HAW HAMBURG

Bachelorarbeit

Adrian Kaßmann

Vergleich von dezentralen Erkundungsstrategien in Multi-Robotersystemen

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department Computer Science

Adrian Kaßmann

Vergleich von dezentralen Erkundungsstrategien in Multi-Robotersystemen

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

Betreuender Prüfer: Prof. Dr. Martin Becke
Zweitgutachter: Prof. Dr. Jan Sudeikat

Eingereicht am: 31. Januar 2023

Adrian Kaßmann

Thema der Arbeit

Vergleich von dezentralen Erkundungsstrategien in Multi-Robotersystemen

Stichworte

Robotik, Erkundungsstrategien, ROS, Simulation, Multi-Agenten-System, Auktionen

Kurzzusammenfassung

In dieser Arbeit werden unterschiedliche Strategien mit mehreren Robotern verglichen, welche gemeinsam ein Gebiet erkunden müssen. Für den Vergleich wird ein eigener Algorithmus vorgestellt, welcher auf einem Auktionsverfahren basiert, um Informationen zwischen den Robotern auszutauschen. Die Resultate des eigenen Algorithmus werden mit der „Ungarischen Methode“ verglichen unter besonderer Berücksichtigung der Dauer und der zurückgelegten Distanz.

Adrian Kaßmann

Title of Thesis

Comparison of decentralised exploration strategies in multi-robot systems

Keywords

Robotics, exploration strategies, ROS, simulation, multi-agent system, auctions

Abstract

This thesis compares different strategies with several robots that have to explore an area together. For the comparison, a new strategy is presented that uses auctions to exchange information between the robots. The results of the new strategy in time and distance covered for the complete exploration are as good as the Hungarian method.

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

Die Navigation von Robotern ist eine klassische Aufgabe in der Robotik. Dabei wird der Roboter oft in einer Umgebung eingesetzt, in der noch keine Informationen bekannt sind und selbst eine Karte erstellt werden muss. Die Erstellung der Karte für unbekannte Umgebungen kann Teil- oder auch die Hauptaufgabe des Roboters sein [21].

Um das Erkunden effizient und schnell umzusetzen, können mehrere Roboter zusammen eingesetzt werden. Eine Strategie legt dabei fest, wie das unbekannte Gebiet auf die Roboter aufgeteilt wird, sodass z.B. die Zeit minimiert werden kann. Die Wahl der Strategie ist dabei abhängig von der Art der Umgebung (Innerhalb oder Außerhalb von Gebäuden), der Art der Roboter (auf dem Boden oder in der Luft [32]) oder auch der Art der Sensoren (Kamera, Lidar, etc.).

Schaut man auf die Anwendungsgebiete[1] [14], die das DFKI [2] in Robotik definiert, kommt in vielen der Gebiete die Aufgabe der Erkundung zum Einsatz. In der Weltraumrobotik gibt es das Aufgabengebiet der planetaren Erkundung. Bei der Unterwasserrobotik kommt es bei der Kartierung des Ozeans zum Einsatz [23]. Das Anwendungsgebiet Search and Rescue & Sicherheitsrobotik ist ein klassisches Beispiel für die Erkundung. Ist es für einen Menschen zu gefährlich, können Roboter eingesetzt werden, um vermisste Personen zu finden [46, 2, 4]. In dem Bereich der Assistenz- und Rehabilitationssysteme sind Staubsaugerroboter ein Beispiel für Erkundungen der Umgebungen. Die neusten Modelle erstellen eine Karte von der Wohnung mit Lidar-Sensoren. Über eine App kann dann der Roboter angewiesen werden, einen bestimmten Raum zu saugen [13].

In den weiteren Anwendungsgebieten des DFKIs ist die Erkundung teilweise enthalten. In der Elektromobilität, Agrarrobotik oder in dem Bereich von Logistik, Produktion und

[1] Weltraumrobotik, Unterwasserrobotik, Elektromobilität, „Logistik, Produktion und Consumer (LPC)“, „Search and Rescue (SAR) und Sicherheitsrobotik“, Assistenz- und Rehabilitationssysteme, Agrarrobotik

[2] Deutsches Forschungszentrum für Künstliche Intelligenz

Consumer wird auch die Navigation in unbekannte Gebiete verwendet. Ein Beispiel sind die Lieferroboter für die „letzte Meile“ [1].

Der Einsatz von mehreren Robotern kann von Vorteil sein, wenn die Aufgabe sehr komplex ist und die Bearbeitung in kleinen Teilen erfolgen kann. Viele simple Roboter haben gegenüber einem leistungsfähigen Roboter den Vorteil, dass diese günstiger, flexibler und fehlertoleranter sein können. Ein Ausfall eines Roboters kann durch das parallele Arbeiten kompensiert werden. [11].

Ziel dieser Arbeit ist einen Vergleich von verschiedenen Strategien zur Koordination von Robotern, die gemeinsam ein unbekanntes Gebiet erkunden. Für das Problem gibt es unterschiedliche Ansätze. Der Fokus dieser Arbeit liegt auf dezentralen Strategien, bei dem jeder Roboter eigenständig agieren kann.

Neben einer zufälligen Zuweisung der Ziele für die Roboter wird eine Strategie auf Grundlage des Zuordnungsalgorithmus „Ungarische Methode“ von Batinovic et al. [6] und einer eigenen Strategie, die auf einem Auktionsmechanismus aufbaut, miteinander verglichen.

In den letzten Jahren wurde wenig zu Auktionsmechanismen in dem Forschungsgebiet geforscht. Die meisten Veröffentlichungen sind von Anfang der 2000er Jahre. Ein Ziel in dieser Arbeit ist daher, das Potential von Auktionen mit aktuellen Erkundungsstrategien zu vergleichen.

Das Kapitel 2 behandelt relevante Publikationen aus diesem wissenschaftlichen Themenbereich. In Kapitel 3 werden die Grundlagen vorgestellt, mit besonderem Fokus auf den Erkundungsstrategien und den verwendeten Software-Paketen.

In Kapitel 4 wird das Konzept des Systems genauer beschrieben, indem die Systemarchitektur und Ausführung erläutert wird. Danach folgt in Kapitel 5 die Durchführung, in dem die Ergebnisse und Resultate der Arbeit dargestellt und miteinander verglichen werden. In Kapitel 6 folgen die Zusammenfassung und der Ausblick auf zukünftige Entwicklung.

2 Verwandte Arbeiten

In der Arbeit von Batinović et al. [6] wurde eine dezentrale Koordinierung mittels der Ungarischen Methode mit der zentralen Koordinierung von Borgard et al. [10] verglichen. Die Resultate zeigen, dass die Erkundung durch den dezentralen Ansatz einen größeren Vorteil bietet, je mehr Roboter eingesetzt werden. Die Versuche werden auf einer einzelnen Karte durchgeführt [6].

Die Arbeit von Kulich et al. [29] vergleicht die benötigte Zeit von verschiedenen Algorithmen mit vier, sechs und acht Robotern auf Karten, die sich in Größe und Strukturen unterscheiden. Dort wurde unter anderem auch die Ungarische Methode verglichen, welche gemeinsam mit dem K-means-Algortihmus [51] am besten abgeschlossen hat. Neben der Ungarischen Methode und dem K-means-Algorithmus, wurden auch noch ein greedy-Algorithmus [55] und „Broadcast of Local Eligibility “ von Werger und Mataric [52] miteinander verglichen.

Kasperski et al. [27] haben mehrere Algorithmen auf größeren Karten verglichen, darunter eine Karte, die ein Gelände außerhalb von Gebäuden abbildet. Hier wurden zur damaligen Zeit verschiedene „state-of-the-art“ Strategien verglichen, darunter auch ein greedy-Algorithmus, MultiWave, MinPos und zwei Utility-Algorithmen. Bei den Utility-Algorithmen wird zum einen die Distanz und zum anderen die Nähe zu anderen Robotern und deren Ziele bei der Wahl des Zielpunktes berücksichtigt [10]. Das ist derselbe Algorithmus, der auch bei Batinović et al. [6] als zentrale Erkundungsstrategie eingesetzt wurde. Bei MultiWave breiten sich Bereiche von den Robotern ausgehend über alle freien Flächen aus. Die Ausbreitung stoppt bei den Bereichen anderer Roboter. Trifft ein Bereich auf ein unerkundetes Gebiet, fährt der Roboter es an [27]. Es wird den geringsten Weg zum nächsten unbekannten Gebiet gesucht.

Die Strategie MinPos [7] plant vom unbekannten Gebiet zum Roboter. Ähnlich wie bei MultiWave, wird mit ausbreitenden Bereichen gearbeitet. Wird ein Roboter getroffen, fährt dieser zum Ursprung des Bereiches.

Elsefy et al. [17] haben in ihrer Arbeit auch verschiedene Strategien verglichen, mit dem Ergebnis, dass der Role-based-exploration-Algorithmus am besten abgeschnitten hat. Zudem hat auch die Ungarische Methode gut abgeschnitten. Als Umgebung wurden 3 Karten verwendet, die unterschiedlich komplex aufgebaut sind [17]. Beim Role-based-exploration-Algorithmus sind die Roboter eines Netzwerkes in einer hierarchischen Struktur eingeordnet. Es gibt einen Base-Station-Roboter, mehrere Relay- und Explore-Roboter. Diese Roboter leiten ihre Informationen weiter zum Base-Station-Roboter. Die Rollen der Roboter können wechseln, wenn dieses die Topologie verbessert [43, 34].

Alle hier erwähnten Arbeiten vergleichen unterschiedliche Erkundungsstrategien miteinander. Batinović et al. [6] und Elsefy et al. [17] fokussieren sich auf die Unterscheidung von zentralen und dezentralen Ansätzen und zeigen den Vorteil der dezentralen Strategien. Für die Untersuchung wurden in allen Vergleichen die Zeit gemessen, bis die Karte zu einem bestimmten Prozentsatz erkundet wurde. In Kasperski et al. [27] wurde als zweite Metrik die Anzahl an Laserscans genommen, die in das Welt-Modell eingefügt wurden, um die Karte zu erstellen. Dabei ist das ein indirekter Faktor für die zurückgelegte Distanz der Roboter. Wenn der Roboter eine bestimmte Distanz zurücklegt oder um einen bestimmten Winkel um die eigene Achse gedreht wurde, wird ein neuer Laserscan in dem Modell berücksichtigt.

Elsefy et al. [17] nutzte auch als zweite Metrik die zurückgelegte Distanz der Roboter, indem die Anzahl an Schritten zur Erkundung gemessen und verglichen wurden.

2.1 Marktorientierte Strategien

Was in den genannten Arbeiten nicht vorgekommen ist, sind Ansätze, die auf marktorientierte Strategien basieren. Daher werden im folgenden Abschnitt ein paar Erkundungsstrategien vorgestellt, die auf Auktionsverfahren basieren.

Yan et al. [56] hat 2013 den Stand der Multi-Roboter Koordination in einer Übersichtsstudie (Survey) zusammengefasst. Darin stellen sie verschiedene Ansätze zur Erkundung mit marktbasierten Strategien vor. Viele Strategien basieren dabei auf dem contract net protocol (CNP) [50], welches ein Task-Share-Protokoll in einem Multi-Agenten-System (MAS) ist. Beispiele dafür sind MURDOCH [18] oder M+ [9].

Für die Erkundung mit mehreren Robotern stellte Zlot et al. [57] im Jahr 2002 eine Strategie vor, bei der die Roboter dezentral über ein marktorientiertes System gesteuert

werden. Jeder Roboter fährt eine Liste an Wegpunkten ab. Die Wegpunkte sind Markierungen, die neue Informationen über das Gebiet enthalten. Ist ein Wegpunkt erreicht, werden neue Wegpunkte erstellt, abhängig von dem erkundeten Gebiet und Anzahl der noch abzufahrenden Wegpunkte [57]. Die Roboter versteigern Ihre Wegpunkte an die anderen Roboter. Bietet ein Roboter einen höheren Preis als der vom versteigernden Roboter geschätzt wurde, bekommt dieser den Zuschlag und der Wegpunkt wird dem neuen Besitzer übertragen [57]. Die beschriebene Kommunikation ist asynchron aufgebaut und kann mit Verbindungsabbrüchen umgehen, sodass im schlimmsten Fall kein Austausch von Wegpunkten möglich ist und jeder Roboter seine eigene Liste abarbeiten kann [57].

Berhault et al. [8] nutzte 2003 Auktionsmechanismen, bei dem die Roboter auf Pakete bieten, die Zielpunkte enthalten. Dabei wird ein zentraler Auktionator eingesetzt, der für den Ablauf der Auktionen verantwortlich ist. Der Höchstbietende zahlt sein gewonnenes Gebot. Zur Effizienzsteigerung zahlen die Roboter zusätzlich noch einen Preis pro zurückgelegte Distanz.

3 Grundlagen

3.1 Multi-Roboter-System

Ein Multi-Roboter-System besteht aus einer Gruppe von mobilen Robotern. Zwischen den Robotern findet ein Austausch von Informationen über ein Kommunikationsnetz statt [16]. Dabei wird zwischen homogene und heterogene Gruppen von Robotern unterschieden. In einer homogenen Gruppe sind alle Roboter vom selben Modell z.B. ein Schwarm von Drohnen [12].

Bei heterogenen Gruppen unterscheiden sich die Robotermodelle. Durch verschiedene Robotertypen kann die Gruppe besser auf die Anforderungen der Aufgaben reagieren. Ein gutes Beispiel dafür ist die Darpa Subterrain Challenge [4], bei der Teams heterogene Gruppen einsetzen, damit z.B. offene Gebiete oder unwegsames Gelände mit einem Quadrocopter, oder enge Passagen mit Landfahrzeuge erkundet werden [3].

Außerdem gibt es spezialisierte Roboter, die an bestimmte Aufgaben angepasst sind. Ein Beispiel ist die Ausrüstung eines Greifarms zum Aufbau einer Mondbasis aus dem Projekt PRO-ACT[20].

In der Simulation dieser Arbeit sind die Roboter eine homogene Gruppe, welche dem Pioneer [26] nachempfunden sind. Der Roboter ist mit einem 2D-Laserscanner ausgerüstet, der 360° abdeckt und bis zu 20 m weite Entfernungen wahrnehmen kann. Die Position der Roboter wird von der Simulation als Odometrie und als absolute Position ausgegeben.

