatensystem.

- *Synchronisierungslevel*
  Beide zuvor genannten Bezüge können sowohl auf lokaler als auch auf übergeordneter und globaler Ebene Anwendung finden. Die einzelnen Sensoren werden in der Regel bereits direkt lokal auf dem Fahrzeug synchronisiert. Nachfolgend müssen aber auch die Fahrzeuge untereinander und mit dem Server synchronisiert werden.

3 Simulation von Sensordaten

Im Projektantrag war ursprünglich vorgesehen, aufgezeichnete Sensordaten aus produktiven Systemen einzusetzen. Nach Rücksprache mit verschiedenen Entwicklern und Anwendern mobiler Roboter wurde dieser Ansatz aus mehreren Gründen verworfen. Derlei Daten werden zur Weiterverwendung aktuell nicht im notwendigen Stil gespeichert, da Anwender mobiler Robotik aus Gründen des Knowhow-Schutzes keine Daten zur Einsatzumgebung herausgeben wollen. Außerdem kollidiert in der Regel der Einsatz von Kameras mit den Anforderungen an den Schutz der Daten der Mitarbeitenden, da Gesichter direkt aufgenommen und gespeichert würden. Eine weitere Herausforderung ergibt sich aus der fehlenden Flexibilität real aufgenommener Daten. Sollen bestimmte Szenarien nachgestellt werden, müssten dazu aufwendig verschiedene Objekte in einer Szene arrangiert und Sensoren präzise platziert und kalibriert werden. Das ist zeit- und kostenintensiv und insgesamt nicht praxistauglich. Mithilfe durch Simulation erzeugter Sensordaten kann diesen Herausforderungen in geeigneter Weise begegnet werden (s. Abschnitt 2.1.1). Aus diesen Gründen wurde die Simulation von Sensordaten zusätzlich in Arbeitspaket drei aufgenommen.

Im folgenden Abschnitt werden die Schritte erläutert, die dazu für dieses Projekt notwendig waren. Nach einer Anforderungsermittlung mittels Expertengespräche wurde erst eine geeignete Simulationssoftware ausgewählt, anschließend wurden die einzelnen Module der Simulation entwickelt und zuletzt ein strukturiertes Datenset für die weiteren Entwicklungsschritte generiert.

3.1.1 Anforderungen and die Simulation

Bevor mit der Entwicklung der Simulation begonnen werden konnte, musste das Ziel genau festgelegt und die Randbedingungen definiert werden. Nur so konnten die generierten Daten später erfolgreich in den nachfolgenden Entwicklungsschritten eingesetzt werden.

Zieldefinition

Für die Entwicklung des kollektiven Umgebungsinformationssystems waren Sensordaten mehrerer simultan betriebener Roboter erforderlich. Darüber hinaus war insbesondere für die Entwicklung der Deep Learning Algorithmen die Erzeugung von Ground-Truth Daten erforderlich. Das Ziel der Simulation ließ sich daher wie folgt festlegen:

Ziel der Simulation ist die Erzeugung eines synthetischen Datensets mit Umgebungssensordaten einer Flotte mobiler Roboter und zugehöriger Ground-Truth Daten zur Entwicklung von Systemen zur Multi-Roboter-Sensordatenfusion.

Anforderungsermittlung durch Expertengespräche

Mithilfe semistrukturierter Experteninterviews wurden erste Informationen zum typischen Aufbau von Industrieumgebungen und auch zum realen Verhalten von Sensorik ermittelt. Dazu wurde ein Fragebogen vorab an acht Interviewpartner verschickt. Als solche wurden sowohl Entwickler als auch Anwender mobiler Roboter ausgewählt, um beide Sichtweisen abbilden zu können. Sie waren zum Zeitpunkt der Interviews bei den Unternehmen Jungheinrich AG, AGILOX Services GmbH, Safelog GmbH, Brose Fahrzeugteile SE & Co. KG, Noyes Robotics GmbH und robominds GmbH überwiegend in den Bereichen Sales, Strategie und Entwicklung angestellt.

Der Fragebogen (s. Anhang A.1) gliedert sich in drei Bereiche:

- Aufbau und Lagerhalle in der Simulationsumgebung
- Objekte in der Lagerhalle
- Sensorik und Entwicklung mobiler Roboter

Die Antworten der Experten können wie folgt zusammengefasst werden:

Aufbau und Lagerhalle in der Simulationsumgebung

Industrieumgebungen sind häufig größere Hallen, die über offene Durchgänge, ansteuerbare Schnelllauf- oder Brandschutztore oder Türen für Personenverkehr miteinander verbunden sind. Dies ergibt sich oft aus der baulichen Historie der Gebäude. Insbesondere neue Gebäude können auch als ein großer durchgehender Raum ausgeführt sein.

Dabei wird die Gesamtfläche in unterschiedliche Zonen untergliedert, die in der Regel z. B. über Linienmarkierungen auf dem Boden voneinander getrennt werden. Es können die Zonen Produktionsbereich, Regallagerbereich, Blocklagerbereich und Übergabebereich von Wareneingang zur Lagerung unterschieden werden.

Zur Ausleuchtung besitzen die Gebäude häufig Wand- und Dachfenster. Allerdings wird die Beleuchtung durch zusätzlich angebrachtes künstliches Licht dominiert, um Arbeitsschutzvorschriften auch bei Nacht einhalten zu können. Dennoch kann die Änderung der natürlichen Beleuchtung im Verlauf des Tag-Nacht-Rhythmus oder verschiedener Wetterlagen Einfluss auf technische Prozesse haben, was bei visuellen Verfahren (z. B. Pick-and-place mit Kamera) zu Problemen führen kann.

Weitere fest installierte Objekte, die in Industrieumgebungen häufig anzutreffen sind, sind Säulen, Regale, Geländer, Anfahrschutzeinrichtungen und Leitplanken, Produktions- und Fördertechnik und Plateaus, auf denen Anlagen stehen können.

Die Anwendung mobiler Roboter findet üblicherweise rein in Innenräumen statt. In Sonderfällen können Hallenwechsel über Außenbereiche durchgeführt werden. Dazu sollten diese allerdings überdacht sein.

Auch eine Nutzung über mehrere Stockwerke ist möglich. Hierzu werden Aufzüge, die die Roboter zum Stockwerkwechsel nutzen können, oder reine Materiallifte eingesetzt, die von den Robotern automatisiert be- und entladen werden. In seltenen Fällen werden Rampen mit sehr geringer Steigung eingesetzt, die in aller Regel eine Anpassung der Sicherheitsfelder erfordern. Üblich ist jedoch der Betrieb einer Flotte auf einem Stockwerk.

Objekte in der Lagerhalle

Von mobilen Robotern werden überwiegend Paletten und Gitterboxen (häufig auf Rolluntersetzern) transportiert. Auch Kleinladungsträger (KLT) und Kartons werden transportiert, diese jedoch häufig auf Paletten, was das Aufnehmen und Absetzen erleichtert und den Transport größerer Mengen ermöglicht. In selteneren Fällen werden auch Großladungsträger (GLT) oder spezielle Regale oder Gestelle (z. B. für Lackierarbeiten) transportiert.

Die Erkennung dynamischer Objekte wurde als besonders wichtig herausgestellt. Dazu gehören Menschen, Gabelstapler, Routenzüge, andere mobile Roboter und auch kleinere Transportwagen. Menschen können sich zudem auf Rollern oder Fahrrädern bewegen.

Daneben sind kleinere statische Hindernisse anzutreffen, wie kleinere Schränke und Regale, Feuerlöscher, Mülleimer / -container, Büroeinrichtung (Tische, Stühle PCs), Baustellenzäune oder Stellwände, Pylonen, herumstehende Kisten/Ladungsträger oder Verpackungsmaterial.

Sensorik und Entwicklung mobiler Roboter

Die größten Herausforderungen bei der Entwicklung mobiler Roboter für die Intralogistik bestehen im Zusammenspiel mit anderen Teilnehmern. Das betrifft die Interaktion mit dynamischen Objekten, v. A. mit Menschen, aber auch mit anderen Robotern. Auch der Mischverkehr mit manuellen Verkehrsteilnehmern und große Verkehrsknotenpunkte mit Querverkehr stellen mobile Roboter vor Herausforderungen. Die Interaktion mit statischen Objekten wird insbesondere dann herausfordernd, wenn dadurch Bewegungen auf engem Raum notwendig werden. Wegen der freizuhaltenden Sicherheitsfelder sind dann häufig ungewollte Stopps der Fahrzeuge die Folge. Solche Situationen müssen daher mit den eingesetzten Sensoren sicher erfasst werden können.

Neben der Interaktion mit anderen Verkehrsteilnehmern sind auch anspruchsvolle Umgebungsbedingungen relevant für die Wahl und Auslegung von Sensoren. Limitierungen, wie z. B. die Anfälligkeit für Dunkelheit oder Blendung durch direkt einfallendes Licht, müssen daher bei der Auslegung berücksichtigt werden. Für den Betrieb ist außerdem sorgfältige Kalibrierung und Synchronisierung notwendig.

Eine bessere Umgebungswahrnehmung wird jedoch von den Experten als Schritt zur Lösung der genannten Herausforderungen gesehen. Dazu muss eine sichere Unterscheidung zwischen statischen und dynamischen Objekten möglich sein. Informationen über Geometrie, Klasse, Ort und Pose und darüber hinaus ggf. eine eindeutige Identifikation über einen Marker müssen dafür verfügbar sein. Außerdem sind aktuelle und historische Bewegungsdaten für die Bewegungsprädiktion dynamischer Objekte wichtig.

Die Verbesserung der Umgebungswahrnehmung könnte möglicherweise durch besser geeignete Umgebungsdaten vorangetrieben werden. Daher wurden die Experten danach gefragt, ob sich perfekte (im Gespräch beschrieben als frei von Fehlern) Umgebungsdaten hypothetisch für eine robustere Wahrnehmung und bessere Lokalisierung nutzen ließen. Diese Frage wurde bejaht. Außerdem wurde genannt, dass sich Roboter in der Folge womöglich besser selbst organisieren und Aufgaben autonom übernehmen könnten, was ebenfalls ein dynamisches Management der Flotte ermöglichen könnte.

Eine darauf aufbauende hypothetische ständige und fehlerfreie Detektion von Objekten im Umfeld des Roboters wurde als sehr wichtig eingeschätzt. Dennoch wäre sie allein nicht ausreichend, um alle offenen Herausforderungen beim Betrieb mobiler Roboter zu lösen.

Im Hinblick auf die Entwicklung der Simulation wurde darüber hinaus danach gefragt, ob Interferenzen unter Sensoren auftreten, um den Effekt gegebenenfalls nachzubilden. Aufgrund technischer Maßnahmen zu deren Mitigierung, sind solche Effekte nach Erfahrung der Experten nicht ausgeschlossen, treten aber sehr selten auf.

Ableitung von Anforderungen an die Simulation aus den Expertengesprächen

Aus den genannten Informationen lassen sich Anforderungen an die Simulation ableiten.

- Um den Standardfall abzubilden, sollte mindestens eine Industriehalle auf einem Stockwerk abgebildet werden, in deren Wände Türen und Tore als Durchgänge zu anderen Hallen vorgesehen sind.
- In der Halle sollten klar ersichtliche Zonen definiert sein: Produktionsbereich, Regallagerbereich, Blocklagerbereich und Übergabebereich von Wareneingang zur Lagerung.

- Um einen realistischen Tag-Nacht-Rhythmus einzuführen und Abweichungen in der Wahrnehmung der Sensoren zu simulieren, sollten Fenster in der Hallenfassade vorgesehen sein. Künstliche Beleuchtung sollte jedoch die Ausleuchtung dominieren.
- Um eine realistische und lebhafte Szenerie zu erreichen, sollten verschiedene Objekte implementiert werden:
    - Fest installierte Halleneinrichtung, wie z. B. Säulen, Regale, Anfahrschutzeinrichtungen und Leitplanken oder Produktions- und Fördertechnik
    - Statische Hindernisse, wie z. B. kleinere Schränke und Regale, Feuerlöscher, Mülleimer, Büroeinrichtung, Baustellenzäune oder Stellwände, Pylonen oder herumstehende Kisten/Ladungsträger oder Verpackungsmaterial
    - Verschiedene Ladungsträger, wie z. B. Paletten, Gitterboxen, KLT, GLT, Kartons
    - Dynamische Objekte, wie z.B. Menschen, Gabelstapler, Routenzüge, andere mobile Roboter, kleinere Transportwagen
- Außerdem sollten verschiedene herausfordernde Szenarien implementiert werden, um gezielte Tests zu ermöglichen:
    - Begegnung mit Menschen
    - Kreuzungssituationen mit Querverkehr, insbesondere Gabelstaplern
- Zukünftig sollen Sensordaten für erweiterte Objekterkennung eingesetzt werden können. Daher muss es die Simulation ermöglichen, Informationen über Geometrie, Klasse, Ort und Pose von Objekten abzuleiten.

Anforderungen aus dem Projektziel

Auch das Projektziel, ein kollektives Umgebungsinformationssystem zu entwickeln, impliziert verschiedene Anforderungen.

- Um die Übertragbarkeit auf Realdaten zu erleichtern, sollten die Sensordaten möglichst realitätsnah simuliert werden können.
- Der Ansatz soll mit aktueller Hardware funktionieren. Das bedeutet, dass aktuell verbaute Sensormodalitäten (2D-LiDAR, Tiefenkamera) simuliert werden sollte, um eine spätere Übertragbarkeit zu gewährleisten. Hier geht es nur um einen Proof of Concept. Für nachfolgende Entwicklungen sollten konkrete Eigenschaften von Sensoren in den Daten abgebildet sein.
- Aber auch zukünftige Sensoren (3D-LiDAR, 2D-RGB-Kamera) sollten mitbetrachtet werden, um aktuelle Trends bei der Entwicklung des Systems berücksichtigen zu können.
- Die Entwicklung eines Systems für mehrere Roboter erfordert die simultane Simulation mehrerer Roboter.
- Die simulierten Daten sollten in gut strukturierter, anwendungsnaher Form und einfach nutzbar vorliegen, um die Weiterverwendung durch Unternehmen und auch die Anwendung auf andere Problemstellungen zu ermöglichen.

Da die Sensordaten nur für die Entwicklung eines Umgebungsmodells eingesetzt werden sollen und abgeleitete Aktionen der Roboter nicht Gegenstand der Betrachtung sind, ist die Entwicklung einer bidirektionalen Schnittstelle (Steuersignale werden an Roboter in der Simulation versendet) nicht erforderlich. Daher ist auch die Erzeugung der Daten in Echtzeit nicht notwendig.

3.1.2 Auswahl der Simulationssoftware

Anforderungen an die Simulationssoftware

Aus den Anforderungen an die Simulation an sich ergeben sich unmittelbar Anforderungen an die eingesetzte Software:

- Realitätsgetreue Sensordaten insbesondere mit 2D-RGB-Kameras erfordern eine gute Visualisierung der simulierten Szenerie.
- Der Import vieler unterschiedlicher Objekte erfordert einfachen Datenimport vor allem für 3D-Modelle. Außerdem lässt sich der Schritt durch vorimplementierte Bausteine stark vereinfachen.
- Die Implementierung verschiedener Sensoren muss möglich sein, sofern vom Hersteller der Software nicht vorgesehen.
- Insbesondere das Verhalten der Sensoren muss außerdem physikalischen Gesetzmäßigkeiten folgen.
- Die Nutzung der simulierten Sensordaten erfordert eine Schnittstelle aus der Simulation heraus oder die Möglichkeit, diese selbst zu implementieren.

Weiter formalisiert lassen sich daraus fünf wesentliche Kriterien ableiten, nach denen Simulationssoftware verglichen werden kann:

Datenimport

Da die Erstellung von simulationsgeeigneten Modellen sehr zeitaufwendig und damit teuer ist, ist es häufig einfacher, vorhandene Modelle zu nutzen. Das können entweder bereits in anderen Programmen selbst erstelle Modelle oder von Marktplätzen erworbene Modelle sein. Die Nutzung solcher Modelle kann die Erstellung der Simulation erheblich beschleunigen. Voraussetzung dafür ist jedoch eine breite Kompatibilität mit verschiedenen Austauschformaten.

Implementierungsaufwand

Einige Softwareumgebungen enthalten standardmäßig bereits verschiedene vorimplementierte Modelle und Funktionen. Gerade in Bezug auf die realistische Abbildung von Sensoren und Sensormodellen können diese vorimplementierten Module den Implementierungsaufwand erheblich senken.

Visualisierung

Die Nutzung optischer Sensoren erfordert ein hohes Maß an visueller Güte der Simulation. Sollen multimodale Sensor-Setups vor allem mit Kameras simuliert werden, müssen die gerenderten Bilder möglichst realitätsgetreu sein, um eine gute Übertragbarkeit auf die reale Anwendung zu ermöglichen.

Physikinteraktion

Ebenso wie die optische Güte ist auch die Güte der physikalischen Interaktion in zweierlei Hinsicht von großer Bedeutung. Einerseits ist die optische Qualität auch von physikalischen Interaktionen abhängig (z. B. Ray-Tracing), andererseits ist die direkte Interaktion von Objekten untereinander für manche Anwendungen wichtig. So kann beispielsweise das Entnehmen einer Palette aus dem Regal simuliert werden, um die bei dieser Aktion anfallenden Umgebungsdaten möglichst realitätsgetreu aufzunehmen.

Datenexport

Am Ende der Simulation müssen die Daten weiter genutzt werden können. Auch wenn manchmal das erwünschte Ergebnis bereits in der Simulationsumgebung angezeigt werden kann, so ist es gerade bei der Simulation von Umgebungsdaten notwendig, diese aus der Simulation in geeigneter Weise exportieren zu können. Nur dann können sie in nachfolgenden Schritten für die Entwicklung von Algorithmen genutzt werden.

Auswahl an Simulationssoftwarelösungen

Um die für das Projekt geeignetste Softwarelösung auszuwählen, wurden verschiedene Simulationsumgebungen verglichen. Dazu wurden als mögliche Softwarelösungen die Programme aus der Veröffentlichung [19] von Collins et al. zugrunde gelegt.

Anschließend wurden MuJoCo, Pybullet, RaiSim, Nvidia Flex und Project Chrono von der Liste entfernt, da es sich lediglich um reine Physik-Engines handelt. Außerdem wurden CARLA, Sim-Grasp, Sofa, UWSim, Flightmare, UUV Simulator, Stonefish, URSim, USVSim, Chai3D, AMBF, jMavSimUnityFlexML und VoxCAD ausgeschlossen, da diese auf andere Szenerien spezialisiert sind und keine Simulation mobiler Roboter erlauben.

Die verbleibenden Programme wurden anhand der Anzahl ihrer Nennung in verschiedenen Veröffentlichungen in SCOPUS von 2019 bis 2024 auf ihre Verbreitung untersucht. Dazu wurde in den Feldern Titel, Abstract und Keywords jeweils nach dem jeweiligen Software-Namen in Kombination mit dem String „mobile robots“ gesucht. In Tabelle 1 sind die Ergebnisse zusammengefasst:

Tabelle 3: Simulationssoftware absteigend sortiert nach Anzahl der Suchtreffer in SCOPUS

Software	**Anzahl Nennung**
Gazebo	482
Unity	84
Webots	62
CoppeliaSim	60
Blender	9
Unreal Engine 4	7
Isaac Sim	6
Airsim	3

Nachfolgend wurden die Programme anhand der zuvor entwickelten Kriterien bewertet. Die Tabelle in Anhang A.2. zeigt eine Zusammenfassung der Bewertung.

Auswahl der geeigneten Lösung

Ausgehend von der Bewertung wurde Unity als geeignete Simulationsumgebung ausgewählt. Unity erlaubt die Erzeugung realitätsnaher Grafik inklusive Ray-Tracing und bringt durch das Unity Perception Package und den Unity Robotics Hub bereits einige vorimplementierte Robotik-Funktionen mit. Fehlende Funktionen können wegen der Systemoffenheit nach eigenen Anforderungen entwickelt werden. Die dafür primär verwendete Programmiersprache C# gilt als verhältnismäßig einfach zu erlernen. Physikalische Interaktionen können durch Nvidia PhysX berechnet werden. Unity kann mit einer großen Bandbreite an Import-Formaten umgehen und Daten entweder live über eine vorentwickelte ROS-Schnittstelle bzw. eigenentwickelte Schnittstellen oder als Datenset im SOLO-Schema ausgeben. Dennoch ist ein hohes Maß an Eigenentwicklung insbesondere bei nicht vorimplementierten Sensoren (z. B. LiDAR) und Modellen notwendig. Die Entwicklung der angesprochenen Packages wurde zudem nach aktuellem Stand eingestellt, was in Zukunft zusätzlichen Implementierungsaufwand nach sich ziehen kann. Außerdem sollte bei der Entwicklung immer berücksichtigt werden, dass diese Art Simulation eine Zweckentfremdung des Produkts darstellt. Unity wurde für die Entwicklung von Videospielen entwickelt und priorisiert demnach standardmäßig zu Teilen andere Funktionen (Darstellung vs. physikalische Korrektheit).

Die Kombination aus Flexibilität und mittlerem Implementierungsaufwand aufgrund der vorgefertigten Funktionen und vielseitigen Importfunktionen führten zu der Entscheidung, Unity in diesem Projekt als Simulationssoftware einzusetzen.

3.1.3 Entwicklung der Module der Simulation

Um die Komplexität der gesamten Simulation zu reduzieren, wurde die Erstellung in drei Schritte unterteilt. Im ersten Schritt wurde die Szenerie erarbeitet. Sie setzt sich aus verschiedenen 3D-Modellen zusammen, die im Zusammenspiel eine belebte Industrieszenerie widerspiegeln. Im nächsten Schritt wurden Roboter entwickelt, die sich in dieser bewegen und mit verschiedenen Sensoren ihre Umgebung wahrnehmen können. Zuletzt wurde eine Schnittstelle zum Export der Sensordaten basierend auf dem Unity SOLO-Schema entwickelt.

Szenerie

Die Szenerie beschreibt die dargestellte Umgebung, also welche Objekte dargestellt werden, wie sie angeordnet sind und ob bzw. wie sie interagieren.

Strukturierung und Layout der Szenerie

Das Layout der Szenerie basiert auf den auf S. 30 festgelegten Zonen Produktion, Regallager, Blocklager und Übergabe von Wareneingang zur Lagerung. In Abbildung 12 ist das Gesamtlayout mit markierten Zonen dargestellt.

Abbildung 12: Hallenlayout aufgebaut aus den Zonen: Produktion, Regallager, Blocklager und Übergabe von Wareneingang zur Lagerung

Hinter diesem Layout steht ein fiktiver Referenzprozess, der eine realitätsnahe Umgebung ermöglichen soll. In der Umsetzung werden allerdings keine Warentransporte simuliert. Der Prozess ist wie folgt: Von der linken Seite erfolgt die Anlieferung der Waren. Diese wird zunächst in den Blocklagern gepuffert und von dort direkt in die Palettenregale eingelagert. Von dort gelangen die Waren durch die Auslagerung in den Produktionsbereich und anschließend zur Verpackung rechts oben. Vor der Abholung werden die fertigen Produkte im rechten Blocklager zwischengelagert. Die Halle ist durch vier Tore mit angrenzenden Hallen verbunden (siehe blaue Tore und Bahnmarkierungen auf der linken Seite). Durch diese Anordnung ergeben sich diverse

Kreuzungen. Vor allem zwischen der Anlieferzone und dem Lager gibt es durch die engen Gassen der Palettenregale beschränkte Sicht auf den Querverkehr. Dies ermöglichen die Aufnahme realitätsnaher oft kritischer Situationen.

