Aufbau und Vergleich von Karten mit Hilfe von annotierten Mittelachsen zur Re-/Lokalisierung

Studienarbeit

Richard Bade
Richard.Bade@student.uni-magdeburg.de

Otto-von-Guericke-Universität Magdeburg
39106 Magdeburg

Unter Betreuung von:
Michael Fiegert
Michael.Fiegert@siemens.com

Siemens AG
CT IC 6
Otto-Hahn-Ring 6
81730 München

SIEMENS

I. Inhaltsverzeichnis

I. INHALTSVERZEICHNIS II

II. ABBILDUNGSVERZEICHNIS III

III. VERZEICHNIS DER ABKÜRZUNGEN IV

1 EINLEITUNG 1

1.1 CT IC 6 2
1.2 AUFGABENSTELLUNG 3
1.3 STAND DER TECHNIK 4

2 GRUNDLAGEN 5

2.1 MITTELACHSEN UND SHOCK GRAPHEN 5
2.2 DATENREPRÄSENTATION 7

3 KARTENAUFBAU 10

3.1 INITIALISIERUNG 11
3.2 AUSWAHL EINER GEEIGNETEN VERBINDUNGSSTELLE 12
3.3 FUSIONIERUNG VON UMGEBUNGSKARTEN 13
3.4 VERGLEICHSMETHODIK 14
3.4.1 VERGLEICH VON KANTEN 15
3.4.2 KNOTENVERGLEICH BASIEREND AUF UMGEBUNGSINFORMATIONEN 17
3.4.3 KNOTENVERGLEICH BASIEREND AUF ORTSINFORMATIONEN 19
3.4.4 FALL-BACK-STRATEGIE 21
3.5 ZUSAMMENFASSUNG 22

4 KARTENMATCHING 23

4.1 IDEE 23
4.2 HYPOTHESENBILDUNG 24
4.3 CANDIDATE ELIMINATION ALGORITHMUS 26
4.4 POSITIONSBESTIMMUNG 27
4.5 ZUSAMMENFASSUNG 27

5 ERGEBNISSE 29

6 IMPLEMENTIERUNG 33

6.1 DATENSTRUKTUREN 33
6.2 DEFINITIONEN 36
6.3 ANWENDERSCHNITTSTELLE 38
6.4 KARTENAUFBAU UND –VERGLEICH 40
6.5 HILFSFUNKTIONEN 45
6.6 SONSTIGE FUNKTIONEN 46

7 ZUSAMMENFASSUNG UND AUSBLICK 49

8 QUELLEN 50

ANHANG – KOMMANDOS FÜR DIE SINAS SIMULATIONSUMGEBUNG 51

II. Abbildungsverzeichnis

ABB. 1: HEFTER ST82R 2
ABB. 2: MOBILER MANIPULATOR 2
ABB. 3: MITTELACHSEN DEFINITION 6
ABB. 4: GLEICHHEIT VON MITTELACHSEN 6
ABB. 5: MIDRANGE-MAP 9
ABB. 6: ZEITLICH AUFEINANDER FOLGENDE MIDRANGE-MAPS 11
ABB. 7: REDUNDANZ DER MIDRANGE-MAPS 11
ABB. 8: DIFFERENZDIAGRAMM VON KANTENVERGLEICHEN 17
ABB. 9: SITUATION – ZWEI ERKUNDUNGSKNOTEN 21
ABB. 10: SITUATION – ZWEI KREUZUNGSKNOTEN 21
ABB. 11: SITUATION – JE EIN ERKUNDUNGS- UND EIN KREUZUNGSKNOTEN 21
ABB. 12: LOKALE KARTE MIT ROBOTERPOSITION 28
ABB. 13: GLOBALE KARTE MIT POSITIONSSCHÄTZUNGEN 28
ABB. 14: KARTE NACH MITTELACHSEN 30
ABB. 15: KARTE MIT GEOMETRISCHER INFORMATION 30
ABB. 16: KARTE MIT DEM GRUNDRISS DER UMGEBUNG UND DER HINDERNISSE 31
ABB. 17: REFERENZKARTE MIT KNOTENHYPOTHESEN 31
ABB. 18: UMGEBUNGSKARTE MIT MARKIERTEM KNOTEN, DER IN DER REFERENZKARTE GEFUNDEN WERDEN SOLL 31

III. Verzeichnis der Abkürzungen

AUK	aktuelle Umgebungskarte
CEA	Candidate Elimination Algorithmus
CT IC 6	Abteilung Corporate Technology - Information and Communication 6
$d(L_{i,A/B})$	relative Richtung des *i*-ten Kantenelementes der Kante A/B
δ_d	Distanzwert der relativen Richtung von zwei Kanten
δ_E	Distanzwert von zwei Kanten (Summe aus δ_d und δ_v)
δ_v	Distanzwert des Voronoi-Levels von zwei Kanten
$E_{A/B}$	Kante eines Knoten A/B
EP	Schätzposition des Knoten V_G
EMA	erweiterte Mittelachsen
GLK	globale Referenzkarte
H_{fixed}	Menge der festgelegten Hypothesen
$L_{i,A/B}$	*i*-tes Kantenelement der Kante $E_{A/B}$
LOK	lokale Karte der Roboterumgebung
LUK	letzte eingefügte Umgebungskarte
$N_{A/B}$	Anzahl der Elemente der Kante $E_{A/B}$
SINAS	Siemens Navigationssystem für autonome Serviceroboter
SLAM	Lokalisierung bei Umgebungsdynamik und Kartenaufbau auf Basis von Mittelachsen
$UK_{A/B}$	Umgebungskarten A/B
$v(L_{i,A/B})$	Voronoi-Level des *i*-ten Kantenelementes der Kante A/B
V_G	Knoten aus der globalen Karte
V_U	aus der aktuellen Umgebungskarte gewählter Knoten
w_d	Normierungsfaktor für den Distanzwert der relativen Richtung
w_v	Normierungsfaktor für den Distanzwert des Voronoi-Levels

1 Einleitung

Im Rahmen meines Informatikstudiums an der Otto-von-Guericke Universität Magdeburg absolvierte ich ein Industriepraktikum bei Siemens am Standort München Perlach. Das Praktikum kam durch ein Praktikantenprogramm zwischen Siemens und der Universität Magdeburg zustande. Die Anfertigung dieser Studienarbeit ist ein Teil des Industriepraktikums. Während der Zeit arbeitete ich in der Abteilung Corporate Technology - Information and Communication 6 (CT IC 6) bei der ich mich an dieser Stelle für die tatkräftige Unterstützung bedanken möchte. Spezieller Dank geht an meinen dortigen Betreuer Michael Fiegert.

Der Inhalt des Praktikums war die Lokalisierung und Relokalisierung von autonomen mobilen Robotern. Das Ziel ist hier, die Erhöhung der Autonomie der Roboter durch eine geeignete Umgebungsabbildung, denn nur dann sind diese in der Lage, mit der Umwelt zu interagieren und komplexe Aufgaben zu erfüllen. Da die derzeitigen Verfahren zur Re-/Lokalisierung auf Kantenerkennung basieren, stoßen sie sehr schnell an ihre Grenzen, wenn keine derartigen Features vorhanden sind. Deshalb sollten Shock Graphen als Basis für das hier beschriebene Verfahren dienen. Shock Graphen nutzen als Erkennungsmerkmal die Freifläche der Umgebung, das heißt die nicht bebaute, zugestellte oder sonst in irgendeiner Form besetzte Fläche. Durch diesen Ansatz ist es möglich, die Arbeitsumgebungen von mobilen Robotern auf ein Gebiet auszudehnen, die mit den aktuell zur Verfügung stehenden Verfahren nur schwer oder gar nicht zugänglich sind. Er wurde von Siemens zum Patent angemeldet, der Autor ist als Miterfinder eingetragen.

Für ein besseres Verständnis, in welchem Zusammenhang meine Aufgabe mit der CT IC 6 stand, werde ich ihr Arbeitsfeld kurz erläutern, bevor ich meine Aufgabe detailliert beschreibe. Daran anschließend findet sich der momentane Stand der Technik zu diesem Thema.

1.1 CT IC 6

Die CT IC 6 beschäftigt sich, als eine der vielen Forschungsabteilungen innerhalb der CT IC, mit dem Thema "intelligente autonome Systeme". Die Arbeitsgruppe ist in zwei Bereiche aufgeteilt - die Software- und die Hardwareagenten. Agenten bedeuten in diesem Zusammenhang autonom agierende Programme. Bezogen auf die Software heißt dies, Aufgaben in der virtuellen Welt der Daten zu übernehmen. Auf der anderen Seite erfüllen Roboter diese Funktionen in der realen physikalischen Welt.

Abb. 1: Hefter ST82R

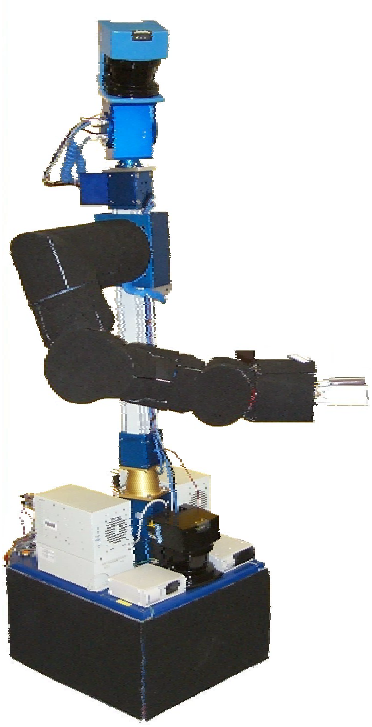

Abb. 2: Mobiler Manipulator

Die zwei Aufgabengebiete der Robotik sind die Navigation und die Manipulation von alltäglichen Umgebungen. Ein Schwerpunkt ist hier das Wahrnehmen der Umwelt mittels verschiedener Sensorik. Ein weiterer das Reagieren auf diese Sensordaten und das Manipulieren der Umwelt. In diesem Zusammenhang entstanden unterschiedliche Systeme, wie z.B. der Putzroboter Hefter ST82 R (Abb. 1) und der mobile Manipulator (Abb. 2).

1.2 Aufgabenstellung

Meine Aufgabe bestand in der Entwicklung und Implementierung einer Strategie, um Karten von Arbeitsumgebungen für den Reinigungroboter Hefter ST82 R zu erstellen, auf deren Grundlage dann eine Lokalisierung und Relokalisierung stattfindet. Die grundlegende Idee auf Shock Graphen aufzubauen, kam durch die Vorarbeit meines Betreuers und des Praktikanten Charles-Marie de Graeve. Sie untersuchten verschiedene Ansätze, von denen dieser am vielversprechendsten klang. Dabei die aktuell eingesetzten Verfahren nicht zu ersetzen, sondern vielmehr zu unterstützen, war das Ziel.

Da diese Methode der Re-/Lokalisierung in der Form noch nicht existierte, war die Entwicklung und Implementierung gleichzeitig mit vielen Experimenten verbunden. Diese sollten letztendlich auch die Korrektheit dieses Ansatzes bestätigen.

Meine Aufgabe teilte sich in folgende Teile auf:

1. Einarbeitung in die zu verwendende Entwicklungsumgebung (Borland C 5.0 mit Bibliotheken des bestehenden Systems).
2. Definition einer geeigneten Datenstruktur für eine Karte.
3. Aufbau einer Karte von der unmittelbaren Umgebung des Roboters. Unmittelbare Umgebung heißt in diesem Fall, die Beschränkung der Karte auf die Reichweite der Sensoren.
4. Aufbau einer globalen Karte durch die Verknüpfung einzelner Umgebungskarten.
5. Vergleich von Karten für die Lokalisierung und Relokalisierung des Roboters.
6. Präsentation und Dokumentation der Arbeit.

Für die Lösung meiner Aufgabe stand mir die SINAS Simulationsumgebung zur Verfügung. Das System simuliert sowohl die Eingangsdaten in Form von Laser- sowie Ultraschallscannern, die Reaktion des Roboters auf die Verarbeitung dieser Daten, sowie das Echtzeitverhalten.

1.3 Stand der Technik

Aktuelle Verfahren zur Selbstlokalisierung von Robotern basieren auf der Erkennung von Objekten. Beispielsweise werden Geraden extrahiert, welche dann, in bestimmten Winkeln angeordnet mit einem vorher gespeicherten Muster der Umgebung verglichen werden. Anstelle von Geraden werden auch Eckpunkte verwendet. Diesen Verfahren liegt die Idee zugrunde, anhand natürlicher Landmarken Objektformen wiederzuerkennen. Einen guten Überblick geben hier Borenstein und Feng [1]. Allerdings muss man hier mit einer großen Einschränkung leben. Diese Methoden setzen voraus, dass die zu extrahierenden Merkmale vorhanden sind. In Umgebungen mit runden Wänden oder Gegenständen, vorstellbar beispielsweise in einem Kaufhaus, ist diese Art von Umgebungswiedererkennung nicht anwendbar. Eine ganz andere Herangehensweise bieten die Freiformflächen[2]. Ausgegangen wird hier, anstelle von gerader Objektform, von der zwischen den Objekten liegenden freien Fläche. Diese kann als Invariante dienen, wenn sie sich zwischen zwei Messungen nicht oder nur wenig ändert. Das ist die Voraussetzung für den Ansatz dieses Verfahrens. Dabei ist es nicht von Bedeutung von welcher geometrischen Form die Objekte sind. Die 1967 entwickelten Voronoi Diagramme [3] werden oft als Grundlage für die Beschreibung der Freifläche durch Mittelachsen eingesetzt. [4] verhilft an dieser Stelle zu einem kurzen Überblick. Mit diesem Verfahren ist es möglich den Bereich der Roboternavigation auf Umgebungen auszudehnen, die bisher nicht oder nur schwer zugänglich waren. Allerdings gibt es auch hier ein Problem, welches in [5] aufgegriffen wird – der Zeitaufwand für den Vergleich der Umgebungsdaten. Wie hier von Larrosa und Valiente aufgeführt wird, kann unter Echzeitbedingungen bezogen auf die Lokalisierung von Robotern diese Methode nicht ohne Modifikationen durchgeführt werden. Alle diese Ideen flossen in die Entwicklung des nachfolgend beschriebenen Verfahrens ein.

Zu Beginn meiner Arbeit waren die Extraktion und die Aufbereitung von Sensordaten des Roboters vorhanden. Diese Daten sind in einer Matrix, der Midrange-Map mit 141*141 Pixeln, gespeichert. Das Zentrum entspricht der aktuellen Weltposition des Roboters. Mit einer Auflösung von 6,4 cm pro Pixel deckt die Midrange-Map damit eine Fläche von 9,024*9,024 m ab. Diese Karte wird alle 300 ms aus der Umgebung erstellt. Für die aktuelle Umgebung werden basierend auf den Daten der Midrange-Map die Mittelachsen generiert, welche dann in einer Struktur zur Verfügung standen.