3.2 Erkundungsstrategien

Für die Aufteilung der Roboter in einem Gebiet gibt es viele Möglichkeiten. Dabei wird in zentrale und dezentrale Erkundungsstrategien unterschieden.

3.2.1 Zentrale Erkundungsstrategien

Bei der Entwicklung der Softwarearchitektur existieren unterschiedliche Ansätze wie die Roboter miteinander kommunizieren. Bei der zentralen Lösung gibt es einen „Anführer“, der die Informationen von allen anderen Robotern (den Erkundern) bekommt und ihnen die nächsten Ziele zuweist. Durch die Hierarchie vereinfacht sich die Koordination, denn die Erkunder geben ihre Informationen weiter und bekommen ein neues Ziel. Der Anführer hält alle Informationen und wertet diese aus z.B. durch SLAM-Algorithmen aus. Die Zuweisung der Ziele ist dadurch zentral gesteuert [33].

3.2.2 Dezentrale Erkundungsstrategien

Dezentrale Strategien können in verteilte Architekturen und hierarchische Architekturen eingeteilt werden. Es existiert kein zentral entscheidender Agent. Vielmehr sind alle Roboter gleichgestellt und treffen ihre eigenen Entscheidungen [56].

Bei den hierarchischen Architekturen sind die Roboter in ein oder mehrere Cluster unterteilt, in denen es einen Anführer gibt, der die Gruppe führt. Diese hybride Struktur ist ein Kompromiss zwischen dem zentralen und dezentralen Ansatz [56].

Ein Vorteil von dezentralen Strukturen ist, dass diese flexibler und adaptiver agieren können. Zudem sollen diese noch verlässlicher und robuster sein, da die ganze Zuordnung nicht auf nur einem Agenten basiert. Die Zuordnungen ist nicht immer perfekt, da die Entscheidungen teilweise nur auf Teilinformationen basieren [56].

3.2.3 Frontier-Ansatz

Yamauchi [54] legte mit dem „Frontier-Ansatz“ die Grundlage für viele Erkundungsstrategien. Jeder Pixel in einer Rasterkarte kann einen von drei Zustände haben. Entweder ist der Zustand an der Position unbekannt, besetzt oder frei. Auf den freien Punkten kann

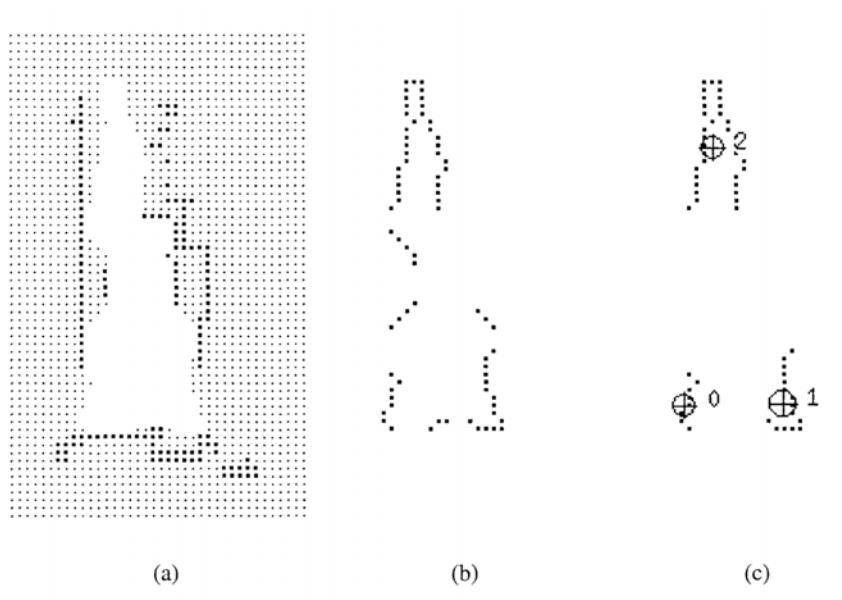


Abbildung 3.1: Ermittlung der Grenzpunkte aus Yamauchi [54]

sich ein Roboter ohne Probleme bewegen. Besetzte Punkte sind ein Hindernis, welches nicht passierbar ist, z.B. eine Wand. Über die unbekannten Umgebungen können noch keine Aussagen getroffen werden. [54].

Um den unbekannten Teil der Karte zu erkunden, muss ein Roboter in diesen Bereich fahren. Da dieser nur auf freien Flächen navigieren kann, werden die Kanten zwischen freien und unbekannten Punkten gesucht. Diese Punkte werden ab einer Mindestgröße (Anzahl an Punkten) zu „Frontier-Regionen“ zusammengefügt [54].

In Abbildung 3.1 ist dieser Ansatz mit drei Karten visualisiert. Auf der linken Seite ist eine Karte mit den Zuständen dargestellt (weiß entspricht einer freien Fläche, die dünnen Punkte sind unbekannt und die dicken Punkte entsprechen einem Hindernis). In der Mitte sind nur die Übergänge zwischen freien und unbekannten Bereichen dargestellt. Auf der rechten Seite werden Regionen (genügend zusammenhängende Punkte) gebildet [54].

Der Mittelpunkt dieser Regionen kann als Navigationsziel für den Roboter genommen werden. Einige der in Kapitel 2 aufgeführten Arbeiten basieren auf diese Frontier-Ansatz [6, 7, 10, 51, 57].

3.2.4 Ungarische Methode

Die Ungarische Methode ist ein Zuordnungsalgorithmus, welcher 1955 von Harold W. Kuhn [28] entwickelt wurde. Mit dieser Methode kann jeder Roboter selbst die Ziele für sich und für die anderen Roboter berechnen, das bedeutet die Zuordnung kann dezentral erfolgen. Dafür müssen nur die aktuelle Position und das aktuelle Ziel der Roboter untereinander kommuniziert werden.

Kostenberechnung

Die Ermittlung der Kosten eines Roboters i für ein Zielpunkt j erfolgt nach Batinović et al. [6].

$$C_{i,j} = \sqrt{(p_{i,x} - y_{j,x})^2 + (p_{i,y} - y_{j,y})^2} \tag{3.1}$$

Der erste Teil $C_{i,j}$, welcher in der Gleichung 3.1 beschrieben ist, definiert die Distanz zwischen dem Ziel (x_j, y_j) und der Roboterposition $(p_{i,x}, p_{i,y})$.

$$U_j = \lambda * k_j \tag{3.2}$$

U_j in Gleichung 3.2 stellt die Kosten für den Informationsgehalt dar. Dabei stellt k_j die Anzahl an nicht erkundeten Pixel in einem Radius um den Punkt j dar. Die Anzahl wird mit einem Faktor λ gewichtet, welcher empirisch ermittelt wurde.

$$F_{i,j} = \begin{cases} \lambda_f e^{-[\frac{y_{jx} - y_{ax})^2}{2\sigma_x^2} + \frac{(y_{j,y} - y_{a,y})^2}{2\sigma_y^2}]}, & \text{if } d(y_j, y_a) < r_f \\ 0 & \text{otherwise} \end{cases} \tag{3.3}$$

$F_{i,j}$ in Gleichung 3.3 bildet zusätzliche Kosten, wenn sich ein potentieller Zielpunkt y_j innerhalb des Radius r_f eines bereits zugewiesenen Punktes y_a befindet. Die Kosten werden über eine 2D-Gauss-Funktion ermittelt werden. Je näher sich die beiden Punkte zueinander befinden, desto höher fallen die Kosten aus. Das Ziel davon ist eine bessere Verteilung der Zielpunkte.

$$W_{i,j} = C_{i,j} - U_j + F_{i,j} \tag{3.4}$$

Die Gesamtkosten W_{ij} jedes Roboters i zu jedem Punkt j wird in die Kostenmatrix eingetragen, wie in Gleichung 3.5 dargestellt wird. Diese dient als Grundlage der Ungarischen Methode.

$$W_{ij} = \begin{bmatrix} W_{0,0} & \dots & W_{0,m} \\ \vdots & \ddots & \vdots \\ W_{n,0} & \dots & W_{n,m} \end{bmatrix} \tag{3.5}$$

Zuordnungsalgorithmus

Für die Zuordnung muss die Kostenmatrix eine quadratische Matrix sein, heißt die Anzahl an Robotern muss der Anzahl an Zielpunkten entsprechen. Das ist bei der Zuordnung der Zielpunkte selten der Fall, da es oft mehr Zielpunkte als Roboter gibt. Als eine mögliche Lösung können virtuelle Roboter in der benötigten Anzahl hinzugefügt werden [6].

Für das Erreichen der Zuordnung müssen folgende Schritte durchgeführt werden: [28]:

1. In jeder Reihe wird das Minimum der Reihe subtrahiert. Dadurch steht in jeder Reihe mindestens eine Null.
2. In jeder Spalte wird das Minimum der Spalte subtrahiert. Auch dadurch steht dann in jeder Spalte mindestens eine Null.
3. Alle Nullen in der Matrix sollen mit so wenig vertikalen und horizontalen Linien markiert werden.
4. Ist das Markieren mit weniger Linien möglich, als die Anzahl der Zeilen/Spalten ist, müssen weitere Schritte so oft wiederholt werden, bis die diese übereinstimmen:
   a) kleinster Wert über Null von allen nicht durchgestrichenen Zeilen subtrahieren.
   b) gleichen Wert auf allen durchgestrichenen Spalten addieren.
5. Zeilen mit nur einer Null wird der entsprechenden Spalte zugeordnet.
6. Alle weiteren Nullen in der Spalte werden gestrichen bzw. auf unendlich gesetzt.

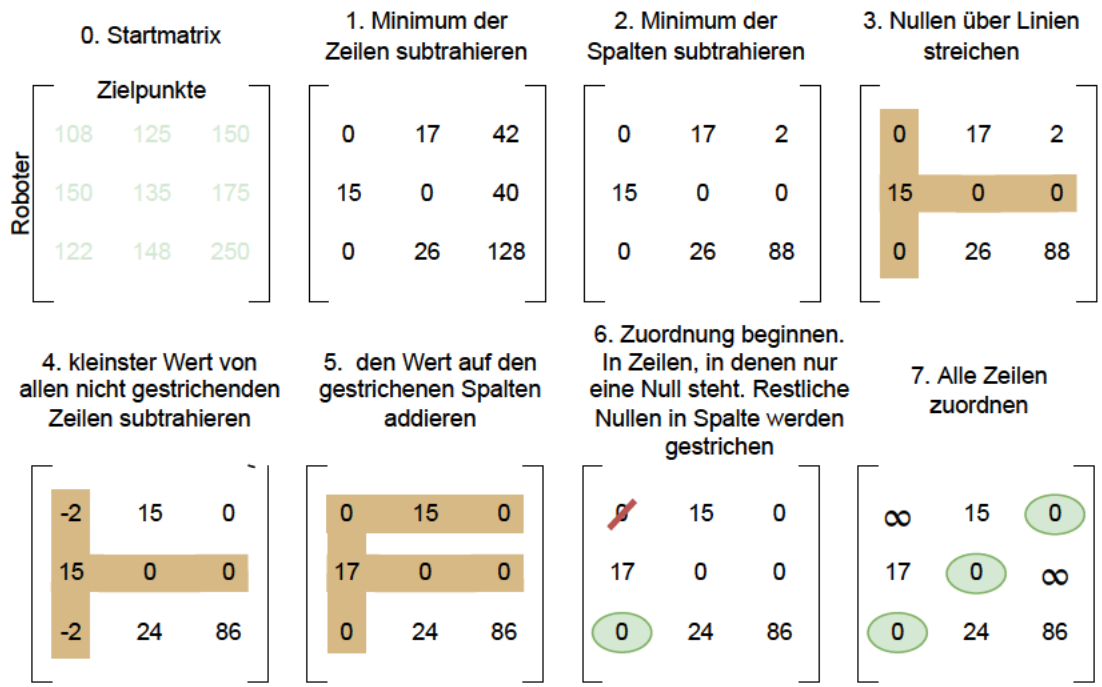


Abbildung 3.2: Ein Beispiel[1] für die Zuordnung mit der Ungarischen Methode.

7. Die letzten beiden Schritte werden wiederholt, bis alle Zeilen zugeordnet sind.

In Abbildung 3.2 ist ein Beispiel zur Veranschaulichung dargestellt, bei dem die Roboter einem Zielpunkt zuordnet werden. Das Markieren der Nullen ist zunächst mit zwei Linien möglich, daher werden die Schritte 4.a) und 4.b) durchgeführt, damit drei Linien benötigt werden, um alle Nullen markieren zu können. Die finalen Zuordnungen aus dem Beispiel sind:

- $\text{Roboter}_0 \rightarrow \text{Zielpunkt}_2$ für die Kosten von 150
- $\text{Roboter}_1 \rightarrow \text{Zielpunkt}_1$ für die Kosten von 135
- $\text{Roboter}_2 \rightarrow \text{Zielpunkt}_0$ für die Kosten von 122

[1] Werte aus diesem Beispiel übernommen: https://brilliant.org/wiki/hungarian-matching/

3.3 Robot-Operating-System

Das Robot-Operating-System (ROS) ist eine Open-Source-Software- und Bibliothekensammlung zur Entwicklung und Steuerung von Robotern. Die Sammlung bietet verschiedene Werkzeuge und Bibliotheken zum Verwalten, Schreiben und Ausführen von Code an. Dazu verfügt ROS über eine 3D-Simulationsumgebung namens Gazebo. Gazebo bildet die Umgebung der Roboter mittels Sensoren und einem realistischen Physikmodell nach [36]. Viele Hersteller für Sensoren oder Aktoren bieten Gerätetreiber für ROS an, um die Entwicklung mit ihrer Hardware zu erleichtern [38].

3.3.1 Architektur

In ROS wird der Aufbau der Software in einzelne Nodes unterteilt. Dabei entspricht jede Node einem eigenen Prozess. Eine Node erfüllt meist nur eine Funktion, wodurch eine Architektur aus vielen kleinen Nodes erschaffen wird. Bei der Software wird viel auf generische Wiederverwendung gesetzt, indem die Schnittstellen generell gehalten werden, um z.B. den Austausch von verschiedenen SLAM-Algorithmen simpel gestalten zu können [37].