Bestandteile der Szenerie

Die einzelnen Modelle, die in der Szenerie eingesetzt werden, können in drei Gruppen unterteilt werden: Gebäude, statische Objekte und dynamische Objekte. Zum Gebäude gehören alle Objekte, die die äußere Hülle der Umgebung bilden (Wand, Boden, Decke), aber auch Säulen und Betriebseinrichtung, wie z. B. die Lüftungsanlage. Statisch Objekte ändern ihren Ort nicht von allein und sind nicht untrennbar mit der Bausubstanz verbunden. Hierzu zählen Regale, Ladungsträger oder Produktionsanlagen. Dynamische Objekte hingegen können ihren Ort von selbst (bzw. durch Steuerung eines Menschen) ändern. Daher fallen in diese Kategorie Menschen, Flurförderzeuge und auch Roboter. In Anhang A.3 sind Objekte der verschiedenen Kategorien genannt und abgebildet. Durch die Mischung der Objekte in Kombination mit dem Layout ergibt sich ein realitätsnaher Aufbau der Szenerie. Somit ist auch die Häufigkeitsverteilung der Objekte in den Daten nah an der Realität.

Relevante Situationen

Situationen bilden Interaktionen von Objekten im Rahmen der Szenerie ab. Für die Ausarbeitung von Situationen ist vor allem die geplante Anwendung bzw. der zu entwickelnde Algorithmus relevant. Dieser soll meist auf bestimmte Situationen abgeprüft werden. Ein vorab erstellter Test-Plan kann daher die Erstellung strukturieren. Außerdem sollte darauf geachtet werden, dass alle Situationen mithilfe der Szenerie dargestellt werden können.

Für die Funktionsprüfung des Umgebungsinformationssystems sind besonders folgende Situationen relevant (s. auch Abschnitt 3.1.1):

- Enge Gänge zwischen den Regalen, die keinen Einblick auf die nachfolgende Kreuzung ermöglichen.
- Bereiche, die nur von einem Roboter eingesehen werden und die anderen Robotern auf ihrer Fahrtroute unbekannt bleiben.
- Verteilte dynamische Objekte, die meist nur von einem Roboter wahrgenommen werden.
- Mobile Roboter, die sich manchmal sehr nah aneinander vorbeibewegen und manchmal weit voneinander entfernt operieren.

Die ersten drei Situationen erfordern den Austausch von Informationen über Hallenbereiche und Objekte, die von einem einzelnen Roboter allein nicht erlangt werden können. Durch die gemeinsame Pflege von Informationen zur Umgebung können die Roboter hier einen Vorteil erlangen. Die vierte Situation ist zur Prüfung des Umgebungsmodells bei unterschiedlich dichter Informationslage. Vor allem bei weit voneinander entfernten Robotern kann so untersucht werden, ob die Dichte an Robotern für die Erstellung eines akkuraten Umgebungsmodells noch ausreicht.

Die Simulation wurde so entwickelt, dass alle beschriebenen Fälle im Laufe der Simulationszeit auftreten.

Robotermodell zur Simulation von Sensordaten

Um Daten aus den richtigen Sensorperspektiven aufnehmen zu können, wurde ein Robotermodell entwickelt, das sich in mehrfacher Ausführung simultan in der Halle bewegen kann. Ziel war, ein möglichst generisches Modell zu erzeugen, das in vielen Anwendungen eingesetzt werden kann. Nachfolgend wird nach der Ermittlung der Anforderungen auf den allgemeinen Aufbau, die Sensorik und die gefahrenen Pfade des Robotermodells eingegangen.

Anforderungsanalyse

Das Konzept für den Aufbau des Robotermodells und die Anordnung der Komponenten wurde basierend auf einer Marktrecherche unter 23 Unternehmen aus Asien, Europa und Nordamerika, die mobile Roboter für die Logistikbranche herstellen, entwickelt. Dazu wurden verschiedene Robotermodelle der Hersteller anhand der Kriterien Typ, Dimension / Gewicht, Antrieb und Sensorik ausgesucht und bewertet.

Typ

Wie in Abschnitt 2.1.4 beschrieben, ist in der Industrie und bei Herstellern der sogenannte Unterfahr-Roboter häufig anzutreffen und wurde daher als Vorlage für den simulierten Roboter gewählt. Dieser Typ kann verschiedene Lasten unterfahren und anschließend anheben. Das direkte Anheben von Paletten vom Boden ist in der Regel nicht möglich, weshalb spezielle Übergabestationen eingesetzt werden. Die Traglast beträgt meist etwa eine Tonne. Unter den Herstellern wurden insgesamt 34 Modelle identifiziert, die die genannten Anforderungen erfüllen.

Daneben wären auch andere Bauformen, wie beispielsweise der Gabelhub-Typ, denkbar. Dadurch änderten sich im Wesentlichen nur die Verbaupositionen der Sensoren, weshalb sich auf den hier vorliegenden Anwendungsfall (Proof of Concept des Gesamtsystems) keine erheblichen Auswirkungen ergäben. Der Einfachheit halber wurde daher auf die Untersuchung anderer Bauformen verzichtet.

Dimensionen

Für die Erstellung des Roboter-Modells sind vor allem die äußeren Abmaße relevant. Unterfahr-Roboter sind meist quaderförmig. Zu den 34 identifizierten Robotern liegen in 31 Fällen Daten zu den Außenabmaßen vor. Abbildung 13 zeigt eine Kastengrafik, die diese Daten in Relation setzt.

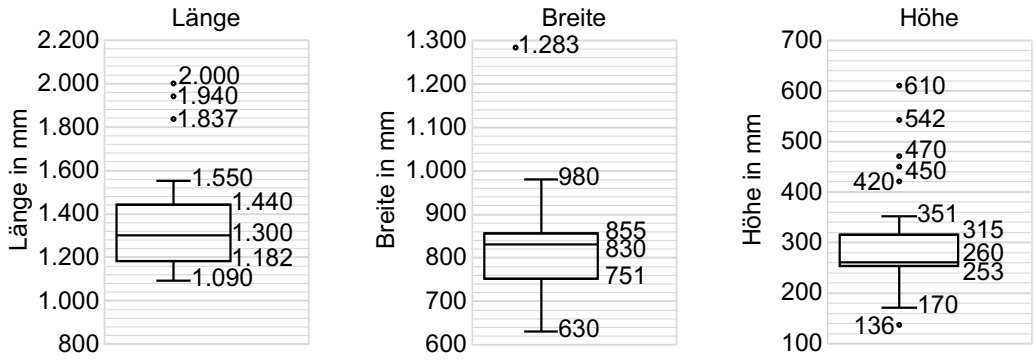


Abbildung 13: Außenmaße der untersuchten Roboter (n = 31)

Man kann erkennen, dass Fahrzeuge im Median die Maße 1300 mm [L] x 830 mm [B] x 260 [H] aufweisen. Da die Daten jedoch von Fahrzeugen stammen, die in ihrem Aspektverhältnis in Abhängigkeit des zu transportierenden Ladungsträgers stark unterschiedlich gestaltet sind, muss bei der Wahl der Außen-Abmaße darauf geachtet werden, dass die Maße insgesamt realitätsnah und auf den Ladungsträger abgestimmt sind.

Antrieb

Alle untersuchten Roboter weisen einen Differentialantrieb, der aus zwei angetriebenen Rädern an der Heckseite, sowie ein oder zwei frei beweglichen Rädern an der Vorderseite besteht. Bei wenigen Robotern sind die angetriebenen Räder auch mittig und die passiven Räder vorne und hinten angeordnet. Der

Differentialantrieb beeinflusst die Manövrierfähigkeit des Fahrzeugs, ist für die Aufzeichnung von Umgebungsdaten jedoch von untergeordneter Relevanz.

Sensorik

Zu 29 der untersuchten Roboter lagen Daten zum Sensor-Setup vor (s. Abbildung 14). In jedem Roboter werden ein oder mehr LiDAR-Sensoren verbaut, da diese eine zuverlässige Lösung für Personenschutz darstellen (vgl.Abschnitt 2.1.5). Auch taktile Sensoren an den Außenseiten finden immerhin bei 10 der 29 Modelle Anwendung. Lediglich bei 4 von 29 Modellen sind Tiefen-Kameras verbaut. 2D-RGB-Kameras wurden nicht verbaut.

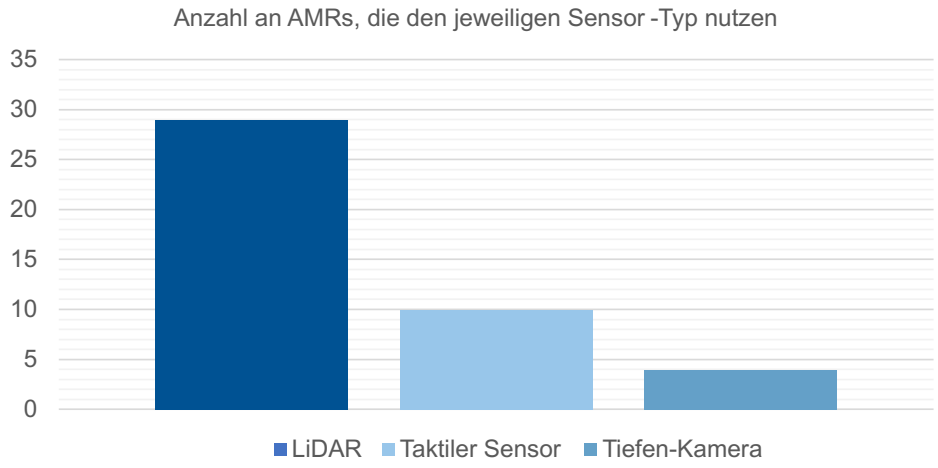


Abbildung 14: Häufigkeit der an Robotern verbauten Sensorik (n=29)

Gespräche mit Experten ergaben darüber hinaus, dass oft zwei LiDAR-Sensoren an den Ecken des Roboters über die Diagonale platziert werden. Auf diese Weise können 360° FOV bei geringer Überschneidung des FOV der Einzelsensoren erzeugt werden. Die Eigenschaften der simulierten Sensoren werden im Abschnitt 3.1.4 angegeben.

Aufbau des Robotermodells

Basierend auf der Marktrecherche wurde im Anschluss ein Robotermodell für die Simulation entworfen. Ziel war, dieses möglichst generisch zu gestalten, sodass eine breite Anwendung der erzeugten Daten möglich ist. Die Außenmaße wurden zu 1540 mm [L] x 710 mm [B] x 310 [H] gewählt, um einerseits möglichst innerhalb der gängigsten Maße zu liegen und andererseits ein vielseitiges und sinnvolles Gesamtkonzept zu erstellen. Damit können Paletten der Länge nach mit Überstand unterfahren werden, um die Erreichbarkeit des Not-Aus-Systems zu gewährleisten. In der Breite sorgen jeweils 45 mm Überstand der Palette für sicheren Stand der Palette auf dem Roboter, ohne die Manövrierbarkeit in verengten Gegebenheiten einzuschränken. Die Höhe von 310 mm erlaubt den einfachen Einbau von LiDAR-Sensoren. Mit diesen Maßen ist neben Paletten auch der Transport von kleineren Ladungsträgern und Trolleys möglich.

Als Antriebskonfiguration wurde ein Differentialantrieb modelliert. Hier ist anzumerken, dass es sich lediglich um eine Entscheidung für die optische Darstellung handelt. Fahrphysik wurde nicht simuliert.

Für den simulierten Roboter wurden zwei LiDAR-Sensoren, platziert an den Ecken über die Diagonale, und eine Kamera als umgebungserfassende Sensorik implementiert. Die Kamera kann sowohl als Tiefen-Kamera als auch als 2D-RGB-Kamera eingesetzt werden. Da keine Interaktion der Umgebung mit dem Roboter

vorgesehen war, wurden taktile Sensoren nicht implementiert. Zusätzlich dazu gibt es eine Lokalisierung relativ zum globalen Koordinatensystem der Simulation.

Das Modell des mobilen Roboters ist in Abbildung 15 mit seinen Außenmaßen dargestellt.

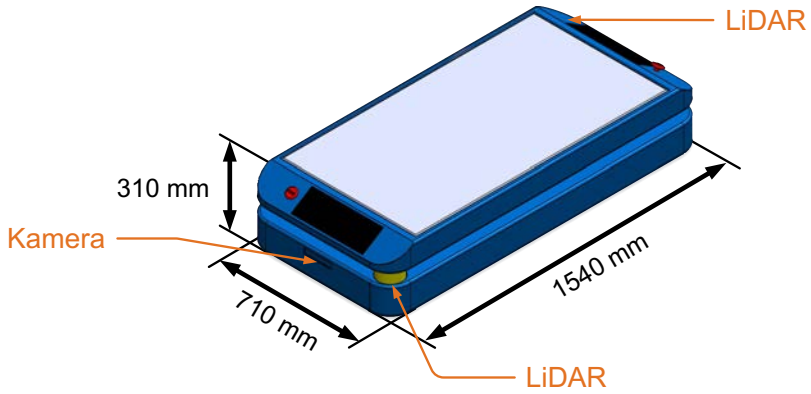


Abbildung 15: Mobiler Roboter mit Außenmaßen

Weitere Maße können in Anhang A.4 gefunden werden.

Bewegung

Um Daten aufzunehmen, bewegen sich die simulierten Roboter auf festen Pfaden durch die Industrieumgebung (s. Abbildung 16). Jeder Roboter verfolgt seinen eigenen Pfad und trifft dabei auf andere Verkehrsteilnehmer in unterschiedlichen Situationen. Dadurch wird eine gewisse Streuung in den Daten erzeugt. Da sich die Pfade nach dem Durchfahren einer vollen Runde allerdings wiederholen, muss darauf geachtet werden, die Simulationslänge in Abhängigkeit der Pfadlänge und Fahrgeschwindigkeit der Roboter zu wählen. Andernfalls ergeben sich zu große Ähnlichkeiten in den Daten und die Gefahr von Overfitting beim Training von Algorithmen mit den Daten steigt.

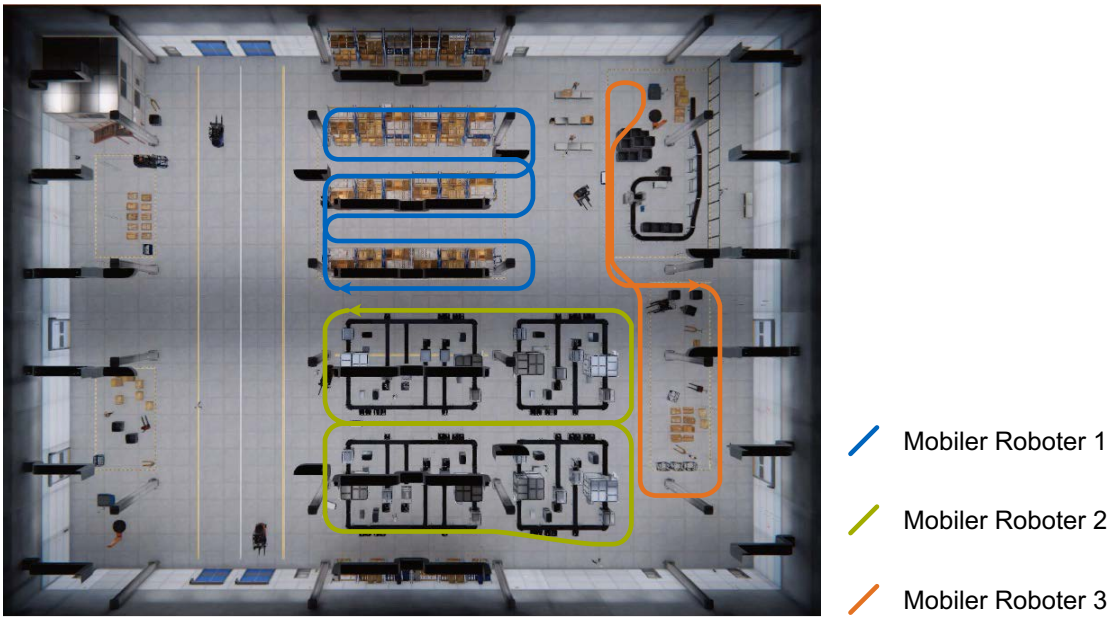


Abbildung 16: Fahrwege (blau, grün, orange) von drei simulierten mobilen Robotern in der Industrieumgebung

3.1.4 Datengenerierung

Nach den beschriebenen Schritten können sich die mobilen Roboter bereits durch die Simulationsumgebung bewegen und diese mit verschiedenen Sensoren wahrnehmen.

Grundsätzlich ist eine direkter Live-Stream von Daten aus der Simulation (z. B. über eine ROS-Schnittstelle) möglich. Dabei ist zu berücksichtigen, dass die Simulation in Echtzeit laufen muss, da sonst nachgelagerte Algorithmen, die auf korrekte Zeitwerte angewiesen sind, u. U. nicht korrekt funktionieren. Da insbesondere die simultane Simulation mehrerer Roboter mit jeweils mehreren Sensoren einen enormen Rechenaufwand darstellt, war diese Möglichkeit nicht umsetzbar. Stattdessen wurden die Daten in Form eines Datensets asynchron abgespeichert. Damit die Sensordaten für die Entwicklung von Algorithmen genutzt werden können, müssen sie in einem sinnvollen Format abgespeichert und zur Erstellung der Ground-Truth gelabelt werden. In den nachfolgenden Abschnitten werden die dazu notwendigen Schritte beschrieben. Außerdem werden verschieden Möglichkeiten zur Optimierung des Datensets diskutiert und zuletzt einige Eckdaten angegeben.

Sensorkonfiguration

Das Robotermodell wurde mit insgesamt fünf Sensoren ausgestattet:

Tabelle 4: Positionierung und Eigenschaften der simulierten Sensorik

LiDAR vorne		*LiDAR hinten*	
Position:	(-275 mm; 690 mm; 140 mm)	Position:	(275 mm; -690 mm; 140 mm)
Rotation:	(0°; 0°; 90°)	Rotation:	(0°; 0°; -90°)
Wiederholrate:	30 Hz	Wiederholrate:	30 Hz
Reichweite:	[0 m; 100 m]	Reichweite:	[0 m; 100 m]
Scanbereich:	[-180°; 90°]	Scanbereich:	[-180°; 90°]
Anzahl Messungen pro Scan:	270	Anzahl Messungen pro Scan:	270

2D-RGB-Kamera		*Tiefen-Kamera*	
Position:	(0 mm; 770 mm; 120 mm)	Position:	(0 mm; 770 mm; 120 mm)
Rotation:	(10°; 0°; 0°)	Rotation:	(10°; 0°; 0°)
Wiederholrate:	10 Hz	Wiederholrate:	10 Hz

Lokalisierung	
Position:	(0 mm; 0 mm; 0 mm)
Rotation:	(0°; 0°; 0°)
Wiederholrate:	30 Hz

Position und Rotation sind gegenüber dem Roboterursprung im Rechtssystem (x, y, z) angegeben. Der Roboterursprung liegt in der Mitte des Fahrzeugs auf Bodenhöhe. In der Zeichnung in Anhang A.5 sind die Positionierungen der Sensoren relativ zum Ursprung eingezeichnet.

Labeling

Wie in Abschnitt 2.3.2 beschrieben sind insbesondere für das Training von Supervised Deep Learning Algorithmen große Menge gelabelter Daten notwendig. Einer der großen Vorteile simulierter Daten besteht darin, dass die generierten Sensordaten automatisiert gelabelt werden können (vgl. Abschnitt 2.1.1).

Es wurden verschiedene Labels implementiert:

- Bounding-Boxen inkl. Occlusion-Filter für Kamerabilder
- Semantische Segmentierung für Kamerabilder
- Punktweise Klassifizierung der LiDAR-Punktwolke

Die Klassen ergeben sich einerseits aus einer Anlehnung an Datensets und Veröffentlichungen in vergleichbaren Domänen ([58], [59]) und andererseits aus der Notwendigkeit zur Realisierung verschiedener Funktionen. Zur Erkennung von Menschen, müssen diese beispielsweise im Datensatz auch als Label ausgewiesen werden. Auf Seite 43 ff. werden in einer Übersicht die verschiedenen Klassen sowohl für Objektdetektion als auch für semantische Segmentierung dargestellt.

Mithilfe einer Occlusion-Metrik (Verdeckung) wurde eine Möglichkeit geschaffen, einzelne Bounding-Boxen anhand der Verdeckung des Objekts auszufiltern. Wird beispielsweise ein Objekt zu 85% verdeckt, ist es für Objekterkennungs-Algorithmen im Allgemeinen schwer, dieses Objekt zu erkennen. Daher können zugehörige Labels aus dem Trainingsdatensatz bei Bedarf ausgeschlossen werden. Abbildung 17 zeigt beispielhaft dieselbe Szene ohne, mit und mit gefilterten Bounding-Boxen.

Abbildung 17: 2D-RGB-Bild (links), überlagerte Bounding-Boxen (mittig), überlagerte durch Occlusion-Filter gefilterte Bounding-Boxen (rechts)

In Abbildung 18 ist die semantische Segmentierung dargestellt. Auf der linken Seite ist das zugrunde liegende 2D-RGB-Bild dargestellt. Auf der rechten Seite ist dieses mit der semantischen Segmentierung durch Transparenz überlagert. Jede Klasse erhält zur Unterscheidung eine eigene Farbe. Beispielsweise ist der Boden rosa eingefärbt und Paletten braun.

2D-RGB-Bild Segmentierung

Abbildung 18: 2D-RGB-Bild (links) und überlagerte Segmentierung (rechts)

Neben den 2D-RGB-Kamera-Bildern wird auch das Bild einer Tiefen-Kamera simuliert. Zu beachten ist, dass sich die Kamera-Labels sowohl auf die 2D-Bilder als auch auf die Tiefen-Bilder anwenden lassen, da sich beide Kameras den gleichen Ursprung teilen. Abbildung 19 zeigt das Tiefen-Bild und beispielhaft die Überlagerung von Bounding-Boxen.

2D-RGB-Bild Tiefen-Bild Tiefen-Bild mit Bounding-Boxen

Abbildung 19: 2D-RGB-Bild (links), Tiefen-Bild (mittig), Tiefen-Bild mit überlagerten Bounding-Boxen (rechts)

Darüber hinaus werden auch die einzelnen Punkte der 2D-LiDAR-Punktwolke klassifiziert. In Abhängigkeit des Objekts, das durch den jeweiligen Laserstrahl getroffen wird, wird dieser Reflektion zusätzlich zu Distanz und Winkel eine Klasse zugeordnet (vgl. Abbildung 20). Im Ergebnis ermöglicht dies eine punktgenaue Klassifizierung jeder Punktwolke in jedem Zeitschritt. Dadurch können beispielsweise Clusteralgorithmen entwickelt werden, die in der Lage sind, Reflektionspunkte von gleichen Objekten zusammenzufassen.