2 Grundlagen

Bevor mit der eigentlichen Arbeit begonnen wurde, musste die spätere Verwendung und der eventuelle Einsatzort genauer definiert werden, da es unmöglich war, eine universelle Lösung für jeden denkbaren Fall zu finden. Darum sind nachfolgend einige notwendige Bedingungen für diese Art der Re-/Lokalisierung aufgezählt.

Die Genauigkeit der Re-/Lokalisierung spielt hinsichtlich der Geschwindigkeit eine signifikante Rolle. Aus diesem Grund gelten folgende Restriktionen. Die Positionsschätzung muss kein genaues Ergebnis liefern. Vielmehr wird erhofft, möglichst schnell eine Umgebungsgleichheit zu finden, um so den Aufenthaltsort des Roboters einzugrenzen. Mit Hilfe anderer Verfahren kann dann in diesem eingeschränkten Bereich eine den Erfordernissen entsprechend genauere Suche durchgeführt werden. Hiermit wird sich eine erhebliche Geschwindigkeitserhöhung bis zur gewünschten Genauigkeit erhofft. Weiterhin muss eine ausreichende Datenmenge vorausgesetzt werden. Denn bei den hier verglichenen Freiflächen sehen, bei beispielsweise zu kurz gefahrenen Strecken, viele Abschnitte auf der Karte ähnlich aus. So ergeben sich zwangsläufig mehrere Positionshypothesen. Der wichtigste Punkt aber ist, dass sich die Umgebung nicht sehr stark ändern darf. Eine Veränderung im Zentrum der Fläche würde eine völlig andere Topologie zur Folge haben und hier nicht erkannt werden. Eine Diplomarbeit "Lokalisierung bei Umgebungsdynamik und Kartenaufbau (SLAM) auf Basis von Mittelachsen" kümmert sich in direkter Nachfolge an diese Studienarbeit um diese Problematik.

Im folgenden Abschnitt wird kurz die Beschreibung der Freiflächen mit Hilfe von Mittelachsen erläutert. Daran schließt sich eine Erläuterung über die Abbildung der Mittelachsen in eine Datenstruktur an.

2.1 Mittelachsen und Shock Graphen

Die von Blum 1967 entwickelten Mittelachsen werden durch Kreismittelpunkte charakterisiert. Diese unterliegen drei Bedingungen. Erstens müssen die Kreise innerhalb der Fläche liegen, zweitens maximal groß sein und drittens den Flächenrand mindestens zweimal tangieren. Abb. 3 zeigt schematisch, wie die Mittelachsen aus den Kreismittelpunkten entstehen. Die Menge dieser Punkte spiegelt die Topologie der Fläche

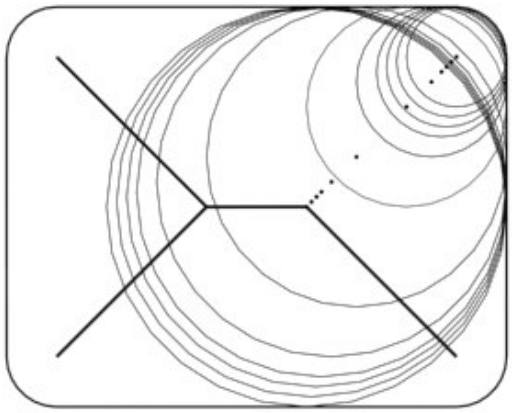

Abb. 3: Mittelachsen Definition

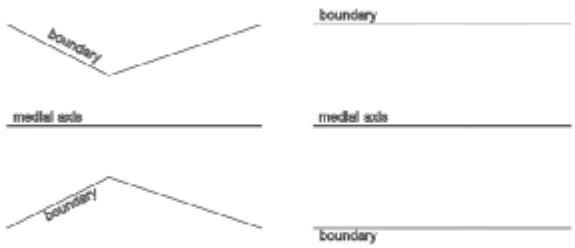

Abb. 4: Gleichheit von Mittelachsen

wieder. Allerdings ist diese Zuordnung nicht umkehrbar eindeutig. In Abb. 4 ist ein einfaches Beispiel von zwei gleichen Mittelachsen dargestellt, die jedoch durch unterschiedliche Umgebungen erzeugt worden sind.

Aufgrund dessen müssen mehr Informationen als nur die der Mittelachsen herangezogen werden, um eine bijektive Zuordnung zu erreichen. Abhilfe schaffen hier Ansätze der Voronoi- oder auch Shockgraphen [6]. Jedem Kreismittelpunkt der Mittelachsen werden spezifische Informationen hinzugefügt. Diese tragen letztendlich zur eineindeutigen Identifizierung der Topologie bei. Die hier eingesetzten Informationen sind der Radius des Kreises (im folgenden Voronoi-Level genannt), nach oben beschriebener Definition und die Lage eines Punktes der Mittelachsen relativ zu seinem Nachbarpunkt. Die so ergänzten Mittelachsen werden im Folgenden erweiterte Mittelachsen (EMA) genannt. Unter Verwendung dieser Informationen, kann die so gespeicherte Freifläche auch eineindeutig wiederhergestellt werden.

Werden die Radien der Punkte der EMA benutzt, um die Kreise wieder herzustellen und werden diese dann ausgefüllt auf einen weißen Hintergrund gezeichnet, so wird ein Negativbild der abgespeicherten Umgebung erstellt. Damit kann auch eine für den Menschen leicht nachvollziehbare Darstellung erzeugt werden.

Nun fehlt noch eine geeignete Repräsentation der EMA in einer Datenstruktur. Im folgenden wird diese hergeleitet und ausführlich beschrieben.

2.2 Datenrepräsentation

Schaut man sich die Definition der Mittelachsen und daraus Hervorgehend die EMA an, so scheint eine verpointerte Liste geradezu prädestiniert, um die Menge der Mittelpunkte mit den dazugehörigen Informationen zu beschreiben. Für einen späteren Vergleich von zwei oder mehr solcher Datenstrukturen ist das allerdings ungeeignet, da der Aufwand bei Auftreten von Verzweigungsstellen dem Lösen graphentheoretischer Probleme gleichkommt. Hinsichtlich der Aufgabenstellung eines Vergleichens von Mittelachsen mit vertretbarem Aufwand wird eine Struktur, die hierarchisches Suchen erlaubt, verwendet. Die erweiterten Mittelachsen werden hierfür in Knoten, Kanten und Kantenelemente zerlegt.

Knoten sind diejenigen Kreismittelpunkte, die sich am Ende der EMA befinden (Endknoten) und diejenigen, in welchen sich drei oder mehr Zweige kreuzen (Kreuzungsknoten). Endknoten werden beim Aufbau einer Karte, also beim Erzeugen der EMA in einer Situation aber zu Erkundungsknoten umdefiniert. Durch die Begrenzung der Reichweite von Laserscannern entstehen am Rand des Scans Bereiche, die durch keine Objekte, beispielsweise Wände begrenzt sind. Das hat zur Folge, dass hier aufgrund fehlender Daten keine EMA berechnet werden können. Allerdings gibt es einen Kreismittelpunkt, der maximal in Richtung dieses unbekannten Bereiches liegt, somit einen Endknoten darstellt. Dieser wird nun zu einem Erkundungsknoten, da der dahinter liegende Bereich noch nicht vollständig aufgeklärt ist. Abb. 5 zeigt die Midrange-Map einen Scans und die dazu berechneten EMA. Die gelben Flächen stellen für den Roboter unbekannte Bereiche dar, Hindernisse sind schwarz gekennzeichnet. Bei den erweiterten Mittelachsen (grün) sind die gelben Punkte Erkundungsknoten, die blauen echte Endknoten und die roten Kreuzungsknoten. Jeder Knoten enthält folgende Informationen. Eine eindeutige ID, den Typ (Kreuzungs-, Erkundungs- oder Endknoten), den Radius des Kreises aus dem er definiert wurde und eine absolute Weltposition. Angeordnet sind alle Knoten in einer Baumstruktur. Der Vorteil gegenüber einem Graphen ist das Vorhandensein eines Wurzelknotens, von dem aus operiert werden kann und das Nichtauftreten von Zyklen. Des Weiteren hat jeder Knoten Zugriff auf alle mit ihm verknüpften Kanten.

Kanten bilden die nächste Stufe in der Datenstruktur. Sie operieren als topologische Verbindungen zwischen benachbarten Knoten. Kanten werden ebenfalls über eine eindeutige ID gekennzeichnet und beinhalten die Start- und Endposition der Kante als absolute Weltkoordinaten, die Größe der Fläche, durch die die geometrische Kante (realer Kantenverlauf zwischen zwei Knoten) verläuft und die Anzahl der Kantenelemente, die diese geometrische Kante bilden. Sie enthält aber nicht die notwendigen Daten, um die Geometrie der freien Fläche wieder herzustellen. Dazu muss in der Hierarchie noch ein Schritt tiefer gegangen und die Stufe der Kantenelemente (grüne Punkte in Abb. 5) betrachtet werden.

Kantenelemente beschreiben die Geometrie der topologischen Verbindungen. Jedes dieser Kantenelemente enthält wieder den Radius des Kreises aus dem der Punkt entstand (Voronoi-Level), sowie die relative Position zum nächsten Element. Der Zugriff auf die Kantenelemente erfolgt über die topologischen Kanten.

Durch die hierarchische Anordnung ist gewährleistet, dass jede Stufe, beginnend mit den Knoten, separat durchsucht werden kann. Da Knoten im Verhältnis zu Kanten und Kantenelementen am wenigsten vertreten sind, enthalten diese aber auch am wenigsten Information. Wenig Information heißt aber gleichzeitig wenig Daten, die miteinander verglichen werden müssen. Darum erlaubt diese Anordnung erste schnelle Vergleiche, zu deren Ergebnissen dann mehr und mehr Informationen hinzugezogen werden können, um einen positiven Match zu erreichen. Zusätzlich sind die Knoten beginnend mit der Wurzel in einer doppelt verkettete Liste organisiert, damit alle Knoten schneller als in einer Baumstruktur durchlaufen werden können.

Von einer globalen Struktur sind letztendlich die Knoten, Kanten und Kantenelemente erreichbar. Diese Struktur stellt die Karte dar. Hier werden ferner die aktuelle Anzahl der jeweiligen Elemente aus einer Hierarchiestufe gespeichert, sowie eine maximale Elementanzahl festgelegt. Das ermöglicht eine effektive Planung des zu verwendenden Speichers.

Nun ist die Grundlage für einen Vergleich von Umgebungen gelegt. Im weiteren Verlauf wird die Datenstruktur für die EMA, die Karte, als Basis für Lokalisierung bzw. Relokalisierung verwendet.

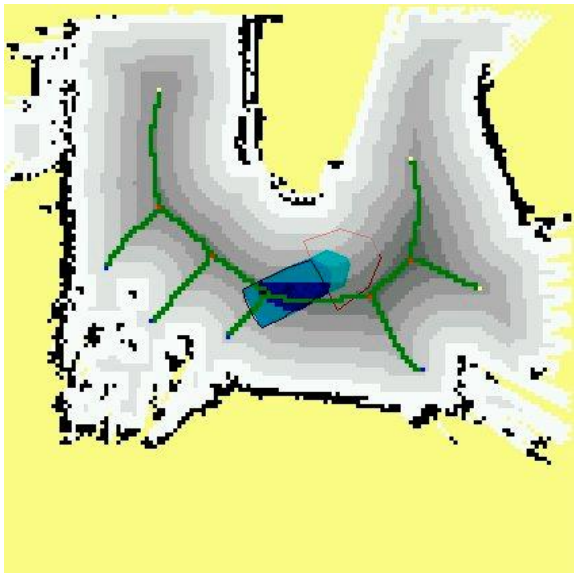

Abb. 5: Midrange-Map

3 Kartenaufbau

In diesem Abschnitt werden die Ideen und deren Umsetzungen, die zum Aufbau einer globalen Karte basierend auf den EMA notwendig sind, erläutert.

Die Ausgangslage ist, wie in den Grundlagen erläutert, zeitlich aufeinander folgende Midrange-Maps. Durch die Bewegung des Roboters auf einer bestimmten Bahn, werden folglich alle unmittelbaren Umgebungen dieser Bahn in den Midrange-Maps erscheinen. Abb. 6 zeigt eine Beispielfolge von nacheinander aufgenommenen Midrange-Maps. Alle diese Karten passend aneinander gelegt, ergeben die insgesamt vom Roboter gesehene Umwelt. Natürlich ist in der so entstandenen globalen Karte sehr viel Redundanz vorhanden (Abb. 7). Die grundlegende Idee ist deshalb, eine geeignete Verknüpfung der in den Midrange-Maps erzeugten EMA zu finden, um Redundanzen in den Umgebungen und den damit verbundenen Speicheraufwand zu vermeiden.

Der erste Schritt besteht in der Speicherung zwei aufeinander folgender EMA als je eine Karte. Darin werden im zweiten Schritt Gemeinsamkeiten gesucht, welche zur Fusionierung benutzt werden können. Das Erzeugen einer Karte aus einer Midrange-Map ist ohne großen Aufwand realisierbar.

Schon an dieser Stelle wird sich zeigen, dass Aufbau und Vergleich von Karten nicht strikt voneinander trennbar sind. Die grundlegenden Ansätze zum Vergleich der lokalen Karten werden sich auch im nachfolgenden Kapitel, dem Kartenmatching, wiederfinden.

Im dritten Schritt werden die Ergebnisse der Suche nach neuer Information aus den zwei lokalen Karten in eine globale überführt. Diese enthält schließlich die gesamte vom Roboter gesehene Umwelt.

Für den Aufbau einer globalen Karte wird eine lokale Karte benötigt, welche durch die Art ihres Gebrauches benannt ist. Im folgenden werden diese Karten, in denen sich die Information der EMA aus einer Midrange-Map befindet als Umgebungskarten bezeichnet. Das sind also lokale Abbildungen der unmittelbaren Umgebung des Roboters.