3.3.2 Kommunikation mit ROS

Das Robot Operating System bietet eine eigene Kommunikationsinfrastruktur zum Austausch zwischen einzelnen Nodes. Diese Nodes können sich über Topics mittels einem publish/subscribe-Modell asynchron austauschen. Für eine synchrone Kommunikation bieten die Nodes Services an, die von anderen Nodes aufgerufen werden [37].

3.4 Stage-Simulation

Stage ist eine Open-Source-2D-Siumlation für Roboter von Player Project. Die Simulation ist ressourcenschonend und erfüllt dabei alle Anforderungen für den Vergleich der Strategien [44].

Die Simulation ist als Software-Paket direkt als Node in ROS verfügbar [47].

Eine .world-Datei beschreibt die Umgebung und die Roboter mit Sensoren. Die zweidimensionale Welt kann über eine .png-Datei in der .world-Datei geladen werden. Somit kann die Karte und Anzahl der Roboter flexibel eingestellt werden [47].

3.5 SLAM - Cartographer

Beim SLAM[2]-Verfahren erstellt ein mobiler Roboter eine Karte von der Umgebung und lokalisiert seine eigene Position in der Karte. Die Karte wird schrittweise erweitert, da bei einer Bewegung des Roboters die neue Position in der Karte lokalisiert werden muss, um die neuen Sensorinformationen korrekt einfügen zu können [15].

Cartographer ist ein SLAM-Verfahren, welches von Google entwickelt und im Jahr 2016 veröffentlicht wurde. Das Verfahren beruht auf Teilkarten (Submaps), die in zeitlicher Abhängigkeit aus Lidarscans erstellt werden. Die Teilkarten werden mit einem Loop-Closure-Verfahren zusammengefügt [25].

Außerdem kann Carthographer Sensordaten von mehreren Robotern parallel nutzen, um Karten daraus zu erstellen. Beim Kreuzen der Roboter-Pfade werden die Karten zu einer gemeinsamen Karte zusammengefügt [19]. Diese Herangehensweise wird in der Simulation zur Generierung der gemeinsamen Karte genutzt.

Teilkarten

Die Teilkarten sind Rasterkarten, die durch Laserscans iterativ erweitert werden. Dazu wird mit der Hilfe von Abgleichen der Scans mit der bestehenden Karte (Scan-Matching) die Scan-Position optimiert. Das führt zu einer Verbesserung der Position und Genauigkeit. Als Zusatzinformation können weitere Sensoren wie eine IMU hinzugenommen werden [25].

Ein Pixel in der Karte hat dabei drei mögliche Zustände, die nach dem Frontier-Approach definiert sind, siehe Unterabschnitt 3.2.3.

[2] englisch für Simultaneous Localization and Mapping

Für jeden Pixel wird eine Wahrscheinlichkeit über den Zustand zugeordnet. Mit neuen Sensordaten erhöhen sich die Wahrscheinlichkeiten der Zustände. Erreicht diese Wahrscheinlichkeit eine bestimmte Grenze, wird die Erstellung der Teilkarte beendet und es wird eine neue Teilkarte angelegt [25].

Loop-Closure-Verfahren

Wenn die Erstellung einer Teilkarte abgeschlossen ist, wird im globalen Loop-Closure-Verfahren die Position der Teilkarte angepasst. Die Orientierung der Karte wird in der X-Y-Ebene und im Winkel angepasst und ermittelt in welcher Position die größte Übereinstimmung mit anderen Teilkarten besteht [25].

Die Optimierung wird nach dem Branch-and-Bound-Verfahren durchgeführt, der aus Baumstrukturen das Optimum zurückgibt. Der Baum hält alle Ergebnisse in den Blättern und die Kosten der Ergebnisse setzen sich aus der Summe der Kanten-Kosten zum Blatt zusammen. Anders als bei einer Breiten- oder Tiefensuche muss nicht jeder Knoten im Baum besucht werden. Bei einer Verzweigung können teilweise Teilbäume von der Suche ausgeschlossen werden, falls die Kantenkosten des Teilbaums größer ist als ein gefundenes Ergebnis [35, 31]. Zur Beschleunigung kann der Algorithmus auch schon früher beendet werden, wenn eine hinreichende Lösung gefunden ist [25].

3.6 Auktionsmechanismen

Auktionen kommen aus der Marktwirtschaft und wurden schon vor Jahrhunderten genutzt, um Waren und Güter zu verkaufen. Ein Käufer hat das Ziel, das Objekt so günstig wie nur möglich zu bekommen, um einen maximalen Gewinn zu erzielen. Dabei muss der Käufer zuvor seinen Wert für das Objekt schätzen. Dieser Wert ist in der Regel geheim und kann nicht von anderen Käufern gesehen werden. Der Verkäufer / Auktionator hat das Ziel, das Objekt so teuer wie nur möglich zu verkaufen, um einen maximalen Gewinn zu erzielen. Der Höchstbietende gewinnt den Zuschlag und bekommt das Objekt [53].

In der Informatik werden Auktionen genutzt, um z.B. Ressourcen zuzuordnen. Dabei wird eine Ressource oder Aufgabe vom Auktionator mehreren Käufern angeboten, die

abhängig von dem Auktionstyp ein Gebot abgeben können [5]. Der Einsatz von Auktionsmechanismen ist in der Informatik durch die simplen Regeln und den eleganten Ergebnissen bewährt [42]. Auktionen sind Modelle, um die Interaktionen zwischen Käufern und Verkäufern hinsichtlich rationalen, strategischen Verhaltens und Interessen abzubilden [30].

3.6.1 Eigenschaften

Für die Modellierung eines Auktions-Algorithmus gibt es einige Eigenschaften, die beachtet werden müssen. Ein Teil ist die Bestimmung des Gewinners, dieses kann ein oder auch mehrdimensional sein, indem mehrere Faktoren wie z.B. Kosten und Qualität die Entscheidung beeinflussen [5].

Außerdem gibt es verschiedene Möglichkeiten, wie der Preis festgelegt wird, welcher vom Höchstbietenden bezahlt werden muss. Bei Erstpreisauktionen wird der gebotene Preis gezahlt. Bei Zweitpreisauktionen muss nur der zweithöchste Preis gezahlt werden. Dadurch zahlt der Sieger den Preis, den er benötigt hätte, um sich gegen die Konkurrenz durchzusetzen, auch wenn die Bereitschaft für ein höheren Preis da gewesen wäre [5]. Dieses Konzept lässt sich auf beliebig viele Positionen der Preise übertragen und wird „k^{th}-price auction“ genannt [48].

In einer offenen Auktion geben die Bietenden ihr Gebot offen ab. Die Mitbietenden bekommen dadurch zusätzliche Informationen, was gegebenenfalls ihr Gebot beeinflussen könnte. Als Alternative zur offenen Auktion gibt es die geheime Auktion. Bei dieser Auktionsart werden die Gebote nur dem Auktionator übermittelt und die anderen Mitbietenden bekommen keine Informationen [48].

3.6.2 Strategien des Bieters in geheimen Auktionen

In geheimen Auktionen gibt es eine grundsätzliche Strategie, wie viel ein Bieter setzt. Jeder Bieter ermittelt einen geschätzten Gegenwert des Objektes. Da jeder Bieter einen Gewinn aus der Auktion ziehen will, sollte das Gebot niedriger sein als der geschätzte Wert.

$$b = \frac{N-1}{N} * v \tag{3.6}$$

Das Gebot setzt sich aus der Multiplikation des Gegenwertes v mit der Anzahl der Bieter N zusammen. Nehmen mehr Bieter an der Auktion teil, ist die Gewinnchance für jeden Bieter geringer. Die Bieter erhöhen ihren Einsatz auf Kosten der Gewinnmarge [42]. Solange alle Bieter die gleiche Strategie verwenden, wird immer der Bieter siegen, der auch den höchsten geschätzten Wert hat.

4 Konzept

4.1 Systemarchitektur

Die Architektur des Systems besteht aus verschiedenen ROS-Nodes, die untereinander Nachrichten über Topics austauschen. In Abbildung 4.1 ist die Architektur abgebildet.

Die gesamte Simulation läuft in der „stage“-Node. Diese stellt die Sensordaten der Roboter zur Verfügung und bietet einen Steuerungsinput für die Roboter über das Topic „/robot_{0..N}/cmd_val“.

Die „cartographer“-Node erstellt aus den Sensordaten der Roboter eine gemeinsame Karte. Die Karte dient als Grundlage für die Aufklärungsstrategien. Aus der Karte werden mittels der „frontier_filter “-Node die möglichen Zielpunkte ermittelt und an die Roboter weitergegeben.

Jeder Roboter ermittelt in Abhängigkeit der Erkundungsstrategie seinen nächsten Wegpunkt für die Erkundung der Karte. Der Wegpunkt, den der Roboter ansteuern soll, wird an die „move_base“-Node weitergegeben. Eine Costmap ist eine Rasterkarte in dem Kosten für die Bewegung des Roboters eingetragen sind. Die move_base-Node navigiert den Roboter mit Hilfe von globalen und lokalen Costmaps zu dem Wegpunkt. Nach Erreichen des Wegpunktes wird der nächste Wegpunkt angesteuert, bis die Karte erkundet ist. Felder in der Nähe von Hindernissen haben hohe Kosten zur Vermeidung einer Kollision.

Die „exploration_tracker“-Node speichert den Fortschritt der Erkundung und die zurückgelegte Distanz der einzelnen Roboter. In der „robot_assigner“-Node unterscheiden sich die Strategien voneinander. In den nächsten Unterabschnitten werden die Abläufe der „robot_assigner“-Node genauer beschrieben.

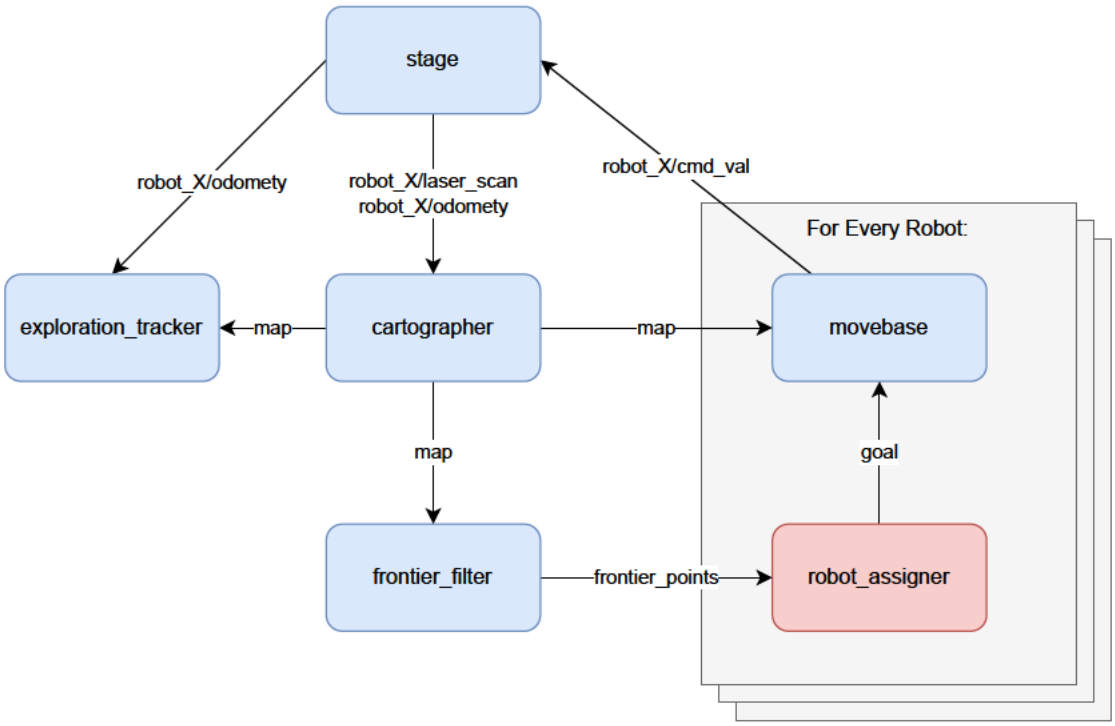


Abbildung 4.1: Systemarchitektur in ROS-Nodes unterteilt. Die Pfeile zwischen den Nodes stellen die ausgetauschten Topics dar.

4.2 Erkundungsstrategien

4.2.1 Ablauf der Zuweisungsstrategien

Der Ablauf des „robot assigner“-Node aus Abbildung 4.1 ist für alle Strategien gleich. In Abbildung 4.2 ist dieser Ablauf als Flussdiagramm grafisch dargestellt.

Der Kontrollfluss wird von zwei Schleifen geleitet. In der Hauptschleife (rote Pfeile) werden alle weiteren Schritte ausgeführt, bis die ROS-Software gestoppt wird. Die innere Schleife (blaue Pfeile) überwacht die Abbruchbedingungen für die Fahrt zum Ziel.

Für die Zielsetzung wird auf die Übermittlung der Zielpunkte gewartet. Mit den verfügbaren Punkten kann das Ziel bestimmt und dem Roboter zugeteilt werden.

Nach der Zuteilung wird dem „move-base“-Node die Zielkoordinaten gesendet und der Roboter startet die Navigation. Bei der Navigation des Roboters zu seinem Ziel gibt es mehrere Bedingungen, die den Abbruch der Fahrt erzwingen können:

- Die Position des Roboters ist weniger als 1m von Ziel entfernt.
  → Der Roboter hat sein Ziel erfolgreich erreicht.
- Die Navigation wurde durch äußere Einflüsse abgebrochen.
  → Stoppen der Simulation.
- Der Roboter hat sich seit acht Sekunden nicht weiterbewegt.
  → Der Roboter steckt vermutlich fest.
- Das Ziel des Roboters wurde in 45 Sekunden nicht erreicht.
  → Das Ziel ist nicht zu erreichen.