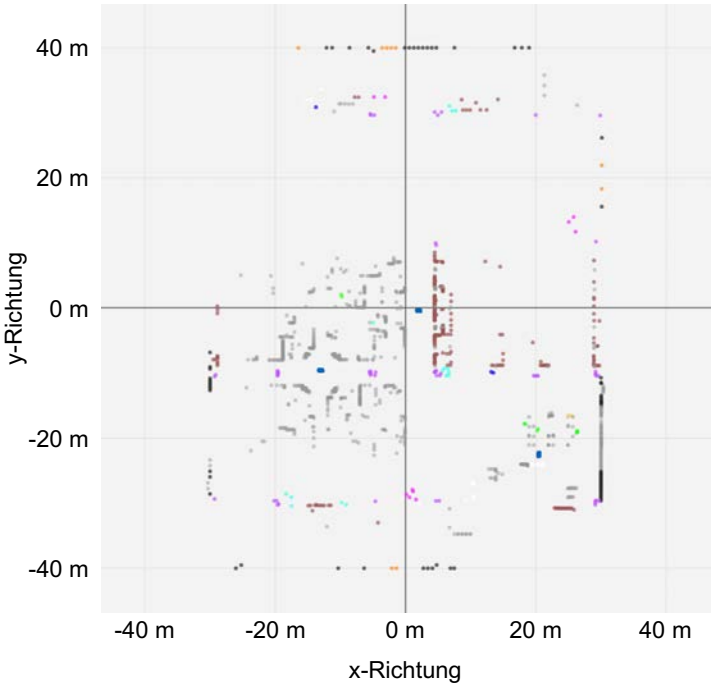


Abbildung 20: Darstellung der kumulierten LiDAR-Punktwolke dreier Roboter; Klassifizierung einzelner LiDAR-Punkte dargestellt durch unterschiedliche Farben (z. B. Gabelstapler in pink)

Datenstruktur

Für eine weitere Verwendung müssen die Daten strukturiert aus der Simulation exportiert werden können. Dafür wurde eine Struktur auf Basis des von Unity entworfenen SOLO-Schemas gewählt. Sensoren, die Unity nicht in seinen Paketen vorgesehen hat (LiDAR, Lokalisierung), wurden simulationsseitig in die Exportskripte integriert, sodass am Ende eine einheitliche Datenbasis entsteht.

Dazu wurde der Sensortyp „type.custom/solo.2DLidar“ eingeführt. Dieser enthält zusätzlich zu den Standard-Feldern (Sensor-Typ, Sensor-ID, Sensor-Beschreibung, globale Sensorrotation und -position) die lokale Position und Rotation des Sensors relativ zum Roboter, Scanreichweite und -Winkel und den neuen Annotationstyp „type.custom/solo.2DLidarClassification“. In diesem werden Distanzen, Winkel, Klassen und Intensitäten (aktuell ungenutzt) der LiDAR-Punktwolke gespeichert.

Für den Lokalisierungssensor reicht die Standard-Sensor-Definition, die die globale Position und Rotation bereits enthält. Eine Anwendung dieses Sensortyps auf einen Roboter kann so die Daten direkt in die Frame-Daten schreiben.

Ebenso wurde die Standard-Struktur auf oberster Ebene um eine Klassen-Definition erweitert (vgl. Abbildung 7). Dadurch können die in der Segmentierungs-Maske angegebenen Farben in Klassen übersetzt werden.

Optimierung des Datensets

Um das finale Datenset zu erreichen, wurden iterativ verschiedene Optimierungen vorgenommen. Im Training von Methoden basierend auf Supervised-Ansätzen ist ein zentrales Problem die ungleiche Verteilung von Klassen. Sind Klassen gegenüber anderen Klassen überrepräsentiert, können diese auch mit höherer Erfolgschance durch einen Mustererkennungsalgorithmus „erlernt“ werden. Klassen hingegen, die unterrepräsentiert sind, werden von solchen Ansätzen im Ergebnis in der Regel schlechter erkannt. In realen Umgebungen ist eine homogene Klassenverteilung durch äußere Restriktionen (z. B. begrenzte Anzahl an Feuerlöschern) meist nicht möglich. Wird eine reale Umgebung in einer Simulation nachgestellt, sind also auch dort Inhomogenitäten impliziert. Allerdings gibt es hier die Möglichkeit, dem in Grenzen entgegenzuwirken. Dazu können seltene Gegenstände häufiger eingesetzt und häufige Gegenstände etwas reduziert werden. Konkret wurde die Anzahl an KLTs, Gabelstaplern, Hubwagen und Feuerlöschern erhöht, und im Gegenzug die ohnehin hohe Anzahl an Paletten etwas reduziert. Außerdem wurde das Layout insgesamt angepasst und weitere Maschinen eingeführt. Das reduziert den unverstellten Blick auf Boden und Wände, die sonst einen überproportional großen Anteil der Bildfläche einnehmen.

Darüber hinaus wurde die Strategie der Datensammlung etwas angepasst. In den ersten Versionen der Simulation waren die einfachen Fahrwege der Roboter deutlich kürzer. Durch eine Verlängerung und Erhöhung der Komplexität der Fahrwege können die Roboter länger fahren, bevor eine Wiederholung eintritt. Das macht die erhobenen Daten vielseitiger.

Zuletzt wurde die Annotation der semantischen Segmentierung der Gitterbox optimiert (s. Abbildung 21). Algorithmen für die semantische Segmentierung sind besonders anfällig für verhältnismäßig feine Strukturen. Die ursprüngliche Art, die Gitterbox zu labeln, führte daher in der Anwendung der Daten zu schlechteren Ergebnissen. Durch Labeln der gesamten Flächen und Reduktion der Detaillierung konnten die Ergebnisse verbessert werden.

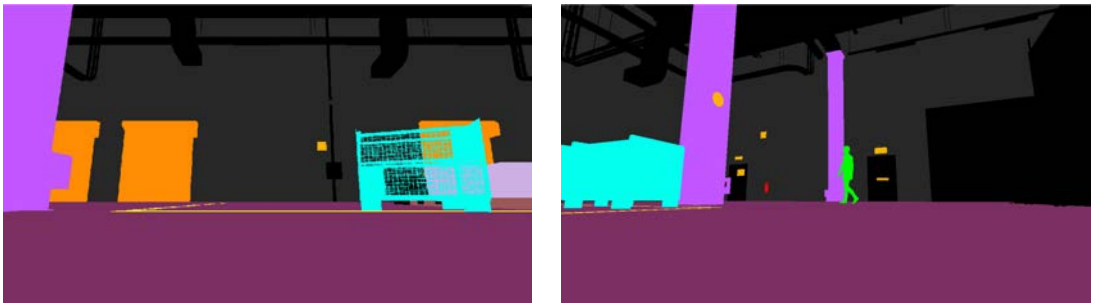


Abbildung 21: Gitterbox (türkis) links mit durchbrochenem Gitternetz, rechts in optimierter Version mit durchgehenden Flächen

Datenset-Statistik

Das fertige Datenset enthält 13 unterschiedliche Klassen für Bounding-Boxen. Diese wurden, wie auf Seite 40 beschrieben, unter Berücksichtigung bestehender Datensätze ausgewählt. Dabei wurde darauf geachtet, die verschiedenen in Abschnitt 2.1.3 genannten Klassen transformierte Objekte, transformierende Objekte und Personen abzubilden. Für die Objektdetektion sind insbesondere Objekt von Interesse, die eine Interaktion mit dem Roboter ermöglichen oder notwendig machen. Daher liegt der Schwerpunkt auf Personen, Gütern und Materialflussmitteln. Ausnahmen sind Kennzeichnungen, da diese für den Roboter relevante Informationen zur Umgebung bereitstellen können. Im Falle eines medizinischen Notfalls könnten so beispielswiese gekennzeichnete Erste-Hilfe-Positionen weiträumig umfahren werden. Außerdem wurden Feuerlöscher gelabelt, die aufgrund ihrer tiefen Position in der Praxis häufig problematisch zu unterfahren sind, da sie im LiDAR-Scan nicht zu erkennen sind. Eine Erkennung von Rolltoren kann für eine automatisierte Ansteuerung darüber hinaus wichtig sein. Tabelle 5 fasst die Label zusammen.

Tabelle 5: Im Datenset ausgewiesene Label für Objektdetektion

Label	Beschreibung	Kategorie
barrel	Fass	Transformiertes Objekt (Gut)
cardboard_box	Karton	Transformiertes Objekt (Gut)
decal	Kennzeichnung	Transformierendes Objekt (Infrastruktur)
fire_extinguisher	Feuerlöscher	Transformierendes Objekt (Infrastruktur)
forklift	Gabelstapler	Transformierendes Objekt (Materialflussmittel)
human	Mensch	Person
large_load_carrier	Großladungsträger	Transformierendes Objekt (Materialflussmittel)
pallet	Palette	Transformierendes Objekt (Materialflussmittel)
pallet_truck	Handhubwagen	Transformierendes Objekt (Materialflussmittel)
robot	Mobiler Roboter	Transformierendes Objekt (Materialflussmittel)
rolling_shutter	Rolltor	Transformierendes Objekt (Infrastruktur)
small_load_carrier	Kleinladungsträger	Transformierendes Objekt (Materialflussmittel)
stillage	Gitterbox	Transformierendes Objekt (Materialflussmittel)

Abbildung 22 zeigt die Verteilung der Klassen für Objektdetektion im erstellten Datenset. Es ist deutlich sichtbar, dass Paletten deutlich überproportional abgebildet sind. Feuerlöscher, Roboter und Gabelstapler sind hingegen gering repräsentiert. Das ergibt sich primär aus der realen Verteilung in vergleichbaren Szenarien. Besonders das Palettenregal sorgt für einen deutlichen Anstieg der Palettenanzahl, der auch bei einer

vergleichbaren realen Szene aufträte. Entsprechend lässt sich auch die Anzahl an Gabelstaplern oder Robotern nicht beliebig erhöhen. Da ein Ziel des Datensatzes die möglichst realitätsnahe Abbildung einer Industrieszenerie ist, ist damit eine Unausgewogenheit der Klassenverteilung unvermeidbar. Darüber hinaus wird die Verteilung auch von den gewählten Fahrtstrecken der Roboter und den darin präsenten Objekten beeinflusst. Daher unterscheidet sich auch der Anteil an der Gesamtanzahl der Labels durch die einzelnen Roboter.

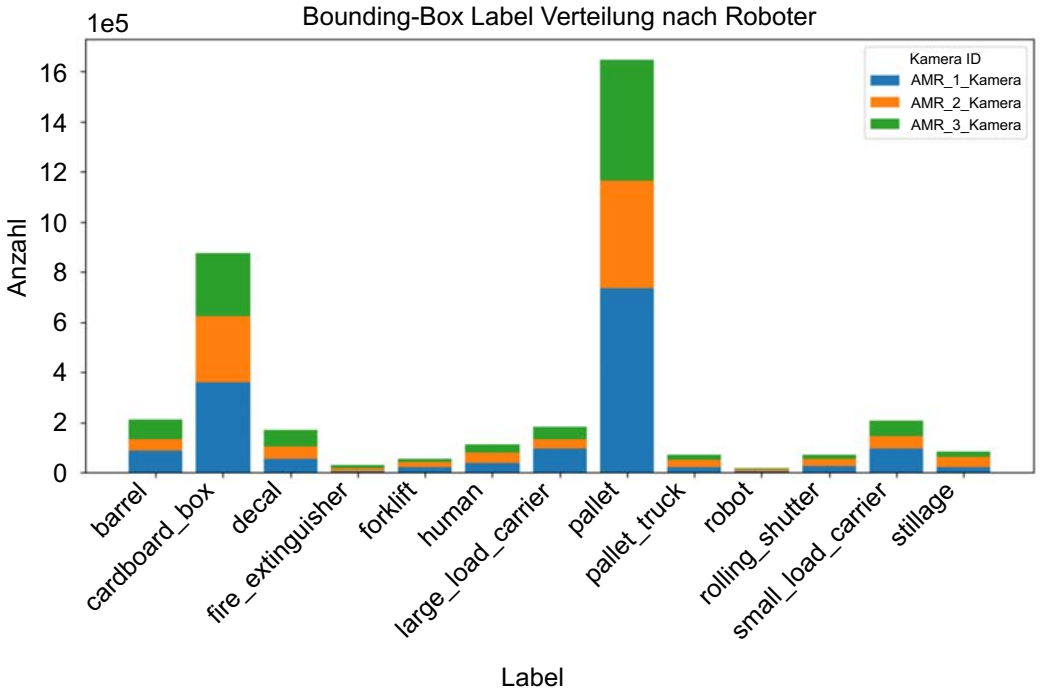


Abbildung 22: Verteilung der Klassen für Bounding-Boxen über das gesamte Datenset aufgeteilt nach dem erfassenden Roboter

Für die semantische Segmentierung wird Liste an Klassen auf insgesamt 17 erweitert (siehe

Tabelle 6). Im Gegensatz zur Objektdetektion sind für die semantische Segmentierung auch Infrastrukturobjekte von größerer Bedeutung. Diese Objekte erfordern wenig oder keine Interaktion mit dem Roboter, liefern aber dennoch für bestimmte Szenarien notwendige Informationen. Außerdem umspannen sie meist große, homogene Bereiche des Bildes (z. B. Boden oder Wände) und sind daher für Objektdetektion ungeeignet. Zusätzlich wurden Wände, Säulen, Boden und Bodenmarkierungen aufgenommen. Durch die Erkennung des Bodens in Kombination mit Bodenmarkierungen kann beispielsweise eine automatisierte Erkennung der befahrbaren Fläche erfolgen. Damit auch Algorithmen rein auf semantischer Segmentierung basierend entwickelt werden können, wurden die Label der Objektdetektion beibehalten.

Tabelle 6: Im Datenset ausgewiesene Label für semantische Segmentierung

Label	Beschreibung	Kategorie
barrel	Fass	Transformiertes Objekt (Gut)
cardboard_box	Karton	Transformiertes Objekt (Gut)
decal	Kennzeichnung	Transformierendes Objekt (Infrastruktur)
decal_floor	Bodenmarkierung	Transformierendes Objekt (Infrastruktur)
fire_extinguisher	Feuerlöscher	Transformierendes Objekt (Infrastruktur)
floor	Boden	Transformierendes Objekt (Infrastruktur)
forklift	Gabelstapler	Transformierendes Objekt (Materialflussmittel)
human	Mensch	Person
large_load_carrier	Großladungsträger	Transformierendes Objekt (Materialflussmittel)
pallet	Palette	Transformierendes Objekt (Materialflussmittel)
pallet_truck	Handhubwagen	Transformierendes Objekt (Materialflussmittel)
pole	Säule	Transformierendes Objekt (Infrastruktur)
robot	Mobiler Roboter	Transformierendes Objekt (Materialflussmittel)
rolling_shutter	Rolltor	Transformierendes Objekt (Infrastruktur)
small_load_carrier	Kleinladungsträger	Transformierendes Objekt (Materialflussmittel)
stillage	Gitterbox	Transformierendes Objekt (Materialflussmittel)
wall	Wand	Transformierendes Objekt (Infrastruktur)

Für das erstellte Datenset ergibt sich für die eben vorgestellten Klassen für semantische Segmentierung die in Abbildung 23 dargestellte Verteilung. Dabei ist zu beachten, dass die Anzahl der semantischen Label im Gegensatz zur Anzahl der Bounding-Boxen, die eine Unterscheidung der einzelnen Objekt-Instanzen erlaubt, nur nach Erscheinen pro Zeitschritt ermittelt werden kann. Erscheint eine Klasse (egal wo und wie oft) in einer Aufnahme, wird die Anzahl um einen Zähler erhöht. Das homogenisiert die Verteilung im Vergleich zur Verteilung der Bounding-Boxen-Label. Dennoch ist erkennbar, dass auch hier die Roboter unterschiedlich stark zu verschiedenen Klassen beitragen. Roboter 1 beispielsweise nimmt signifikant weniger Feuerlöscher auf, als Roboter 1 und 2.

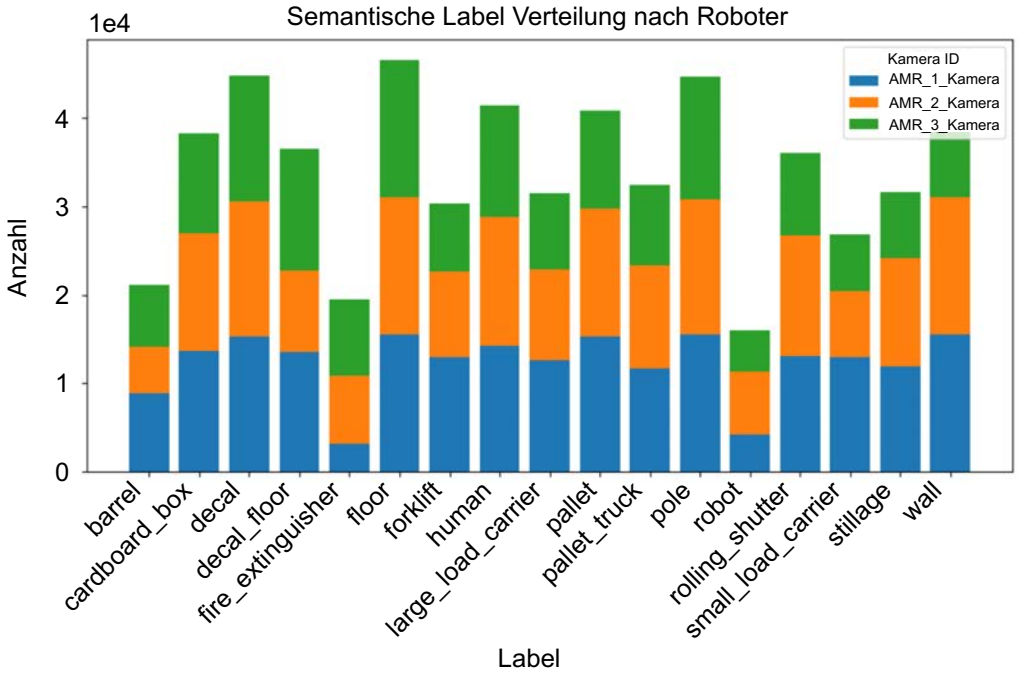


Abbildung 23: Absolute Verteilung der Klassen für semantische Segmentierung über das gesamte Datenset aufgeteilt nach dem erfassenden Roboter

Abbildung 24 zeigt zudem eine relative Häufigkeits-Verteilung der Klassen auf Pixel-Ebene im gesamten Datenset über alle Roboter kumuliert. Wie auch bei den Labels für Objektdetektion, sind einige Klassen stark überrepräsentiert. Dass bestimmte Klassen überproportional stark repräsentiert sind (Wand, Säule, Boden), ist dadurch zu erklären, dass diese häufig anzutreffen sind und im Bild zusätzlich große Flächen einnehmen. Andere Klassen sind im Verhältnis dazu deutlich seltener und kleiner. Daher ist die durch sie belegte Anzahl an Pixeln deutlich geringer.

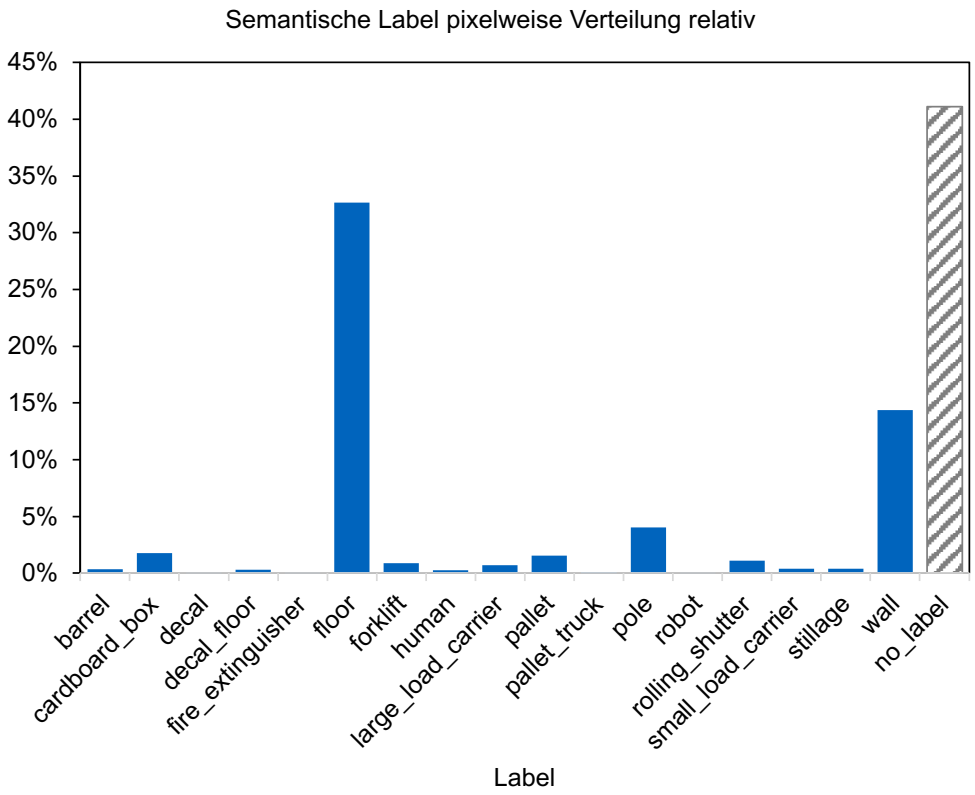


Abbildung 24: Relative Verteilung der Klassen für semantische Segmentierung über das gesamte Datenset und alle Roboter

Darüber hinaus weist das Datenset folgende Daten auf:

Tabelle 7: Datenset-Statistik

Laufzeit (RT)	~ 1550 s
Anzahl Frames	48305
Anzahl Klassen Bounding-Boxen	13
Anzahl Klassen Semantische Segmentierung	17
Anzahl Roboter	3
Sensoren	2 x LiDAR (v, h) 1 x RGB-Kamera 1 x Tiefenkamera 1 x Lokalisierung
Wiederholrate LiDAR	30 Hz
Wiederholrate Kamera	10 Hz
Wiederholrate Lokalisierung	30 Hz
Speichergröße	~ 123 GB

4 Kollektives Umgebungsinformationssystem

Ziel des kollektiven Umgebungsformationssystems ist es, die Auswertung von Sensordaten einer Flotte mobiler Roboter auf einen zentralen Server zu verlagern, um Synergieeffekte nutzbar zu machen (vgl. Abschnitt 1.2). Um das zu ermöglichen, müssen verschiedene Funktionen erfüllt werden, die wiederum von verschiedenen Modulen übernommen werden. In den nachfolgenden Abschnitten wird zunächst ein Überblick über die Gesamtarchitektur gegeben. Anschließend werden die einzelnen Funktionsmodule näher erläutert.