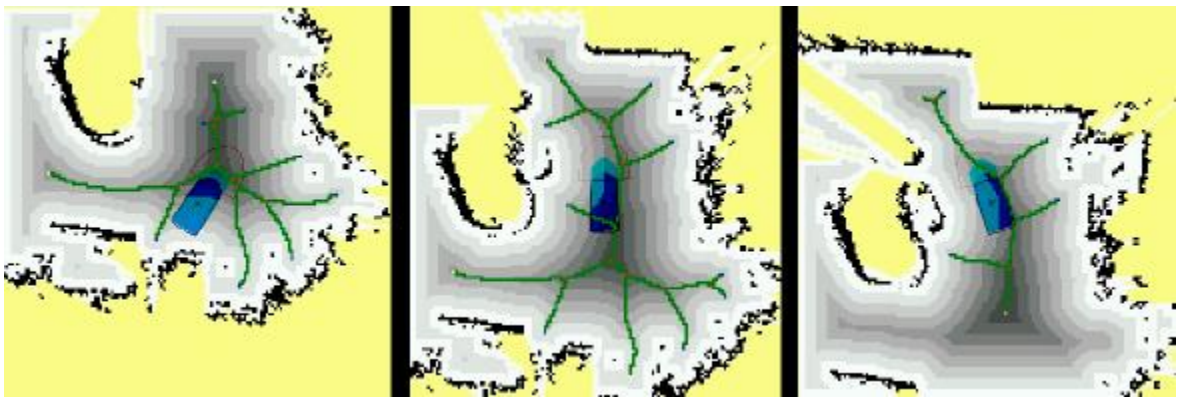

Abb. 6: zeitlich aufeinander folgende Midrange-Maps

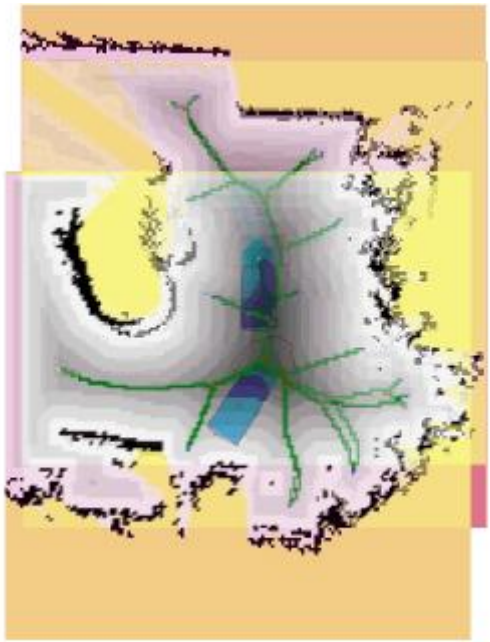

Abb. 7: Redundanz der Midrange-Maps

Nachfolgend soll detailliert auf den Prozess des Aufbaus der globalen Karte eingegangen werden.

3.1 Initialisierung

Um eine Karte zu erzeugen, werden die als Liste gespeicherten Punkte der EMA aus der Midrange-Map in die in Abschnitt 2.2 beschriebene Datenstruktur überführt. Dies geschieht zusätzlich zum Aufbau der Baumstruktur durch einfaches Kopieren der Information jedes Punktes der Mittelachsen. An dieser Stelle werden die Punkte nach den

Regeln der Datenstruktur in Knoten und Kantenelemente getrennt und es entsteht eine Umgebungskarte.

Alle nachfolgenden Operationen – der Vergleich von zwei Umgebungskarten, das Finden geeigneter Verbindungsstellen und anschließendes Kopieren in die globale Karte – erfolgen nun auf Basis der Knoten der EMA.

Ein initialer Vergleich von zwei Umgebungskarten wird möglich, indem die erste Umgebungskarte des Roboters ohne Änderungen in die globale Karte übernommen wird. Um sicher zu stellen, dass diese nicht zu wenig Information enthält, wird sie erst erstellt, wenn der Roboter mindestens die Hälfte der Reichweite der Midrange-Map (ca. 4,5 m) zurückgelegt hat. Diese Karte wird als zuletzt eingefügte Umgebungskarte (LUK) gespeichert. Zusätzlich sind die Weltkoordinaten, an der die Karte aufgenommen wurde (die Position des Roboters), von Interesse. Sie werden als Position der LUK für eine spätere Fall-Back-Lösung genutzt. Jede folgende Umgebungskarte wird nun mit der zuletzt in die globale Karte eingefügten auf Ähnlichkeiten hin untersucht. Diese Ähnlichkeiten bilden die Grundlage für eine Verknüpfung der beiden Karten.

3.2 Auswahl einer geeigneten Verbindungsstelle

Da die Verknüpfung, wie erwähnt, anhand der Knoten erfolgt, bilden diese auch den Ausgangspunkt für einen Ähnlichkeitstest. Aus der Umgebungskarte des aktuellen Scans wird ein ‚geeigneter' Knoten für den Test gewählt. Natürlich kann auch ein beliebiger Knoten herausgenommen werden, jedoch vereinfacht eine gute Wahl den Vergleich, da dadurch bestimmte Annahmen gemacht werden können. Kreuzungsknoten beispielsweise besitzen mehr Informationen als Erkundungs- und Endknoten, da sie mit mehr als einer Kante verbunden sind. Aus diesem Grund werden nur Kreuzungsknoten extrahiert, um sicher zu stellen, dass der gewählte Knoten genug individuelle Merkmale besitzt. Der gewählte Knoten sollte möglichst nahe zum und hinter dem Roboter liegen. Der Grund für die Nähe ist die Stabilität der Topologie im zentralen Gebiet der Umgebungskarten. So könnten Kreuzungsknoten im äußeren Bereich mit Kanten verbunden sein, deren zweiter Knoten ein Erkundungsknoten ist. Diese Art von Konstellation birgt wenig verlässliche Information und kann damit leicht vermieden werden.

Die Wahl eines Knotens, der sich hinter dem Roboter befindet, begründet sich mit dem zeitlichen Verlauf der Umgebungskarten. Angenommen es wurden bereits zwei EMA verknüpft, also zwei Umgebungskarten fusioniert, dann sind aufgrund der Ortsveränderung des Roboters in Bewegungsrichtung immer neue topologische Informationen vorhanden. Die zuletzt verbundene Umgebungskarte kann daher von der aktuellen sehr stark differieren. Entgegen der Bewegungsrichtung, im Allgemeinen also hinter dem Roboter, sind die Umgebungen andererseits konstanter. Die Wahrscheinlichkeit an dieser Stelle zwei gleiche Knoten zu finden, ist daher höher.

Es ist natürlich auch möglich, dass derselbe Kreuzungsknoten bei dieser Art von Selektion mehrere Male nacheinander gewählt wird, speziell während langsamer Roboterbewegung. Bei genauerer Betrachtung der Fusionierung von zwei Umgebungskarten ist das aber durchaus sinnvoll.

3.3 Fusionierung von Umgebungskarten

Um Umgebungskarten fusionieren zu können, ist es notwendig, einen Knoten in der globalen Karte zu finden, welcher dem gewählten Knoten aus der aktuellen Umgebungskarte (AUK) entspricht. Der Knoten aus der AUK wurde, wie in 3.2 beschrieben, gewählt. Zugunsten der Geschwindigkeit wird er nur mit ‚ausgewählten' Knoten der globalen Karte verglichen. Die Menge dieser ‚ausgewählten' Knoten entspricht dabei dem Bereich, den der Roboter maximal befahren haben kann, seit der letzten Fusionierung. Durch die Begrenzung der Maximalgeschwindigkeit auf einem Meter pro Sekunde liegt damit ein zugehöriger Knoten mindestens auf der LUK. Da diese Karte gespeichert ist, braucht nur deren Knotenliste für den Vergleich herangezogen werden.

Konnte der Knoten aus der globalen Karte (die LUK ist durch den letzten Fusionierungsschritt Teil der globalen Karte) eindeutig einem Knoten aus der AUK zugewiesen werden, werden die EMA an dieser Stelle verknüpft. Dazu werden alle nachfolgenden Knoten, des aus der globalen Karte ausgewählten Knotens einschließlich diesem und aller zugehörigen Kanten entfernt. An ihre Stelle treten die Informationen der AUK. Das heißt, von dem als gleich identifiziertem Knoten werden aus der Baumstruktur ab diesem alle Knoten und Kanten in die globale Karte kopiert. Damit ist auch klar, was geschieht, wenn im nächsten Schritt als Verknüpfungspunkt wieder die gleiche Stelle gewählt wird. Die neuen Daten

werden, vorausgesetzt es erfolgt eine eindeutige Zuordnung, genommen, um die alten zu ersetzen. Dadurch kann beim Zusammenbau vor der Umgebung des Roboters die globale Karte sehr stark schwanken. Da aber die Informationen über die Umgebung um so besser werden, je mehr sich der Roboter dieser nähert, sichert dieses Vorgehen eine höhere Integrität der endgültigen Karte.

Für den Fall, dass der aus der AUK gewählte Knoten auf keinen Knoten der globalen Karte abgebildet werden konnte, wird eine Fall-Back-Strategie (siehe Kapitel 3.4.4) ausgeführt, wenn folgende Bedingung gilt. Die Entfernung zwischen der aktuellen Position des Roboters und der Position der LUK überschreitet ein Drittel der Reichweite der Midrange-Map (ca. 3,0 m). Die Notwendigkeit dieser Vorgehensweise begründet sich durch die stetige Änderung der Umgebung des sich bewegenden Roboters. Es besteht nie wieder die Möglichkeit einer Korrektur oder Vervollständigung der globalen Karte, wenn der Roboter die Umgebung, an der sie ‚abriss', nicht noch einmal befährt. Die globale Karte würde, da kein Knoten mehr auf die LUK abgebildet werden könnte, nicht ‚weitergebaut' oder nur mit einer falschen Abbildung durch eine ähnliche Umgebung fehlerhaft vervollständigt werden. Gilt die Bedingung nicht, werden in diesem Durchlauf keine neuen Informationen an die globale Karte angefügt.

3.4 Vergleichsmethodik

Für die Suche nach Gemeinsamkeiten zwischen dem gewählten Knoten aus der AUK (V_U) und dem Knoten aus der globalen Karte (V_G) wurden zwei Verfahren implementiert. Der Ansatz des ersten basiert auf Umgebungsinformationen. Die Knoten V_U und V_G werden miteinander verglichen, indem die Informationen der mit ihnen verknüpften Kanten untersucht werden. Die zweite Methode greift auf die Ortsinformation der Knoten zurück.

Beide Strategien sind in einer hierarchischen Suche organisiert. Das heißt, sollte die Zuordnung nach der ersten Methode scheitern, versucht der zweite Algorithmus ein Ergebnis zu finden. Greift auch diese Vorgehensweise nicht, tritt eine Fall-Back-Lösung in Kraft.

Bei der ersten Methode ist der Vergleich der Knoten über die Kanten realisiert. Aus diesem Grund ist im Folgenden der Kantenvergleich detailliert beschrieben.

3.4.1 Vergleich von Kanten

Auf Grund der Datenstruktur, ist über die Kanten der Zugriff auf die einzelnen Kantenelemente möglich. Da diese als Gesamtheit die geometrische Information der Kante beinhalten, stellt ein direkter Vergleich der Kantenelemente sicher, dass gleiche Strukturen als solche erkannt werden.

Die Kantenelemente werden als doppelt verkettete Liste mit Startpunkt am ausgehenden Knoten gespeichert. Bedingt durch den Aufbau der Bäume, welche direkt von der Fahrtrichtung des Roboters abhängen, können neben gleichen Kanten mit gleicher Orientierung auch Kanten auftreten, die sich in der Orientierung unterscheiden. aber dennoch die gleiche Umgebung beschreiben. Am Beispiel zwei gleicher Kanten E_A und E_B lässt sich die Sachlage leicht erläutern. Die Kantenelemente $L_{i,A}$ (lies: i-tes Element der Kante E_A) der Kante E_A mit $i=0..N_A$ (N_A ist die Anzahl der Elemente der Kante E_A) sind aufgrund der umgekehrten Orientierung entgegengesetzt zu den Kantenelementen ($L_{i,B}$) der Kante E_B angeordnet. Das heißt, dass das erste Element von E_A dem letzten Element von E_B, das zweite dem vorletzten und so weiter entspricht. Das ist beispielsweise dann der Fall, wenn der Roboter eine Umgebung mehrere Male in unterschiedlicher Richtung abfährt. Deshalb wird vor dem direkten Vergleich zweier Kanten zunächst bestimmt, in welche Richtung die Kantenelemente gelesen werden müssen. Ist die Orientierung unterschiedlich (z.B. bei der Kante E_A von dem Knoten weg und bei E_B zum Knoten hin), werden die Elemente einer Kante entgegengesetzt gelesen, ansonsten werden beide Kanten in Laufrichtung verarbeitet.

Am Ende des Vergleiches der Kanten E_A und E_B steht ein Distanzwert δ_E, welcher den Unterschied der beiden Kanten charakterisiert. In δ_E gehen die Differenzwerte für die relative Richtung (δ_d) und das Voronoi-Level (δ_v) der Elemente $L_{i,A/B}$ ein. Da die Kanten E_A und E_B meist unterschiedlich lang sind, werden nur so viele Elemente in die Rechnung einbezogen, wie der kürzeren Kante zur Verfügung stehen. Die überschüssigen Elemente der längeren Kante werden ignoriert. Damit ist zu mindestens gewährleistet, dass zwei gleiche Kanten, von denen eine Kante unvollständig ist, nicht schlechter bewertet werden. Damit unterschiedliche Längen von Kanten keine Auswirkungen auf den Distanzwert haben, ist zusätzlich eine Normierung notwendig. δ_v und δ_d werden damit wie folgt berechnet:

$$\delta_v = \frac{\sum_{i=1}^{MIN(N_A,N_B)} \left(v(L_{i,A}) - v(L_{i,B})\right)^2}{MIN(N_A, N_B)} \quad (1)$$

$$\delta_d = \frac{\sum_{i=1}^{MIN(N_A,N_B)} \left(d(L_{i,A}) - d(L_{i,B})\right)^2}{MIN(N_A, N_B)} \quad (2).$$

Die Funktionen $v(L_{i,A/B})$ und $d(L_{i,A/B})$ charakterisieren die Eigenschaften (Voronoi-Level bzw. Richtungsinformation) des i-ten Kantenelementes.

Die zwei Differenzen δ_v und δ_d müssen nun noch geeignet verknüpft werden. Hierfür wurde ein Experiment durchgeführt, bei dem der Nutzer jeweils zwei gleiche Kanten E_A und E_B aus zwei überlappenden Umgebungskarten (UK_A, UK_B) vorgibt. Die Werte δ_v und δ_d werden dann für das jeweilige Kantenpaar berechnet. Da E_A nur zu E_B nicht aber zu den restlichen Kanten gleich ist, bilden alle Kantenpaare zwischen E_A und den restlichen Kanten aus UK_B sowie alle Kantenpaare zwischen E_B und den restlichen Kanten aus UK_A Beispiele für unterschiedliche Kantenpaare. Für diese werden dann ebenfalls δ_v und δ_d berechnet.

Das Experiment wurde mit 42 gleichen Kantenpaaren durchgeführt. Die Normierungsfaktoren w_v und w_d wurden so ermittelt, dass δ_v und δ_d in etwa gleich gewichtet in δ_E eingehen. Das Diagramm in Abb. 8 veranschaulicht Voronoi-Level- und Richtungsdifferenzen für Kantenvergleiche, die jeweils im Bereich von 0 bis 20 auf der Abszisse bzw. der Ordinate abgetragen sind. Rote Quadrate kennzeichnen die Werte für zwei unterschiedliche Kanten, grüne hingegen stellen die Werte zweier gleicher Kanten dar. Es ist erkennbar, dass alle grünen Quadrate idealerweise in der Nähe des Ursprungs liegen, die roten jedoch nicht. Als Normierungsfaktor w_v wurde das Doppelte der größten auftretenden Voronoi-Differenz von gleichen Kanten gewählt, für w_d entsprechend das Doppelte der maximalen Richtungsdifferenz. Durch folgende Formel wird dann der Differenzwert δ_E berechnet:

$$\delta_E = \sqrt{\frac{1}{w_v^2}\delta_v^2 + \frac{1}{w_d^2}\delta_d^2} \quad (3).$$

Die hier berechneten Kantendifferenzen eines Knotenpaares werden nun eingesetzt, um zwei Knoten zu vergleichen, was im folgenden Abschnitt beschrieben wird.