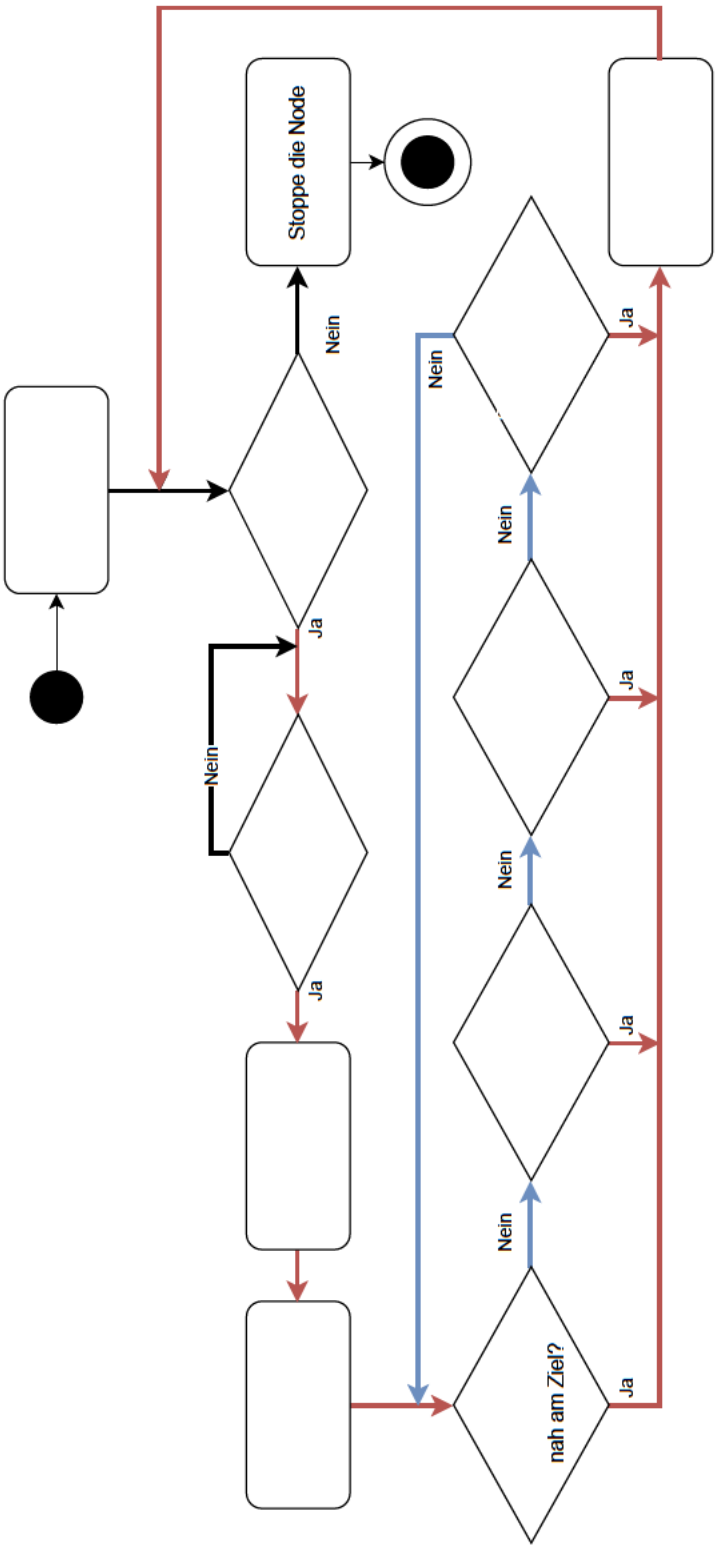


Abbildung 4.2: Allgemeiner Ablauf der Node „robot assigner“. Die Strategien unterscheiden sich nur in der Berechnung für das Ziel.

4.2.2 Zufällige Zuweisung

Bei der zufälligen Strategie wählt jeder Roboter sein eigenes Ziel zufällig aus der Liste verfügbarer Punkte aus. Die Wahl der Roboter könnte auf dieselben Ziele fallen, da keine Rücksicht auf eine Aufteilung genommen wird.

4.2.3 Ungarische Methode

Die Umsetzung der Ungarischen Methode basiert auf der Veröffentlichung von Batinović et al. [6]. Der Ablauf des Zuordnungsalgorithmus ist im Unterabschnitt 3.2.4 beschrieben.

Mit dieser Methode berechnet jeder Roboter die optimale Zuweisung alle Roboter und kommt bei gleichen Voraussetzungen (Gleiche Positionen der Roboter und gleiche Grenzpunkte) auf ein identisches Ergebnis.

Vor- und Nachteile

Weil jeder Roboter seinen eigenen Zielpunkt berechnet, ist dieser scheinbar unabhängig von anderen Robotern. Leider ist diese Berechnung nur scheinbar unabhängig, da für die Berechnung die Position und das aktuelle Ziel der anderen Roboter benötigt werden. Ohne diese Informationen kann die Kostenmatrix nicht vollständig erstellt werden. Die Anzahl der Roboter könnte für die Berechnung variabel gehalten werden, um auf Ausfällen von Robotern reagieren zu können.

Zu den Nachteilen gehört der erhöhte Rechenaufwand, da jeder Roboter die Ziele aller Roboter berechnen muss. In der zentralen Version, könnte ein Roboter die Berechnung übernehmen und allen anderen Robotern Ziele zuweisen.

4.2.4 Auktions-Algorithmus

Die Roboter ersteigern als Bieter die Zielpunkte. Bei der Versteigerung kann ein Roboter mehrere Zielpunkte gleichzeitig erlangen. Die ersteigerten Punkte werden in einer Warteschlange nacheinander abgearbeitet.

Ein Roboter kann zwei Rollen einnehmen, die des Auktionators und die des Bieters, bei dem sich die Abläufe unterscheiden. Die Roboter, die die Rolle des Auktionators einnehmen sind auch gleichzeitig Bieter und können Zielpunkte ersteigern.

Auktionator

In Abbildung 4.3 ist der Ablauf eines Auktionators abgebildet. Die Auktion der Grenzpunkte läuft dezentral über die Roboter ab. Hat ein Roboter keinen zugewiesenen Zielpunkt, eröffnet dieser eine neue Auktion, in der er als Auktionator fungiert. Existiert schon eine laufende Auktion, wird keine neue eröffnet.

Zum Start einer Auktion werden allen Robotern die verfügbaren Zielpunkte vom Auktionator zugeschickt, damit alle Roboter mit den gleichen Daten arbeiten. Danach wird gewartet, bis Gebote von allen Robotern eingegangen sind oder die Gebotszeit abgelaufen ist.

Für die Auswertung werden alle Gebote in einer Matrix gesammelt und für jeden Zielpunkt wird das Höchstgebot ermittelt.

$$B_{ij} = \begin{Bmatrix} b_{0,0} & \dots & b_{i,0} \\ \vdots & \ddots & \vdots \\ b_{0,j} & \dots & b_{i,j} \end{Bmatrix} \begin{matrix} \max() \rightarrow \\ \max() \rightarrow \\ \max() \rightarrow \end{matrix} \begin{Bmatrix} i \\ \vdots \\ i \end{Bmatrix} = \text{results}, \begin{matrix} i \in n\_\text{roboter} \\ j \in \text{Grenzpunkte} \end{matrix} \tag{4.1}$$

In Gleichung 4.1 ist B_{ij} die Matrix mit allen Geboten der Roboter 0 bis i und der Grenzpunkte 0 bis j. Der Index des höchstbietenden Roboters wird im *results*-Tupel gespeichert und das Tupel wird an allen Robotern übermittelt.

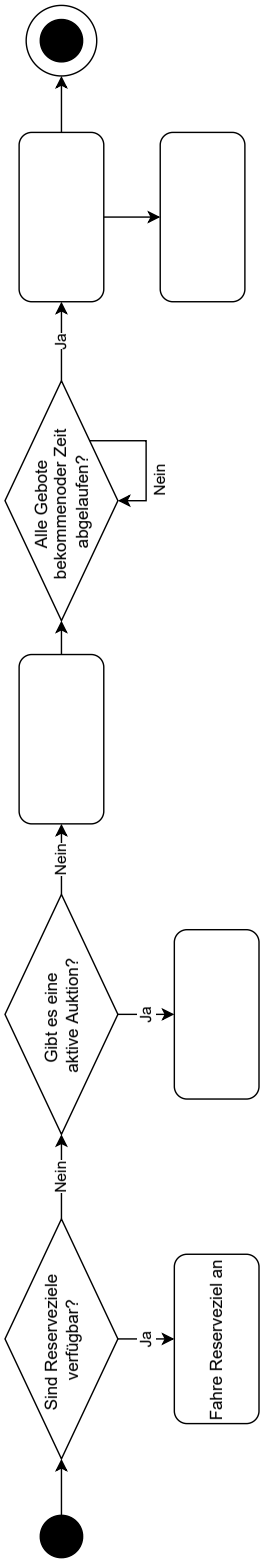


Abbildung 4.3: Ablauf des Auktionators

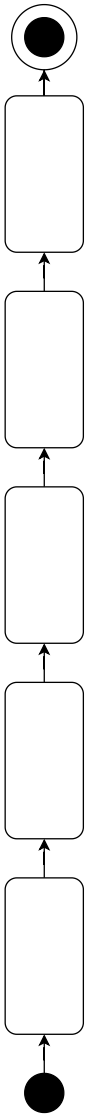


Abbildung 4.4: Ablauf des Bieters

Bieter

Die Bieter prüfen beim Start der Auktion, ob diese Auktion auch valide ist. Wurde z.B. zuvor schon ein Auktionsstart empfangen, wird geschaut welcher Auktionator den niedrigeren Index hat und damit höherer priorisiert wird. Dadurch werden zwei parallele Auktionen vermieden.

Damit der Bieter einen angemessenen Wert bieten kann, ermittelt dieser zunächst seinen persönlichen Gegenwert mittels einer Kostenfunktion. Die Kostenfunktion ähnelt der der ungarischen Methode, welche in Unterabschnitt 3.2.4 beschrieben ist. Nach der Bewertung aller Zielpunkte werden nur die günstigsten weiter betrachtet und das Gebot für diese Punkte berechnet.

Für jede Auktion steht jedem Roboter ein Budget zur Verfügung. Falls der Roboter noch weitere Zielpunkte in Reserve hat, wird dem Budget ein Teil abgezogen (Gleichung 4.3).

Für die Bewertung der Zielpunkte wird das Budget abhängig von den Kosten auf die günstigsten Zielpunkte aufgeteilt (Gleichung 4.4). Daher wird mit dem Kehrwert der Kosten durch die Summe aller Kehrwerte der Kosten dividiert, um dem optimalen Zielpunkt dem größten Anteil des Budgets zu geben.

Zuletzt wird auf dem geschätzten Gegenwert die Strategie angewendet, die in Unterabschnitt 3.6.2 genauer erläutert wird. Daraus wird das finale Gebot auf den Zielpunkt berechnet.

$$k_j = C_{ij} - U_j + F_{ij} \tag{4.2}$$

$$\text{Kapital} = K = 10000 - 3000 \cdot \text{Anzahl Reservepunkte} \tag{4.3}$$

$$\text{Bewertung} = b_j = K \cdot \frac{k_j^{-1}}{\sum_{n=0}^{N} k_n^{-1}}, N = \text{Anzahl Grenzpunkte} \tag{4.4}$$

$$\text{Gebot} = g_j = b_j \cdot \frac{n_{robots} - 1}{n_{robots}} \tag{4.5}$$

Vor - Nachteile

Der Auktionsmechanismus läuft dezentralisiert ab und jeder Roboter muss nur die Kosten für sich berechnen, im Gegensatz zur Ungarischen Methode, bei der jeder Roboter die

Kosten aller anderen Roboter auch berechnen muss. Der Rechenaufwand skaliert mit der Anzahl der möglichen Zielpunkte, daher dauert die Berechnung auf größeren Karten länger. Für die Steigerung der Effizienz fahren die Roboter während einer Auktion nahe Zielpunkte ab, um nicht warten zu müssen.

Durch die Verzögerung der systematischen Abarbeitung der Zielpunkte, kann der Zielbereich schon durch andere Roboter erkundet worden sein. Um die Verzögerung zu begrenzen, ist eine maximale Anzahl an Punkten in der Warteschlange erlaubt. Zudem werden nach jeder Auktion die alten Punkte aus der Warteschlange aktualisiert. Das ermöglicht die Neuvergabe von Zielpunkten an höherbietende Roboter.

4.3 Metrik zur Bewertung

Zur Bewertung der Strategien werden zwei unterschiedliche Metriken verwendet. Primär wird nach der Zeit verglichen, die die Roboter für die Erkundung benötigen. Die Messung der Zeit ist das wichtigste Vergleichskriterium zwischen den unterschiedlichen Strategien.

Sekundär wird die Strecke verglichen, die die Roboter zurücklegen. Eine hohe Distanz spricht für das mehrfache Abfahren vieler Abschnitte, die bereits erkundet sind. Daher ist der Vergleich mit der zurückgelegten Strecke ein Indiz, ob ein Algorithmus die Roboter häufiger durch erkundete Abschnitte fahren lässt. Ein weiterer Grund für die Berücksichtigung der zurückgelegten Strecke ist der Energieverbrauch. Jede Bewegung eines Roboters verbraucht Energie, welche sich durch intelligente Wegplanung reduzieren lässt. Die Betrachtung des Verbrauches gewinnt immer mehr an Bedeutung in den Entscheidungen von Systemstrukturen, wie z.B. im Entwurf von Robotern [33].

Die prozentuale Abdeckung der Karte errechnet sich aus der Größe der Karte und der Anzahl der freien und besetzten Punkte. Da es bei der Erstellung der Teilkarten und dem Loop-Closure-Verfahren es zu kleinen Abweichung kommt, kann die vollständige Abdeckung der Karte von 100 Prozent abweichen.