4.1 Gesamtarchitektur

Wie in Abschnitt 1.3 beschrieben besteht das kollektive Umgebungsinformationssystem aus drei Hauptbestandteilen:

- *Umgebungsmodell*
  Repräsentation der Gesamtumgebung in einem einheitlichen Modell
- *Datenübertragung*
  Übertragung der Sensordaten von den mobilen Robotern zum Umgebungsmodell und relevanter Modelldaten vom Umgebungsmodell zu den mobilen Robotern
- *Flotte mobiler Roboter*
  Aufnahme der Umgebung durch Sensoren und Übermittlung der Daten an das Modell

Diese Bestandteile werden im nachfolgenden Diagramm in einzelne Funktionsmodule und Datenflüsse unterteilt:

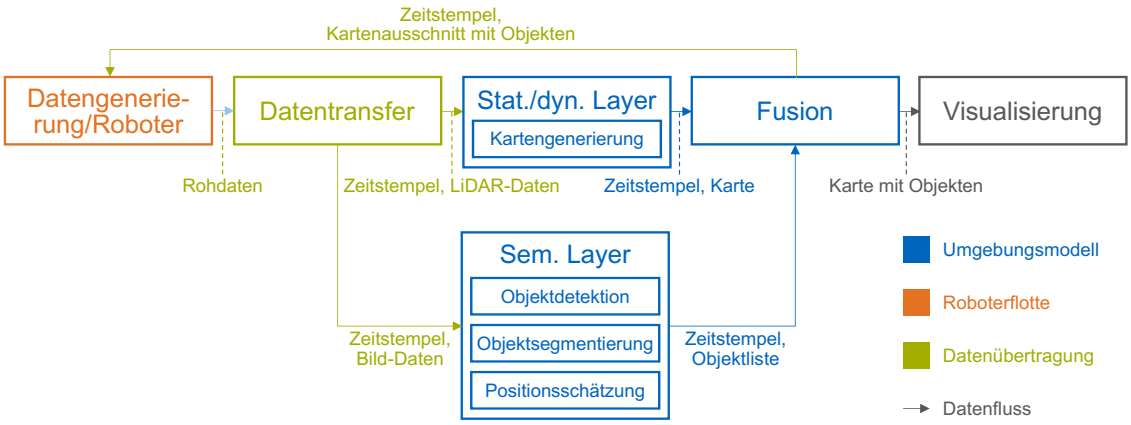


Abbildung 25: Funktionsmodule des kollektiven Umgebungsinformationssystems verbunden durch Datenflüsse

Das erste Modul *Datengenerierung / Roboter* befasst sich mit der Erzeugung der Daten und wurde im vorherigen Abschnitt 3 behandelt. Die dort erzeugten Daten werden im Anschluss über den *Datentransfer* an die Generierung des Umgebungsmodells übermittelt. Diese setzt sich zusammen aus dem *statischen und dynamischen Layer* einerseits und dem *semantischen Layer* andererseits. Während in Ersterem eine Karte erzeugt wird, die vor allem die Grundlage für Aufgaben der Eigen- und Objektlokalisierung und der Pfadplanung bildet, wird diese Karte im semantischen Layer um weitere Informationen angereichert. Zu wissen, welche Objekte den Roboter umgeben, und sich nicht blind auf Sicherheitsfelder verlassen zu müssen, kann in vielen Fällen den Prozess beschleunigen oder die Sicherheit erhöhen. Bspw. ist es dann möglich statische Paletten

von dynamischen Gabelstaplern gezielt zu unterscheiden und somit Ausweichmanöver präziser zu planen und auf die Anforderungen der Situation abzustimmen. Beide Layer werden im Anschluss im Schritt der *Fusion* in ein einheitliches Modell überführt. Das heißt, semantische Informationen werden in der Karte verankert. Z. B. werden erkannte Objekte zur richtigen Zeit am richtigen Ort in der Karte mit den Informationen der Karte kombiniert. Relevante Informationen können nach dem Update des Modells an einzelne Roboter zurückgespiegelt werden. Muss beispielsweise für einen neuen Auftrag ein Pfad geplant werden, können so potenzielle Blockaden frühzeitig erkannt und in die Route mit eingeplant werden. Zuletzt wird das Ergebnis für den Nutzer mithilfe einer *Visualisierung* dargestellt.

4.2 Datentransfer

Um die Daten von den mobilen Robotern auf den zentralen Server zu übertragen, auf dem die Berechnung des kollektiven Umgebungsmodells stattfindet, braucht es ein geeignetes Konzept vornehmlich für drei Bereiche:

- *Übertragungstechnologie*
  Über welchen Funkstandart werden die Daten übertragen?
- *Synchronisierung*
  Wie können Daten nach Zeit und Ort differenziert werden?
- *Queuing*
  Wie kann eine Kollision der Daten trotz Zentralisierung effektiv verhindert werden und wie kann ihre Reihenfolge eingehalten werden?

In den folgenden Abschnitt werden die drei Teilbereiche näher erläutert. Da eine praktische Lösung dieser Fragen jedoch aufgrund der Nutzung einer Simulation für die Datenerzeugung meist irrelevant war, werden hier lediglich theoretischer Aspekte erläutert.

4.2.1 Übertragungstechnologie

In Abschnitt 2.4.1 wurden bereits die Grundlagen zu verschiedenen Übertragungstechnologien dargelegt. Demnach kommen für die Umsetzung des Umgebungsinformationssystems vornehmlich 3GPP oder Wi-Fi 6 (IEEE 802.11ax) infrage. Das resultiert insbesondere aus den Anforderungen an die Bandbreite aufgrund der großen Menge an zu übertragenden Sensordaten. Wi-Fi 7 (IEEE 802.11be) ist daher für zukünftige Anwendungen ebenfalls denkbar, aktuell jedoch aufgrund der geringeren Verfügbarkeit von notwendigem Equipment noch wenig verbreitet. Entscheidungsrelevant sind darüber hinaus häufig der Aufwand zur Installation und Entwicklung einer Übertragungslösung, da hier hohe Kosten verursacht werden. Auch wenn die Implementierung einer Wi-Fi-Umgebung im Detail aufwendig ist, so ist 3GPP 5G aufgrund technologischer und regulatorischer Hürden nach wie vor erheblich komplexer umzusetzen. Wie im Rahmen von Expertengesprächen bestätigt wurde, kommt daher im Bereich mobiler Roboter fast ausschließlich (IEEE 802.11ax) zum Einsatz.

Außerdem wurde darauf hingewiesen, dass bei der Implementierung einer Funkübertragung aus technischer Sicht vor allem geringe Latenz, große Abdeckung und der reibungslose Wechsel von Netzzugangspunkten (Handover zwischen Access-Points) die größten Herausforderungen darstellen. Diese Punkte sollten daher bei der Wahl der Übertragungstechnologie besonders berücksichtigt werden.

4.2.2 Sensordatensynchronisierung

Wie in Abschnitt 2.4.2 beschrieben, ist es notwendig, Sensordaten im Zuge der Fusion zu synchronisieren, um Inkonsistenzen in der Modellbildung zu verhindern. Diese Anforderung wurde bei der simulativen Generierung der Sensordaten ebenfalls berücksichtigt.

Für den zeitlichen Bezug wurde jeder Messung ein Zeitstempel angefügt. Durch eine Ergänzung um die lokale Position des Sensors relativ zum Roboter-Koordinatensystem und einen dedizierten Lokalisierungssensor, der das Roboterkoordinatensystem im globalen Koordinatensystem verortet, können die Sensordaten auch räumlich in Relation gesetzt werden. Als Rückfallebene werden zusätzlich zu jeder Messung die globalen Koordinaten des zugehörigen Sensors ergänzt. Das ist in der Realität nicht mit vertretbarem Aufwand durchführbar, ermöglicht jedoch eine einfache Überprüfung der Transformationen.

4.2.3 Queuing

Nach der Übertragung erreichen die Sensordaten den Server in der Regel ungeordnet und werden kurzfristig in einer sogenannten Queue gespeichert. Eine Queue ist ein Kurzzeitspeicher, der nach verschiedenen Prinzipien gefüllt und entleert werden kann (vgl. Gumm et al. [60], S. 351-354).

Nach kurzem Zwischenspeichern in einer Queue werden die Daten durch Synchronisierungsmechanismen geordnet und in korrekter Reihenfolge in eine neue Queue eingespeichert. Von dieser können sie dann an die Modellgenerierung übergeben werden. Auf diese Weise können Datengenerierung und -übertragung von der Weiterverarbeitung entkoppelt werden.

Abbildung 26 veranschaulicht den Prozess im Gesamtkontext. Auf der linken Seite werden Sensordaten durch die Roboterflotte generiert. Diese werden anschließend an die erste Queue übergeben. Aufgrund verschiedener Einflüsse, wie z. B. unterschiedliche Verbindungsqualitäten, erreichen die Daten den Server in falscher zeitlicher Reihenfolge (t_{n-1}, t_{n+1}, t). Nach Einspeichern in die erste Queue werden die Sensordaten sortiert (t_{n-1}, t, t_{n+1}) und in die nächste Queue eingespeichert. Von dort gelangen Sie in korrekter Reihenfolge zum Umgebungsmodell.

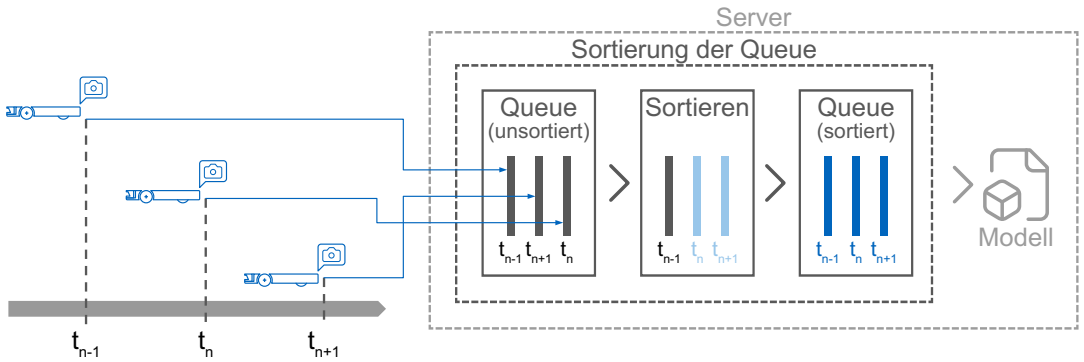


Abbildung 26: Generierung der Sensordaten durch die Roboterflotte (links) und Übertragung an die erste Queue; anschließend Sortieren, erneutes Abspeichern und Übergabe an Umgebungsmodell

4.3 Statischer und dynamischer Layer

Das Lokalisieren von Hindernissen erfordert die Erfassung der Umgebung in einer Karte. Im Bereich der mobilen Roboter haben sich in der Vergangenheit metrische Karten durchgesetzt, da diese eine verglichen mit topologischen Ansätzen einfache Routenplanung ermöglichen. Durch die Kombination einer statischen mit einer dynamischen Karte können beide so parametriert werden, dass sie unterschiedliche Umgebungsobjekte möglichst gut repräsentieren können. In Abschnitt 2.1.3 wird bereits eine Unterscheidung von Objekten vorgenommen. Eine weitere Unterscheidung kann nach der Dynamik von Objekten erfolgen (vgl. Abschnitt 3.1.3). Regale beispielsweise sind in der Regel fest verbaut und können daher als statisch angesehen werden. Menschen und Gabelstapler sind üblicherweise ortsveränderlich und weisen daher eine Eigendynamik auf. Sollen beide Gruppen sensorisch erfasst und in einem Modell repräsentiert werden, muss üblicherweise mit unterschiedlichen Parametersätzen für die Modellbildung gearbeitet werden.

Für dieses Projekt wurde erst der in Abschnitt 0 beschriebene Grid-basierte Ansatz von Wolf und Sukhatame [61] gewählt. Anschließend wurde der Kartierungsalgorithmus von Vandorpe et al. [62] implementiert, um der ressourcenintensiven Grid-basierten Lösung einen ressourcenschonenderen Ansatz entgegenzustellen. Beide Algorithmen wurden im Rahmen des Projekts weiterentwickelt, um den Anwendungsfall besser abzubilden.

4.3.1 Grid-basierter Ansatz

Um statische von dynamischen Objekten zu unterscheiden werden nach dem Ansatz von Wolf und Sukhatame [61] zwei separate Grid-Karten S und D geführt. S repräsentiert alle statischen Elemente der Umgebung, während D die Belegungswahrscheinlichkeit dynamischer Hindernisse abbildet. Ein Hindernis wird als dynamisch eingestuft, wenn es sich in der Vergangenheit mindestens einmal gegenüber dem letzten Zeitschritt bewegt hat.

Da beide Karten unabhängig voneinander aktualisiert werden, bedeutet eine freie Zelle in S lediglich, dass dort kein statisches Hindernis zu erwarten ist, dynamische Hindernisse aber dennoch vorhanden sein können. Umgekehrt bedeutet eine freie Zelle in D nur die Abwesenheit dynamischer Hindernisse. Statische Hindernisse können trotzdem auftreten.

Die übergeordneten Schritte des Kartierungs-Algorithmus sind in Abbildung 27 dargestellt:

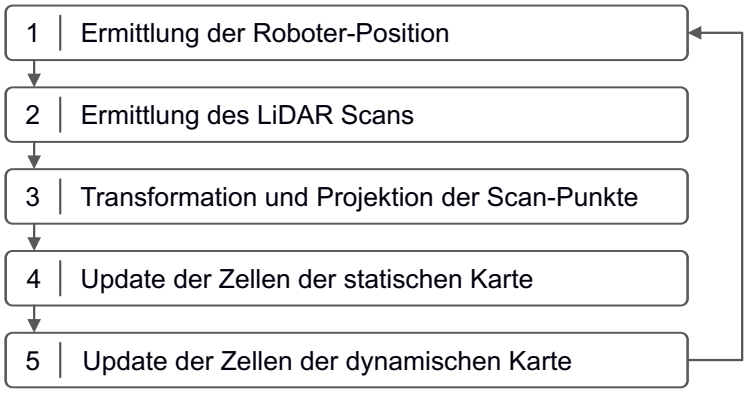


Abbildung 27: Schritte des implementierten Grid-basierten Kartierungs-Algorithmus

Zunächst muss die Position des Roboters ermittelt werden (Schritt 1). Da sich das kollektive Umgebungsinformationssystem nur mit der Modellbildung befasst, wird die Lokalisierung nicht betrachtet. Stattdessen

wird angenommen, dass sich der Roboter selbstständig lokalisieren kann und dafür nicht auf das Modell angewiesen ist.

Anschließend werden die LiDAR-Scans des Roboters erzeugt (Schritt 2). In Schritt 3 werden diese Punktwolken in das globale Karten-Koordinatensystem transformiert und die einzelnen Punkte in die Karte projiziert. Basierend darauf werden in Schritt 4 und 5 zuerst die statische Karte S und dann die dynamische Karte D aktualisiert bevor der Prozess erneut gestartet wird.

Im Detail verlaufen die Schritte wie folgt.

Ermittlung der Roboterposition

Aus der Simulation sind die idealen Positionsdaten des Roboters bekannt. Da die Eigen-Lokalisierung im Rahmen des Projekts nicht betrachtet wird, werden diese Idealdaten für die weiteren Berechnungen genutzt.

Ermittlung des LiDAR-Scans

Ebenfalls der Simulation entstammen die LiDAR-Scans. In Abschnitt 3.1.4 wird der Aufbau der Sensorik und der Prozess der simulativen Sensordatengenerierung näher erläutert.

Transformation und Projektion der Scan-Punkte

Ziel dieses Schrittes ist es, die LiDAR-Scans der Roboter so zu prozessieren, dass sie für die Berechnung der Belegungswahrscheinlichkeiten der Karten herangezogen werden können. Dazu wird die Position des LiDAR-Sensors und sukzessive jeder Scan-Punkt anhand der Roboter-Pose aus dem lokalen Roboter-Koordinatensystem in das globale Karten-Koordinatensystem transformiert und in das Gitter projiziert. Mithilfe des Bresenham Algorithmus können dann die Zellen bestimmt werden, die der Lichtstrahl vom Sensor bis zum Objekt zurückgelegt hat. Diese Zellen können als frei angenommen werden, während die Zelle des Scan-Punkts als belegt markiert wird. Alle anderen Zellen bleiben unverändert. In Abbildung 28 ist der Prozess dargestellt.

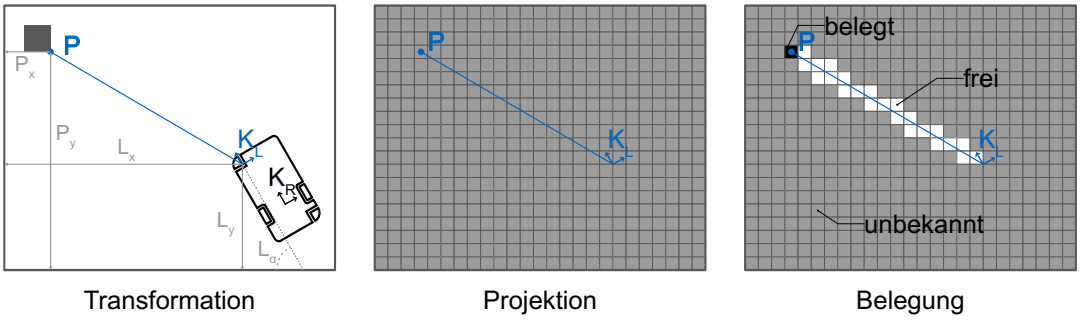


Abbildung 28: Schritte von der Transformation der Messung P ausgehend vom Koordinatensystem K_L des LiDARs in globale Koordinaten (links) über die Projektion ins Gitter (mittig) bis zur Belegung des Gitters (rechts)

Update der Zellen der statischen Karte

Ziel ist, für jeden Zeitschritt t aus den vorherigen Beobachtungen $\mathbb{P}(S^t|o^1 \dots o^t)$ den aktuellen Zustand S^t der statischen Karte zu bestimmen. Da sich statische Hindernisse nie über die Zeit ändern, können diese durch einen Vergleich mit dem letzten Zeitschritt S^{t-1} von dynamischen Hindernissen unterschieden werden. Damit muss insgesamt $\mathbb{P}(S^t|o^1 \dots o^t, S^{t-1})$ bestimmt werden.

Dafür wird folgende Gleichung angewendet:

$$\log\left(\frac{\mathbb{P}(S^t|o^1 \dots o^t, S^{t-1})}{1-\mathbb{P}(S^t|o^1 \dots o^t, S^{t-1})}\right) = \log\left(\frac{\mathbb{P}(S^{t-1})}{1-\mathbb{P}(S^{t-1})}\right) + \log\left(\frac{\mathbb{P}(S^t|o^t, S^{t-1})}{1-\mathbb{P}(S^t|o^t, S^{t-1})}\right)$$

In [61] ist die Herleitung im Detail erläutert. Zu erwähnen ist, dass die Log-Odds-Form ausgenutzt wurde, um numerischen Instabilitäten bei Wahrscheinlichkeiten nahe Null oder Eins entgegenzuwirken. Damit sind die Belegungswahrscheinlichkeiten der Karte $\mathbb{P}(S^t)$ abhängig vom vorherigen Zustand der Karte $\mathbb{P}(S^{t-1})$ und dem sogenannten inversen Sensormodell $\mathbb{P}(S^t|o^t, S^{t-1})$, das bestimmt, welchen Einfluss die Beobachtungen im aktuellen Zeitschritt auf die Belegungswahrscheinlichkeiten haben.

Folgende Tabelle fasst die Auswirkungen des inversen Sensormodells auf den Zustand der statischen Karte zusammen:

Tabelle 8: Inverses Sensormodell für das Update der statischen Karte [61]

$\mathbb{P}(\boldsymbol{S^{t-1}})$	$\boldsymbol{o^t}$	$\mathbb{P}(\boldsymbol{S^t}\|\boldsymbol{o^t}, \boldsymbol{S^{t-1}})$
Frei	Frei	↓
Unbekannt	Frei	↓
Belegt	Frei	↓
Frei	Belegt	↓
Unbekannt	Belegt	↑
Belegt	Belegt	↑

Die erste Spalte beschreibt den Zustand der statischen Karte im letzten Zeitschritt, die mittlere Spalte die Information des Sensors aus der aktuellen Messung und die rechte Spalte zeigt den Einfluss des inversen Sensormodells auf die aktuelle Karte. „Frei“ bedeutet in diesem Zusammenhang, dass die Belegungswahrscheinlichkeit $\mathbb{P}(S^{t-1})$ unter einem festgelegten Grenzwert liegt. „Belegt“ bedeutet entsprechend, dass die Belegungswahrscheinlichkeit über einem festgelegten Grenzwert liegt. Unbekannt ist jeder Zustand zwischen den Grenzwerten.

Wird durch den Sensor kein Hindernis erkannt (Zeile eins bis drei), wird die Wahrscheinlichkeit der jeweiligen Zelle verringert (↓). War die Zelle im letzten Zeitschritt in der Karte als frei markiert (Zeile vier), wird durch den Sensor jedoch als belegt angezeigt, geht das Modell davon aus, dass es sich um ein dynamisches Objekt handelt. Entsprechend wird die Wahrscheinlichkeit der statischen Karte durch das inverse Sensormodell verringert. Ist der Zustand einer Zelle unbekannt und der Sensor deutet auf eine belegte Zelle hin (Zeile fünf), wird so lange davon ausgegangen, dass es sich um ein statisches Hindernis handelt, bis es als bewegt erkannt wird. Daher wird die Belegungswahrscheinlichkeit in diesem Fall durch das Modell erhöht (↑). Auch wenn die Annahme einer belegten Zell durch die Messung des Sensors bestätigt wird, erhöht sich deren Wahrscheinlichkeit (Zeile sechs).

Nach diesem Ansatz würden die Belegungswahrscheinlichkeiten unendlich wachsen bzw. schrumpfen. Um eine ausreichende Berechnungseffizienz zu gewährleisten, wurden daher die Grenzen auf $[1 - p_{max}, p_{max}] = [1\%, 99\%]$ festgelegt.

Um die Werte des inversen Sensormodells für Erhöhung (↑) und Reduzierung (↓) der Belegungswahrscheinlichkeit zu bestimmen, wurde ausgehend von einer Zelle mit unbekannter Belegungswahrscheinlichkeit die Wahrscheinlichkeitsänderung ermittelt, die zu einer Erreichung des Grenzwerts p_{max} innerhalb von 60 Zeitschritten notwendig ist. D. h. wird ein Hindernis innerhalb von 60 aufeinanderfolgenden Zeitschritten als statisch wahrgenommen, ist der Grenzwert p_{max} und damit die maximal mögliche Belegungswahrscheinlichkeit erreicht. Die Werte ergeben sich zu:

↑: 0,519 ↓: 0,481 in Log-Odds-Form.