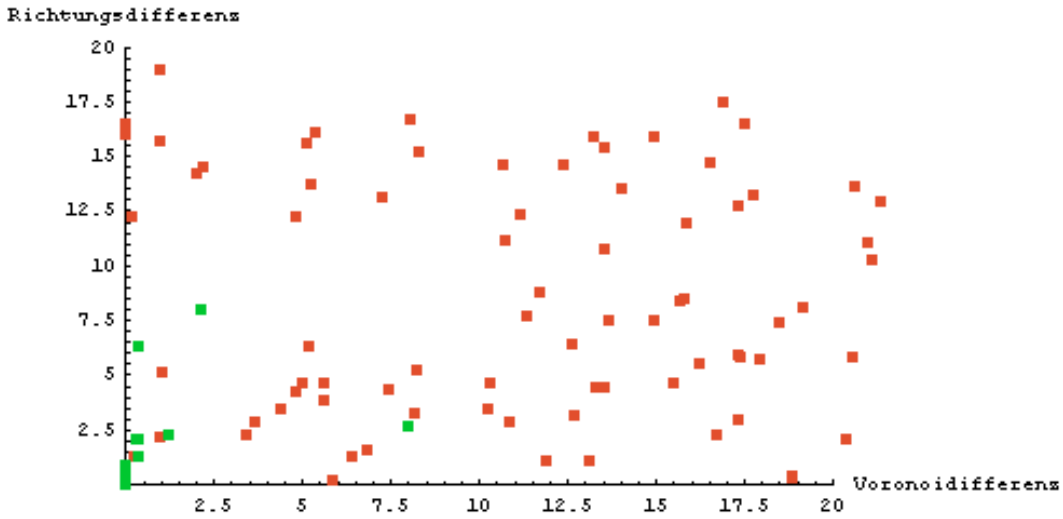

Abb. 8: Differenzdiagramm von Kantenvergleichen

3.4.2 Knotenvergleich basierend auf Umgebungsinformationen

Bei dieser Methode wird die Gleichheit zweier Knoten wie folgt definiert. Zwei Knoten V_U und V_G sind gleich, wenn die Kanten von V_U eindeutig auf die Kanten von V_G abgebildet werden können. Anhand der Kantendifferenzen wird die beste Zuordnung gesucht. Zusätzlich werden die unmittelbaren Nachbarknoten des Knotens V_G mit berücksichtigt. Dieser Mehraufwand macht den Knotenvergleich insgesamt robuster gegen das Auftreten von Störungen, hauptsächlich das Wegfallen von Knoten aus der Umgebungskarte.

Für die Kantenzuordnung wurden zwei Methoden implementiert. Zum einen die vollständige Suche und zum anderen ein ‚gieriger' Ansatz. Eine weitere Möglichkeit ist der ungarische Algorithmus [7]. Dieser findet für das Zuordnungsproblem ein optimales Ergebnis in akzeptabler Zeit. Aufgrund des Implementierungsaufwandes wurde er aber zu Gunsten der Näherungslösung verworfen.

Der gierige Ansatz sucht in jedem Schritt das am besten passende Kantenpaar und legt sich darauf fest. Ein Revidieren dieser Entscheidung ist nicht möglich. Dieser Ansatz kann somit die beste Lösung verfehlen, spart aber Laufzeit, da nicht alle Möglichkeiten ausprobiert werden. Es gilt jetzt abzuschätzen, welcher Ansatz zu wählen ist.

Die Laufzeit der vollständigen Suche kann folgendermaßen abgeschätzt werden. Da die Ausgangsdaten direkt der Midrange-Map entnommen sind, bei der jeder Pixel der Karte entweder einem Knoten oder einem Kantenelement entspricht, sind die Knoten mit höchstens acht Kanten verbunden. Das trifft auch dann zu, wenn Knoten direkt nebeneinander liegen, da diese dann durch eine Kante verbunden sind, die kein Kantenelement enthält. Eine Zuordnung der Kanten von V_U auf V_G erfordert damit höchstens *8*8* Kantenvergleiche. Das Berücksichtigen der Nachbarn von V_G erhöht diese von acht auf maximal *64*, da für jeden Nachbarknoten von V_G nochmals *7* neue Kanten hinzukommen. Insgesamt ergibt das theoretische *8*8+8*8*7 = 512* Kantenvergleiche pro Knoten. Hinzu kommen ca. $1{,}78 * 10^{14}$ Operationen für das Finden der optimalen Konfiguration. Praktisch sind die Knoten aber mit höchstens drei Kanten verbunden, was *27* Kantenvergleichen und *504* Suchoperationen entspricht. Obwohl es in der Praxis nicht häufig vorkommt, ist der theoretisch mögliche maximale Aufwand aber nicht zu vertreten. Deshalb wurde hinsichtlich der Erweiterbarkeit mit praktisch mehr als nur drei Kanten umzugehen, eine weitere Methode entwickelt. Als Preis für weniger Operationen muss allerdings der Kompromiss eingegangen werden, nicht immer die beste Lösung zu finden. Es kann aber von einem lokalen Optimum ausgegangen werden.

Der Algorithmus arbeitet folgendermaßen. Die Ergebnisse der Kantenvergleiche von V_U und V_G werden gegeneinander in einer Tabelle eingetragen. Aus jeder Spalte und jeder Zeile wird das Minimum gesucht. Für jedes Minimum werden die Indizes zusammen mit der Differenz zum jeweils zweitkleinsten Wert der Spalte bzw. Zeile abgespeichert. Aus dieser Menge beschreiben die Indizes die Kantenzuordnungen. Die Differenz charakterisiert den Fehler, der gemacht wird, wenn anstelle des kleinsten Wertes der zweitkleinste genommen wird. Dabei ist der Fehler um so größer, je größer die Differenz ist. Die Auswahl einer Kantenzuordnung fällt somit auf die größte Differenz, um den Fehler so gering wie möglich zu halten. Die Zeile und Spalte des angenommenen Wertes werden aus der Tabelle gestrichen, nachdem die Zuordnung gespeichert ist. Nachfolgend werden neue Minima mit den dazugehörigen Differenzen zwischen den verbleibenden zwei kleinsten Werten berechnet. Dann erfolgt die Auswahl der nächsten Zuordnung. Der Vorgang wird solange wiederholt, bis entweder alle Spalten oder alle Zeilen weggestrichen sind. Je nachdem, ob die Kanten von V_U (maximal acht, im Gegensatz zu V_G mit 64 möglichen Kanten) auf die Spalten oder Zeilen abgebildet werden, ist deren Anzahl auch zuerst aufgebraucht.

Die Betrachtung der Laufzeit ergibt maximal 512 Kantenvergleiche und eine obere Schranke von etwa 25000 Operationen pro Knoten. Die Anzahl der Vergleiche bestimmt sich wie bei der vollständigen Suche. Die Operationen setzen sich wie folgt zusammen. Die Schleife wird, aufgrund der maximal acht zuzuordnenden Kanten, acht mal durchlaufen. Pro Schleife finden höchstens 3072 Operationen statt, welche aus der Minimumsuche nebst Differenzbildung und der Speicherung zusammengesetzt sind. Diese Zahlen reduzieren sich in der Praxis pro Knoten (durch maximal drei Kanten pro Knoten) auf 27 Vergleiche und 486 Operationen.

Im Vergleich zur vollständigen Suche ist diese Vorgehensweise vor allem bei mehr als drei Kanten bei weitem schneller. Allerdings wird momentan auf Grund der Tatsache von praktisch nur drei Kanten pro Knoten und der Garantie die beste Lösung zu finden die vollständige Suche verwendet.

Dieser umgebungsorientierte Ansatz des Knotenvergleiches schlägt fehl, wenn sich die Topologie zwischen LUK und AUK zu stark unterscheiden. Tritt dieser Fall ein, erfolgt der Versuch zwei gleiche Knoten zu finden allein über Odometrieinformationen.

3.4.3 Knotenvergleich basierend auf Ortsinformationen

Dieser Ansatz verwendet folgende Definition der Gleichheit. Zwei Knoten sind gleich, wenn sie sich an der gleichen Position im Raum befinden. Im besten Fall kann diese Methode daher im Gegensatz zur vorhergehenden nur die Odometrie widerspiegeln. Da sie aber nur aufgerufen wird, wenn der Vergleich über die Kanten kein Ergebnis lieferte, stellt sie höchstens einen ungenaueren Kartenabschnitt dar, der gegenüber keiner neuen Karteninformation sicher vorzuziehen ist. Diese neuen Informationen können im nächsten Verknüpfungsschritt durchaus wieder durch Anwendung des ersten Verfahrens ausgebessert werden.

Um eine höhere Ergebnissicherheit mit dieser Methode zu erzielen, werden aus der aktuellen Umgebungskarte nicht nur wie in 3.2 beschrieben Kreuzungsknoten, sondern zusätzlich noch Erkundungsknoten berücksichtigt. Endknoten, das ergaben Tests, schwanken in ihrer Position zu sehr, als das sie sich für diese Vorgehensweise eigneten. Erschwerend kommt hinzu, dass aufgrund von Datenrauschen Endknoten in einem Scan

vorhanden sind, im nächsten jedoch wegfallen. In der AUK ergeben sich dadurch nur Kreuzungs- und Erkundungsknoten als mögliche Verbindungsstellen V_U.

In Abhängigkeit davon, ob V_U Kreuzungs- bzw. Erkundungsknoten ist, wird die Position, an der der Knoten V_G liegen kann, abgeschätzt. Dadurch ist es möglich die Roboterbewegung herauszurechnen. Wenn V_U ein Erkundungsknoten ist, wird die Position *EP*, an der V_G vermutet wird, auf Basis des zurückgelegten Weges zwischen AUK und LUK mit folgender Formel berechnet:

$$EP_{x,y} = V_{U\,x,y} - AUK_{x,y} + LUK_{x,y} \qquad (4).$$

Ist V_U jedoch ein Kreuzungsknoten, entspricht die Position von V_U der vermuteten Position von V_G. Diese Annahme kann aufgrund der geeigneten Wahl der Verbindungsstelle (siehe 3.2) getroffen werden. V_G ist der Knoten mit der geringsten euklidischen Distanz zur abgeschätzten Position.

Mit dem Knoten V_U aus der AUK und dem Knoten V_G aus der globalen Karte ergeben sich unter diesen Voraussetzungen drei Szenarien, die einer näheren Betrachtung wert sind. V_U und V_G können beide Erkundungsknoten sein oder V_U ein Kreuzungsknoten und V_G dazu ein Kreuzungs- oder Erkundungsknoten. Alle anderen Konstellationen sind nicht aussagekräftig genug, um verlässliche Aussagen treffen zu können. Die Abbildungen Abb. 9, Abb. 10 und Abb. 11 stellen Beispiele dieser Konstellation dar. In Abb. 9 ist schematisch dargestellt, dass aus der globalen Karte der Knoten V_G entfernt wird (rote Umrandung) und an dessen Stelle neue Kantenelemente angefügt werden und ein neuer Erkundungsknoten (V_U) hinzukommt (grüne Umrandung). Das Kantenelement mit dem kleinsten euklidischen Abstand zu V_G kennzeichnet das Element der Kante aus der AUK, ab welchem diese an die Kante der globalen Karte gebunden wird. Im Gegensatz dazu werden, wenn V_G und V_U Kreuzungsknoten sind, der Knoten V_G und alle im Baum nachfolgenden Elemente gelöscht und V_U mit seinen Nachfolgern an dessen Stelle angefügt. Die dritte Möglichkeit ist eine Kombination der ersten beiden. V_G ist ein Erkundungs- und V_U ein Kreuzungsknoten. Der Knoten V_G wird durch das entsprechende Kantenstück und V_U ersetzt, alle nachfolgenden Informationen aus der AUK schließen sich daran an.

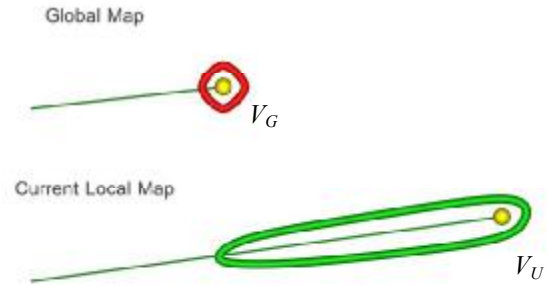

Abb. 9: Situation – zwei Erkundungsknoten

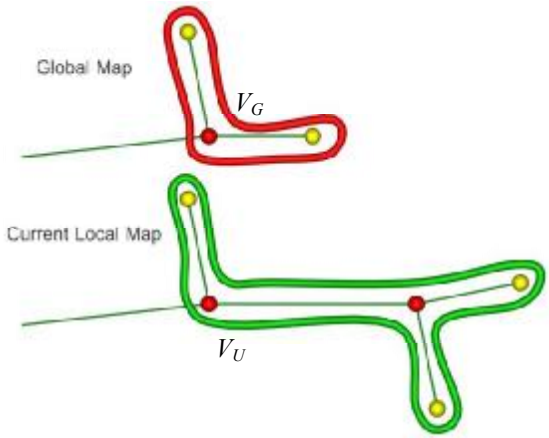

Abb. 10: Situation – zwei Kreuzungsknoten

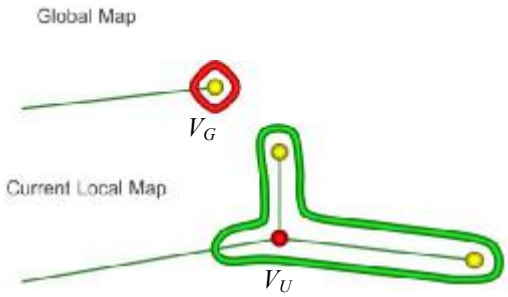

Abb. 11: Situation – je ein Erkundungs- und ein Kreuzungsknoten

Ein Fehlschlag auch dieser Vergleichsmethode hat zur Folge, dass ein Fall-Back-Verfahren durchgeführt wird, wenn die Bedingung dafür erfüllt ist.