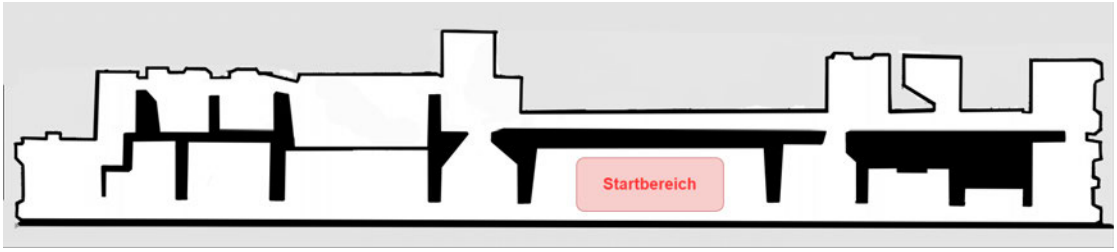


Abbildung 4.5: Belgioioso Castle - Maße ca. 30m x 10m

4.4 Wahl der Karten

Die Strategien werden auf zwei unterschiedlichen Karten getestet, um Unterschiede zwischen den Algorithmen hervorzuheben. Auf beiden Karten starten die Roboter in einem gemeinsam in einen Bereich, welcher auf den beiden Abbildungen markiert ist.

Die erste Karte „Belgioioso Castle “ (Abbildung 4.5) hat Heahnel erstellt [22] und wurde auch von Batinović et al. verwendet. Dort sind viele Räume miteinander verbunden und alle nah beisammen. Die Schwierigkeiten bei dieser Karte stellen vor allem die engen Übergänge dar, die zwischen den Räumen zu finden sind. Der Roboter fährt für die Erkundung von Raum zu Raum. Durch die Offenheit der Räume ist der Raum beim Betreten des Roboters vollständig erkundet.

Die zweite Karte „Area “ (Abbildung 4.6) ist selbst gezeichnet und umfasst ein größeres Gebiet mit längeren Gängen und einzelnen Räumen. Dadurch müssen die Roboter oft mehrere Wegpunkte in einem Gang abfahren, um in den nächsten Raum zu kommen. Die größeren Räume sind durch Wände oder Hindernisse unterteilt, sodass die Roboter für die Erkundung des gesamten Raums weiterfahren müssen. Außerdem sind die Gänge etwas breiter angelegt, sodass die Roboter es einfacher haben auszuweichen, falls sie in entgegen gesetzten Richtungen fahren. Hier soll die Komplexität darin liegen, dass sich die Roboter zusätzlich in die Räume bewegen müssen und nicht nur dran vorbei zum nächsten Raum navigieren müssen.

Die Ergebnisse von Kulich et al. [29] und Kasperski et al. [27] zeigen, dass die Performance von Erkundungsstrategien auch von der Komplexität der Karte abhängt. Daher werden für eine bessere Untersuchung der Algorithmen zwei verschiedenen Karten verwendet.

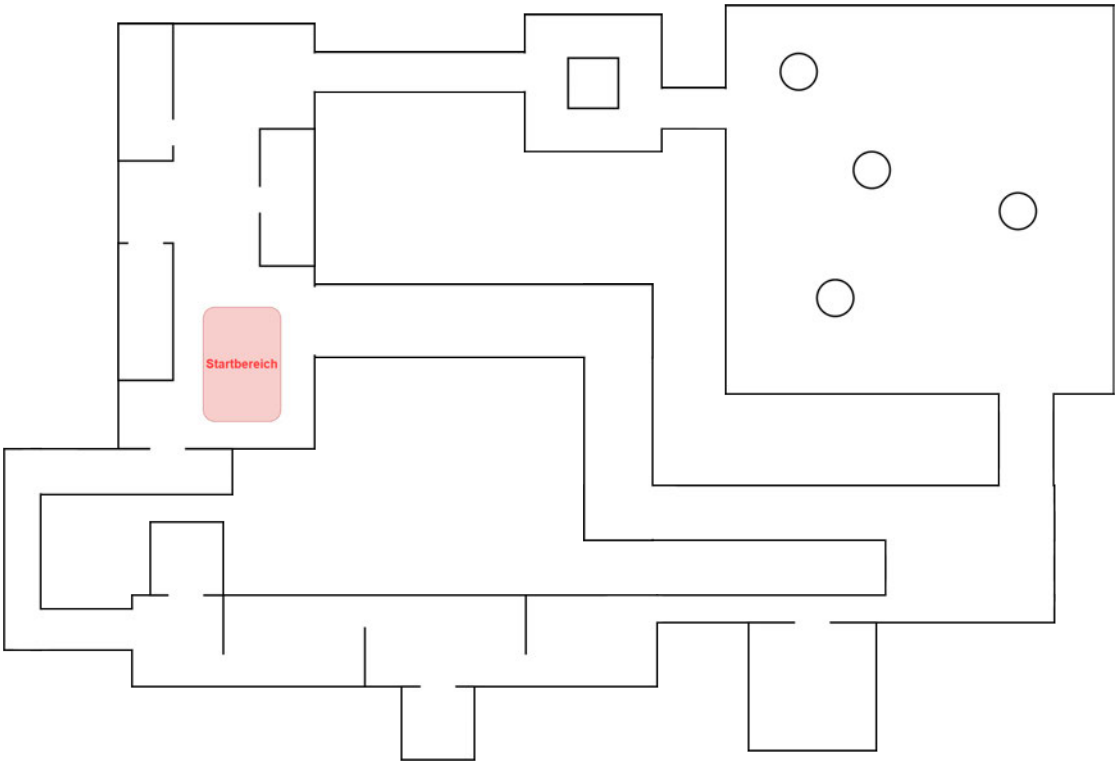


Abbildung 4.6: Area - Maße ca. 45m x 25m

4.5 Unterschied zwischen der Simulation und der realen Welt

Da die Erkundung in einer Simulation abläuft, bei der die Roboter die absolute Position zur Karte haben, spiegelt dieses nicht die reale Welt dar. Für eine Umsetzung in die reale Welt, wäre eine absolute Orientierung, wie z.B. über GPS für Außeneinsätze von Vorteil. Innerhalb von Gebäuden wäre eine absolute Orientierung über Landmarken möglich. Landmarken sind Besonderheiten (features) in der Umgebung, die aus den Sensordaten extrahiert und gut wiedererkannt werden können. Die Besonderheiten könnten unter anderem Schilder oder bestimmte Objekte wie z.B. eine auffällige Farbe. [49]

Durch den zentralen SLAM-Algorithmus ist die Umsetzung in der Simulation vereinfacht. Bei einer dezentralen Umsetzung des SLAM-Algorithmus kann der Austausch von Daten und Positionen schwieriger werden, da jeder Roboter über sein eigenes Koordinatensystem verfügt.

5 Durchführung

5.1 Automatisierung

Um den Aufwand der Simulationen gering zu halten, sollen diese automatisiert in der CI-Pipeline von Github durchgeführt werden. Dadurch ist es möglich, dass z.B. nach jedem Push oder abhängig von der Tageszeit die Simulationen durchgeführt und die Ergebnisse als Tabelle oder direkt in Diagrammen gespeichert werden.

5.1.1 ROS-Tests

Für die Automatisierung in ROS können ROS-Tests verwendet werden. Diese basieren für C++ auf dem gtests[1]- und für Python auf dem unittests[2]-Framework. Die rostest-Dateien sind Erweiterungen von Launch-Dateien, mit denen das System und die einzelnen Nodes zusammen gestartet werden können [41]. Mit diesem System können einzelne Nodes, aber auch ganze Systemtests und Integrationstests erstellt werden [40]. Als Beispiel könnten Daten aus ROS-Bags[3] zum Testen spezieller Bedingungen abgespielt werden. Außerdem könnten auch ganze Simulationen zum Testen des Systems gestartet werden.

Für die Testklassen stehen „assert“-Funktionen zum Abfragen von Bedingungen bereit [45]. Als Bedingung kann eine Variable oder auch ein ROS-Topic abgefragt werden. Erst wenn diese Bedingung erfüllt ist, ist der Test beendet[4]. In diesem Fall kann die Bedingung sein, dass 99 Prozent der Karte erkundet sein muss. Ist diese Bedingung nach 600 Sekunden immer noch nicht eingetreten, schlägt der Test fehl und wird abgebrochen.

Für die Automatisierung startet ein Skript den Test und gibt die Karte, die Strategie und die Anzahl der Roboter vor. Dazu lässt sich im Skript die Anzahl der Testdurchläufe

[1] GoogleTest Framework: https://github.com/google/googletest
[2] Standard Bibliothek von Python: https://docs.python.org/3/library/unittest.html
[3] Aufgezeichnete ROS-Topics, die erneut abgespielt werden können, um z.B. Sensoren zu simulieren.
[4] Zusätzlich kann noch ein Zeitlimit vorgegeben werden

einstellen, sodass jeder Test erneut mit z.B. zufälligen Parameter ausgeführt wird. Die Metriken (Distanz und Zeit) werden von einer ROS-Node für die spätere Auswertung in eine CSV-Datei abgespeichert.

5.1.2 Grenzen des Systems

Die von Github zur Verfügung gestellte Rechenleistung für die flüssige Ausführung der Simulation ist leider nicht ausreichend. Die kostenlosen Leistungen[5] begrenzen sich auf zwei Kerne, 7 GB RAM und 14 GB SSD Festplattenspeicher. Da es mit mehreren Robotern zu Problemen bei der Fahrt und der Orientierung in der Simulation kommt, wurde die Ausführung auf einen lokalen Desktop-PC umgestellt.

5.2 Ergebnisse

Es wurden Simulationen auf zwei Karten mit drei verschiedenen Algorithmen und vier verschiedenen Anzahlen an Robotern durchgeführt. Daraus ergeben sich 24 verschiedene Szenarien. Auf der Karte Belgioioso wurde jedes Szenario dreißigmal durchgeführt, auf der eigenen Karte „Area “ zehnmal. So ergeben sich daraus insgesamt 480 Durchgänge.

Der Grund für die unterschiedliche Anzahl an Durchgängen bei den Karten liegt an deren Größe. Die erkundbare Fläche von „Area“ ist um den Faktor 3,5 größer als Belgioioso. weshalb die Erkundung mehr Zeit in Anspruch nimmt. Durch die größere Karte sind bei der Berechnung mehr Zielpunkte vorhanden, was die Zeit für die Zuordnung erhöht.

5.2.1 Validierung der Simulation

Bedingungen

Da diese Arbeit auf Simulation ausgelegt ist, müssen die Ergebnisse validiert werden können. Die Simulation in dieser Arbeit basiert auf der Veröffentlichung von Batinović et al. [6].

[5] https://docs.github.com/en/actions/using-github-hosted-runners/about-github-hosted-runners#supported-runners-and-hardware-resources

Der Vergleich wurde mit derselben Karte (Abbildung 4.5), mit demselben Algorithmus und derselben Anzahl an Robotern durchgeführt.

Die Hardware für die Ausführung der Simulation und die Startposition der Roboter unterscheiden sich in dem Vergleich. Bei diesem Versuch starten die Roboter in einem gemeinsamen Raum, während die Roboter bei Batinović et al. verteilt auf der Karte starten.

Das realistischere Startszenario für die Roboter ist ein gemeinsamer Start in einem Raum oder am Anfang einer Karte. Dort werden die Roboter eingeschaltet und suchen nach Wegen und kartieren die Umgebung. Der Start der Roboter an verschiedenen Orten wäre ein Sonderfall und in der Realität wesentlich komplexer, da die Roboter die absoluten Positionen der anderen Roboter nicht kennen. Bei einem gemeinsamen Start hätten die Roboter eine Referenz über das gemeinsame Umfeld.

Außerdem sollte der gemeinsame Start eine bessere Unterscheidung zwischen den Strategien liefern. Für die effiziente Erforschung der Karte müssen sich die Roboter aktiv für verschiedene Richtungen entscheiden. Bei einem verteilten Start könnten die Roboter zum gleichen Wegpunkt steuern und trotzdem würde ein großer Teil der Karte erkundet werden.

Vergleich

In Abbildung 5.1 sind die Ergebnisse von Batinović et al. und in Abbildung 5.2 sind die Ergebnisse aus der eigenen Durchführung dargestellt. Für den Vergleich interessant sind in Abbildung 5.1 nur die Ergebnisse der dezentralisierten Durchläufe, da diese mit dem gleichen Algorithmus durchgeführt wurden. In den Diagrammen sind die Zeiten für das Erreichen von 50, 75, 90 und 98 Prozent als Boxplots abgebildet. Dazu wird zwischen 2, 3 und 5 Roboter unterschieden.

Der Vergleich mit fünf Robotern zeigt, dass die Erkundungen in Abbildung 5.1 etwas schneller die Abdeckungsgrade erreichen. Der Grund könnte in den unterschiedlichen Startpositionen liegen, denn die Erkundung von 50 Prozent der Karte wird schon innerhalb von Sekunden erreicht. Bei verteilten Startpositionen wird mit erhöhter Anzahl an Robotern der Vorteil des Abdeckungsgrades größer, da jeder Roboter neue Informationen ohne Fortbewegung direkt beitragen kann. In Abbildung 5.1 ist dieses bei 50-prozentiger

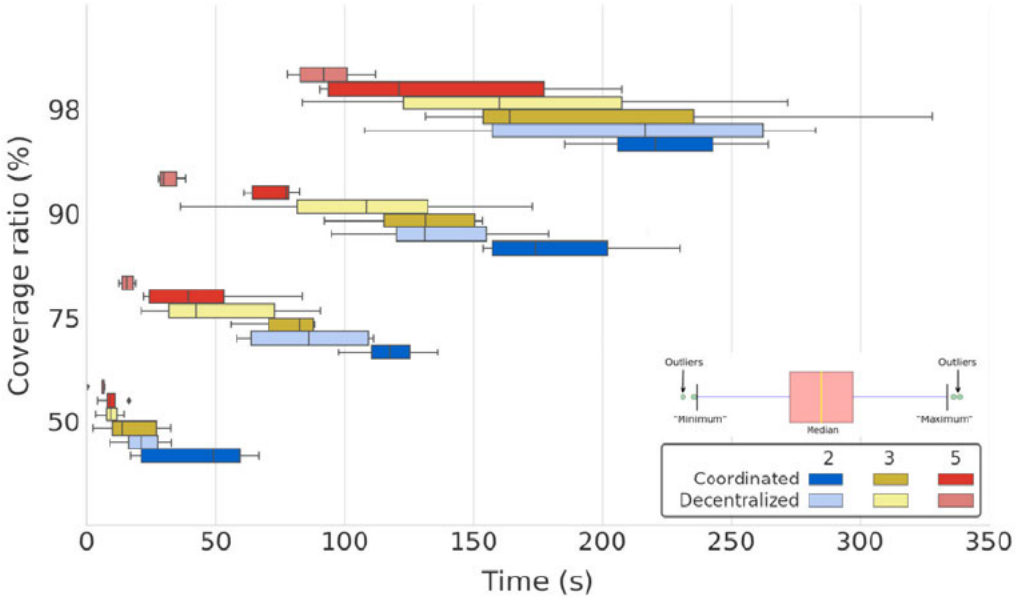


Abbildung 5.1: Ergebnisse von Batinović et. al [6]. Unterscheidung zwischen koordinierten und dezentralen Strategien, die mit 2,3 und 5 Robotern durchgeführt wurden. Dabei sind die helleren Boxplots die relevanten für den Vergleich zwischen den Simulationen.