Update der Zellen der dynamischen Karte

Analog zur statischen Karte erfolgt das Update der dynamischen Karte nach der Gleichung:

$$\log\left(\frac{\mathbb{P}(D^t|o^1 \dots o^t, S^{t-1})}{1-\mathbb{P}(D|o^1 \dots o^t, S^{t-1})}\right) = \log\left(\frac{\mathbb{P}(D^{t-1})}{1-\mathbb{P}(D^{t-1})}\right) + \log\left(\frac{\mathbb{P}(D^t|o^t, S^{t-1})}{1-\mathbb{P}(D^t|o^t, S^{t-1})}\right)$$

Das inverse Sensormodell für die dynamische Karte ist wie folgt:

Tabelle 9: Inverses Sensormodell für das Update der dynamischen Karte [61]

$\mathbb{P}(\boldsymbol{S^{t-1}})$	$\boldsymbol{o^t}$	$\mathbb{P}(\boldsymbol{D^t}\|\boldsymbol{o^t}, \boldsymbol{S^{t-1}})$
Frei	Frei	↓
Unbekannt	Frei	↓
Belegt	Frei	↓
Frei	Belegt	↑
Unbekannt	Belegt	↓
Belegt	Belegt	↓

Wie bei der statischen Karte wird auch im dynamischen Fall unabhängig vom Zustand der statischen Karte aus dem vorherigen Zeitschritt bei einer Sensormessung, die „Frei“ anzeigt (Zeilen eins bis drei), die Belegungswahrscheinlichkeit der dynamischen Karte reduziert (↓). Ist jedoch die Zelle als „Frei“ markiert und die Messung indiziert eine Belegung (Zeile vier), erhöht das inverse Sensormodell die Belegungswahrscheinlichkeit der dynamischen Karte (↑). In diesem Fall wird davon ausgegangen, dass es sich um ein dynamisches Hindernis handelt. Bei unbekanntem oder belegtem Zustand der statischen Karte wird bei einer Messung, die einen belegten Zustand anzeigt (Zeilen fünf und sechs), die Belegungswahrscheinlichkeit der dynamischen Karte reduziert. Im ersten Fall kann keine Aussage über den Hindernistyp getroffen werden und im zweiten Fall wird von einem statischen Hindernis ausgegangen, solange keine Bewegung wahrgenommen wird.

Ebenfalls analog zum statischen Fall ergeben sich unter der Annahme von fünf Zeitschritten, die sich ein Hindernis in einer freien Zelle befinden muss, bevor es als dynamisches Objekt eingestuft wird, folgende Werte für das inverse Sensormodell:

↑: 0,751 ↓: 0,385 in Log-Odds-Form.

In Abbildung 29 ist ein beispielhaftes Ergebnis dargestellt. Wieder sind Zellen mit einer hohen Belegungswahrscheinlichkeit dunkel dargestellt, während helle Zellen als frei betrachtet werden. Es ist deutlich erkennbar, dass statische Hindernisse als dunkle Umrandung mit grauer Füllung in der statischen Karte dargestellt werden, da keine Aussagen über den Zustand innerhalb der Hindernisse getroffen werden kann (statisch wie dynamisch). In der dynamischen Karte sind zusätzlich zwei bewegte Hindernisse sichtbar (nachträglich blau markiert). Es handelt sich um einen Gabelstapler (oben) und einen Roboter (unten), die den Fahrweg des aufnehmenden Roboters kreuzen. Durch die Reflektionslinie, die sich mit der Zeit durch die Bewegung ändert und die Trägheit der Bildung der Karte (fünf notwendige Zeitschritte), erzeugen dynamische Objekte in der Karte immer eine Art Schleppe aus mehreren Linien.

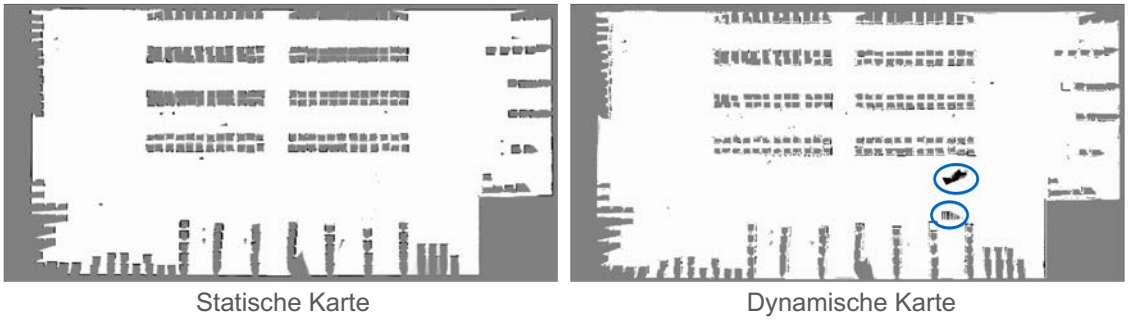


Abbildung 29: Beispielergebnis des Grid-basierten Ansatzes aus einer Simulation; Statische Karte (links) und dynamische Karte (rechts); blau umrandet sind dynamische Objekte

4.3.2 Kartierung mittels geometrischer Primitive

Wie bereits in Abschnitt 2.2.1 erwähnt, haben Grid-basierte Ansätze den Nachteil eines verglichen mit topologischen Ansätzen hohen Speicherbedarfs, da unabhängig von der tatsächlichen Komplexität der Umgebung immer eine gleichmäßige Auflösung beibehalten werden muss. Damit steigt der benötigte Speicherplatz mit der Größe der Umgebung. Außerdem sind für das Verfolgen von Objekten immer noch weitere Prozessschritte erforderlich, die belegte Zellen zusammenfassen und semantische Konsistenz über mehrere Zeitschritte herstellen (Welche Zellen gehören zu einem Objekt und in welche Richtung bewegt es sich?).

Es gibt verschiedene alternative Lösungsansätze, diesen Herausforderungen zu begegnen. Einer davon basiert auf einer Reduzierung der Komplexität mithilfe geometrischer Primitive (Linien und Kreise) nach Vandorpe et al. [62]. Diese brauchen nur dann Speicherplatz, wenn ein Hindernis erkannt wird. Außerdem können sie direkt dynamischen Objekten zugeordnet werden.

Im Folgenden wird zunächst der Ansatz nach Vandorpe et al. [62] beschrieben und anschließend um eine Optimierung für dynamische Objekte erweitert.

Der Ablauf ist wie folgt:

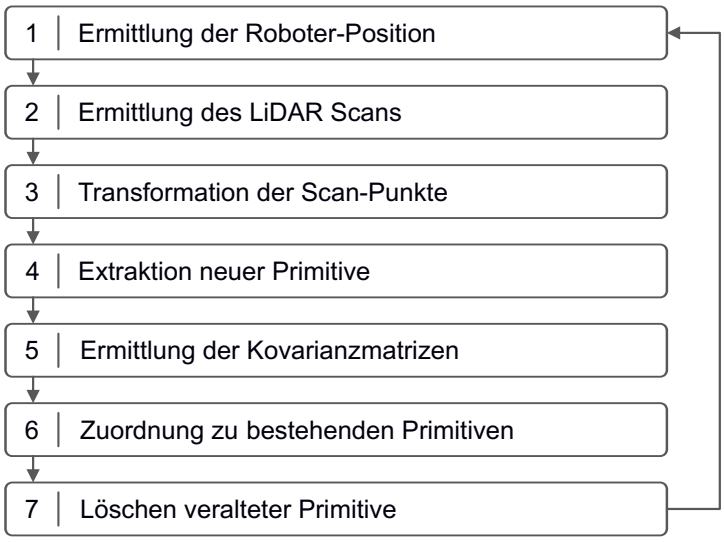


Abbildung 30: Schritte der hier implementierten Kartierung mit geometrischen Primitiven

Die ersten beiden Schritte verlaufen identisch zum Grid-basierten-Ansatz (vgl. 4.3.1). Schritt drei erfordert zusätzlich die Bestimmung einer Kovarianzmatrix für jeden einzelnen Scan-Punkt, um die Unsicherheiten der Messung abbilden zu können. In Schritt vier werden basierend auf den transformierten Punktwolken neue

Primitive ermittelt, zu denen dann in Schritt fünf jeweils eine Kovarianzmatrix ermittelt wird. Anschließend können die neuen Primitive mit bereits in der Karte bestehenden Primitiven verglichen werden. Ist eine ausreichende Übereinstimmung vorhanden, wird angenommen, dass es sich um dasselbe Hindernis handelt und die Primitive werden kombiniert. Andernfalls wird ein neues Hindernis als Primitiv abgespeichert. Zuletzt werden in Schritt sieben veraltete Primitive gelöscht. Das betrifft Primitive, die innerhalb des vom Sensor als frei ermittelten Bereichs liegen.

Transformation der Scan-Punkte

Wie beim Grid-basierten Ansatz werden die einzelnen Scan-Punkte $\boldsymbol{S}$ mithilfe der Vorschrift $\boldsymbol{X}_p = f(\boldsymbol{X}_r, \boldsymbol{S})$ in das globale Koordinatensystem transformiert, wobei $\boldsymbol{X}_r$ die Pose des Roboters beschreibt.

Anschließend wird für jeden Punkt $\boldsymbol{X}_p$ die Kovarianzmatrix $\boldsymbol{C}_p$ ermittelt, die dessen Unsicherheit beschreibt. Zur Bestimmung von $\boldsymbol{C}_p$ ist die Unsicherheit $\boldsymbol{C}_{Xr}$ der Roboterpose $\boldsymbol{X}_r$ (aus der Lokalisierung) und die Unsicherheit zu jedem Punkt $\boldsymbol{X}_p$ notwendig. Letztere wird durch die Kovarianzmatrix $\boldsymbol{C}_s$ abgebildet:

$$\boldsymbol{C}_s = \begin{bmatrix} \sigma_{rr} & 0 \\ 0 & \sigma_{\alpha\alpha} \end{bmatrix}$$

Da die Elemente nicht korrelieren, handelt es sich um eine Diagonalmatrix.

Damit kann für jeden Scan-Punkt die Unsicherheit in Form der Kovarianzmatrix $\boldsymbol{C}_p$ ermittelt werden zu:

$$\boldsymbol{C}_p = \boldsymbol{F}\boldsymbol{C}_{Xr}\boldsymbol{F}^T + \boldsymbol{G}\boldsymbol{C}_S\boldsymbol{G}^T$$

mit $\boldsymbol{F}$ als Jacobi-Matrix von f nach $\boldsymbol{X}_r$ und $\boldsymbol{G}$ als Jakobi-Matrix von f zu $\boldsymbol{S}$.

Extraktion neuer Primitive

Für jeden LiDAR-Scan werden die neuen Punkte der Punktwolke zu geometrischen Primitiven zusammengefasst. Der Ansatz beschränkt sich bei der Form der Primitive auf Linien und Kreise. Der Algorithmus versucht erst Linien zu bilden. Liegen die Scan-Punkte jedoch zu nah beieinander (z. B. bei einem Stuhlbein), würde dies in sehr kurzen Linien resultieren. In diesem Fall wird daher aus den Punkten ein Kreis gebildet.

Für die Entscheidung, ob der nächste Scan-Punkt $\boldsymbol{X}_{p,i} = (x_{p,i}, y_{p,i})$ zur aktuell entstehenden Linie gehört, gelten folgende Regeln:

1. Die Distanz des Punkts $\boldsymbol{X}_{p,i}$ zum zuletzt zur Linie hinzugefügten Punkt $\boldsymbol{X}_{p,i-1}$ ist kleiner als der Grenzwert d_{critp}.
2. Die Distanz von $\boldsymbol{X}_{p,i}$ zur Linie darf den Grenzwert d_{critl} nicht überschreiten.
3. Die Winkelabweichung $\Delta\alpha$ zwischen dem gemessenen Winkel α_i des neuen Punkts $\boldsymbol{X}_{p,i}$ und dem Winkel α_{i-1} des vorherigen Punkts $\boldsymbol{X}_{p,i-1}$ darf den Grenzwert α_{crit} nicht überschreiten.

Wird eine der Vorschriften verletzt, wird die Extraktion der Linie unter der Voraussetzung abgeschlossen, dass die Linie lang genug ist ($> ll_{min}$) und genug Punkte ($> lp_{min}$) aufweist.

Andernfalls wird versucht einen Kreis zu generieren. Einzige Bedingung für das Hinzufügen eines neuen Punkts $\boldsymbol{X}_{p,i}$ zu einem aktuell entstehenden Kreis ist, dass die Distanz von $\boldsymbol{X}_{p,i}$ zum Zentrum des Kreises unter dem Grenzwert d_{crit} liegen muss. Wird dieses Kriterium nicht erfüllt und liegt die Anzahl der Punkte des Kreises unter dem Grenzwert cp_{min} , wird die Generierung abgebrochen.

Linien werden schlussendlich wie in Abbildung 31 dargestellt über die Parameter ρ, θ und die Endpunkte (x_b, y_b) und (x_e, y_e) definiert. Dazu wird nach jedem neu zur Linie hinzugefügten Punkt (s. o) eine lineare

Regression angewendet und nach Abschluss des Prozesses die Linie in eine Gerade, definiert über ρ und θ, umgewandelt. Abschließend wird das Liniensegment über Projektion des ersten und letzten der Gerade zugeordneten Punkts bestimmt.

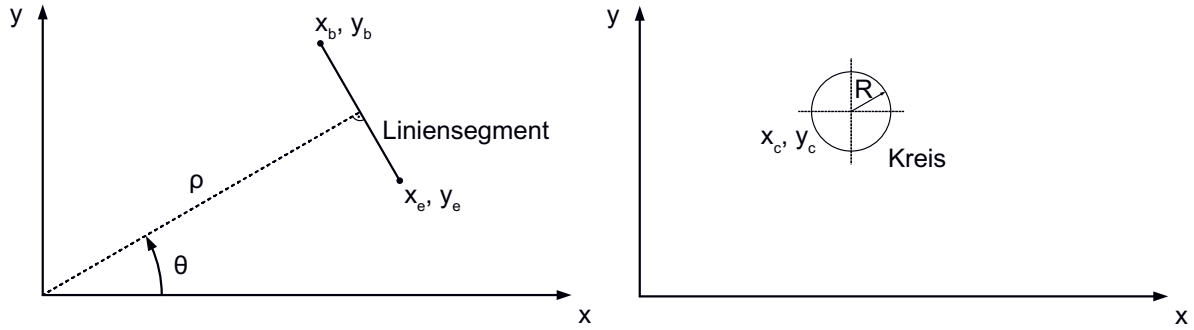


Abbildung 31: Links - Definition eines Liniensegments über die Parameter ρ, θ und die Endpunkte (x_b, y_b) und (x_e, y_e); Rechts - Definition eines Kreises über den Radius R und das Zentrum (x_c, y_c); nach [62]

Kreise werden wie in Abbildung 31 gezeigt über das Kreiszentrum (x_c, y_c) und den Radius R defniert.

Ermittlung der Kovarianzmatrizen

Abgeleitet von den Kovarianzen der Einzelpunkte werden auch für die Linien Kovarianzen für die spätere Zuordnung zu bestehenden Linien berechnet.

Die Berechnung erfolgt in zwei Schritten:

1. Auf Basis der linearen Regression wird die Kovarianzmatrix $\boldsymbol{C}_v$ der ermittelt.
2. Für die Form, beschrieben durch ρ und θ, kann daraus die Kovarianzmatrix $\boldsymbol{C}_l$ abgeleitet werden.

Die Unsicherheit in den Kreisen wird durch die Varianzen der Koordinaten des Kreismittelpunkts σ_x und σ_y abgebildet. Der Radius ergibt sich zu $R = \sqrt{\sigma_x^2 + \sigma_y^2}$.

Zuordnung zu bestehenden Primitiven

Bei der Zuordnung der in den Schritten zuvor generierten Primitive zu bereits in der Karte vorhandenen Primitiven können zwei Fälle auftreten. Entweder es gibt eine Übereinstimmung mit einem vorhandenen Primitiv. Dann werden diese kombiniert, um die Genauigkeit der Karte zu erhöhen. Oder es gibt keine Übereinstimmung und das neue Primitiv wird ohne Korrektur in die Karte übernommen.

Die Übereinstimmung von Linien wird zunächst über Grenzwerte für Winkeldifferenz und Distanz ermittelt. Liegen die Linien hier nah genug beieinander, wird eine normalisierte Distanz zwischen den Kovarianzmatrizen von neuer und vorhandener Linie gebildet. Der Grenzwert wird so gewählt, dass die Chance auf zusammengehörige Linien, die einander nicht zugeordnet werden, kleiner oder gleich 10% ist. Zuletzt wird überprüft, ob der Unterschied zwischen den Distanzen zwischen den Endpunkten der Linien unter dem Grenzwert d_{critll} liegt.

Wird eine Übereinstimmung ermittelt, werden Position und Kovarianzmatrix der vorhandenen Linie über einen statischen Kalman-Filter aktualisiert. Die Endpunkte werden bestimmt, indem die Endpunkte der neuen Linie auf die aktualisierte Linie projiziert werden. Die projizierten Punkte ersetzen die alten Endpunkte, wenn die Linie dadurch länger wird.

Für Kreise muss für eine Zuordnung der Abstand der Mittelpunkte lediglich unter dem Grenzwert d_{critcc} liegen. Im Falle der Zuordnung werden aktualisierter Mittelpunkt und Radius über die Mittelung von neuem und vorhandenem Kreis gebildet.

Löschen veralteter Primitive

Um die Karte auch bei sich verändernder Umwelt stets aktuell zu halten, müssen veraltete Primitive gelöscht werden können. Dafür wurde im Ansatz von [62] eine „Löschform“ entwickelt. Diese Löschform löscht alle Teile der Primitive, die mit der Form überlappen. Ausgehend vom Sensor wird die Form über jede neue Linie gebildet (s. Abbildung 32).

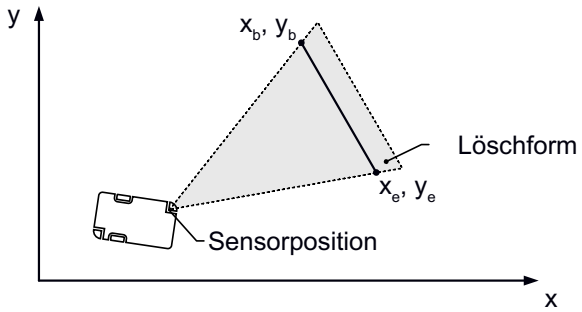


Abbildung 32: Konstruktion der Löschform ausgehend vom Sensor über die Endpunkte jeder Linie

Erweiterungen zur Abbildung dynamischer Objekte und Verbesserung der Löschform

Der Ansatz von Vandorpe et al. [62] ist nur für statische Objekte ausgelegt. Dynamische Objekte erzeugen aufgrund der formalisierten Beschreibung der Primitive immer mehrere Linien hintereinander wie eine Schleppe. Durch eine Erweiterung der Zustandsvektoren und Kovarianzmatrizen um Geschwindigkeiten können zusätzlich Positionsänderungen prädiziert werden.

Linien werden dann durch $(\rho, \theta, \dot{\rho}, \dot{\theta})$ und Kreise durch $(x_c, y_c, \dot{x_c}, \dot{y_c})$ beschrieben. Die Kovarianzmatrizen werden um die entsprechenden Einträge erweitert.

Das ermöglicht im Aktualisierungsschritt anstelle des statische Kalman-Filters einen klassischen Kalman-Filter anzuwenden und damit in der Karte vorhandene Primitive vor der Zuordnung in den aktuellen Zeitschritt zu prädizieren. Im Ergebnis unterbindet dies die Bildung der beschriebenen Schleppen.

Ein weiterer Punkt für Optimierung betrifft die Erzeugung der Löschform. Der ursprünglich von Vandorpe et al. vorgeschlagen Ansatz bedeutet einen hohen Effizienzverlust, da die Form für jede Linie einzeln erzeugt und angewendet wird. Ausgehend von der Lidar-Punktwolke wird daher die Löschform global erstellt. Ähnlich wie bei der Grid-basierten Karte wird davon ausgegangen, dass zwischen Sensor und erkanntem Hindernis kein Hindernis liegen kann. Aus der Außenkontur der Punktwolke kann daher direkt die Löschform abgeleitet werden. Um die Komplexität der Form zu reduzieren, wird zuvor jedoch die Anzahl der Punkte entfernt und zur Vermeidung von Ambiguitäten beim Löschen von Linien werden mithilfe des Douglas-Peucker-Algorithmus fluchtende Punkte entfernt.

In Abbildung 33 wird ein beispielhaftes Ergebnis der Kartierung mit geometrischen Primitiven angewendet auf Daten aus der Simulation gezeigt. Die Karte ist zum aufgenommenen Zeitpunkt noch im Aufbau begriffen. Bereits zu diesem Zeitpunkt können jedoch schon die Konturen der Anlagen deutlich erkannt werden. Außerdem ist im unteren Bereich beim Roboter der sich bewegende Gabelstapler gut in der Karte zu erkennen.

Aufgrund der nicht ausschließlich frontalen Annäherung des Gabelstaplers in Richtung Sensor ist eine leichte Schleppenbildung zu beobachten.

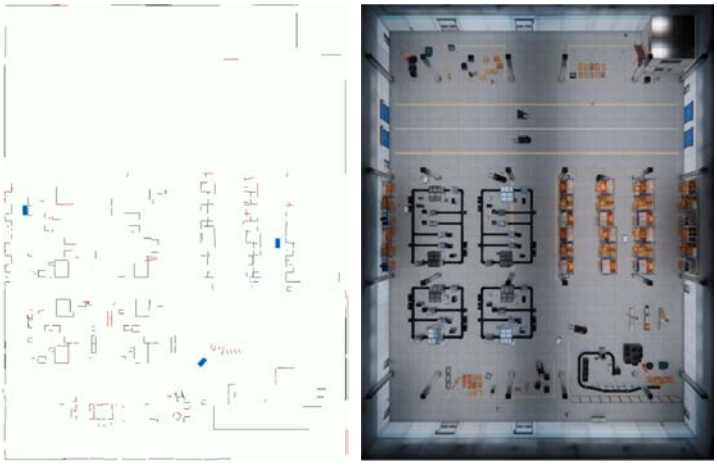


Abbildung 33: Kartierung mit geometrischen Primitiven mit aufzeichnenden Robotern in blau (links), Simulation zum Vergleich (rechts)

4.4 Semantischer Layer

Derzeit verlassen sich mobile Roboter für die Navigation auf eine statische Karte und für die Kollisionsvermeidung auf Sicherheitsfelder der LiDAR-Sensoren. Wird ein Schutzfeld verletzt, bremst der Roboter schrittweise ab und bleibt dann ganz stehen [22, S. 226]. Daher sind komplexere auf die Situation und die Art des Hindernisses abgestimmte Ausweichmanöver nicht möglich.

Mit dem beschriebenen statischen und dynamischen Layer allein können Hindernisse zwar erkannt werden, es ist jedoch nicht möglich zu bestimmen, um welches Art Hindernis es sich handelt und welche weiteren Eigenschaften es zusätzlich zur aktuellen Dynamik besitzt. Um daher weitere Funktionen, wie z. B. situationsabhängige Ausweichmanöver oder eine von vornherein effizientere Routenplanung, zu ermöglichen, müssen semantische Informationen in der Karte verankert werden (vgl. 2.2.4).