3.4.4 Fall-Back-Strategie

Die Fall-Back-Routine sorgt für eine zusammenhängende globale Karte. Schlagen nämlich die zwei vorher beschriebenen Knotenvergleichsmethoden fehl, heißt das, dass die globale Karte nicht weiter vervollständigt wird. Wenn sowohl die topologischen Unterschiede als auch die Entfernungen zwischen der LUK und der AUK so groß werden, dass die Strategien basierend auf Umgebungs- und Ortsinformation Knotengleichheit nachzuweisen nicht mehr gelingen, ist es nur schwer möglich die globale Karte fortzusetzen. Die einzige Möglichkeit, den Kartenaufbau normal wiederaufzunehmen, besteht darin den Roboter nochmals über das Gebiet fahren zu lassen, in welchem die globale Karte nun endet. Um

das zu vermeiden prüft der Fall-Back-Ansatz nach jedem Fehlschlag die Entfernung zwischen der LUK und der AUK. Überschreitet diese den Wert von einem drittel der maximalen Scannerreichweite, wird die AUK in die globale Karte übernommen und durch eine ‚leere' Kante mit dem Rest der globalen Karte verknüpft. Dadurch wird neues Umgebungsmaterial in die Karte übernommen, welches gleichzeitig in die Struktur eingebunden ist. Eine ‚leere' Kante besitzt keine Informationen, im Sinne von Kantenelementen. Sie ist einfach eine strukturelle Verbindung, um einen ‚Abriss' zu reparieren. Mit den neuen Karteninformationen haben die Vergleichsstrategien wieder genug Informationen, um anhand der Umgebung des Weges des Roboters weitere Kartenelemente an die globale Karte anzuknüpfen.

3.5 Zusammenfassung

In diesem Kapitel wurden die Probleme und Lösungen des Kartenaufbaus erläutert. Die Motivation für diese Art des Kartenaufbaus, war die Redundanz der Midrange-Maps. Der Grund die Redundanzen der einzelnen Midrange-Karten zu beseitigen, liegt zum einen im erhöhten Speicheraufwand und zum anderen in der Garantie die Eindeutigkeit der einzelnen Umgebungen in der globalen Karte zu gewährleisten. Unter der Bedingung einer geeigneten Wahl der Verbindungsstelle einer Umgebungskarte, finden die in einer Hierarchie angeordneten Knotenvergleichsmethoden entsprechende Fusionierungspunkte in der globalen Karte. Die zwei Methoden basieren einerseits direkt auf den Umgebungsinformationen sowie andererseits nur auf Ortsinformationen von Knoten. Sind beide Strategien mit den ihnen zur Verfügung stehenden Informationen fehlgeschlagen, so greift eine Fall-Back-Lösung. Dies geschieht unter der Bedingung, dass der Roboter eine zu große Wegstrecke zurücklegt, nach der es keine Möglichkeit mehr für die zwei Methoden gibt, eine Verknüpfung zwischen der Umgebungskarte und der globalen Karte zu finden. Die Fall-Back-Lösung sorgt für den Zusammenhang der globalen Karte, sodass die Struktur ohne Unterbrechung gespeichert ist.

Das Ergebnis des Kartenaufbaus ist eine Baumstruktur, die globale Karte, welche die Informationen der Midrange Maps enthält, sodass möglichst wenig Redundanzen vorhanden sind. Dabei beinhaltet die globale Karte die komplette vom Roboter besuchte Umwelt. Diese Karte wird nun als Grundlage für eine Re-/Lokalisierung des Roboters benutzt.

4 Kartenmatching

Dieses Kapitel beschreibt die Vorgehensweise der Re-/Lokalisierung eines Roboters auf Basis der Karten, die mit Hilfe der Mittelachsen (siehe Kapitel 3) erstellt worden sind.

4.1 Idee

Um sich zu orientieren, wird eine Referenzkarte benötigt, welche die Umgebung beschreibt, in der der Roboter agieren soll. Solch ein Referenzsystem stellt die globale Karte aus dem letzten Kapitel dar. Die grundlegende Idee für das Finden der Roboterposition besteht in dem Vergleich der lokalen Umgebung (LOK) des Roboters mit der Referenzkarte (GLK). Dabei bilden ähnlich, wie in Kapitel 3.3 beschrieben, die Knoten der Baumstruktur den Ansatzpunkt für einen Vergleich. Jedoch werden im Unterschied dazu nicht zwei gleiche Knoten gesucht, vielmehr wird versucht mehrere Knoten der lokalen Umgebungskarte gleichzeitig auf die globale Karte abzubilden.

Die Knoten der LOK müssen sich dazu topologisch an den gleichen Positionen, wie die Knoten der Referenzumgebung befinden. Aufgrund der Definition des Aufbaus der Karten durch die EMA (siehe 2.1) ist dieser Punkt gesichert.

Die Vorgehensweise ist dabei folgende. Für jeden Knoten aus der LOK wird eine Liste erstellt, in der alle möglichen Kandidaten aus der GLK enthalten sind, die dem Knoten der LOK entsprechen könnten. Diese Hypothesen werden durch verschiedene Gleichheitsmaße gewichtet. Der Knoten mit der besten Hypothese kann dadurch als Ausgangspunkt für einen Candidate Elimination Algorithmus, der auf die Idee von Mitchell [8] zurückgeht, verwendet werden. Das Ergebnis ist eine vollständige Knotenzuordnung der LOK auf die GLK. Auf Grund dieser Abbildung kann die Position der LOK aus der Referenzkarte berechnet werden. Folglich ist die Roboterposition, die relativ zur LOK bekannt ist, leicht nachvollziehbar.

Die Hypothesen zu bilden, ist der erste Schritt in Richtung Re-/Lokalisierung. Diese so gut es geht abzuschätzen, ist Aufgabe des folgenden Abschnittes.

4.2 Hypothesenbildung

Das Ziel der Hypothesenbildung ist, für jeden Knoten der LOK herauszufinden, welchem Knoten der GLK er entsprechen könnte. Danach werden die wahrscheinlichsten Kandidaten für jeden Knoten in einer Liste gespeichert.

Gute Hypothesen zu finden, ist Voraussetzung für das Gelingen des Candidate Elimination Algorithmus. Dazu gehört, wie gesagt möglichst zum Knoten aus der LOK passende Kandidaten aus der GLK zu finden. Zum anderen und das ist die zweite Bedingung, darf die richtige Lösung nicht ausgeschlossen werden. Würde sie an dieser Stelle schon ausgeschlossen, kann die richtige Lösung nicht gefunden werden. Es ist wenig erfolgversprechend, wenn ein scheinbar gleicher Knoten einen nicht so guten Kandidaten verdrängt, obwohl dieser das Ergebnis liefern würde.

Die erste Bedingung – die Gleichheit von Knoten – ist durch zwei Definitionen gekennzeichnet. Eine ist bereits aus Kapitel 3.4.2 bekannt: zwei Knoten sind gleich, wenn die Kanten des einen eindeutig auf die Kanten des anderen abgebildet werden können. Wie das Gleichheitsmaß berechnet wird, ist dementsprechend in diesem Kapitel beschrieben. Die zweite hier benutzte Definition geschieht über das Voronoi-Level: ist das Voronoi-Level zweier Knoten gleich, so stimmen sie überein. Dieses Maß ist natürlich sehr ungenau und es werden mehr Knoten in eine Klasse gelegt, als gleich sind. Aber, und das ist der entscheidende Punkt, der ‚richtige' Knoten kommt ebenfalls in der Lösungsmenge vor. Der Unterschied zu der schon besprochenen Methode ist der Zeitaufwand für das Berechnen der Gleichheitsmaße. Das verhältnismäßig einfache Vergleichen des Voronoi-Levels (über die quadratische Differenz) braucht sehr viel weniger als der Vergleich über die Gesamtheit der Kanten.

Während der Tests hat sich die Voronoi-Level Methode als robuster erwiesen, vor allem gegenüber lokalen Karten mit fehlenden Informationen. Allerdings liefert im Gegensatz dazu die Kanten Methode genauere Positionen des Roboters. Das ist damit zu begründen, dass durch die fehlenden Informationen in der LOK diese durch das Voronoi-Level auf mehrere Gebiete der GLK abgebildet werden kann. Dadurch entstehen natürlich mehr Positionsschätzungen.

Im Hinblick auf die Geschwindigkeit für die Benutzung der Kanten Methode, ist sie um eine Technik erweitert worden. Diese dient dazu absolut falsche Knoten schon vorher auszuschließen, damit wirklich nur interessante Kandidaten den langen Weg der Kantenvergleiche gehen müssen. Dazu wurde für die Knoten eine Kennzahl eingeführt, die die Fläche beschreibt, in welcher die mit diesem Knoten verknüpften Kanten liegen. Die Fläche wird folgendermaßen berechnet. Ausgehend von dem Knoten werden in Richtung jeder Kante 50 Kantenelemente für die Flächenberechnung genutzt. Sollte eine Kante aus weniger als 50 Elementen bestehen, wird von dem Knoten, an der sie ankam, die robusteste Kante ermittelt und weiter verwendet. Das wird solange fortgeführt, bis 50 Elemente verwendet worden sind, oder ein Endpunkt erreicht wurde. Robust heißt an dieser Stelle, dass die Kante das größte Fläche pro Kantenelement-Verhältnis besitzt, was wiederum bedeutet, dass die Kante in jedem Kartenaufbau mit großer Wahrscheinlichkeit an der selben Position erscheint. Die Kennzahl wird nun ermittelt, indem jedes Kantenelement mit der Pixelauflösung der Midrange-Map multipliziert und zur Gesamtfläche aufsummiert wird. Eine nachfolgende Sortierung ermöglicht es die besten Kandidaten herauszunehmen und den Rest zu verwerfen.

Als Hypothesen werden für jeden Knoten der LOK die 20 besten Kandidaten verwendet. Nur 20 Hypothesen zu verwenden, hat sich im Test mit Karten, die aus ca. 60 Knoten und etwa 95 Kanten bestanden, bewährt. Hier gilt es natürlich einen Mittelweg zwischen genug Hypothesen und nicht zu vielen Informationen zu finden. Genug Hypothesen, damit die Lösung dabei ist, aber nicht soviel, dass es vieldeutige Abbildungen der LOK auf die GLK gibt.

Eine zusätzliche Bedingung ergab sich im Test. Die Erkundungsknoten, sowie deren direkte Nachbarn bekommen als Hypothesen alle Knoten der GLK. Der Grund dafür sind die mangelnden Informationen der angrenzenden Kanten. Die richtige Lösung wird damit sichergestellt.

Für das nachfolgend beschriebene Verfahren des Candidate Elimination Algorithmus wurde die Methode der Kantenvergleiche mit der Vorsortierung durch die Flächen verwendet.

4.3 Candidate Elimination Algorithmus

Die Idee des Candidate Elimination Algorithmus (CEA) beruht auf dem Gedanken ausgehend von einer Startbedingung solange Kandidaten, in diesem Fall Hypothesen auszuschließen, bis die Lösung übrig bleibt. Dabei ist eine gute Ausgangsbedingung wichtig, da es ansonsten auch zu keinem Ergebnis kommen kann, was natürlich besser ist, als ein falsches.

Die Ausgangsbedingung ist die beste Hypothese, die für die Knoten der lokalen Karte nach Kapitel 4.2 gefunden wurde. Sie stellt den passendsten Kandidaten dar und wird als gegeben vorausgesetzt. Dadurch ist gesichert, mit hoher Wahrscheinlichkeit eine Lösung zu finden. Sollte das dennoch nicht der Fall sein, wird Backtracking angewendet. Das heißt, das mit dem zweitbesten Kandidaten initialisiert und der CEA neu gestartet wird.

Der CEA arbeitet nun wie folgt. Die direkten Nachbarknoten des Knotens mit der Starthypothese aus der lokalen Karte werden als ‚zu verarbeiten' markiert. Die Menge der festgelegten Hypothesen H_{fixed} wird mit der Starthypothese initialisiert. Hier ist anzumerken, dass wenn von Hypothesen gesprochen wird, nur Knoten der globalen Karte gemeint sein können. Die markierten Knoten werden anschließend auf ihre Hypothesen hin überprüft und eventuell angepasst. Alle Hypothesen, die nicht direkte Nachbarn der Knoten aus der Menge H_{fixed} sind, werden gelöscht. Die übrig gebliebenen Hypothesen der Knoten werden in die Menge H_{fixed} übernommen. Ist die Menge der Hypothesen eines Knotens geändert worden, so werden dessen direkte Nachbarn wiederum markiert. Mit den markierten Knoten wird nun wieder wie oben beschrieben umgegangen. Dieser Vorgang wird so lange wiederholt, bis keine Knoten mehr markiert sind.

Im besten Fall, hat nun jeder Knoten genau einen Kandidaten aus der globalen Karte bekommen. In der Praxis blieben jedoch meist zwei oder drei Hypothesen pro Knoten übrig. Im schlechtesten Fall wurden alle Hypothesen verworfen. Das kann geschehen, wenn die Startbedingung schlecht gewählt worden ist. In diesem Fall wird, wie oben bereits erwähnt, die zweitbeste Hypothese für die Initialisierung verwendet und im Falle des wiederholten Verwerfens aller Hypothesen die nächste.

Natürlich ist, im Einsatz unter Echtzeitbedingungen auf die vorhandene Restzeit zu achten, wenn ein neuer Durchlauf des Candidate Elimination Algorithmus gestartet werden soll.

4.4 Positionsbestimmung

Nachdem der Candidate Elimination Algorithmus durchgelaufen ist, müssen aus den restlichen Hypothesen die möglichen Roboterpositionen extrahiert werden. Hierfür werden alle noch vorhandenen Hypothesen verwendet. Die relative Position des Roboters gegenüber den Knoten der lokalen Karte ist bekannt. Durch die restlichen Hypothesen sind die lokalen Knoten auf die der globalen Karte abgebildet worden. Die Roboterposition ist nun relativ zu den Knoten der globalen Karte berechenbar. Das Referenzsystem gibt damit die absolute Position des Roboters preis.

4.5 Zusammenfassung

Das hier verwendete Kartenmatching ist der Versuch durch geeignete Features, in diesem Fall sind es Knoten, eine Abbildung einer lokalen Karte eines mobilen Roboters auf eine Referenzkarte zu finden. Der Kern ist ein Candidate Elimination Algorithmus, der auf Basis vorher aufgestellter Hypothesen operiert. Die Hypothesenmengen für die Knoten der lokalen Karte werden durch Vergleichen mit den Knoten des Referenzsystems ermittelt. Das Ergebnis des Candidate Elimination Algorithmus sind Abbildungen der lokalen Karte auf Teile der globalen Karte. Durch diese lassen sich mögliche Roboterpositionen abschätzen.

Am Ende des Kartenvergleiches stehen mehrere Positionen, an denen sich der Roboter befinden könnte. Das ist aufgrund mehrerer vorhandener Hypothesen der Fall. Die Abbildungen Abb. 12 und Abb. 13 stellen diesen Sachverhalt in der Praxis dar. Durch die lokale Karte Abb. 12 wurden in der Referenzkarte Abb. 13 mehreren Schätzpositionen (rote Punkte) berechnet. Eine Häufung tritt offensichtlich an der realen Roboterposition (blauer Punkt in Abb. 12) auf, kann aber mit Sicherheit in einem Verarbeitungsschritt nicht garantiert werden. Daher muss zusätzlich eine Wahrscheinlichkeitsverteilung über die Schätzpositionen erfolgen, beispielsweise durch eine Markov Lokalisierung. Das ist aber nicht Teil dieser Arbeit und dient daher nur als Verbesserungsvorschlag.