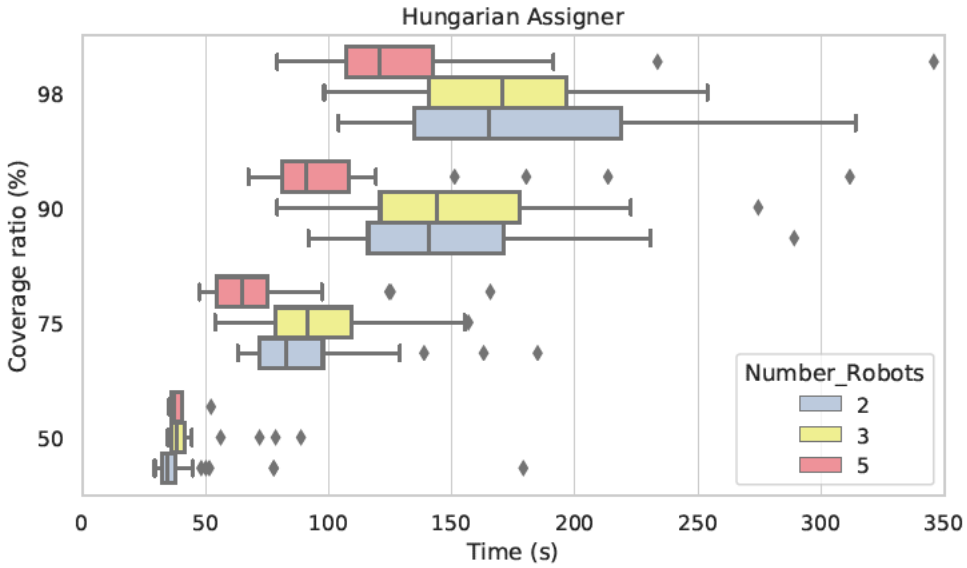


Abbildung 5.2: Eigene Durchführung der Ungarischen Methode auf der Karte Belgioioso (Abbildung 4.5), welche mit 2,3 und 5 Robotern durchgeführt wurde.

Abdeckung zu erkennen, wo sich die Zeit von ca. 25 Sekunden bei zwei Robotern auf ca. fünf Sekunden bei fünf Robotern reduziert.

Der Performanceunterschied mit zwei Robotern ist zwischen Batinović et al. [6] und eigener Simulation marginal. Über alle prozentualen Abdeckungen weichen die mittleren Zeiten ca. zehn Sekunden voneinander ab. Bei zwei Roboter und 98-prozentiger Abdeckung liegen die Zeiten bei beiden Simulationen etwa zwischen 100 und 300 Sekunden. Im Mittel des Box-Plots ist die eigene Simulation aber ungefähr 25 Sekunden schneller.

Die Varianz steigt proportional zu der Zunahme der Kartenabdeckung. Das Aufdecken der letzten 98 Prozent ist viel Zeitintensiver als das Erreichen von 50 Prozent, da die Roboter auch den letzten Raum erkundet haben müssen. Was ungewöhnlich an den Zeiten der 98-prozentigen Abdeckung der Karte mit fünf Robotern in Abbildung 5.1 erscheint, ist die geringe Varianz der Zeit. Ein Anstieg der Varianz zur Abdeckung der Karte ist zu erkennen, aber im Vergleich zwischen der Varianz bei 3 Robotern und bei 5 Robotern zu klein.

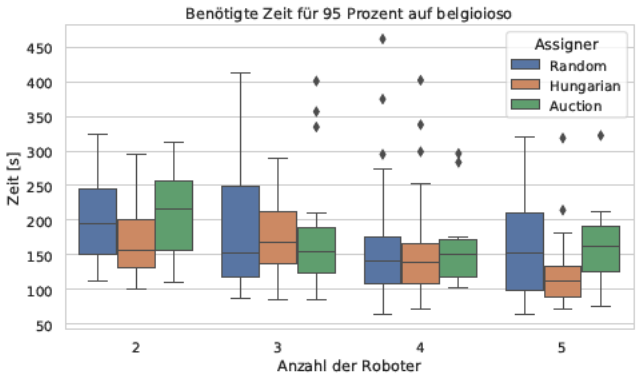
Trotz der genannten Auffälligkeiten kommen es bei beiden Versuchen zu gleichen Ergebnissen. Die Resultate der Arbeit von Batinović et al. sind mit dem gleichen Algorithmus in der eigenen Simulation reproduzierbar. Daher sind die Ergebnisse dieser Arbeit als plausibel einzuschätzen.

5.2.2 Performance auf Belgioioso-Castle

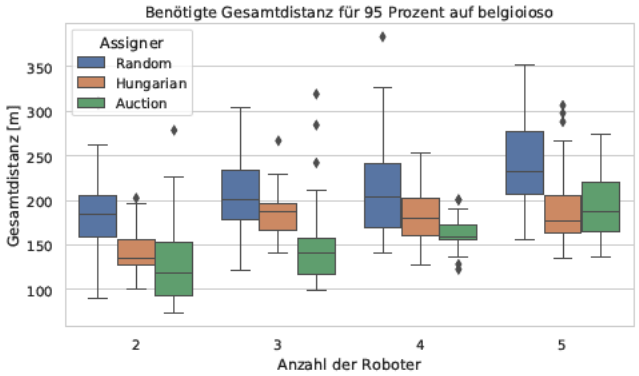
In Abbildung 5.3a sind die Zeiten abgebildet, welche die Roboter für die Aufdeckung von 95 Prozent der Karte benötigt haben. Anders als in Unterunterabschnitt 5.2.1 wurde hier die 95 Prozent Aufdeckung gewählt, da eine 98 Prozent Aufdeckung nicht in allen Versuchen erreicht wurde. Daher sind die 95 Prozent ein Kompromiss für eine Erreichung einer möglichst großen Abdeckung der Karte.

Der Vergleich zwischen den Algorithmen zeigt, dass die Zeiten für die zufällige Zuordnung durchschnittlich nur etwas höher liegen, aber die Varianz stark nach oben abweicht.

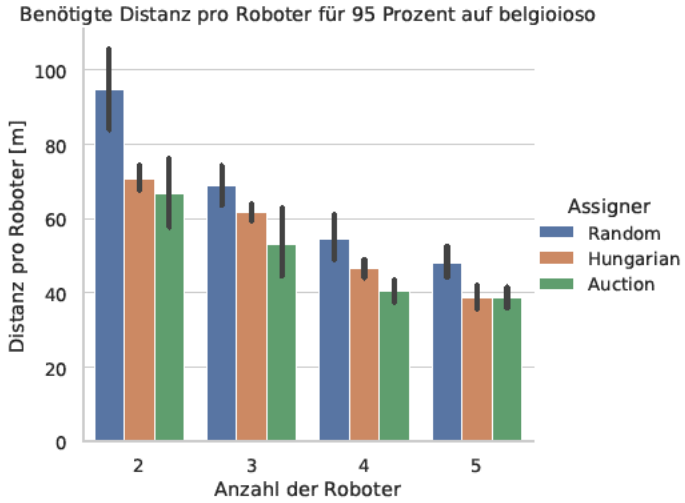
Durch den Einsatz von mehreren Robotern verbessern sich die Zeiten pro weiteren Roboter um durchschnittlich 15 Sekunden. Eine Ausnahme davon bildet der Auktions-Algorithmus, welcher sich bei fünf Robotern etwas verschlechtert. Ein Grund dafür könnte sein, dass die Performance der Simulation aufgrund des aufwendigeren Auktions-Prozesses verlangsamt wird. Der Zeitaufwand des Algorithmus sollte mit der Anzahl der



(a) Benötigte Zeit bis 95% der Karte Belgioioso erkundet ist.



(b) Benötigte Distanz bis 95% der Karte Belgioioso erkundet ist.



(c) Zurückgelegte Distanz pro Roboter auf der Karte Belgioioso. Die schwarzen Säulen am oberen Ende zeigen den Min-Max-Bereich der Durchläufe an.

Abbildung 5.3: In den Diagrammen sind die Ergebnisse der Karte Belgioioso dargestellt.

Roboter wenig skalieren, denn der größte Aufwand liegt in der Abgabe des Gebots, die jeder Roboter parallel berechnet. Eine genaue Analyse zu den Skalierungen über die Anzahl der Roboter ist in Unterabschnitt 5.2.5 nachzulesen. Die Resultate der Ungarischen Methode sind dagegen immer besser, wenn sich die Anzahl der Roboter erhöht.

Neben der Zeit ist die zweite Metrik die zurückgelegte Distanz, bis die Karte erkundet ist. In Abbildung 5.3b sind die Distanzen als Box-Plot aufgeführt, die die Roboter zurückgelegt hatten. Der zufällige Algorithmus hat bei allen Anzahlen an Robotern die höchste zurückgelegte Distanz. Die geringsten Distanzen sind vom Auktions-Algorithmus. Bei den Durchführungen mit fünf Robotern sind die Distanzen zwischen Auktions-Algorithmus und Ungarischer Methode relativ gleich.

In Abbildung 5.3c wird die durchschnittliche Distanz pro Roboter für die verschiedenen Algorithmen dargestellt. Je mehr Roboter eingesetzt werden, desto kleiner wird die zurückgelegte Distanz pro Roboter. Die Abbildung 5.3b zeigt die Zunahme der Gesamtdistanz mit Anzahl der Roboter. Außerdem ist zu erkennen, dass die Distanz pro Roboter beim Auktions-Algorithmus am niedrigsten liegt, wenn auch diese nur marginal kleiner ist im Vergleich zu der Ungarischen Methode.

Der Auktions-Algorithmus ist langsamer als die Ungarische Methode, aber dafür legen die Roboter eine kürzere Distanz zurück. Die durchschnittliche Geschwindigkeit pro Roboter ist in Abbildung 5.4 abgebildet. Die Roboter des Auktions-Algorithmus haben eine langsamere durchschnittliche Geschwindigkeit als bei den anderen Algorithmen. Ein Grund könnten längere Pausen zur weiteren Streckenplanung sein. Andererseits wird beim Auktions-Algorithmus eine geringere Gesamtdistanz zurückgelegt, sodass trotz der niedrigeren Geschwindigkeit eine ähnliche Dauer zur Erkundung der Karte benötigt wird. Dies könnte auch auf eine effizientere Aufteilung der Ziele unter den Robotern hinweisen.

5.2.3 Performance auf der eigenen Karte „Area“

In Abbildung 5.5a sind die benötigten Zeiten für die Aufdeckung von 95 Prozent der Karte „Area“ aufgeführt. Auffällig dabei ist, dass die benötigte Zeit nicht proportional mit der Anzahl der Roboter skaliert. Bei dem zufälligen Algorithmus ist zu erkennen, dass sich mit erhöhter Anzahl an Robotern der Zeitaufwand verringert, jedoch ist bei der

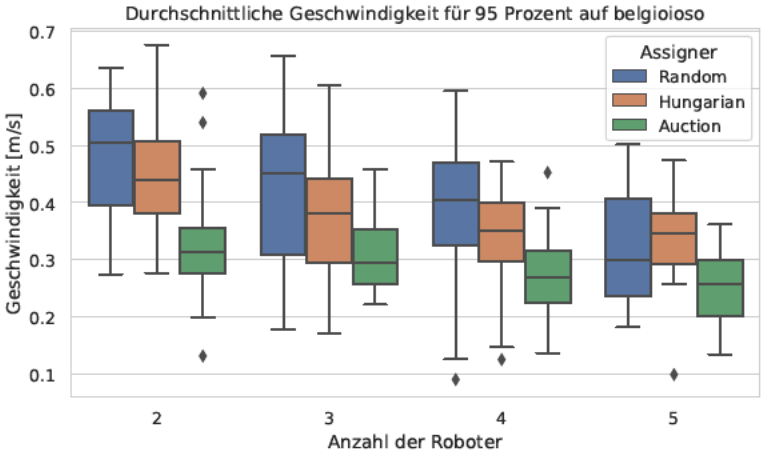


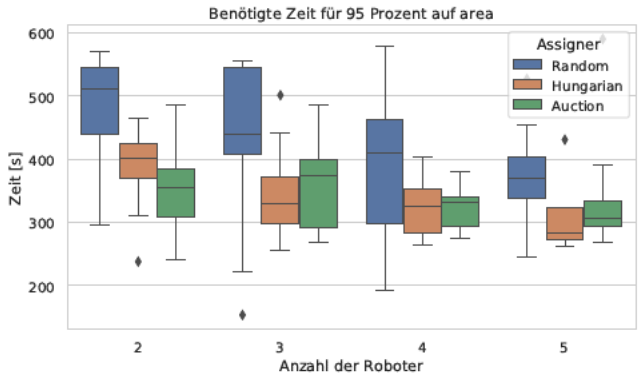
Abbildung 5.4: Durchschnittliche Geschwindigkeit eines Roboters auf der Karte Belgioioso.

zufälligen Strategie auch die Standardabweichung am größten. Die anderen beiden Algorithmen verbessern sich mit der Anzahl der Roboter um durchschnittlich 20 Sekunden pro Roboter.