Ziel des semantischen Layers ist, die im vorherigen Abschnitt beschriebenen Karten um erkannte Objekte zu erweitern. Dazu müssen zunächst Objekte detektiert, also klassifiziert und lokalisiert, und dann deren Position in das globale Koordinatensystem transformiert werden. Dazu wird im Kamera-Bild mittels eines Deep-Learning-Modells eine Objektdetektion durchgeführt. Die Positionsschätzung der Objekte wird dann mithilfe einer ebenfalls Deep-Learning-basierten semantischen Segmentierung optimiert.

Darüber hinaus soll der durch den Roboter befahrbare Bereich in der Karte angezeigt werden können. Mithilfe der semantischen Segmentierung wird dazu das mögliche Areal bestimmt und in globale Koordinaten transformiert.

Daneben sind eine Vielzahl weiterer Funktionen vorstellbar, wie z. B. die Erfassung der Verbindungsstärke oder die Qualität des Bodens, die jedoch über den Rahmen des Projekts hinausgehen und daher nicht näher untersucht wurden.

4.4.1 Abbildung von Objekten im semantischen Layer

Das Abbilden von Objekten im semantischen Layer umfasst die zwei Schritte Klassifizieren und Lokalisieren. Erst wird bestimmt, um welche Klasse es sich bei einem bestimmten Objekt handelt, dann kann das potenzielle Objekt in der Karte verankert werden.

Da die Informationsdichte der aktuell üblichen 2D-LiDAR-Sensoren in der Regel nicht für eine Objekt-Klassifizierung ausreicht, werden für den hier beschriebenen Ansatz 2D-RGB-Kameras eingesetzt. Für die Feinpositionierung wird später zusätzlich auf Tiefen-Kameras zurückgegriffen.

Nachfolgend wird beschrieben, wie aus den Bildern der 2D-RGB-Kamera erst Objektklassen und In-Bild-Positionen extrahiert werden und wie diese dann anschließend in globale Koordinaten transformiert werden.

Objektdetektion mit Yolov11

Die Grundlagen zur Objektdetektion mittels Deep-Learning wurden bereits in Abschnitt 2.3.2 erläutert. Für dieses Projekt wurde Yolov11 in der Version „l“ als Objektdetektions-Algorithmus mit folgenden Parametern gewählt:

- Bildgröße: 1280 x 720 px
- Batch-Size: 8
- Epochen: 100

Die Bild-Größe entspricht der nativen Auflösung des in der Simulation berechneten Kamerabildes; die Batch-Size wird durch den Speicher der Grafikkarte auf maximal 8 limitiert. Vortests haben ergeben, dass der Algorithmus meist innerhalb von 100 Trainings-Epochen konvergiert. Findet dann keine Konvergenz statt, muss das Training mit anderen Parametern wiederholt werden.

Für das Training wird dem Algorithmus das 2D-RGB-Bild einer Roboter-fixierten Kamera übergeben. Daraus wird eine Liste mit Bounding-Boxen und Klassen bestimmt. Damit ist sowohl die Position von Objekten in Bild-Koordinaten als auch die jeweilige Klasse des Objekts, das durch die Bounding-Box umgrenzt wird (vgl. Abbildung 34), bekannt.

Abbildung 34: Beispielergebnis der Objektdetektion; Bounding-Boxen in verschiedenen Farben für verschiedene Objekte (bspw. Rolltore in lila)

Der Algorithmus erreicht folgende Werte:

- mAP_{50}: 96,80 %
- $mAP_{50\text{-}95}$: 89,07 %

Vergleicht man die Werte mit den auf dem COCO-Datensatz erreichbaren Werten (54,5% $mAP_{50\text{-}95}$ für die größte Version „x“) [63], so fallen sie signifikant höher aus. Das ist unter anderem dadurch erklärbar, dass

die simulierten Bilder erheblich weniger Varianz aufweisen und damit für den Algorithmus besser zu erkennen sind. Auch reale Effekte, wie Sensorrauschen, entfallen und erleichtern damit die Erkennbarkeit. Möglicherweise trägt auch fehlende Generalisierung zu dem Ergebnis bei. Grundsätzlich ist dieser Effekt als unerwünscht einzustufen, da die Übertragbarkeit auf andere Daten verloren geht. Hier handelt es sich jedoch nur um einen Proof of Concept für das Gesamtsystem, weshalb fehlende Generalisierung toleriert werden kann.

Transformation in globale Koordinaten mittels semantischer Segmentierung und Tiefen-Bildern

Anschließend wird mittels des DeepLabv3+-Algorithmus das Bild semantisch segmentiert. Ausgang ist wieder das 2D-RGB-Bild der Roboter-fixierten Kamera. Das Ergebnis ist eine Segmentierungsmaske, die die einzelnen Objekt-Klassen unterscheidet (vgl. Abbildung 35).

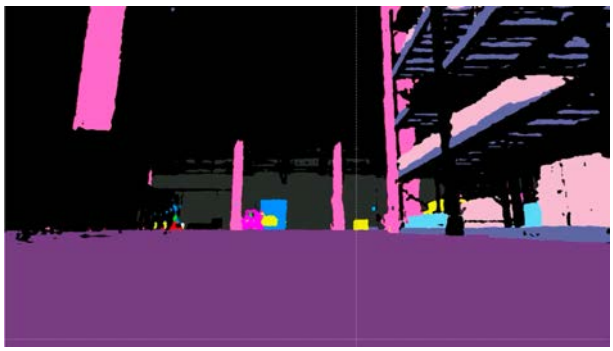


Abbildung 35: Beispielergebnis semantische Segmentierung; Klasse in verschiedenen Farben (Säulen bspw. pink)

Folgende Parameter wurden gewählt:

- Backbone: ResNet-50
- Batch-Size: 8
- Epochen: 15

Damit wurden folgendes Ergebnis erreicht:

- Mean IoU: 97,99 %
- Overall Accuracy: 95,23 %
- Mean Accuracy: 87,50 %

Auf dem PASCAL VOC 2012 Testset erreicht DeepLabv3+ Mean-IoU-Werte von bis zu 89,0 %. Auch hier sind die hohen auf den Simulationsdaten erreichten Werte durch die einfache Struktur der Daten und eine möglicherweise zu geringe Generalisierung zu erklären.

Durch eine Kombination von Bounding-Boxen mit Segmentierung und Tiefen-Bild kann nun die Distanz von einzelnen Objekten bestimmt werden. Die vorangestellte Objektdetektion ermöglicht hier zusätzlich das Unterscheiden von einzelnen Objektinstanzen, was bei reiner semantischer Segmentierung nicht möglich wäre.

Für die Distanzschätzung wird jede Bounding-Box mit dem passenden Ausschnitt der Segmentierungsmaske verglichen. Die Pixel im Ausschnitt, die derselben Klasse wie der der Bounding-Box entsprechen, werden dann über das Bild der Tiefen-Kamera gelegt. Durch eine Mittelung der entsprechenden Pixel im Tiefenbild kann so die Distanz geschätzt werden. Das ist für kleine Objekte hinreichend genau. Zu beachten ist auch, dass bei dieser Methode eine sehr genaue Kalibrierung von 2D-RGB- und Tiefen-Kamera gegeben sein muss. Durch die Verwendung der simulierten Daten entfällt die Kalibrierung. In einer späteren Version soll durch eine

Erweiterung des Neuronalen Netzes die Distanzbestimmung über die Tiefen-Kamera direkt über die Segmentierung im 2D-RGB-Bild erfolgen.

Nachdem die Objekte in der Bildebene mit zugehöriger Distanz bestimmt wurden, müssen diese Werte in globale Koordinaten transformiert werden, um in der Karte dargestellt werden zu können. Dabei wird nur das Zentrum des jeweiligen Objekts betrachtet.

In Anbetracht der begrenzten Projektlaufzeit wurde der Schwerpunkt im Laufe der Bearbeitung jedoch auf die anderen Projektbestandteile gelegt. Auch wenn die Transformation der Objektposen in globale Koordinaten für den Gesamtansatz notwendig ist, so sind für die Erbringung des Proof of Concept Kartenbildung und Objektdetektion bzw. semantische Segmentierung relevanter. Um daher eine gute Qualität der anderen Module zu gewährleisten, wurde auf eine Implementierung der Transformation verzichtet.

In Abbildung 36 ist der gesamte Prozess dargestellt.

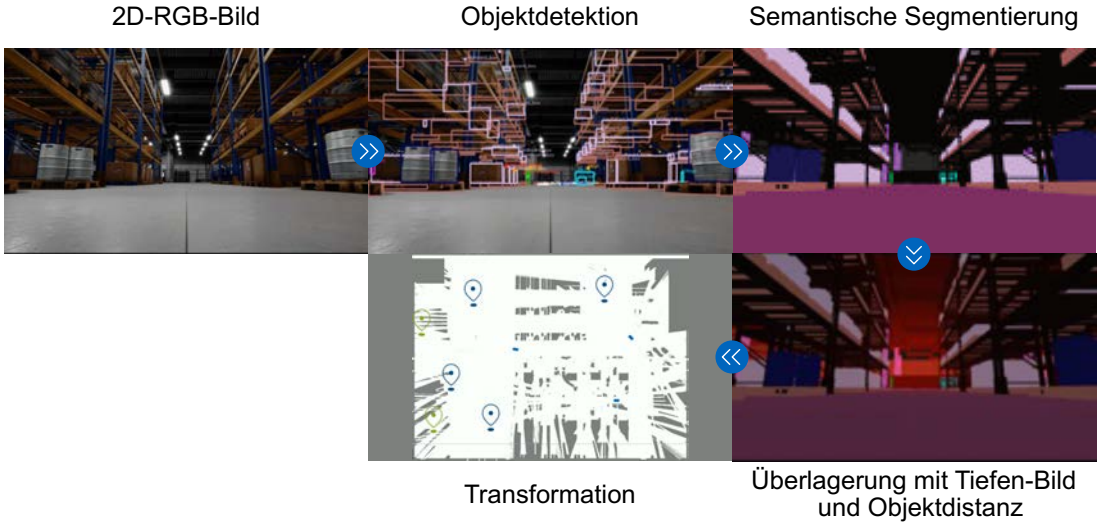


Abbildung 36: Prozess der Objektdistanzschätzung ausgehend vom 2D-RGB-Bild (links) über die Schritte Objektdetektion, semantsiche Segmentierung und Ermittlung der Objektdistanz mittels Tiefen-Bilder und Transformation in globale Koordinaten zu Objekten dargestellt in der Karte (mittig, unten)

4.4.2 Erkennung des befahrbaren Bereichs

Aufbauend auf der bereits vorgestellten semantischen Segmentierung soll zusätzlich eine Kennzeichnung des für mobile Roboter befahrbaren Bereichs in der Karte erfolgen. Zur Bestimmung des befahrbaren Bereichs wird die Klasse „Boden“ herangezogen. Die innerhalb der Klasse liegenden segmentierten Pixel werden dafür in globale Koordinaten transformiert und in den statischen und dynamischen Layer übertragen.

Im Gegensatz zu einer reinen Freiraumerkennung wie bei der Kartierung, können hier weitere Umgebungsmerkmale berücksichtigt werden. Beispielsweise kann der Bereich unter Regalen frei sein und prinzipiell befahren werden. Bei der Routenplanung sollte dieser Bereich dennoch gemieden werden.

Außerdem wäre eine Präzisierung über Markierungen am Boden in der Klasse „Decal_floor“ denkbar. Dadurch könnten bestimmte Bereiche durch Anbringen von Bodenmarkierungen ausgeschlossen werden, ohne dass diese vorab händisch in die Karte eingetragen werden müssen.

Da es sich um eine Zusatzfunktion handelt, die für den reinen Proof of Concept nicht notwendig ist, wurden die anderen Projektteile priorisiert.

4.5 Fusion und Visualisierung

Zuletzt werden die einzelnen Bestandteile zusammengefügt:

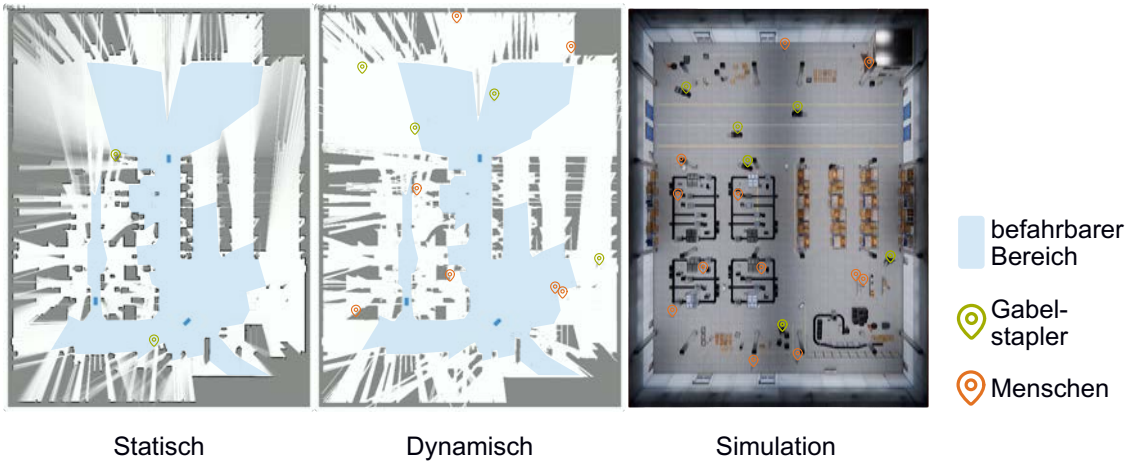


Abbildung 37: Mock-Up der Fusion der einzelnen Bestandteile zu einem einheitlichen Modell der Umgebung; Statische und dynamische Karte links/mittig; Objekte werden je nach Klasse durch Pfeile in anderer Farbe dargestellt (Beispiel: Gabelstapler, Menschen); durch mobile Roboter befahrbarer Bereich in hellblau; zum Vergleich: Simulation (rechts)

Die zusätzlichen semantischen Informationen werden entsprechend ihrer Dynamik in statische oder dynamische Karte übertragen. Dadurch liegt ein einheitliches Modell vor, das die Umgebung repräsentiert und von allen Robotern aktualisiert werden kann.

Bei der Darstellung der Objekte in der Karte handelt es sich zum Zeitpunkt des Berichts nur um ein Mock-Up, da der letzte Schritt der Transformation von erkannten Objekten in globale Koordinaten aufgrund der Projektprioritäten nicht umgesetzt wurde.

5 Fazit und Ausblick

Ziel des Forschungsprojekts KolUmBot war die prototypische Entwicklung eines kollektiv erstellten und lesbaren Umgebungsmodells für mobile Roboter. Dazu wurde das Gesamtsystem in die Funktionsmodule Datengenerierung, Datentransfer, statischer / dynamischer Layer, semantischer Layer, Fusion und Visualisierung aufgeteilt.

Insbesondere der Bereich Datengenerierung beinhaltete unvorhergesehene Herausforderungen und fiel daher im Vergleich zur ursprünglichen Planung erheblich umfangreicher aus. Nachdem die Sensordaten umgebungserfassender Sensoren mobiler Roboter in der Regel nicht ohne weiteres verfügbar sind, mussten diese für die weitere Entwicklung zunächst simulativ erzeugt werden. Gründe dafür sind z. B. der rechtliche Rahmen beim Datenschutz oder unklare Eigentums- bzw. Nutzungsrechte zwischen Roboterhersteller und -nutzer.

Nach einer Ziel- und Anforderungsdefinition wurde zunächst eine geeignete Simulationsumgebung gesucht. Auf Basis der ermittelten Anforderungen fiel die Wahl auf die Game-Engine Unity, da diese einen guten Kompromiss aus Flexibilität und durch verschiedene Erweiterungen bereits vorimplementierten Bausteinen für die Simulation von Sensordaten bietet. In Unity wurde dann ein für die Anwendung möglichst vielseitig nutzbares Modell aufgebaut. Dazu wurde eine Umgebung entwickelt, die viele gängige Bestandteile einer Logistik- und Industrieumgebung enthält. Diese wurden anhand von Expertenwissen und Literaturrecherche ermittelt. Außerdem wurde ein Robotermodell entworfen, das real eingesetzten Robotern möglichst nahekommen soll. Die Wahl fiel daher auf die Bauform des Unterfahr-Roboters, wobei Dimensionen und Sensorauswahl bzw. -positionierung auf Basis einer umfassenden Marktrecherche ermittelt wurden. Wichtig war dabei, dass nicht nur aktuell gängige Sensoren abgebildet wurden (z. B. 2D-LiDAR), sondern auch derzeit noch wenig verbreitete, aber zukünftig relevante Sensoren, wie beispielsweise 2D-RGB-Kamera. Damit soll die Zukunftsfähigkeit der Daten, aber auch der darauf aufbauenden Algorithmen sichergestellt werden. Um die Daten für die Algorithmen-Entwicklung nutzbar zu machen, wurde ein automatisiertes Labeling implementiert, das die Erstellung eines Datensets inklusive Ground-Truth für Supervised-Learning-Algorithmen ermöglicht. Hierbei zeigte sich, dass sich durch verschiedene Optimierungen die mit dem Datenset erzielbaren Ergebnisse noch weiter verbessern ließen. Daher wurden gezielt Maßnahmen ergriffen, um bspw. die Verteilung der auftretenden Klassen zu homogenisieren. Aufgrund der geringen Performance der gleichzeitigen Simulation mehrerer Roboter wurde auf eine Echtzeit-Schnittstelle verzichtet.

Durch die Simulationsdaten war eine tiefergehende praktische Betrachtung der Datenübertragung und deren Mechanismen nicht mehr erforderlich. Daher wurde die Umsetzbarkeit vornehmlich theoretisch betrachtet. Die drei Teilbereiche Übertragungstechnologie, Synchronisierung, Queuing wurden als besonders kritisch eingestuft und daher näher untersucht. Dabei wurde festgestellt, dass sich für die große Menge der zu übertragenden Umgebungsdaten in Kombination mit der erforderlichen Reichweite 3GPP 5G und Wi-Fi 6 (IEEE 802.11ax) eignen. Damit die Daten in das Modell in der richtigen Reihenfolge eingespeist werden, müssen sie bei Ankunft auf dem Server nach Zeit und Ort synchronisiert werden. Andernfalls können Inkonsistenzen auftreten. Für das notwendige Zwischenspeichern der Daten sollte ein Queuing-System eingesetzt werden.

Auf Basis der generierten Daten wurde dann ein Funktionsmodell für das kollektive Umgebungsinformationssystem entwickelt. Für statischen und dynamischen Layer wurden zwei potenzielle Algorithmen näher untersucht. Der erste basiert auf einem Occupancy-Grid, speichert also für jede Gitterzelle einen Wahrscheinlichkeitswert. Im Unterschied zu gängigen Verfahren, werden beim gewählten Ansatz zwei Karten gleichzeitig

geführt, wovon eine für statische Objekte und eine für dynamische Objekte optimiert ist. Diese Form der Umgebungsrepräsentation erfordert die Speicherung große Datenmengen und häufige Aktualisierungen bei neuen Messungen. Um diesen Herausforderungen zu begegnen, wurde ein zweiter Ansatz untersucht, der die Umgebung über geometrische Primitive (Linien und Kreise) abbildet. Dafür müssen nicht mehr alle Gitterzellen gespeichert und aktualisiert werden, was die Performanz erhöhen kann.

Da metrische Informationen allein kein umfassendes Bild der Umgebung ermöglichen, wurde ein semantischer Layer eingeführt. Dieser enthält vornehmlich Informationen zu Objektklassen. Wird beispielsweise ein Gabelstapler identifiziert, wird diese Information in dem darunterliegenden dynamischen Layer verankert, um z. B. die Pfadplanung unter Berücksichtigung dieser Zusatzinformation zu optimieren. Die Detektion von Objekten wird mittels des Deep-Learning-Algorithmus YOLO11 auf simulierten Kamerabildern durchgeführt. Die guten Ergebnisse der Detektion machen den Einsatz dieser Technik vielversprechend. Anschließend wurde mittels des DeepLabv3+-Algorithmus eine pixelweise semantische Segmentierung durchgeführt. In Kombination mit Bildern der Tiefen-Kamera kann so eine gute Schätzung der Distanz von Objekten erfolgen. Nach einer Transformation aus der Bild-Ebene zusammen mit der Distanzschätzung in globale Koordinaten können die Objekte in den statischen und dynamischen Layer übertragen und für nachfolgende Prozesse genutzt werden. Die Transformation wurde bis zum Ende der Projektlaufzeit nicht umgesetzt und daher nur als Mock-Up gezeigt. Bei der Bewertung der Ergebnisse sollte außerdem berücksichtigt werden, dass alle zugrunde liegenden Daten simuliert sind. Eine Übertragbarkeit auf Realdaten ist daher nicht ohne zusätzlichen Aufwand möglich.

Insgesamt wurde das kollektive Umgebungsinformationssystem für mobile Roboter prototypisch umgesetzt. Eine Evaluation erfolgte nur auf Basis simulierter Daten nicht jedoch im produktiven Einsatz.

5.1 Abgleich von Zielen / Anforderungen mit den Ergebnissen

Ein Abgleich der eingangs aufgestellten Ziele und Anforderungen mit den erreichten Projektergebnissen zeigt eine überwiegende Erfüllung:

- *Definition Bestandteile*
  Die Bestandteile des Umgebungsinformationssystems wurden klar definiert.
- *Semantik*
  Das Umgebungsmodell kann neben metrischen auch semantische Informationen abbilden.
- *Datentransfer*
  Es wurde eine Strategie zur Datenübertragung vorgestellt. In Abhängigkeit der konkreten Gegebenheiten sollte vor allem auf die Wahl der Übertragungstechnologie, Synchronisierung und Queuing geachtet werden.
- *Zugriffskonzept*
  Durch das beim Datentransfer beschriebene Queuing kann einem Verlust von Daten durch Überlagerung wirkungsvoll begegnet werden.
- *Größe des Umgebungsmodells*
  Aktuell ist die Größe des Modells stark von den vorhandenen Ressourcen des Server-Systems abhängig. Große Umgebungen können daher entweder nicht oder nur mit größeren Verzögerungen berechnet werden. Um dem entgegenzuwirken könnte eine Kombination von topologischen mit metrischen Karten in Betracht gezogen werden.

- *Umgang mit verschiedenen Sensormodalitäten*
  Das Modell nutzt 2D-LiDAR, 2D-RGB-Kamera und Tiefen-Kamera für die Generierung von metrischen und semantischen Informationen.
- *Nachhaltige Betrachtung von Sensoren*
  Zusätzlich zu aktuell häufig eingesetzten 2D-LiDAR-Sensoren, nutzt das System 2D-RGB-Kameras, um Objekterkennung und semantische Segmentierung umzusetzen. Allerdings fehlt die Umsetzung von 3D-LiDAR-Sensoren, die dem Modell eine dritte Dimension hinzufügen könnten.
- *Abdecken relevanter Situationen*
  Der entwickelte Datensatz enthält verschiedene relevante Situationen, wie z. B. verteilte mobile Roboter oder enge Gänge mit schlechter Sicht. Diese werden jedoch nicht explizit ausgewiesen und sind lediglich in den Daten enthalten. Spezielle Datensätze, um bestimmte Situationen abzutesten, sollten zukünftig abgeleitet werden.