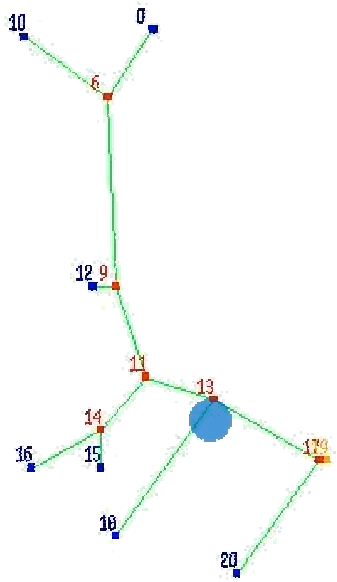

Abb. 12: lokale Karte mit Roboterposition

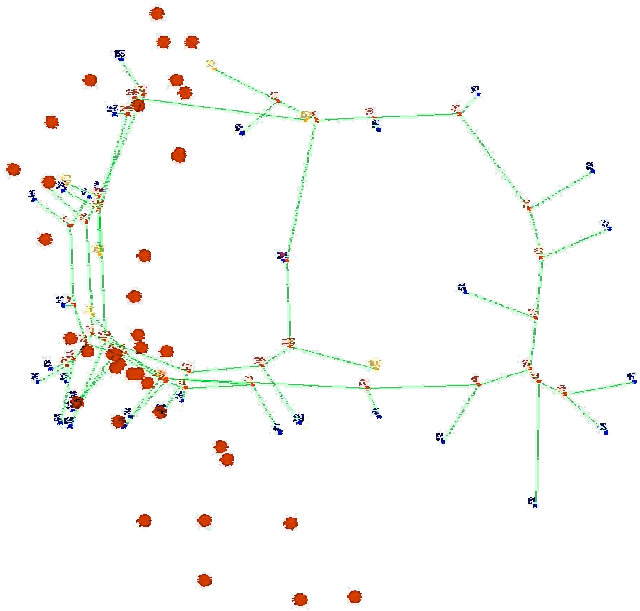

Abb. 13: globale Karte mit Positionsschätzungen

5 Ergebnisse

In diesem Kapitel finden sich die Resultate des Kartenaufbaus und im Anschluss daran des –vergleiches. Abschließend werden die Einsatzfähigkeiten dieser Methoden besprochen.

Die Aufgabe eine globale Karte aus den Mittelachsen zu erstellen, ist im grundlegenden abgeschlossen. Weiterführende Probleme sind als Folgeprojekte vorgesehen. So sind bei dem hier beschriebenen Verfahren doppelt vorkommende Pfade durchaus noch zulässig. Sie entstanden dadurch, dass der Roboter diese Strecken mehrmals befahren hat. Die Lösung dieses Problems sollte als weiterführenden Aufgabe bearbeitet werden, um den Kartenvergleich zu erleichtern.

Es zeigte sich, dass für unterschiedliche Umgebungen unterschiedliche Parameter benutzt werden müssen. Ausschlaggebend ist die Genauigkeit des Voronoi-Levels, mit welchem die Mittelachsen und von ihnen abgeleitet die Karten aufgebaut sind. Abb. 14 zeigt die Karte eines ca. 15x7 m großen Raumes. Knoten (rote, blaue und gelbe Punkte) und Kanten (grüne Verbindungen) sind hier einschließlich der Knoten-ID's zu erkennen. Die feineren Informationen (Geometrie), welche die Kantenelemente bieten, sind in Abb. 15 ablesbar. Je heller die Kanten sind, desto kleiner sind die in den Kantenelementen enthaltenen Voronoi-Werte, das heißt desto geringer ist der Abstand zum nächsten Hindernis. Abb. 16 zeigt die aus der Karte rekonstruierte Grundfläche des Raumes mit den dazugehörigen Hindernissen. Die entsprechenden Parameter, die für den Aufbau der Karte verwendet wurden, sind dem Kapitel 6.2 zu entnehmen. Welche Parameter für welche Umgebungen notwendig sind, ist von der jeweiligen Aufgabe des Roboters abhängig. Generell gilt, dass dem höheren Detailreichtum der Umgebung immer ein größerer Speicherverbrauch der Karte zum einen und ein höherer Algorithmusaufwand zum anderen gegenüber steht.

Beim Kartenvergleich konnten folgende Beobachtungen gemacht werden. Mit der Karte aus Abb. 17 als gültiges Referenzsystem wurde der markierte Knoten aus der Karte in Abb. 18 an den dunkel markierten Stellen der Referenzkarte wiedergefunden. Je dunkler die Markierungen sind, um so höhere Gleichheit wurde erkannt. Das zeigt, dass der Ansatz des Vergleiches über die Mittelachsen funktioniert. Es stellte sich jedoch auch heraus, dass die Karten aufgrund von Messungenauigkeiten der Sensoren für eine Re-/Lokalisierung in einigen Fällen zu ungenau waren. Abb. 13 verdeutlicht dieses Problem. Die

Schätzpositionen sind über das ganze Höhenspektrum verteilt. Es ist daher für ein sicheres System unabdingbar die Verteilung der Postitionshypothesen über mehrere Verarbeitungsschritte zu betrachten.

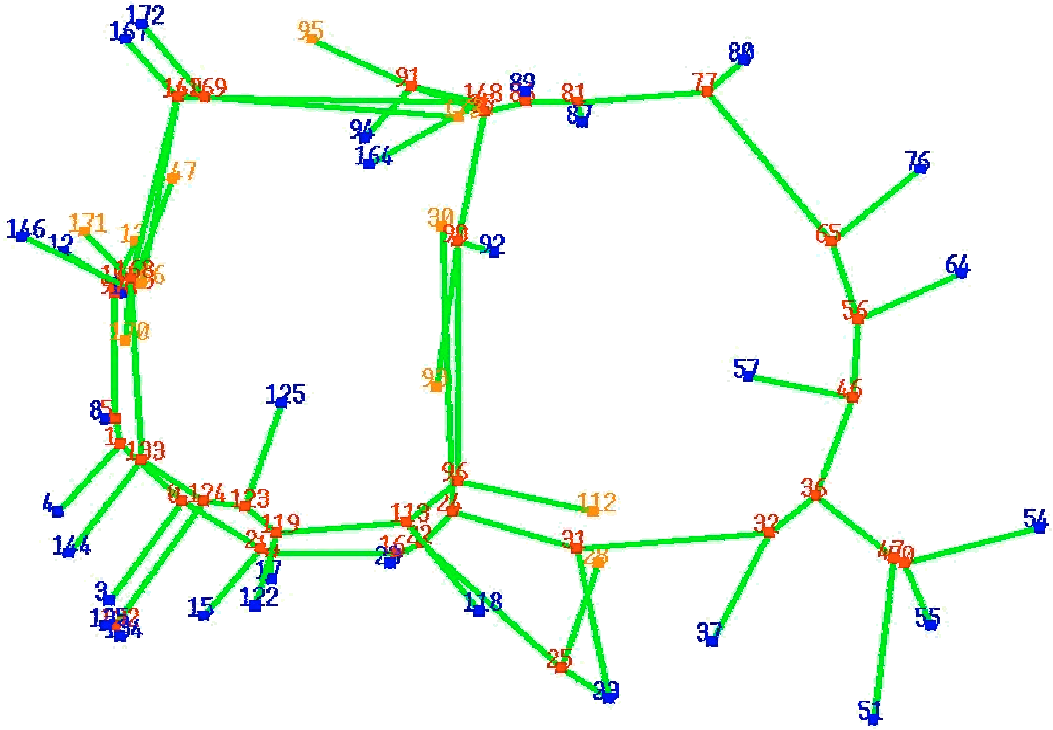

Abb. 14: Karte nach Mittelachsen

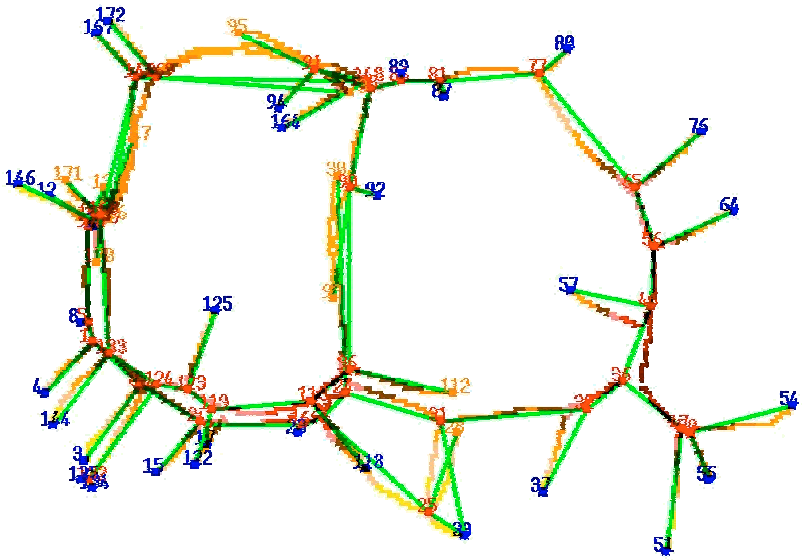

Abb. 15: Karte mit geometrischer Information

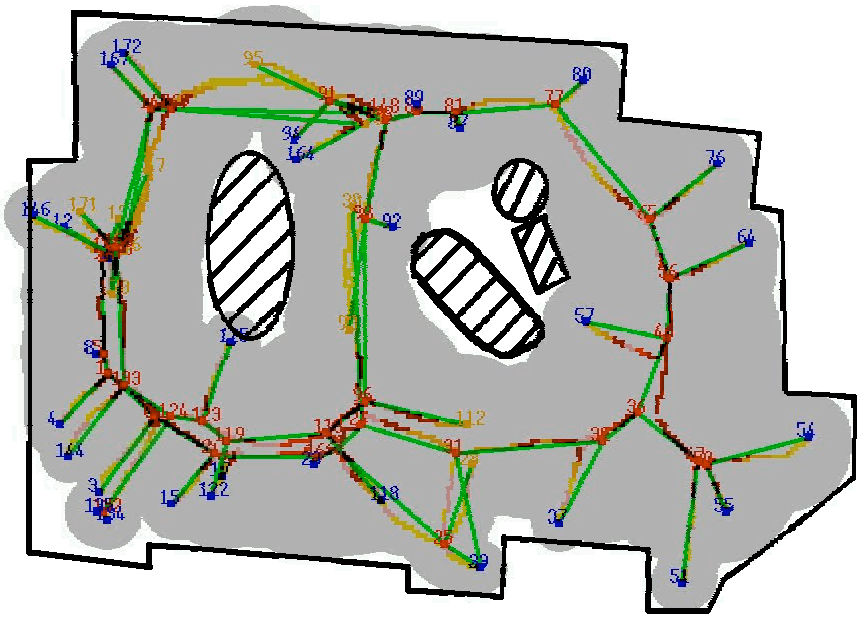

Abb. 16: Karte mit dem Grundriss der Umgebung und der Hindernisse

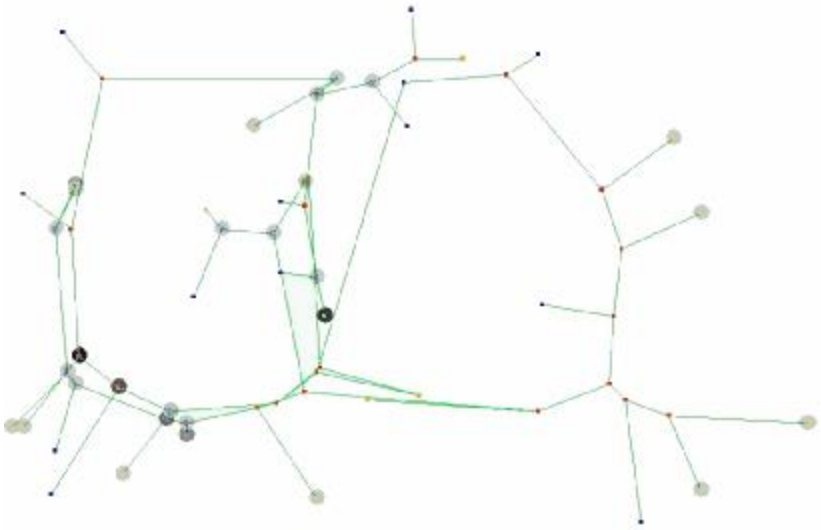

Abb. 17: Referenzkarte mit Knotenhypothesen

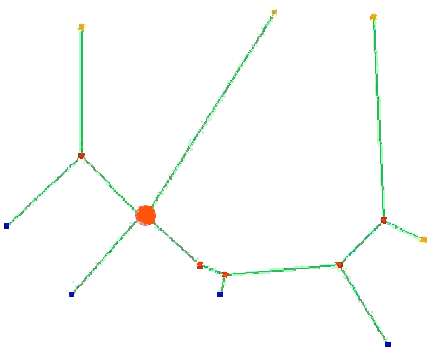

Abb. 18: Umgebungskarte mit markiertem Knoten, der in der Referenzkarte gefunden werden soll

Der Einsatz der Re-/Lokalisierung auf Mittelachsen Basis sollte auch mit den angesprochenen Verbesserungen nicht als universelle Lösung betrachtet werden, sondern zusätzlich zu den bisherigen Methoden arbeiten. Lange Koridore bleiben für die Mittelachsen genauso wie für die kantenbasierten Verfahren ein Problem. Die Querrichtung kann leicht durch die Punkte der Mittelachse bzw. den Abstand zu den Wänden herausgefiltert werden, in Längsrichtung ist aber höchstens die Odomietriegenauigkeit möglich. Daher sollten bisherige Verfahren die Ergebnisse mit den Mittelachsen im Zeitverlauf fusionieren, um wenigsten den Ausgangspunkt der Messung basierend auf Odometriedaten so genau wie möglich zu halten. Im Gegensatz dazu sind runde Säulen in der Umgebung für die Mittelachsen kein Problem, kantenbasierte Verfahren scheitern jedoch, wenn außer diesen Säulen keine Bezugspunkte vorhanden sind. Die Mittelachsen können auch der Verbesserung der Wegplanung dienen. Der Roboter kann genau entlang der geometrischen Kante dirigiert werden, sofern der Platz ausreicht, um hindurch zu fahren. Denn die geometrische Kante liegt immer in der Mitte der Freifläche. Der Platz kann einfach über das Voronoi-Level überprüft werden. Ist dieses größer als die halbe benötigte Breite des Roboters ist eine Durchfahrt möglich.

Der Bereich der Roboternavigation, so kann zusammenfassend bestätigt werden, wird durch das auf Mittelachsen basierte Verfahren erleichtert und Roboter können damit auch in bisher schwer oder unzugänglichen Umgebungen agieren.