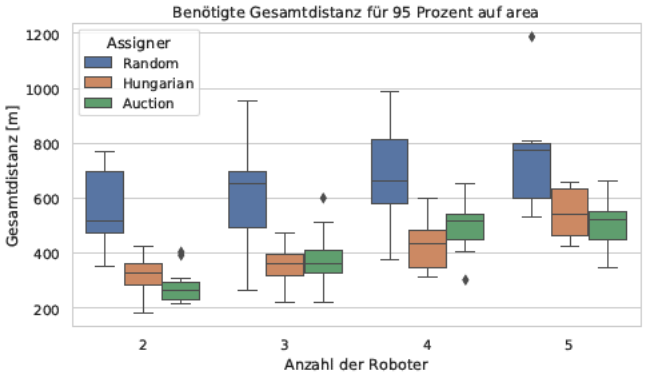
Der Unterschied zwischen dem Auktions-Algorithmus und der Ungarischen Methode ist marginal. Abgesehen von dem Zeitaufwand mit zwei Robotern ist die Ungarische Methode leicht besser als der Auktions-Algorithmus.

In Abbildung 5.5b sind die benötigten Distanzen aufgeführt. Hier ist deutlich zu erkennen, dass der zufällige Algorithmus fast die doppelte Distanz für die Erkundung der Karte benötigt. Durch die Größe der Karte steigern sich die Distanzen zwischen den Zielpunkten. Bei der zufälligen Zuweisung müssen die Roboter weite Wege zurücklegen und im schlimmsten Fall über die gesamte Karte fahren. Außerdem begünstigt der Aufbau der langen Gänge und wenigen Verbindungsräumen die großen Distanzen zusätzlich. Bei einer ungünstigen Zuweisung muss der Roboter zunächst zu einem Raum zurückfahren, um den Zielpunkt im neben liegenden Gang anfahren zu können. Die Luftlinie weicht stark von der Länge der Wegstrecke ab.

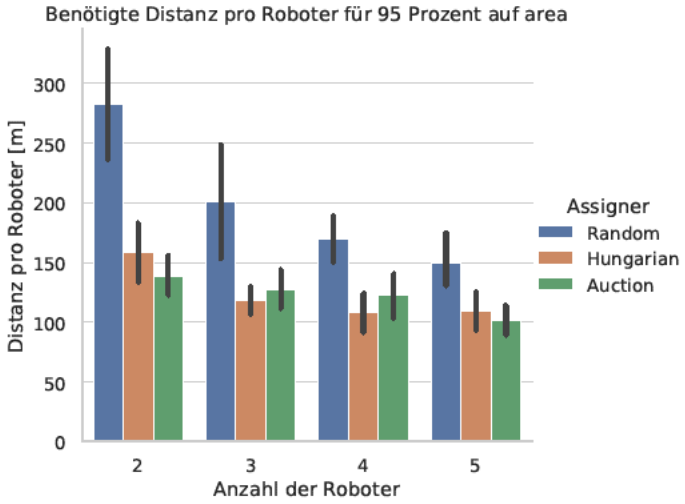
Im Vergleich zwischen dem Auktions-Algorithmus und der Ungarischen Methode wechseln sich die beiden Strategien bei der kürzesten Distanz ab. Daher lassen sich die beiden Strategien in der Gesamtdistanz als gleich gut bewerten.



(a) Benötigte Zeit bis 95% der Karte „Area“ erkundet ist.



(b) Benötigte Distanz bis 95% der Karte „Area“ erkundet ist.



(c) Zurückgelegte Distanz pro Roboter auf „Area“

Abbildung 5.5: In den Diagrammen sind die Ergebnisse der Karte „Area“ dargestellt. Achtung, die Skalierung der Y-Achse unterscheiden sich zu den Ergebnissen von Belgioioso.

5.2.4 Optimale Roboter Anzahl

Im optimalen Fall fährt ein Roboter durchgehend mit der vorgegebenen Geschwindigkeit, ohne eine Pause für die Berechnung des nächsten Ziels einzulegen. Damit wäre die Distanz proportional zur benötigten Zeit, da die Geschwindigkeit konstant wäre.

Für die optimale Erkundung ist eine minimale Distanz zurückzulegen, bei der kaum Bereiche mehrfach befahren werden. Gibt es mehrere Sackgassen, ist eine Fahrt durch erkundetes Gebiet unumgänglich. Bei einer optimalen Fahrt eines einzelnen Roboters sollten die wenigsten Überschneidungen mit erkundeten Gebieten auftreten.

Beim Einsatz mehrere Roboter sollten die Überschneidungen häufiger auftreten. Selbst bei einer optimalen Fahrt mit mehreren Robotern kann das Bewegen auf erkundetem Gebiet nicht vermieden werden. Beim Start teilen sich die Roboter optimal auf die Wege auf. Ist die Anzahl an Roboter größer als die Anzahl der Wege, sind doppelte Fahrten unvermeidlich. Ähnlich ist es auch für den letzten unerkundeten Raum auf der Karte. Die Zielpunkte in diesem Raum werden von allen Robotern angefahren. Mit steigender Anzahl von Robotern konvergiert die benötigte Distanz pro Roboter gegen ein Minimum.

Daraus könnte eine optimale Anzahl an Robotern ermittelt werden, die eingesetzt werden sollte. Dieses ist von der Größe der Karte und der Sichtweite des Roboters abhängig. Die Ermittlung der optimalen Anzahl an Robotern könnte in einer weiterführenden Arbeit betrachtet werden.

5.2.5 Zeitlicher Vergleich

In der Abbildung 5.6 ist der Zeitaufwand eines Roboters für die Bestimmung des nächsten Zielpunkts aufgeführt. Beim Auktions-Algorithmus ist es die benötigte Zeit des Auktionators vom Start der Auktion bis zur Verkündung der Gewinner.

Die Skalierungen aller Strategien sind im Diagramm gut zu erkennen. Die zufällige Zuordnung ist unabhängig von der Anzahl der Roboter, diese Zeit weist einen konstanten Wert um ca. 200 ms auf. Der Rechenaufwand der zufälligen Strategie kann mit $\mathcal{O}(1)$ beschrieben werden.

Der Aufwand bei der Ungarischen Methode steigt mit jedem weiteren Roboter immer stärker an. Der Rechenaufwand hat mindestens einen quadratischen Aufwand $\mathcal{O}(n^2)$. Jeder einzelne Roboter berechnet die optimalen Zielpunkte für alle Roboter mit dem

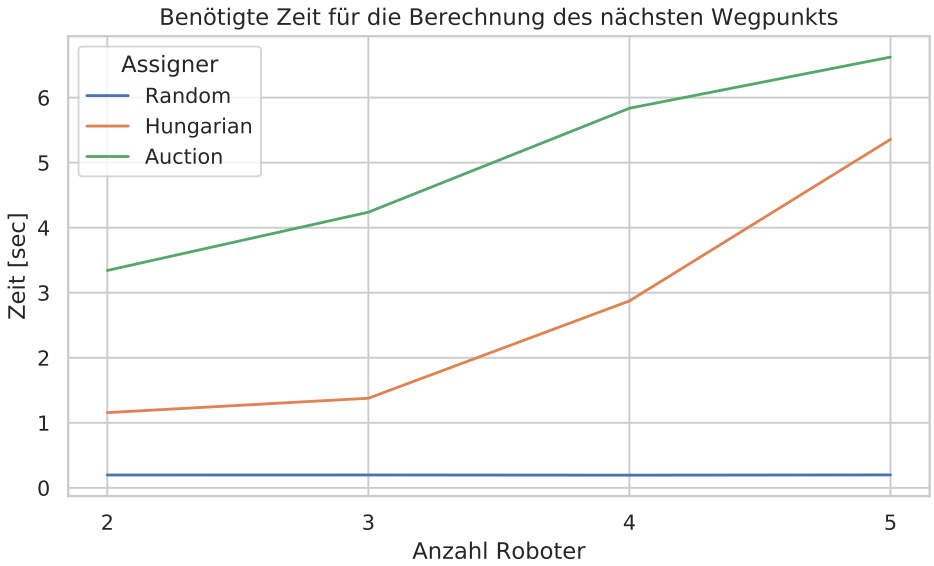


Abbildung 5.6: Benötigte Zeit der Algorithmen, um den nächsten Zielpunkt zu bestimmen.

Zuordungsalgorithmus. Steigt die Anzahl der Roboter wird das Aufstellen der Kostenmatrix und das Lösen mit der Ungarischen Methode aufwendiger. Beim Extrapolieren der Kurven würde die benötigte Zeit der Ungarischen Methode die Zeit des Auktions-Algorithmus überholen.

Die zeitliche Zunahme mit der Anzahl an Robotern steigt bei der Auktionsstrategie linear an und kann mit $\mathcal{O}(n)$ beschrieben werden. Grund hierfür ist, dass die Kostenberechnung weitestgehend auf die einzelnen Roboter parallel durchgeführt wird und der Auktionator bei der Auswertung der Gebote abhängig von der Anzahl der Roboter ist.

Dabei ist aber auch die Kostenberechnung mit einem linearen Aufwand verbunden. Für die Kosten werden die Positionen und Ziele der anderen Roboter mit in die Berechnung einbezogen. In den Rohdaten ist die steigende Zeit zu erkennen, die der Auktionator wartet, um die Gebote zu bekommen. Da alle Teile nur linear steigen, ist der gesamte Aufwand $\mathcal{O}(n)$.

5.2.6 Probleme bei der Durchführung

Bei der Durchführung der Simulationen sind immer wieder Probleme aufgetreten. Das größte Problem waren Kollisionen zwischen zwei Robotern oder mit einer Wand. Bei einer Kollision eines Roboters stoppt dieser und lässt sich nicht mehr navigieren. Wenn von fünf Roboter zwei am Start miteinander kollidieren werden die Ergebnisse durch die resultierende Zeit verfälscht. Daher werden die Durchläufe, in denen Kollision auftreten, herausgefiltert und nicht in den Ergebnissen berücksichtigt.

Bei der Pfadplanung der Roboter ist die Kollision mit anderen Robotern nicht berücksichtigt. Befindet sich ein anderer Roboter auf Kollisionskurs, wird dieser Roboter über den Lidarsensor erkannt. Diese Information wird aber erst als Karte über die „Carthogarpher“-Node an die Wegplanung übergeben. Die Verzögerung des Informationsflusses ist oft so lang, dass beide Roboter miteinander kollidieren.

Die Kollisionen treten vor allem dann auf, wenn sich beide Roboter um eine Ecke bewegen, da die Roboter schnell reagieren müssten. Ein Ausweichen in großen Räumen ist dagegen oft erfolgreich. Wie oft eine Kollision vorkommt, liegt an der Anzahl der Roboter und an dem Aufbau der Karte. Schmale kurvige Abschnitte bieten keine Ausweichmöglichkeiten.

Zum Lösen des Problems müsste die Pfadplanung der Roboter angepasst werden, indem z.B. eine andere / eigene Pfadplanung statt „Move_Base“-Bibliothek [39] verwendet wird, welche bei der Planung besser auf bewegende Hindernisse reagieren könnte. Ein Beispiel wäre die Arbeit von Heinemann et al. [24], jedoch würde die Implementierung den Umfang dieser Arbeit übersteigen.

6 Fazit

6.1 Zusammenfassung

In dieser Arbeit wurde der Vergleich zwischen zwei dezentralen Erkundungsstrategien aufgezeigt. Dabei wurden zunächst der aktuelle Forschungsstand beschrieben und es wurde auf die Grundlagen zu den Erkundungsstrategien eingegangen. Danach wurden das Konzept und der Aufbau eines solchen Systems für die Simulation erläutert. In der Durchführung wurden die Resultate vorgestellt und die Probleme behandelt, welche während der Simulation aufgetreten sind.

Zudem wurde eine eigene Strategie entwickelt, bei der die Roboter gleichgestellt sind und für die Zuordnung der unbekannten Gebiete wechselnd die Rolle eines Auktionators einnehmen.

Als Ergebnis lässt sich aus der Arbeit schließen, dass die Nutzung eines Auktions-Algorithmus eine ähnliche zeitliche Performance zu der vorgestellten Ungarischen Methode von Batinović et al. bietet. Bei der zurückgelegten Distanz pro Roboter agiert der eigene Auktions-Algorithmus teilweise etwas besser als die ungarische Methode. Die Resultate aus der Analyse des Zeitaufwands in Abbildung 5.6 zeigt das Potential von Auktions-Algorithmen für größere Gruppen an Robotern.

Jedoch müssen auch die möglichen Nachteile von Auktions-Algorithmen für die Umsetzung in der realen Welt betrachtet werden. Grade bei größeren Gebieten kann die Verlässlichkeit von Kommunikation nicht vorausgesetzt werden. Das führt vor allem bei Auktionen zu Problemen, da das Senden von Geboten oder die Verkündung des Höchstbietenden essenziell ist.

6.2 Ausblick

Für die Umsetzung in der realen Welt ist eine Abwandlung des Konzeptes unabdingbar. In der Simulation lief der SLAM-Algorithmus zentral ab. Für die Umsetzung in der realen Welt gibt es keine zentrale Einheit. Jeder Roboter baut eine eigene Karte auf, in der er selbst navigiert. Dazu müssen auch noch die Probleme der realen Welt berücksichtigt werden, wie z.B. die Verlässlichkeit der Kommunikation in Höhlen oder Tiefgaragen.

Außerdem waren die Positionsinformationen der Roboter in der Simulation perfekt. Durch die „Stage“-Node musste keine Umwandlung in ein gemeinsames Koordinatensystem erfolgen. Zudem gab es keine Ungenauigkeiten, wie ein Rauschen oder Drift der Positionen, was in der realen Welt beachtet werden müsste.

In dem Aufbau des Auktions-Algorithmus liegt auch noch Potential zur Verbesserung. Dort könnte der Ansatz von Zlot et al. [57] mit einfließen, bei dem es mehrere Auktionatoren gibt, die ihre Zielpunkte an die anderen Roboter versteigern. Deckt ein Roboter ein neues Gebiet auf, werden die neuen möglichen Zielpunkte zu seinen eigenen hinzugefügt. Die Versteigerungen werden regionaler durchgeführt werden und es müssen nicht alle Roboter adressiert werden, da die Interessenten für die Zielpunkte auch nur in der Nähe des Auktionators sind. Daher sind die Auktionen resistenter gegen Kommunikationsprobleme.