5.2 Ausblick

Einige Funktionen konnten bis zum Ende der Projektlaufzeit nicht umgesetzt werden. So fehlt der letzte Schritt der Transformation ermittelter semantischer Daten. Außerdem fand keine Anwendung auf reale Daten statt. Gerade für die Nutzung in produktiven Umgebungen ist es jedoch unabdingbar, das System zunächst mit Realdaten umzusetzen. Dabei zeigt die Praxis, dass insbesondere die Übertragbarkeit von auf simulierten Daten trainierten Deep-Learning-Algorithmen, wie sie hier eingesetzt wurden, auf Realdaten häufig Nacharbeiten in erheblichem Umfang bedeuten. Meist sind neue Datensets und ein neues Training der Algorithmen erforderlich.

Darüber hinaus gibt es Möglichkeiten, das System weiter zu verbessern oder um zusätzliche Funktionen zu erweitern. In einer nächsten Version könnte bei der Segmentierung durch eine Erweiterung des Algorithmus bereits direkt eine Distanzschätzung auf Basis des 2D-Bilds erfolgen. Außerdem sollen detektierten Objekten Eigenschaften zugewiesen werden. Beispielsweise kann die erwartete Dynamik eines Objekts in späteren Prozessschritten, wie z. B. der Pfadplanung, für genauere Ergebnisse genutzt werden. Denkbar wäre auch eine typische Höhe zu betrachten, um daraus eine 2.5D-Karte zu erstellen. Zusätzlich können weitere Metainformationen im semantischen Layer gespeichert werden. Beispielsweise könnten die Bodenbeschaffenheit oder die Verbindungsstärke zum nächsten Access Point gespeichert werden. Diese Informationen können Aufschluss über Optimierungspotenziale für die Industrieumgebung bieten (z. B. Entfernung von Bodenunebenheiten oder Einrichtung zusätzlicher Access Points), aber auch zur Optimierung des Gesamtprozesses herangezogen werden. Da große Umgebungen viel Speicher und Rechenleistung für die Aktualisierung der Karte benötigen, kann das System in diesen Fällen an seine Grenzen stoßen. Es wäre daher von Interesse, einen effizienteren Kartierungs-Algorithmus einzusetzen, der z. B. topologische mit metrischen Karten verbindet, um Aktualisierungen zu beschleunigen. Außerdem sollten einzelne Datensätze für bestimmte Situationen erstellt werden, um diese gezielt abprüfen zu können.

6 Ressourcenverwendung und Ergebnistransfer

6.1 Verwendung der Zuwendung

6.1.1 Wissenschaftlich-technisches Personal und studentische Hilfskräfte

		Jahr 1				Jahr 2				Jahr 3
Arbeitspakete		Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1
AP 1	Anforderungsdefinition, Analyse des Stands der Technik und Forschung	3								
AP 2	Konzeption des kollektiven Informationssystems für mobile Roboter		3							
AP 3	Entwicklung des Datenmodells zur Zusammenführung der Umgebungsdaten inkl. zusätzlich Entwicklung Simulationsumgebung			3	3	2				
AP 4	Entwicklung von Strategien für das Management der Ein- und Ausgangsdaten im Modell					1	3			
AP 5	Demonstratorische Umsetzung des Informationssystems							3	2	
AP 6	Evaluation des Informationssystems									1
AP 7	Projektmanagement und Transfer der Ergebnisse in die Praxis					2				
										Summe 26

Abbildung 38: Arbeitspakete und Verteilung auf die einzelnen Monate

In Summe wurde ein wissenschaftlicher Mitarbeiter für 26 Monate eingesetzt. Wissenschaftliche Hilfskräfte wurden nach Bedarf eingesetzt.

6.1.2 Geräte (Einzelansatz B des Finanzierungsplans)

Es wurden keine Finanzmittel für Gerätebeschaffungen beantragt.

6.1.3 Leistungen Dritter (Einzelansatz C des Finanzierungsplans)

Es wurden keine Leistungen für die Arbeit Dritter beantragt.

6.2 Notwendigkeit und Angemessenheit der geleisteten Arbeit

Ziel des Forschungsvorhabens war die Entwicklung eines kollektiven Umgebungsinformationssystems für mobile Roboter. Für einen effizienten Betrieb von Robotern ist eine möglichst umfassende Kenntnis der Umgebung ein entscheidender Faktor. Der Ansatz eines gemeinsam aktualisierbaren Modells der Umgebung an zentraler Stelle im Gesamtsystem wurde bisher nicht umgesetzt. Parallel zur Entwicklung eines Funktionsdemonstrators wurden das Vorgehen und die im Umgebungsmodell enthaltenen Informationen in Expertengesprächen validiert und die Praxisnähe kontinuierlich sichergestellt.

Um die nötige wissenschaftliche Gründlichkeit bei der Bearbeitung zu garantieren, mussten wissenschaftliche Mitarbeiter eingesetzt werden. Die Arbeiten wurden entsprechend der Notwendigkeit und ihres Umfangs von den eingesetzten wissenschaftlichen Mitarbeitern durchgeführt. Zusätzlich wurden die Arbeiten von studentischen Hilfskräften und den Beteiligten des projektbegleitenden Ausschusses unterstützt.

6.3 Darstellung des wissenschaftlich-technischen und wirtschaftlichen Nutzens der erzielten Ergebnisse insbesondere für KMU sowie ihres innovativen Beitrags und ihrer industriellen Anwendungsmöglichkeiten

Das Umgebungsinformationssystem kann die Performanz einer Flotte mobiler Roboter durch intensiveren Austausch von Daten erhöhen. Das gemeinsam gepflegte Modell der Umgebung kann präzisere Routenplanung oder auf die Situation abgestimmte Ausweichmanöver ermöglichen. Das erhöht die Produktivität und ermöglicht auch KMU einen effizienteren Betrieb ihrer Roboterflotten. Die Aufnahme weiterer Informationen, wie z. B. der Verbindungsqualität zum nächsten Access Point, erlaubt außerdem eine Optimierung der Infrastruktur. Darüber hinaus können kritische Situationen global erkannt und möglicherweise vermieden werden, was die Betriebssicherheit verbessert.

Besonders KMU haben oft keine Kapazitäten größere Simulationen aufzubauen. Die im Rahmen des Projekts entwickelten Simulationsdaten zu Umgebungssensoren sollen nach Abschluss des Projekts veröffentlicht und auch der Industrie zur Verfügung gestellt werden. Damit haben auch KMU die Möglichkeit, die Daten für Entwicklungs- und/oder Testzwecke zu nutzen.

Außerdem wurde durch die Einführung des semantischen Layers Möglichkeiten aufgezeigt, in welcher Form semantische Informationen zukünftig in ein Umgebungsmodell integriert werden können. Das eröffnet Chancen für die Entwicklung neuer Teilfunktionen oder ganzer Produkte.

6.4 Wissenstransfer in die Wirtschaft

Während der Projektlaufzeit wurden verschiedene Maßnahmen zum Transfer der Zwischen- und Endergebnisse in die Wirtschaft durchgeführt. Sie sind in Tabelle 10 aufgelistet.

Tabelle 10: Durchgeführte Maßnahmen zum Transfer der Projektergebnisse in die Wirtschaft während der Projektlaufzeit

Ziel	Rahmen		Zeitraum
Maßnahme A: Projektbegleitender Ausschuss			
Fortlaufende Diskussion und Abstimmung des Forschungsfortschrittes mit Unternehmen des projektbegleitenden Ausschusses	**A1**	Vorstellung des Projekts und Diskussion der geplanten Arbeiten	28.02.2023
	A2	Vorstellung des entwickelten Datenmodells	20.11.2023
	A3	Vorstellung der Strategien zur Bearbeitung der Ein- und Ausgangsdaten und Ausblick auf das Informationssystem	06.11.2024
	A4	Vorstellung der Projektergebnisse und der demonstratorischen Anwendung	26.02.2025
Maßnahme B: Präsentation auf Kongressen und Konferenzen			
Präsentation von (Teil-) Ergebnissen des Projekts auf Fachtagungen vor Wirtschaft und Wissenschaft	**B1**	Präsentation des Datenmodells auf einer Konferenz mit Logistikbezug (bspw. WGTL-Kolloquium)	In Planung
	B2	Präsentation des Demonstrators auf einer Konferenz mit Robotikbezug (bspw. IEEE International Conference on Robotics and Automation)	In Planung

Maßnahme C: Elektronischer Newsletter und Internetauftritt

Elektronische Verbreitung der Forschungsinhalte und -ergebnisse, Gewinn weiterer interessierter Unternehmen	**C1**	Das Projekt wird auf den Kanälen der sozialen Medien (bspw. LinkedIn) vorgestellt.	1. Quartal
	C2	Der aktuelle Projektfortschritt wird regelmäßig über die sozialen Medien (bspw. LinkedIn) veröffentlicht.	5. Quartal 8. Quartal
	C3	Frei zugänglicher Internetauftritt des Forschungsprojekts über die Homepage des fml.	Ab 1. Quartal fortlaufend
Maßnahme D: Veröffentlichungen			
Ergebnistransfer in die Wirtschaft	**D1**	Vorstellung des Projekts in der Fachzeitschrift „Technische Logistik, Hebezeuge Fördermittel“	13.06.2023
	D2	Publikation der demonstratorischen Umsetzung in der Fachzeitschrift „ZWF Zeitschrift für wirtschaftlichen Fabrikbetrieb“ (oder vergleichbar) im Umfang von ca. 8 Seiten	In Planung
Maßnahme E: Übernahme in die Lehre			
Einbringung der erarbeiteten Ergebnisse in den Lehrbetrieb	**E1**	Mitarbeit studentischer Hilfskräfte	Gesamte Projektlaufzeit
	E2	Anfertigung von Studienarbeiten im Rahmen des Projekts	Gesamte Projektlaufzeit
Maßnahme F: MFK – Deutscher Materialflusskongress			
Ergebnistransfer in die Wirtschaft1	**F1**	Vortrag und Vorführung im Rahmen des an der Forschungsstelle jährlich stattfindenden Materialflusskongresses	Vorstellung im Rahmen eines Industrie-abends am Lehrstuhl

6.5 Geplante spezifische Transfermaßnahmen nach der Projektlaufzeit

Die Verbreitung der Forschungsergebnisse wird auch nach Projektende fortgeführt. Die dafür geplanten Maßnahmen sind in Tabelle 11 aufgelistet.

Tabelle 11: Geplante Maßnahmen zum Transfer der Forschungsergebnisse in die Wirtschaft nach Ende der Projektlaufzeit

Ziel	Rahmen		Zeitraum
Maßnahme G: Beratung			
Ergebnistransfer an Unternehmen ohne eigene Forschungsaktivität	**H1**	Unterstützung von Firmen bei der eigenständigen Anwendung der Ergebnisse durch das Angebot von Inhouse-Seminaren oder Beratungsleistungen	Ab Quartal III 2025
Maßnahme H: Veröffentlichungen			
Ergebnistransfer in die Wirtschaft	**I1**	Veröffentlichung des Abschlussberichts auf der Homepage des Lehrstuhls fml	Quartal III 2025
	I2	Vorstellung der Evaluationsergebnisse in der Fachzeitschrift „FTS-World“ (oder vergleichbar) im Umfang von ca. 6 Seiten	Quartal III 2025
	I3	Vorstellung des Demonstrators bei Lehrstuhlbesichtigungen und öffentlichen Workshops der Wirtschaft	Ab Quartal III 2025
Maßnahme I: Konzepttransfer			
Möglichkeit der Nutzung der Ergebnisse	**J1**	Das entwickelte Konzept inkl. Demonstrator steht allen Interessierten diskriminierungsfrei zur Begutachtung bereit. Die geplanten Veröffentlichungen während und nach der Projektlaufzeit werden für die Bewerbung des Demonstrators ebenso genutzt wie die an der Forschungsstelle regelmäßig stattfindenden Veranstaltungen, darunter bspw. Show-Cases für Unternehmen und Veranstaltungen im Rahmen des Mittelstands 4.0-Kompetenzzentrums Augsburg.	Ab Quartal III 2025

6.6 Veröffentlichungen

Im Rahmen des Projekts wurde die folgende Veröffentlichung angefertigt:

Zeitschriftenartikel „Im Kollektiv zu genaueren Umgebungsmodellen“ in „Technische Logistik, Hebezeuge Fördermittel“ in der Ausgabe 06/23

Für die weitere Verbreitung der Projektergebnisse ist eine Journal-Veröffentlichung für das Quartal III 2025 in Erstellung.

6.7 Studienarbeiten

Im Rahmen des Projekts wurden neun Studienarbeiten durchgeführt (s. Tabelle 12).

Tabelle 12: Im Rahmen des Projekts durchgeführte Studienarbeiten

Name	Typ	Titel	Zeitraum
Eren Yildirim	Bachelorarbeit	Konzeptionierung eines Kartenmodells zur Anwendung in der kollaborativen Robotik	01.11.2022-28.04.2023
Jonas Wagner	Semesterarbeit	Entwicklung einer Simulationsumgebung für mobile Roboter in der Intralogistik	01.07.2023-31.12.2023
Junze Zhou	Semesterarbeit	Simulation of Sensors for Environment Perception in Unity	03.07.2023-
Jerome Schneider	Semesterarbeit	Simulation mobiler Logistik-Roboter in Unity	16.10.2023-16.04.2024
Théo Karst	Interdisziplinäres Projekt	Kartierung von dynamischen Objekten durch mobile Roboter in der Intralogistik	18.03.2024-17.09.2024
Konstantin Zeck	Bachelorarbeit	Objekterkennung im Kontext der Lagerlogistik: Ein Vergleich aktueller Modelle	01.04.2024-31.08.2024
Ahmed Ammari	Bachelorarbeit	Entwicklung einer Methodik für die Sensordaten-Simulation im industriellen Umfeld	01.06.2024-02.12.2024
Jonas Nebl	Semesterarbeit	Topologisches Tracking von Menschen mit einer AMR-Flotte	23.08.2024-17.01.2025
Fadi Eshak	Interdisziplinäres Projekt	Semantic segmentation of simulated data in an industrial context	14.10.2024-17.03.2025

7 Literaturverzeichnis

[1] Sandra Marković, „Robotics: market data & analysis“, Statista, Jan. 2024. Zugegriffen: 20. März 2025. [Online]. Verfügbar unter: https://www.statista.com/study/116601/robotics-market-data-analysis-and-forecasts/

[2] „Size of the European market for autonomous mobile robots (AMR) from 2016 to 2021, with a forecast through 2028“, Inkwood Research, Graph, Nov. 2022. Zugegriffen: 20. März 2025. [Online]. Verfügbar unter: https://www.statista.com/statistics/1285864/autonomous-robots-market-size-europe/

[3] „Kompatibilität von Fahrerlosen Transportsystemen (FTS) Leitsteuerung für FTS“, Beuth Verlag GmbH, Berlin, Okt. 2005. [Online]. Verfügbar unter: https://www.beuth.de/de/technische-regel/vdi-4451-blatt-7/78321538

[4] R. T. Silva, M. Brilhante, H. Sobreira, D. Matos, und P. Costa, „AGVs vs AMRs: A Comparative Study of Fleet Performance and Flexibility“, in *2024 7th Iberian Robotics Conference (ROBOT)*, Madrid, Spain: IEEE, Nov. 2024, S. 1–6. doi: 10.1109/ROBOT61475.2024.10797381.

[5] *VDA 5050 - Interface for the communication between automated guided vehicles (AGV) and a master control*, Berlin., 2025.

[6] B. R. Kiran *u. a.*, „Deep Reinforcement Learning for Autonomous Driving: A Survey“, *IEEE Trans. Intell. Transp. Syst.*, Bd. 23, Nr. 6, S. 4909–4926, Jan. 2021, doi: 10.1109/TITS.2021.3054625.

[7] E. Datteri und V. Schiaffonati, „Robotic Simulations, Simulations of Robots“, *Minds Mach.*, Bd. 29, Nr. 1, S. 109–125, März 2019, doi: 10.1007/s11023-019-09490-x.

[8] M. Santos Pessoa De Melo, J. Gomes Da Silva Neto, P. Jorge Lima Da Silva, J. M. X. Natario Teixeira, und V. Teichrieb, „Analysis and Comparison of Robotics 3D Simulators“, in *2019 21st Symposium on Virtual and Augmented Reality (SVR)*, Rio de Janeiro, Brazil: IEEE, Okt. 2019, S. 242–251. doi: 10.1109/SVR.2019.00049.

[9] H. Choi *u. a.*, „On the use of simulation in robotics: Opportunities, challenges, and suggestions for moving forward“, *Proc. Natl. Acad. Sci.*, Bd. 118, Nr. 1, S. e1907856118, Jan. 2021, doi: 10.1073/pnas.1907856118.

[10] J. Collins, R. Brown, J. Leitner, und D. Howard, „Traversing the Reality Gap via Simulator Tuning“, 3. März 2020, *arXiv*: arXiv:2003.01369. doi: 10.48550/arXiv.2003.01369.

[11] E. Todorov, T. Erez, und Y. Tassa, „MuJoCo: A physics engine for model-based control“, in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, S. 5026–5033. doi: 10.1109/IROS.2012.6386109.

[12] E. Coumans und Y. Bai, „PyBullet, a Python module for physics simulation for games, robotics and machine learning“. 2016. [Online]. Verfügbar unter: http://pybullet.org

[13] J. Li und D. J. Cappelleri, „Sim-Grasp: Learning 6-DOF Grasp Policies for Cluttered Environments Using a Synthetic Benchmark“, *IEEE Robot. Autom. Lett.*, Bd. 9, Nr. 9, S. 7645–7652, Sep. 2024, doi: 10.1109/lra.2024.3430712.

[14] B. O. Community, *Blender - a 3D modelling and rendering package*. Stichting Blender Foundation, Amsterdam: Blender Foundation, 2018. [Online]. Verfügbar unter: http://www.blender.org

[15] „Gazebo“. Zugegriffen: 2. Mai 2025. [Online]. Verfügbar unter: https://gazebosim.org/home

[16] „Isaac Sim“, NVIDIA Developer. Zugegriffen: 2. Mai 2025. [Online]. Verfügbar unter: https://developer.nvidia.com/isaac/sim

[17] Unity Technologies, *Unity Engine*. (2023). [Online]. Verfügbar unter: https://unity.com/

[18] Epic Games, *Unreal Engine*. (2019). [Online]. Verfügbar unter: https://www.unrealengine.com

[19] J. Collins, S. Chand, A. Vanderkop, und D. Howard, „A Review of Physics Simulators for Robotic Applications“, *IEEE Access*, Bd. 9, S. 51416–51431, Jan. 2021, doi: 10.1109/ACCESS.2021.3068769.

[20] M. Ten Hompel, T. Schmidt, und J. Dregger, *Materialflusssysteme: Förder- und Lagertechnik*. in VDI-Buch. Berlin, Heidelberg: Springer Berlin Heidelberg, 2018. doi: 10.1007/978-3-662-56181-2.

[21] P. Rawal, M. Sompura, und W. Hintze, „Synthetic Data Generation for Bridging Sim2Real Gap in a Production Environment“, 10. Mai 2024, *arXiv*: arXiv:2311.11039. doi: 10.48550/arXiv.2311.11039.

[22] G. Ullrich und T. Albrecht, *Fahrerlose Transportsysteme: Eine Fibel - mit Praxisanwendungen - zur Technik - für die Planung*, 3., Vollständig überarbeitete Auflage. Wiesbaden; Heidelberg: Springer Vieweg, 2019.
[23] T. Lackner, J. Hermann, C. Kuhn, und D. Palm, „Review of autonomous mobile robots in intralogistics: state-of-the-art, limitations and research gaps“, *Procedia CIRP*, Bd. 130, S. 930–935, 2024, doi: 10.1016/j.procir.2024.10.187.
[24] M. Haun, *Handbuch Robotik: Programmieren und Einsatz intelligenter Roboter*, 2. Aufl. in VDI-Buch. Berlin; Heidelberg: Springer Vieweg, 2013.
[25] M. Ben-Ari und F. Mondada, *Elements of Robotics*. Cham: Springer International Publishing, 2018. doi: 10.1007/978-3-319-62533-1.
[26] *DIN EN ISO 13849-1:2023-12, Sicherheit von Maschinen - Sicherheitsbezogene Teile von Steuerungen - Teil 1: Allgemeine Gestaltungsleitsätze (ISO 13849-1:2023); Deutsche Fassung EN ISO 13849-1:2023*. doi: 10.31030/3435657.
[27] *DIN EN ISO 3691-4:2023-12, Flurförderzeuge - Sicherheitstechnische Anforderungen und Verifizierung - Teil 4: Fahrerlose Flurförderzeuge und ihre Systeme (ISO 3691-4:2023); Deutsche Fassung EN ISO 3691-4:2023*. doi: 10.31030/3386408.
[28] Unity Technologies, „Unity Perception Package“. 2020. [Online]. Verfügbar unter: https://github.com/Unity-Technologies/com.unity.perception
[29] „Synthetic Optimized Labeled Objects (SOLO) Dataset Schema | Perception Package | 1.0.0-preview.1“. Zugegriffen: 11. April 2025. [Online]. Verfügbar unter: https://docs.unity3d.com/Packages/com.unity.perception@1.0/manual/Schema/SoloSchema.html
[30] „Synthetic Optimized Labeled Objects (SOLO) Dataset Schema | Perception Package | 1.0.0-preview.1“. Zugegriffen: 21. Mai 2025. [Online]. Verfügbar unter: https://docs.unity3d.com/Packages/com.unity.perception@1.0/manual/Schema/SoloSchema.html
[31] S. Thrun, „Robotic mapping: a survey“, in *Exploring Artificial Intelligence in the New Millennium*, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2003, S. 1–35.
[32] X. Han, S. Li, X. Wang, und W. Zhou, „Semantic Mapping for Mobile Robots in Indoor Scenes: A Survey“, *Information*, Bd. 12, Nr. 2, S. 92, Feb. 2021, doi: 10.3390/info12020092.
[33] H. Moravec und A. Elfes, „High resolution maps from wide angle sonar“, in *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, Bd. 2, 1985, S. 116–121. doi: 10.1109/ROBOT.1985.1087316.
[34] S. Thrun und A. Bü, „Integrating grid-based and topological maps for mobile robot navigation“, in *Proceedings of the Thirteenth National Conference on Artificial Intelligence - Volume 2*, in AAAI'96. Portland, Oregon: AAAI Press, 1996, S. 944–950.
[35] C. Gomez *u. a.*, „Hybrid Topological and 3D Dense Mapping through Autonomous Exploration for Large Indoor Environments“, in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, Paris, France: IEEE, Mai 2020, S. 9673–9679. doi: 10.1109/ICRA40945.2020.9197226.
[36] J. Crespo, J. C. Castillo, O. M. Mozos, und R. Barber, „Semantic Information for Robot Navigation: A Survey“, *Appl. Sci.*, Bd. 10, Nr. 2, S. 497, Jan. 2020, doi: 10.3390/app10020497.
[37] G. Rebala, A. Ravi, und S. Churiwala, *An Introduction to Machine Learning*. Cham: Springer International Publishing, 2019. doi: 10.1007/978-3-030-15729-6.
[38] R. Szeliski, *Computer Vision: Algorithms and Applications*. in Texts in Computer Science. Cham: Springer International Publishing, 2022. doi: 10.1007/978-3-030-34372-9.
[39] A. Torralba, P. Isola, und W. T. Freeman, *Foundations of Computer Vision*, Null., Bd. null. in null, no. null, vol. null. Cambridge, Massachusetts: The MIT Press, 2024. [Online]. Verfügbar unter: https://research.ebsco.com/linkprocessor/plink?id=09d07d2a-08fd-3fa8-8e2b-5fdfbaf6ae73
[40] T.-Y. Lin *u. a.*, „Microsoft COCO: Common Objects in Context“, 21. Februar 2015, *arXiv*: arXiv:1405.0312. doi: 10.48550/arXiv.1405.0312.
[41] J. Redmon, S. Divvala, R. Girshick, und A. Farhadi, „You Only Look Once: Unified, Real-Time Object Detection“, 2015, *arXiv*. doi: 10.48550/ARXIV.1506.02640.
[42] Ultralytics, „YOLO11 🚀 NEU“. Zugegriffen: 25. April 2025. [Online]. Verfügbar unter: https://docs.ultralytics.com/de/models/yolo11