6 Implementierung

Für die Implementierung wurde die Entwicklungsumgebung von Borland C++ 5.0 verwendet. Als Programmiersprache war C durch das bereits vorhandene System vorgegeben.

Auf Grund der Größe des Systems ist es im Sinne der Übersichtlichkeit in einzelne Module unterteilt. Sie bestehen aus einer oder mehreren in logische Funktionseinheiten zusammengefassten Dateien. So sind Module vorhanden, die für die Wegplanung, für den Laserscanner oder für die Reinigung zuständig sind. Durch die Modularisierung wird außerdem die Möglichkeit geschaffen, den Funktionsumfang einfach zu erweitern oder bestehende Teile zu ersetzen. So ist beispielsweise das Modul für die Steuerung der Bürsten für die Reinigung bei einem Reinigungsroboter sinnvoll, bei einem Transportroboter aber sicherlich überflüssig. Die einzelnen Module können untereinander über definierte Schnittstellen Daten austauschen. Des Weiteren besitzt jedes Modul, welches vom Taskmanager berücksichtigt werden soll, Funktionen, um den Zugriff darauf zu ermöglichen. Für das Echtzeitsystem notwendige Informationen über die Laufzeit der einzelnen Module werden in diesen über eine entsprechende Datenstruktur zur Verfügung gestellt. Das System läuft auf dem Echtzeitsystem RMOS. Unter Linux und Windows existiert ein Simulator, der Umgebungensdaten abspielt und sowohl Sensorinformationen liefert, als auch Roboterreaktionen simuliert und dabei das Echtzeitverhalten berücksichtigt. Die Umgebungsdaten sind zuvor mit dem realen Roboter über die Sensoren aufgenommen worden. Dadurch wird der Test von neuen Funktionen erheblich erleichtert.

Im Rahmen dieser Arbeit wurden der Kartenaufbau und das Kartenmatching in das vorhandene Modul SymMap integriert. Dieses besteht aus der Datei c_symmap.h, sowie c_symmap.c. Die Hauptdatentypen, sowie -definitionen, die Anwenderschnittstelle, die Funktionen innerhalb des Moduls und die damit verbundenen Tools werden an dieser Stelle in tabellarischer Form aufgeführt und kurz beschrieben.

6.1 Datenstrukturen

Der Hauptdatentyp ist ein Baum, welcher die Karte der Umgebung nach dem Mittelachsen Schema darstellt. Dieser kann beliebig groß, im Rahmen des vorhandenen Speicherplatzes werden. Zur genaueren Beschreibung wird auf das Kapitel 2.2 verwiesen. Des Weiteren

sind hier alle Datenstrukturen, die zur Gestaltung der Karte notwendig und keine C-Basisdatentypen sind, aufgeführt.

Strukturname	Komponenten	Beschreibung
EdgeTypes	Dieser Datentyp beschreibt alle Arten von Pixeln auf der Midrange Map. Dazu gehören die Knoten als auch die Kanten. Folgende Belegungen sind möglich: *edgePoint* (Kantenelement), *endPoint* (Endknoten), *explorePoint* (Erkundungsknoten) und *crossingPoint* (Kreuzungsknoten).	
RT_Edge	uint8 TypeID struct rt_Edge *prev struct rt_Edge *next	Die ersten drei Variablen sind für die interne Listenverwaltung notwendig.
	uint32 ID	Eindeutige Kantennummer
	RT_EdgeElement *edgeElements	Zeiger auf die Kantenelemente der Kante
	struct rt_Vertice *nextVertice	Zeiger auf den nachfolgenden Knoten
	uint32 numberOfEdgeElements	Anzahl der Kantenelemente, die zu dieser Kante gehören
	float area	Fläche, in der diese Kante liegt
	RT_FConf posFirstElement	Position des ersten Kantenelementes
	RT_FConf posLastElement	Position des letzten Kantenlementes
RT_EdgeElement	uint8 TypeID struct rt_EdgeElement *prev struct rt_EdgeElement *next	Die ersten drei Variablen sind für die interne Listenverwaltung notwendig.
	float voronoiWidth	Voronoi-Level des Elementes
	float curvature	Biegung der Kante an diesem Element (nicht benutzt)
	uint8 dirNextElement	Kennzeichnet die Richtung des nächsten Kantenelementes in dieser Kante (..0-oben..2-links..)

RT_FConf	float x float y float beta	Position im in der Ebene mit entsprechender Ausrichtung
RT_GlobalGraph	RT_Vertice *vertices RT_Vertice *verticeList uint32 verticeCounter uint32 edgeCounter uint32 edgeElementCounter uint32 verticeID uint32 edgeID uint32 maxEdgeElements uint32 maxEdges uint32 maxVertices	Zeiger auf den ersten Knoten Knotenliste der Karte über die interne Listenverwaltung Anzahl der Knoten auf der Karte Anzahl der Kanten auf der Karte Anzahl der Kantenelemente auf der Karte Eindeutige Nummer für den zuletzt eingefügten Knoten Eindeutige Nummer für die zuletzt eingefügte Kante Maximal zulässige Knotenanzahl Maximal zulässige Kantenanzahl Maximal zulässige Kantenelementanzahl
RT_MatchVerticeID	uint8 TypeID struct rt_MatchVerticeID *prev struct rt_MatchVerticeID *next uint32 verticeID bool valid	Die ersten drei Variablen sind für die interne Listenverwaltung notwendig. Nummer des passenden Knoten Kennzeichnet einen möglichen Treffer beim Vergleich von Karten
RT_RobPosHyp	uint8 TypeID struct rt_RobPosHyp *prev struct rt_RobPosHyp *next float match_value RT_FConf robPos	Die ersten drei Variablen sind für die interne Listenverwaltung notwendig. Vergleichswert von diesem und dem'vertice_id' Knoten Position, an der die Umgebung aufgenommen wurde (Roboterposition)

	uint32 vertice_id	Nummer des Knoten, mit dem verglichen wurde
RT_StackElement	Diese Struktur stellt die Nutzung eines Stacks bereit, um die Baumstruktur der globalen Karte topologisch zu durchlaufen. Diese Vorgehensweise ist bei der Verfolgung eines Pfades im Baum notwendig.	
RT_Vertice	uint8 TypeID struct rt_Vertice *prev struct rt_Vertice *next struct rt_Vertice *prevVertice uint32 ID float voronoiWidth RT_FConf absPoint RT_Edge *edges EdgeTypes edgeType RT_MatchVerticeID *matchList float errorValue bool searchAgain bool seen bool allHypotheses RT_RobPosHyp *robPosHyp	Die ersten drei Variablen sind für die interne Listenverwaltung notwendig. Zeiger auf den vorhergehenden Knoten im topologischen Sinne Eindeutige Knotennummer Voronoi-Level des Knotens Absolute Weltposition Zeiger auf die abgehenden Kanten Typ des Knotens (Erkundungs-, Kreuzungs- oder Endknoten) Liste der Knoten der möglichen Kandidaten der Vergleichskarte Summe des Fehlerwertes für den matchList Parameter Kennzeichnet die Notwendigkeit eines erneuten Betrachtens Dieser Knoten ist abgearbeitet Kennzeichnet, dass für diesen Knoten alle Hypothesen gelten Positionshypothese

6.2 Definitionen

Nachfolgend sind alle im Modul SymMap für den Kartenaufbau und den –vergleich notwendigen Definitionen aufgeführt und erläutert. Die definierten Werte können und sollen natürlich im Rahmen der Anwendung angepasst werden. Sie dienen hier nur als Vorschlag für die getesteten Umgebungen (siehe Kapitel 5).

Name	Belegung	Beschreibung
GG_MAX_SEARCH_RUNS	5000	Maximale Durchläufe bei der vollständigen Suche während des Knotenvergleiches basierend auf Umgebungsinformationen
MAX_G_EDGES	60.000	Maximale Kantenanzahl
MAX_G_EDGE_ELEMENTS	200.000	Maximale Kantenelementanzahl
MAX_G_MATCHVERTICEID	MAX_G_VERTICES * 10	Maximale Knotenhypothesenanzahl
MAG_G_POSLIST_ELEMENTS	1000	Maximale Roboterpositions-schätzungen
MAG_G_STACK_SIZE	5000	Maximale Stackgröße
MAX_G_VERTICES	30.000	Maximale Knotenanzahl
MAX_GG_COUNTER	20	Maximale Länge der Knoten-hypothesenliste pro Knoten
MAX_GG_CROSSWAYS	8	Maximale Kanten, die von einem Knoten ausgehen können
MAX_GG_EDGES	MAX_G_EDGES / 2	Maximale Kantenanzahl pro Referenzkarte
MAG_GG_EDGE_ELEMENTS	MAX_G_EDGE_ELEMENTS / 2	Maximale Kantenelementanzahl pro Referenzkarte
MAX_GG_EETOCOMPARE	50	Anzahl der Kantenelemente, die bei einem Knotenvergleich in jede Kantenrichtung überprüft werden
MAX_GG_MISS_MATCH_VALUE	2	Anzahl der Knoten, die nach einem Kartenvergleich nicht übereinstimmen müssen, um noch ein positives Ergebnis zu erhalten
MAX_GG_VERTICES	MAX_G_VERTICES / 2	Maximale Knotenanzahl pro Referenzkarte

MAX_MG_EDGES	MAX_G_EDGES / 4	Maximale Kantenanzahl pro Vergleichskarte
MAX_MG_EDGE_ELEMENTS	MAX_G_EDGE_ELEMENTS / 4	Maximale Kantenelementanzahl pro Vergleichskarte
MAX_MG_VERTICES	MAX_G_VERTICES / 4	Maximale Knotenanzahl pro Vergleichskarte
MAX_SG_EDGE_ELEMENTS	MAX_G_EDGE_ELEMENTS / 8	Maximale Kantenelementanzahl pro Umgebungsgraphen
MAX_SG_EDGES	MAX_G_EDGES / 8	Maximale Kantenanzahl pro Umgebungsgraphen
MAX_SG_VERTICES	MAX_G_VERTICES / 8	Maximale Knotenanzahl pro Umgebungsgraphen
RT_HALF_MIGRID_RANGE	(mi_DIM * MiGrid_GetCellSize()) / 2	Die Hälfte der Reichweite der Midrange-Map (4,512 m)
RT_QUARTER_MIGRID_RANGE	(mi_DIM * MiGrid_GetCellSize()) / 4	Ein Viertel der Reichweite der Midrange-Map (2,256 m)
RT_THIRD_MIGRID_RANGE	(mi_DIM * MiGrid_GetCellSize()) / 3	Ein Drittel der Reichweite der Midrange-Map (3,008 m)

6.3 Anwenderschnittstelle

Die Schnittstelle, die dem Anwender für den Aufbau und für das Matching geboten wird, umfasst vier Funktionen, welche nachfolgend erläutert sind. Die zwei ursprünglich geplanten Funktionen *Add* und *Search* wurden um die zwei Funktionen *SymMap_GetRobPosHyp* und *SymMap_DeleteRobPosHyp* erweitert. Im Laufe der Entwicklung zeigte sich, dass Verarbeitungsmöglichkeiten für die Positionshypothesen

bereit gestellt werden müssen, die mehr als die Funktion *Search* übernehmen. Hier ist die Schnittstelle für eine weiterführende Betrachtung der gefundenen Positionsschätzungen.

Funktionsname	Parameterliste	Beschreibung
Add	LocalGraph, GlobalGraph	Diese Funktion setzt die in Kapitel 3 beschriebene Funktionalität des Graphaufbaus um. Der Parameter LocalGraph ist die aktuelle Umgebungskarte des Roboters, GlobalGraph stellt die globale Karte dar. GlobalGraph wird in der Funtion manipuliert, sodass hier auch das Ergebnis des Anfügens der Umgebungskarte an die globale Karte zurückgegeben wird.
Search	MatchGraph, GlobalGraph	Search liefert die beste Positionsschätzung zurück, die sie aufgrund des ersten besten Matches ermittelt. Da sie nur eine Schätzung zurückgibt, kann hier nur schwer ein weiteres Verfahren angesetzt werden. GlobalGraph ist die Referenzkarte und MatchGraph die Karte für einen Vergleich.
SymMap_ DeleteRobPosHyp	Vertice_list	Löscht die aus der Funktion SymMap_GetRobPosHyp erhaltene Knotenliste einschließlich der Hypothesen.
SymMap_ GetRobPosHyp	Migrid_pos, Vertice_list, Cm_id	Diese Funktion ermittelt alle Knoten der aktuellen Vergleichskarte mit den dazu-gehörigen Knotenhypothesen aus dem Referenzsystem. Cm_id liefert die Möglichkeit verschiedene Vergleichsmethoden zu wählen. Das sind CM_0_MAG (Vergleich basierend auf das Voronoi-Level der Knoten), CM_1_AREA (basierend auf die Fläche, die die Kanten des Knoten einnehmen), CM_2_EDGE (basierend auf den Kanteninformationen des Knoten) und CM_3_AREA_EDGE (eine Mischung aus

		CM_1_AREA und CM_2_EDGE).

6.4 Kartenaufbau und –vergleich

Die nachfolgend beschriebenen Funktionen entstanden während der Entwicklung des hier beschriebenen Verfahrens für den Kartenaufbau und den –vergleich. Es wird knapp erläutert, welche Aufgabe sie erfüllen. Für detaillierte Informationen muss der Quelltext konsultiert werden.