Bei großen Gruppen von Robotern offenbart sich ein Potential in hierarchischen Erkundungsstrategien. Dort könnte der Austausch zwischen den lokalen Robotern über einen Auktions-Algorithmus durchgeführt werden, bei dem keine großen Distanzen zu erwarten sind und eine vollvermaschte Topologie möglich ist. Bei dem Austausch zwischen den lokalen Gruppen wäre ein anderer Ansatz möglich, der robuster gegen Kommunikationsproblemen wäre.

Literaturverzeichnis

[1] Ackerman, Evan: *Startup Developing Autonomous Delivery Robots That Travel on Sidewalks*. IEEE Spectrum. November 2015. – URL https://spectrum.ieee.org/starship-technologies-autonomous-ground-delivery-robots. – Zugriffsdatum: 17.12.2022

[2] Ackerman, Evan: *Help Rescuers Find Missing Persons With Drones and Computer Vision*. IEEE Spectrum. November 2019. – URL https://spectrum.ieee.org/help-rescuers-find-missing-persons-through-emergency-response-contest. – Zugriffsdatum: 17.12.2022

[3] Ackerman, Evan: *Your Guide to the DARPA SubT Finals: Meet the Teams*. IEEE Spectrum. September 2021. – URL https://spectrum.ieee.org/darpa-subterranean-challenge?utm_campaign=post-teaser&utm_content=cfptaphk. – Zugriffsdatum: 02.09.2022

[4] Ackerman, Evan: *DARPA Subterranen challenge*. IEEE Spectrum. April 2022. – URL https://spectrum.ieee.org/darpa-subterranean-challenge-2657170650. – Zugriffsdatum: 02.09.2022

[5] Baranwal, Gaurav ; Kumar, Dinesh ; Raza, Zahid ; Vidyarthi, Deo P.: *Auction Based Resource Provisioning in Cloud Computing*. Cambridge, UK : Springer Singapore, 2018. – ISBN 978-981-10-8737-0

[6] Batinović, Ana ; Oršulić, Juraj ; Petrović, Tamara ; Bogdan, Stjepan: Decentralized Strategy for Cooperative Multi-Robot Exploration and Mapping. In: *IFAC-PapersOnLine* 53 (2020), Nr. 2, S. 9682–9687. – URL https://www.sciencedirect.com/science/article/pii/S2405896320333760. – 21st IFAC World Congress. – ISSN 2405-8963

[7] Bautin, Antoine ; Simonin, Olivier ; Charpillet, François: MinPos : A Novel Frontier Allocation Algorithm for Multi-robot Exploration. In: Su, Chun-Yi (Hrsg.) ; Rakheja, Subhash (Hrsg.) ; Liu, Honghai (Hrsg.): *Intelligent Robotics and Applications*. Berlin, Heidelberg : Springer Berlin Heidelberg, 2012, S. 496–508. – ISBN 978-3-642-33515-0

[8] Berhault, M. ; Huang, H. ; Keskinocak, Pinar ; Koenig, S. ; Elmaghraby, W. ; Griffin, Paul ; Kleywegt, A.: Robot exploration with combinatorial auctions, 11 2003, S. 1957 – 1962 vol.2. – ISBN 0-7803-7860-1

[9] Botelho, S.C. ; Alami, R.: M+: a scheme for multi-robot cooperation through negotiated task allocation and achievement. In: *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)* Bd. 2, 1999, S. 1234–1239 vol.2

[10] Burgard, W. ; Moors, M. ; Stachniss, C. ; Schneider, F.E.: Coordinated multi-robot exploration. In: *IEEE Transactions on Robotics* 21 (2005), Nr. 3, S. 376–386

[11] Cao, Y. U. ; Fukunaga, Alex S. ; Kahng, Andrew B. ; Meng, Frank: Cooperative mobile robotics: Antecedents and directions. 1995. – Forschungsbericht

[12] Cieslewski, Titus ; Choudhary, Siddharth ; Scaramuzza, Davide: Data-Efficient Decentralized Visual SLAM. In: *CoRR* abs/1710.05772 (2017). – URL http://arxiv.org/abs/1710.05772

[13] DEEBOT: *DEEBOT - Advanced Mapping & Navigation*. https://meetdeebot.com/advanced-mapping-navigation

[14] DFKI: *DFKI - Anwendungsfelder in der Robotik*. – URL https://robotik.dfki-bremen.de/de/forschung/anwendungsfelder/. – Zugriffsdatum: 12.12.2022

[15] Durrant-Whyte, H. ; Bailey, T.: Simultaneous localization and mapping: part I. In: *IEEE Robotics & Automation Magazine* 13 (2006), Nr. 2, S. 99–110

[16] Elshenawy Elsefy, Ayman ; Mohamed, Khalil ; Harb, Hany: Exploration Strategies of Coordinated Multi-Robot System: A Comparative Study. (2018), 03, S. 48–58

[17] Elshenawy Elsefy, Ayman ; Mohamed, Khalil ; Harb, Hany: An Evaluation of Multi-robot Systems Exploration Algorithms. In: *Journal of Al-Azhar University Engineering Sector* 14 (2019), 04

[18] Gerkey, Brian ; Matari, Maja: MURDOCH: Publish/Subscribe Task Allocation for Heterogeneous Agents. (2000), 06

[19] Google-Cartographer: *Cartographer - Multi-trajectories SLAM.* – URL https://google-cartographer-ros.readthedocs.io/en/latest/going_further.html#multi-trajectories-slam. – Zugriffsdatum: 07.12.2022

[20] Govindaraj, Shashank ; Nieto, Irene ; But, Alexandru ; Brinkmann, Wiebke ; Dettmann, Alexander ; Danter, Leon ; Aouf, Nabil ; Sotoodeh Bahraini, Masoud ; Zenati, Abdelhafid ; Savino, Heitor ; Stelmachowski, Jakub ; Colmenero, Francisco ; Heredia Aguado, Enrique ; Alonso, Mercedes ; Purnell, Joe ; Picton, Kevin ; Lopes, Luís: Multi-Robot Cooperation for Lunar Base Assembly And Construction, 10 2020

[21] Gul, Faiza ; Rahiman, Wan ; Alhady, Syed Sahal N.: A comprehensive study for robot navigation techniques. In: *Cogent Engineering* 6 (2019), Nr. 1, S. 1632046. – URL https://doi.org/10.1080/23311916.2019.1632046

[22] Haehnel, Dirk: *Karte: Belgioioso Castle.* 2014. – URL http://www2.informatik.uni-freiburg.de/~stachnis/datasets.html. – Zugriffsdatum: 19.10.2022

[23] Hampson, Michelle: *Underwater Robots Get a Boost in Mapping the Ocean.* IEEE Spectrum. August 2022. – URL https://spectrum.ieee.org/mapping-unchartered-waters. – Zugriffsdatum: 17.12.2022

[24] Heinemann, Tonja ; Villinger, Moritz ; Lechler, Armin ; Verl, Alexander: Dynamic Obstacle Layer: A new concept to avoid moving obstacles in local path planning using ROS. In: *ISR 2020; 52th International Symposium on Robotics*, 2020, S. 1–8

[25] Hess, Wolfgang ; Kohler, Damon ; Rapp, Holger ; Andor, Daniel: Real-Time Loop Closure in 2D LIDAR SLAM. In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, S. 1271–1278

[26] IEEE: *Pioneer Robot.* robots.ieee.org. – URL https://robots.ieee.org/robots/pioneer/. – Zugriffsdatum: 02.09.2022

[27] Kasperski, Sebastian ; Wollenberg, Johannes ; Eich, Markus: Evaluation of cooperative exploration strategies using full system simulation. In: *2013 16th International Conference on Advanced Robotics (ICAR)*, 2013, S. 1–6

[28] Kuhn, H. W.: The Hungarian method for the assignment problem. In: *Naval Research Logistics Quarterly* 2 (1955), Nr. 1-2, S. 83–97. – URL https://onlinelibrary.wiley.com/doi/abs/10.1002/nav.3800020109

[29] Kulich, M. ; Juchelka, T. ; Přeučil, L.: COMPARISON OF EXPLORATION STRATEGIES FOR MULTI-ROBOT SEARCH. In: *Acta Polytechnica* 55 (2015), Nr. 3, S. 162–168

[30] Kumar Dinesh, Vidyarthi Deo P.: A Survey on Auction based Approaches for Resource Allocation and Pricing in Emerging Edge Technologies. (2021), 12

[31] Land, A. H. ; Doig, A. G.: An Automatic Method of Solving Discrete Programming Problems. In: *Econometrica* 28 (1960), Nr. 3, S. 497–520. – URL http://www.jstor.org/stable/1910129. – Zugriffsdatum: 2022-09-13. – ISSN 00129682, 14680262

[32] Mahdoui, Nesrine ; Fremont, Vincent ; Natalizio, Enrico: Communicating Multi-UAV System for Cooperative SLAM-based Exploration. In: *Journal of Intelligent & Robotic Systems* 98 (2020), 05

[33] Mahdoui, Nesrine ; Frémont, Vincent ; Natalizio, Enrico: Cooperative Frontier-Based Exploration Strategy for Multi-Robot System. In: *2018 13th Annual Conference on System of Systems Engineering (SoSE)*, 2018, S. 203–210

[34] Mohamed, Khalil ; El Shenawy, Ayman ; Harb, Hany: A Hybrid Decentralized Coordinated Approach for Multi-Robot Exploration Task. In: *The Computer Journal* 62 (2018), 10, Nr. 9, S. 1284–1300. – URL https://doi.org/10.1093/comjnl/bxy107. – ISSN 0010-4620

[35] Morrison, David R. ; Jacobson, Sheldon H. ; Sauppe, Jason J. ; Sewell, Edward C.: Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning. In: *Discrete Optimization* 19 (2016), S. 79–102. – URL https://www.sciencedirect.com/science/article/pii/S1572528616000062. – ISSN 1572-5286

[36] Open-Robotics: *Gazebo*. https://gazebosim.org/home

[37] Open-Robotics: *Konzept*. http://wiki.ros.org/de/ROS/Concepts

[38] Open-Robotics: *ROS Introduction*. https://wiki.ros.org/action/fullsearch/de/ROS/Introduction

[39] Open-Robotics: *ROS Package Move_Base*. http://wiki.ros.org/move_base

[40] Open-Robotics: *Rostests*. – URL http://wiki.ros.org/rostest. – Zugriffsdatum: 02.12.22

[41] Open-Robotics: *Wrinting ROS tests*. – URL http://wiki.ros.org/rostest/Writing. – Zugriffsdatum: 02.12.2022

[42] Pak-Sing Choi, Felix Munoz-Garcia: *Auction Theory*. Springer Cham, 2021. – 29–38 S. – ISBN 978-3-030-69574-3

[43] Pal, Anshika ; Tiwari, Ritu ; Shukla, A.: Multi-Robot Exploration in Wireless Environments. In: *Cognitive Computation* 4 (2012), 12

[44] Playerstage: *Stage Simulation*. http://playerstage.sourceforge.net/index.php?src=stage

[45] Python-Software-Foundation: *Asserts im Unittest framework*. – URL https://docs.python.org/3/library/unittest.html#assert-methods. – Zugriffsdatum: 02.12.2022

[46] Queralta, Jorge P. ; Taipalmaa, Jussi ; Can Pullinen, Bilge ; Sarker, Victor K. ; Nguyen Gia, Tuan ; Tenhunen, Hannu ; Gabbouj, Moncef ; Raitoharju, Jenni ; Westerlund, Tomi: Collaborative Multi-Robot Search and Rescue: Planning, Coordination, Perception, and Active Vision. In: *IEEE Access* 8 (2020), S. 191617–191643

[47] ROS-Simulation: *Stage-ROS-Package*. https://github.com/ros-simulation/stage_ros

[48] Shoham, Yoav ; Leyton-Brown, Kevin: *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge, UK : Cambridge University Press, 2009. – ISBN 978-0-521-89943-7

[49] Siciliano, Bruno ; Khatib, Oussama: *Springer Handbook of Robotics*. Berlin, Heidelberg : Springer-Verlag, 2007. – 864 S. – ISBN 354023957X

[50] Smith: The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver. In: *IEEE Transactions on Computers* C-29 (1980), Nr. 12, S. 1104–1113

[51] Solanas, A. ; Garcia, M.A.: Coordinated multi-robot exploration through unsupervised clustering of unknown space. In: *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)* Bd. 1, 2004, S. 717–721 vol.1

[52] Werger, B. B. ; Mataric, M. J.: Broadcast of Local Eligibility for Multi-Target Observation. In: *Distributed Autonomous Robotic Systems 4* (2001), S. 347–356

[53] Wilson, Robert: Chapter 8 Strategic analysis of auctions. Elsevier, 1992, S. 227–279. – URL https://www.sciencedirect.com/science/article/pii/S1574000505800116. – ISSN 1574-0005

[54] Yamauchi, B.: A frontier-based approach for autonomous exploration. In: *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97. 'Towards New Computational Principles for Robotics and Automation'*, 1997, S. 146–151

[55] Yamauchi, Brian: Frontier-Based Exploration Using Multiple Robots. In: *Proceedings of the Second International Conference on Autonomous Agents*. New York, NY, USA : Association for Computing Machinery, 1998 (AGENTS '98), S. 47–53. – URL https://doi.org/10.1145/280765.280773. – ISBN 0897919831

[56] Yan, Zhi ; Jouandeau, Nicolas ; Cherif, Arab A.: A Survey and Analysis of Multi-Robot Coordination. In: *International Journal of Advanced Robotic Systems* 10 (2013), Nr. 12, S. 399. – URL https://doi.org/10.5772/57313

[57] Zlot, R. ; Stentz, A. ; Dias, M.B. ; Thayer, S.: Multi-robot exploration controlled by a market economy. In: *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)* Bd. 3, 2002, S. 3016–3023 vol.3

Erklärung zur selbstständigen Bearbeitung

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

Ort	Datum	Unterschrift im Original