[43] R. Girshick, „Fast R-CNN“, 27. September 2015, *arXiv*: arXiv:1504.08083. doi: 10.48550/arXiv.1504.08083.
[44] S. Ren, K. He, R. Girshick, und J. Sun, „Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks“, 6. Januar 2016, *arXiv*: arXiv:1506.01497. doi: 10.48550/arXiv.1506.01497.
[45] J. Kim, J.-Y. Sung, und S. Park, „Comparison of Faster-RCNN, YOLO, and SSD for Real-Time Vehicle Type Recognition“, in *2020 IEEE International Conference on Consumer Electronics - Asia (ICCE-Asia)*, Seoul, Korea (South): IEEE, Nov. 2020, S. 1–4. doi: 10.1109/ICCE-Asia49877.2020.9277040.
[46] G. Csurka, R. Volpi, und B. Chidlovskii, „Semantic Image Segmentation: Two Decades of Research“, 13. Februar 2023, *arXiv*: arXiv:2302.06378. doi: 10.48550/arXiv.2302.06378.
[47] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, und H. Adam, „Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation“, 22. August 2018, *arXiv*: arXiv:1802.02611. doi: 10.48550/arXiv.1802.02611.
[48] Y. Mo, Y. Wu, X. Yang, F. Liu, und Y. Liao, „Review the state-of-the-art technologies of semantic segmentation based on deep learning“, *Neurocomputing*, Bd. 493, S. 626–646, Juli 2022, doi: 10.1016/j.neucom.2022.01.005.
[49] Z.-H. Zhou, „A brief introduction to weakly supervised learning“, *Natl. Sci. Rev.*, Bd. 5, Nr. 1, S. 44–53, Jan. 2018, doi: 10.1093/nsr/nwx106.
[50] I. Goodfellow, Y. Bengio, und A. Courville, *Deep Learning*. MIT Press, 2016.
[51] A. Seferagić, J. Famaey, E. De Poorter, und J. Hoebeke, „Survey on Wireless Technology Trade-Offs for the Industrial Internet of Things“, *Sensors*, Bd. 20, Nr. 2, S. 488, Jan. 2020, doi: 10.3390/s20020488.
[52] *ISO/IEC 2382*. [Online]. Verfügbar unter: https://www.iso.org/obp/ui/#iso:std:iso-iec:2382:ed-1:v2:en
[53] S. Kounev, K.-D. Lange, und J. Von Kistowski, *Systems Benchmarking: For Scientists and Engineers*. Cham: Springer Nature Switzerland, 2025. doi: 10.1007/978-3-031-85634-1.
[54] E. J. Oughton, W. Lehr, K. Katsaros, I. Selinis, D. Bubley, und J. Kusuma, „Revisiting Wireless Internet Connectivity: 5G vs Wi-Fi 6“, *Telecommun. Policy*, Bd. 45, Nr. 5, S. 102127, Juni 2021, doi: 10.1016/j.telpol.2021.102127.
[55] J. John, Md. Noor-A-Rahim, A. Vijayan, H. V. Poor, und D. Pesch, „Industry 4.0 and Beyond: The Role of 5G, WiFi 7, and Time-Sensitive Networking (TSN) in Enabling Smart Manufacturing“, *Future Internet*, Bd. 16, Nr. 9, S. 345, Sep. 2024, doi: 10.3390/fi16090345.
[56] Wi-Fi Alliance, „Wi-Fi CERTIFIED 7 Technology Overview“. Wi-Fi Alliance, 1. Januar 2024.
[57] H. Kabir, M.-L. Tham, und Y. C. Chang, „Internet of robotic things for mobile robots: Concepts, technologies, challenges, applications, and future directions“, *Digit. Commun. Netw.*, Bd. 9, Nr. 6, S. 1265–1290, Dez. 2023, doi: 10.1016/j.dcan.2023.05.006.
[58] C. Mayershofer, D.-M. Holm, B. Molter, und J. Fottner, „LOCO: Logistics Objects in Context“, in *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*, Miami, FL, USA: IEEE, Dez. 2020, S. 612–617. doi: 10.1109/ICMLA51294.2020.00102.
[59] C. A. Akar, J. Tekli, D. Jess, M. Khoury, M. Kamradt, und M. Guthe, „Synthetic Object Recognition Dataset for Industries“, in *2022 35th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, Natal, Brazil: IEEE, Okt. 2022, S. 150–155. doi: 10.1109/SIBGRAPI55357.2022.9991784.
[60] H.-P. Gumm, M. Sommer, und W. Hesse, *Einführung in die Informatik*, 9., Vollst. überarb. Aufl. München: Oldenbourg, 2011. doi: 10.1524/9783486704587.
[61] D. F. Wolf und G. S. Sukhatme, „Mobile Robot Simultaneous Localization and Mapping in Dynamic Environments“, *Auton. Robots*, Bd. 19, Nr. 1, S. 53–65, Juli 2005, doi: 10.1007/s10514-005-0606-4.
[62] J. Vandorpe, H. Van Brussel, und H. Xu, „Exact dynamic map building for a mobile robot using geometrical primitives produced by a 2D range finder“, in *Proceedings of IEEE International Conference on Robotics and Automation*, Minneapolis, MN, USA: IEEE, 1996, S. 901–908. doi: 10.1109/ROBOT.1996.503887.
[63] R. Khanam und M. Hussain, „YOLOv11: An Overview of the Key Architectural Enhancements“, 23. Oktober 2024, *arXiv*: arXiv:2410.17725. doi: 10.48550/arXiv.2410.17725.

A Anhang

A.1 Fragebogen zur Anforderungsermittlung für Simulationsdaten

Fragebogen zum Thema: Synthetische Sensordatengenerierung in Unity

Im Rahmen meiner Semesterarbeit beschäftige ich mit der Frage, wie eine Simulationsumgebung für mobile Roboter in der Intralogistik aussehen muss, um synthetische Sensordaten zu generieren, die für die Entwicklung mobiler Roboter verwendbar sind. Dazu möchte ich Ihre Expertise in den Bereichen Intralogistik, sowie Entwicklung mobiler Roboter für mein Vorgehen nutzen.

Vielen Dank für ihre Teilnahme!

Frage 1: Nachfolgend sind mehrere Tätigkeitsbereiche aufgelistet. Bitte kreuzen Sie an, welcher dieser Bereiche ihre Arbeit am ehesten beschreibt.

☐ Entwicklung: Simulation

☐ Entwicklung: Versuche/Prototyping

☐ Geschäftsführung

☐ Sales

☐ Sonstiger Tätigkeitsbereich:

Fragen 2-8: Aufbau der Lagerhalle in der Simulationsumgebung

Frage 2: Wo werden Ihre mobilen Roboter eingesetzt?

☐ Indoor

☐ Outdoor

☐ Indoor + Outdoor

Frage 3: Wie viele Stockwerke umfasst der Bereich, in welchem die mobilen Roboter eingesetzt werden?

☐ ein Stockwerk

☐ mehrere Stockwerke, Details:

Frage 4: Sofern der Einsatzbereich der mobilen Roboter mehr als ein Stockwerk umfasst: Wie gelangt der Roboter von einem Stockwerk ins andere? (Aufzug, …)

Frage 5: Ist die Lagerhalle in mehrere Räumlichkeiten unterteilt? Sofern ja, beschreiben Sie die Raumaufteilung.

☐ ein großer Raum

☐ mehrere Räume, Anzahl:

Frage 6: Sofern die Lagerhalle aus mehreren Räumlichkeiten besteht, wie sind diese Räume untereinander verbunden? (Tür, Rolltor, etc.)

Frage 7: Beschreiben Sie die vorhandenen Lichtquellen in der Lagerhalle.

Natürliche Lichtquelle (Fensterarten):

Künstliche Lichtquelle (Beleuchtungsart):

Frage 8: Gibt es sonstige Strukturen in der Lagerhalle? (z. B. Säulen)

Frage 9: Lässt sich der Einsatzbereich der mobilen Roboter in verschiedene Zonen unterteilen? (Lagerraum, Be-/Entladezone, …) Beschreiben sie die einzelnen Zonen.

Platz für sonstige Anmerkungen zum Hallendesign:

Fragen 10-12: Objekte in der Lagerhalle

Frage 10: Welche Objekte werden von Ihren Robotern transportiert?

☐ Palette

☐ Gitterbox

☐ KLT-Box (Kleinladungsträger)

☐ Karton

☐ Sonstige:

Frage 11: Welche weiteren mobilen Objekte kommen im Einsatzbereich ihrer mobilen Roboter vor?

☐ Menschen

☐ Gabelstapler

☐ mobiler Roboter

☐ Transportwagen

☐ Wagenheber

☐ Sonstige:

Frage 12: Welche der weiteren Gegenstände kommen in Ihren Lagerhallen vor?

☐ Schrank

☐ Regal

☐ Feuerlöscher

☐ Mülleimer

☐ Sonstige:

Fragen 13-19: Sensorik und Entwicklung mobiler Roboter

Frage 13: Welche technischen Probleme bestehen aus ihrer Sicht bei der Entwicklung autonomer mobiler Roboter in der Intralogistik?

Frage 14: Kann verbesserte Umgebungswahrnehmung des Roboters helfen, diese Probleme zu lösen?

Frage 15: Welche Informationen über die Umgebung des Roboters sind dazu notwendig? (z. B. um welches Objekt es sich handelt, Geometrische Abmaße, etc.)

Frage 16: Ihnen stehen unbegrenzte Sensordaten mobiler Roboter zur Verfügung. Wie lassen sich diese nutzen, um den Autonomiegrad mobiler Roboter zu erhöhen?

Frage 17: Dem Roboter ist es möglich zu jeder Zeit alle Objekte in seiner Umgebung identifizieren zu können. Lassen sich aktuelle Probleme beheben, sofern der Roboter diese Fähigkeit besitzt?

Frage 18: Beeinflusst sich die Sensorik verschiedener Fahrzeuge im realen Betrieb gegenseitig? (z. B. durch Interferenzen zwischen verschiedenen Lidar Sensoren)

Frage 19: Haben Sie sonstige Hinweise, Tipps oder Ratschläge für die Umsetzung der Simulationsumgebung, die nicht in den Fragen berücksichtigt wurden?

A.2 Bewertungsmatrix Simulationssoftware

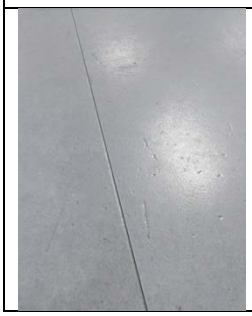
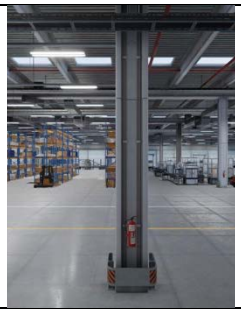
Datenimport	Implementierungsaufwand	Visualisierung	Physikinteraktion	Datenexport
Gazebo Wegen seiner tiefen Integration in das ROS-Framework, ist Gazebo der de-facto-Standard für die Simulation von Problemen mit ROS-Bezug ohne hohe Anforderungen an die Grafik. Gazebo ist open-source und enthält eine große Bandbreite an vorimplementierten Sensoren.				
◑	◑	○	◑	◑
URDF, stl, OBJ, Collada	Vorimplementierte Sensoren	Einfaches grafisches Rendering und Licht basierend auf der OGRE-Engine	DART-Engine, Zusätzliche Physik-Engine-Plugins; Trivial Physics Engine mit Schwerpunkt auf großen kinematischen Umgebungen	ROS-Integration Kein Datenset-Export
Unity Unity wurde als Spiel-Engine entwickelt und verfügt über eine Physik-Engine und umfangreiche Grafikmöglichkeiten. Unity wird zunehmend für verschiedene Arten von maßgeschneiderten Simulationen verwendet.				
●	◑	●	◑	●
URDF, FBX, OBJ, Collada, 3ds, dxf, .blend	Der Unity Robotics Hub und das Unity Perception Package bieten verschiedene Robotik- und Computer Vision-bezogene Erweiterungen, die jedoch nicht mehr gepflegt werden; keine vorimplementierten Sensormodelle	Highfidelity-Rendering, einschließlich hochwertiger Beleuchtung, Schatten, Texturen und Ray-Tracing	Nvidia PhysX	ROS-Integration SOLO-Format
Webots Webots ist ein open-source 3D-Roboter-Simulator, der verschiedene veränderbare Modelle, Roboter, Sensoren und Aktoren bereitstellt.				
○	◑	◑	◑	◑
Collada, STL, OBJ	Viele vor-implementierte Modelle, Sensoren, und Aktoren; Bildsegmentierung ist möglich	Fortgeschrittene Grafik basierend auf dem Webots Renderer	Open Dynamics Engine für Starrkörperdynamik	ROS-Integration Kein Datenset-Export
CoppeliaSim				

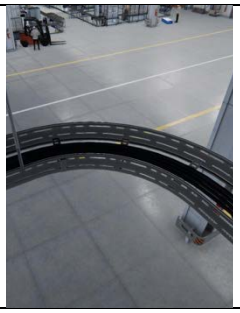
CoppeliaSim ist ein 3D-Roboter-Simulator für stationäre und mobile Robotik. Er enthält verschiedene Physik-Engines und vor-implementierte Algorithmen.				
◑	◑	○	●	◑
URDF/SDF OBJ, DXF, STL, 3DS, Collada	Verschiedene Modelle und Algorithmen sind enthalten; keine automatische Ground-Truth-Generierung	Einfache Grafik ohne viele visuelle Details	Bullet Physikbibliothek, Open Dynamics Engine, MuJoCo Physik-Engine, Vortex Studio Engine, Newton Dynamics Engine; 3D Kollisionserkennung, Starrkörperdynamik, Dynamik elastischer Körper	ROS-Integration Kein Datenset-Export
Blender Blender wird für Modellierung, Texturierung und Animation von 3D-Modellen eingesetzt. Blender ist open-source und wird in jüngster Zeit öfter genutzt, um synthetische Trainingsdaten für das Training von Deep-Learning-Algorithmen zu erzeugen.				
●	○	●	◑	○
Collada, Alembic, USD, PLY, FBX, glTF, gLBOBJ, STL, 3DS, X3D	Nahezu alle Robotik-bezogenen Funktionen müssen von Grund auf selbst entwickelt werden	Highfidelity-Rendering, einschließlich hochwertiger Beleuchtung, Schatten, Texturen und Ray-Tracing	Starrkörper, Elastische Körper, Fluid- und Partikel-Simulation	Kann nur Bilder speichern
Unreal Engine Wie Unity ist auch Unreal eine Spiel-Engine. Sie ist bekannt für ihre High-Fidelity-Rendering-Fähigkeiten und wird in letzter Zeit für verschiedene Arten von Simulationen verwendet.				
●	○	●	◑	◑
FBX, OBJ, glTF, GLB	Nahezu alle Robotik-bezogenen Funktionen müssen von Grund auf selbst entwickelt werden	Highfidelity-Rendering, einschließlich hochwertiger Beleuchtung, Schatten, Texturen und Ray-Tracing	Nvidia PhysX	Keine offizielle ROS-Integration
Isaac Sim Um den Prozess von der Modellierung über die Simulation bis hin zur Erzeugung synthetischer Sensordaten vollständig abzudecken, hat Nvidia Isaac Sim als Teil von Nvidia Omniverse entwickelt. Es verfügt über verschiedene Umgebungs- und Robotermodelle sowie eine Physik-Engine und umfangreiche Grafikfunktionen.				
◑	●	●	◑	●
USD, FBX, OBJ, glTF	Große Bibliothek für Modelle, Roboter, Sensoren; automatische	Highfidelity-Rendering, einschließlich hochwertiger Beleuchtung,	Nvidia PhysX	ROS-Integration; Datenset-Export mittels vorgefertigter und

	Ground-Truth-Generierung (Bounding-Boxen, Segmentierung)	Schatten, Texturen und Ray-Tracing		maßgeschneiderter Writer
Airsim Airsim wurde von Microsoft entwickelt und ist ein Open-Source-Simulator für verschiedene Arten von mobilen und fliegenden Robotern. Er basiert auf Unreal oder Unity und verfügt über realitätsnahe Grafik sowie vorimplementierte Umgebungen und Sensoren.				
◑	●	●	◑	●
Siehe Unreal/Unity Import Umgebung voll, Roboter nur von Gazebo	Vorimplementierte Modelle, Roboter, Sensoren; automatische Segmentierung	Siehe Unreal/Unity nutzt Highfidelity-Modelle	Nvidia PhysX	ROS-Integration; Proprietäres Aufnahme-Format

○ = Kriterium ist nicht erfüllt	◑ = Kriterium ist zum Teil erfüllt	● = Kriterium ist voll erfüllt

A.3 Umgebungsobjekte in der Simulationsszenerie

Boden	Wand	Säule	Beleuchtung	
			Dachfenster	Künstliches Licht
				

Gebäude (Fortsetzung)			
Büro	Betriebseinrichtungen		
	Lüftung	Rohr	Kabel
			

Statische Objekte			
Regal	Verpackungsanlage	Produktionsanlage	Hubwagen
			

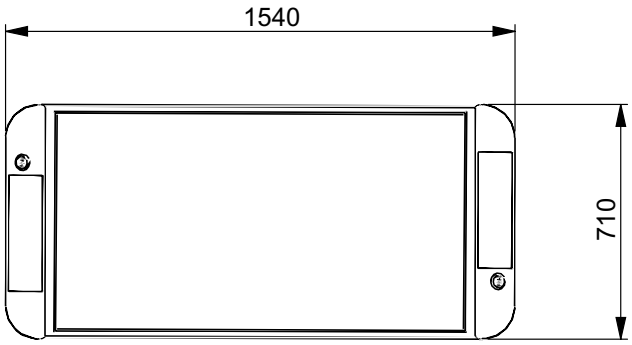
Statische Objekte (Fortsetzung 1)			
Tisch	Werkzeugwagen	Markierungen	
		Schilder	Bodenmarkierung
			

Statische Objekte (Fortsetzung 2)					
Ladungsträger					
Palette	KLT	GLT	Gitterbox	Fass	Karton
					

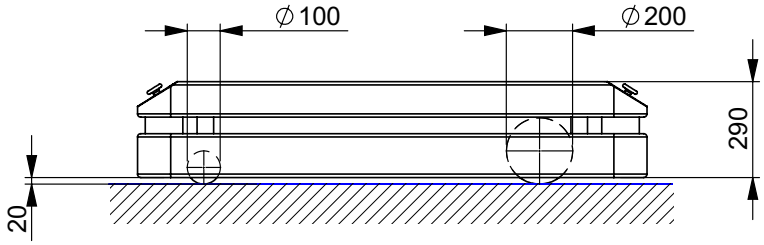
Dynamische Objekte						
Menschen				Flurförderzeuge		Roboter
Mensch 1	Mensch 2	Mensch 3	Mensch 4	Gabelstapler	Schmalgang-stapler	
						

A.4 Maße des simulierten Roboters

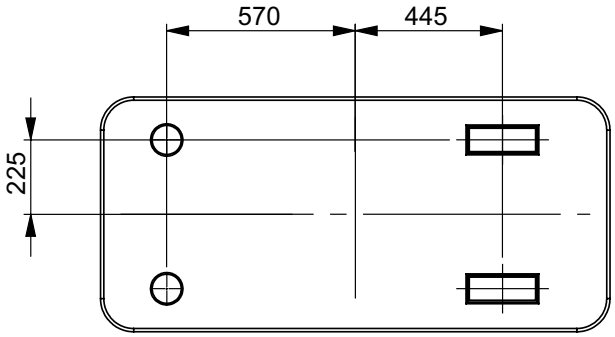
Draufsicht



Seitenansicht

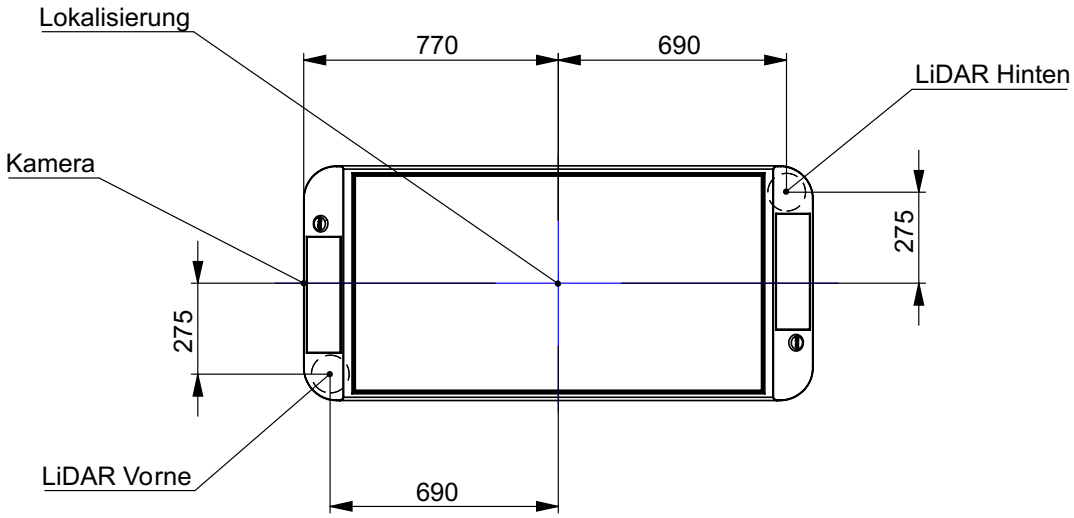


Untersicht



A.5 Positionierung der Sensoren auf dem Roboter

Draufsicht



Seitenansicht