Funktionsname	Parameterliste	Beschreibung
AddEdge	Edge, Vertice, id	Fügt eine Kante mit der Nummer id dem Knoten Vertice hinzu. Ist eine Kante Edge vorhanden, wird die neue Kante an Edge angefügt, ansonsten wird eine neue Kantenliste erzeugt.
AddEdgeElement	EdgeElement, Edge, VoronoiEdge	Fügt der Kante Edge ein neues Kantenelement aus den Informationen von VoronoiEdge hinzu. Wenn EdgeElement existiert, so wird das neue Element angefügt, ansonsten wird eine neue Liste erstellt.
AddEdgeElements	GlobalGraph, Edge, VoronoiEdge	Fügt neue Kantenelemente aus der Mittelachsen Struktur VoronoiEdge an die Kante Edge der Karte GlobalGraph an.
AddVertice	GlobalGraph, Vertice, Edge, VoronoiEdge, id	Fügt der Karte GlobalGraph einen neuen Knoten mit der Nummer id an. Vertice ist der Vorgängerknoten, Edge eine mögliche Kante, die auf den neuen Knoten zeigt. VoronoiEdge ist das Element, aus dem der neue Knoten entsteht. Sollte Vertice nicht existieren, wird eine neue Knotenliste erstellt.
Bind	GlobalGraph, Vertice, VoronoiEdge, EdgeComingFrom	Fügt neue Informationen der Karte GlobalGraph an den Knoten Vertice aus der gegebenen Mittelachsen Datenstruktur VoronoiEdge an. EdgeComingFrom markiert den schon verarbeiteten Bereich.

cmp_MatchStats	El1, El2	Vergleichsfunktion für Quicksort. Vergleicht zwei Elemente RT_StatisticsElement Elemente El1 und El2 hinsichtlich ihrer stats Variablen.
CompareGlobalEdges	Vertice1, Edge1, Vertice2, Edge2	Vergleicht zwei Kanten Edge1 und Edge2 in der Richtung, in der sie auf die Knoten Vertice1 und Vertice2 zeigen.
CompareGlobal Vertices	Graph1, Vertice1, Graph2, Vertice2	Vergleicht zwei Knoten Vertice1 und Vertice2 aus zwei Karten Graph1 und Graph2 nach einer in der Variable m.CmpGblVtcsMethod gewählten Methode (CMP_GLVTYPE_MINEDGE und CMP_GLVTYPE_FULLSEARCH). Diese beschreibt, wie die Knoten verglichen werden sollen. (für Näheres siehe Kapitel 3.4.2)
CompareGlobal VerticesFullSearch	Graph1, Vertice1, Graph2, Vertice2	Wird ausgeführt, wenn CMP_GLVTYPE_-FULLSEARCH gesetzt ist. (siehe Compare-GlobalVertices)
CompareGlobal VerticesMinEdge	Graph1, Vertice1, Graph2, Vertice2	Wird ausgeführt, wenn CMP_GLVTYPE_-MINEDGE gesetzt ist. (siehe CompareGlobal-Vertices)
CompareWithAll EdgesAround	Edge1, Vertice1, Graph2, Vertice2, Matrix	Vergleicht Kante Edge1, welche auf den Knoten Vertice1 zeigt mit allen Kanten des Knotens Vertice2 aus der Karte Garph2 und allen Kanten der Nachbarknoten von Vertice2. Die Ergebnisse sind in Matrix gespeichert.
ComputeAreaAround	Graph_in, Vertice_in, Element_count _out	Berechnet die Fläche des Knotens Vertice_in der Karte Graph_in, in der die angrenzenden Kanten liegen.
ComputeDifference	Graph1, Vertice1,	Vergleicht die Kanten Vertice1 und Vertice2 der Karten Graph1 und Graph2 mit der in dem

	Graph2, Vertice2, Method_id	Parameter Method_id gewählten Methode (area_diff – Differenz der Fläche, welche durch die Kanten der Knoten belegt ist, mag_diff – Differenz der Voronoi-Level der Knoten).
ComputeGraph Configurations	MatchGraph, GlobalGraph	Sucht nach beendetem Candidate Elemination passende Stellen für die Karte MatchGraph in der Karte GlobalGraph.
ComputeMatch Possibilities	MatchVertice, MatchGraph, GlobalGraph, Method_id	Berechnet einen Vergleichswert zwischen dem Knoten MatchVertice aus der Karte MatchGraph und den Knoten aus der Karte GlobalGraph. Die Vergleichsmethode bestimmt der Parameter Method_id. Dieser kann folgendermaßen belegt werden: mag_compare (nur das Voronoi-Level bestimmt den Vergleich), area_compare (nur die Fläche, in der die Kanten des Knoten liegen, bestimmen den Vertleich) und edge_compare (nur die Kanten bestimmen den Vergleich).
ConstraintPropagation Blops	StartMatchVertice, ActualHypothesis, MatchGraph, GlobalGraph	Alle Knoten der Karte MatchGraph die noch verarbeitet werden müssen, durchlaufen einen Iterationsschritt des in Kapitel 4.3 beschriebenen Candidate Elimination.
ConstraintPropagation Wave	StartMatchVertice, ActualHypothesis, MatchGraph, GlobalGraph	Diese Funktion vollzieht den kompletten Weg des Candidate Elimination für die Knoten der Karte MatchGraph ausgehend vom Knoten StartMatchVertice.
CopyFirstLocalInto Global	LocalGraph, GlobalGraph	Initialisiert eine Karte durch die Werte der gegebenen Mittelachsen Datenstruktur LocalGraph. Liefert die initialisierte Karte GlobalGraph und den ersten Knoten.
EqualConf	Arr1, Arr2, Count	Wird in der vollständigen Suche beim Knotenvergleich eingesetzt, um nach der

		Zielstellung zu suchen.
ExtractPossible Positions	StartMatchVertice, MatchGraph, GlobalGraph, OnlyOneHypPerVert, Difference	Berechnet von allen Knoten mit nur noch einer Hypothese die mögliche Position des Roboters und liefert alle diese Ergebnisse als Liste zurück.
FillDirection	EdgeElement, From, To	Fügt dem Kantenelement EdgeElement die Richtung des nächsten Kantenelementes hinzu.
FindEdgeComing From	GlobalGraph, BindingVertice, DistinctEdge	Liefert die Kante aus der Mittelachsen Struktur DistinctEdge, welche schon verarbeitet ist. Zum Vergleich wird der Knoten BindingVertice aus der Karte GlobalGraph herangezogen.
FindFirstDistinctPoint InLocal	LocalGraph	Findet den ersten Knoten aus der gegebenen Mittelachsen Datenstruktur LocalGraph.
FirstMatchBuild Method	Global_graph	Erweitert die Karte Global_graph durch Methoden aus Kapitel 3.4.2.
FindMatchPoint InGlobal	GlobalGraph, DistinctEdge, ProbMatchPos	Liefert einen Knoten aus der Karte GlobalGraph, welcher dem Knoten DistictEdge (aus der Mittelachsen Struktur) am nächsten liegt. Wenn der Knoten DistinctEdge ein Erkundungsknoten ist, dann muss die Position ProbMatchPos gegeben sein, welche die mögliche Position des zu findenden Knoten angibt.
FindMatchVertice	GlobalGraph, LastGraph, ActualGraph, LastVerticeReturn, ActualVerticeReturn	Findet den Knoten in der Karte GlobalGraph, um die neue Umgebungskarte ActualGraph anzuschließen.
FindNearestEdge	From, To,	Liefert das am Element zwischen From und To,

InLocalRegion	ProbMatchPos	welches der Position ProbMatchPos am nächsten liegt.
GetMostRobustEdge	Graph, Vertice, Last_edge, Return_edge	Liefert die robusteste Kante des Knotens Vertice. Das heißt die Kante mit dem größten Voronoi-Level pro Kantenelement.
GetRobPosHyp	Migrid_pos, Vertice_list, Method_id	Diese Funktion wird für das Kapseln der Methode SymMap_GetRobPosHyp aus der Anwenderschnittstelle benutzt. Für eine Beschreibung siehe SymMap_GetRobPosHyp. Der Parameter Method_id entspricht der Belegung von Cm_id. (CM_0_MAG – mag_compare, CM_1_AREA – area_compare, CM_2_EDGE – edge_compare, CM_3_AREA_EDGE – area_compare und edge_compare)
GetVerticesAround	GlobalGraph, GlobalVertice, VerticesAround	Liefert alle Nachbarn des Knoten GlobalVertice der Karte GlobalGarph.
IncrementConf	Array, Count, Increment_index, Overflow	Wird in der vollständigen Suche beim Knotenvergleich eingesetzt, um das nächste Element zu generieren.
InitializeHypothesis Lists	Graph, Value	Initialisiert die Hypothesenliste der Karte Graph auf den boolschen Wert Value.
InitializeSeenVariable	Graph, Value	Initialisiert die ‚verarbeitet'-Varialbe seen der Knoten der Karte Graph auf den boolschen Wert Value.
SecondMatchBuild Method	LocalGraph, GlobalGraph, ActualRobPosition	Erweitert die Karte GlobalGraph durch Methoden aus Kapitel 3.4.3.
SumStatConf	Array, Count	Wird bei der vollständigen Suche beim

		Knotenvergleich eingesetzt, um die Kosten zu berechnen.
ToWorldCoord	Center, Value	Transformiert lokale Migrid-Map Koordinaten in Weltkoordinaten.

6.5 Hilfsfunktionen

Die folgenden Funktionen bieten die Möglichkeit mit den aufgenommenen Karten zu arbeiten. Dazu gehört sowohl das Speichern und das Löschen einer Karte als auch die graphische Veranschaulichung dieser, wenn die Gelegenheit dazu besteht. Sie werden für den eigentlich Kartenaufbau und den –vergleich nicht benötigt, sind aber nötiges Handwerk für den schnellen Umgang mit den Karten.

Funktionsname	Parameterliste	Beschreibung
CalculateGraph StatisticMatrix	Graph	Vergleicht alle Knoten der Karte Graph miteinander und schreibt sie als Matrix in die Datei „graph_stat.sta“.
CopyFromVertice	Graph1, Vertice1, ReturnVertice1, ReturnEdge1, Graph2, Vertice2	Kopiert die Karte Graph1 von dem Knoten Vertice1 an den Knoten Vertice2 der Karte Graph2. Alle Informationen nach dem Knoten Vertice2 gehen verloren. Die Topologie der Karte Graph1 vom Knoten Vertice1 wird in die Karte Graph2 kopiert. Hypothesenlisten werden nicht mit kopiert.
CopyGlobalGraph	Graph1, Graph2	Kopiert die Karte der Struktur Graph1 in die Struktur Graph2.
CountFromVertice	Vertice_in	Zählt alle in der Baumstruktur nachfolgenden Knoten vom Knoten Vertice_in.
DeleteFromVertice	GlobalGraph, Vertice	Löscht die Karte GlobalGraph ausgehend von dem Knoten Vertice einschließlich diesem.
DeleteGlobalGraph	GlobalGraph	Löscht die Karte GlobalGraph.
GetEdge	From, To	Liefert die Kante zwischen den Knoten From und To.
GetEdgeByID	GlobalGraph,	Liefert die Kante aus der Karte GlobalGraph mit

	Id	der Nummer id.
GetPrevVertice	GlobalGraph, Vertice	Liefert den Vorgängerknoten des KnotensVertice aus der Karte GlobalGraph.
GetVerticeByID	GlobalGraph, Id	Liefert den Knoten aus der Karte GlobalGraph mit der Nummer Id.
MoveGlobalGraph	Graph1, Graph2	Verschiebt die Karte aus der Struktur Graph1 in die Struktur Graph2.
PopElement	Element	Holt ein Strukturelement vom Stack.
PrintEdgeElements	GlobalGraph, RT_SimLayer	Wenn graphischer Modus möglich ist, werden alle Kantenelemente der Karte GlobalGraph auf den RT_SimLayer ausgegeben.
PushElement	Element	Packt ein Strukturelement auf den Stack.
ReadGlobalGraph	Filename, GlobalGraph	Liest eine Karte aus der Datei Filename in die Struktur GlobalGraph.
ShowFreeSpace	GlobalGraph, RT_SimLayer	Wenn graphischer Modus möglich ist, wird die Freiformfläche der Karte GlobalGraph auf den RT_SimLayer ausgegeben.
ShowGlobalGraph	GlobalGraph, RT_SimLayer	Wenn graphischer Modus möglich ist, wird die komplette Topologie der Karte GlobalGraph auf den RT_SimLayer ausgegeben.
WriteGlobalGraph	Filename, GlobalGraph	Speichert die Karte GlobalGraph in die Datei Filename.

6.6 Sonstige Funktionen

Diese Sammlung von Funktionen dient zum einen der Ermittlung von Parametern für die Umgebungen, in denen dieses Verfahren eingesetzt werden soll. Zum anderen stellt es einen Debug Modus zur Verfügung, der es ermöglicht, die Aufbau- und Vergleichsfunktionen schrittweise durchführen zu lassen.

Funktionsname	Parameterliste	Beschreibung
CalculateEdgeStatistic	Keine Param.	Gleiche Berechnung wie in WriteEdgeStatistics, nur dass der Prozess hier halbautomatisch

		abläuft. Alle Kanten, deren Vergleichswert weit unter einer Schwelle liegt, werden als ‚gleich' betrachtet, weit darüber liegende als ‚ungleich'. Bei Unsicherheit über die Beziehung der Kanten, wird die Benutzereingabe verlangt. Die Ergebnisse werden in den Dateien „match.txt" und „miss_match.txt" gespeichert.
CalculateVertice Statistic	Keine Param.	Gleiches Verhalten wie bei CalculateEdgeStatistic, nur für Knoten. (für Näheres siehe CalculateEdgeStatistics)
OneDebugCycle	Keine Param.	Muss aufgerufen werden, um jegliche Form der Statistik Funktionen ausführen zu können.
mousehan_Choose Vertice	Type, X, Y	Über das SimWindow ist ein Knoten per Doppelklick mit der Maus wählbar. Dieser wird mit sich selbst verglichen. Die Parameter sind Rückgabewerte (X und Y sind in Weltkoordinaten, Type ein interner Parameter).
mousehan_Choose Vertices	Type, X, Y	Über das SimWindow sind zwei Knoten per Doppelklick mit der Maus wählbar, welche miteinander verglichen werden. Die Parameter sind Rückgabewerte (X und Y sind in Weltkoordinaten, Type ein interner Parameter).
mousehan_Compare Vertices	Type, X, Y	Über das SimWindow kann ein Knoten per Doppelklick mit der Maus gewählt werden, welcher mit allen anderen Knoten verglichen wird. Die Parameter sind Rückgabewerte (X und Y sind in Weltkoordinaten, Type ein interner Parameter).
WriteEdgeStatistic	Keine Param.	Berechnet die Vergleichswerte von zwei gewählten Kanten aus zwei Karten. Die Kanten werden durch Tastatureingabe gewählt, wenn der Debugmodus für das Vergleichen von Kanten gewählt wurde. Die gewählten Kanten sollten

		möglichst gleich sein, da nach automatischer Fortsetzung alle anderen Kanten der zwei Karten verglichen werden und ihre Vergleichswerte in die Kategorie ‚ungleich' eingeordnet werden. Die gleichen Werte werden in der Datei „match.txt", die ungleichen in der Datei „miss_match.txt" gespeichert.
WriteVerticeStatistic	Keine Param.	Gleiches Verhalten wie bei WriteEdgeStatistics, nur für Knoten. (für Näheres siehe WriteEdgeStatistics)

7 Zusammenfassung und Ausblick

Der vorangegangene Text gab einen Einblick in die Entwicklung eines Moduls für die Re-/Lokalisierung zur Navigationsunterstützung des Roboters Hefter ST82 R. Die Entwicklung begann unter der Annahme, dass Mittelachsen als Grundlage für den Aufbau einer globalen Karte verwendet werden können und in einer entsprechenden Datenstruktur für eine Wiedererkennung von Teilen dieser Karte geeignet sind. Als Ergebnis konnte der Kartenaufbau bestätigt und die Machbarkeit der Re-/Lokalisierung über die Mittelachsen nachgewiesen werden.

Zur Visualisierung der Ergebnisse können die Karten in den SINAS Simulator geladen und angezeigt werden. Während der Entwicklung wurde darauf geachtet, dass die einzelnen Funktionen des Moduls einfach erweitert oder durch andere Algorithmen ersetzt werden können. Daher sollte diese Arbeit auch als Referenz für künftige Projekte nutzbar sein. So ist es möglich das Kreisschlussproblem mit in das SymMap Modul einzubinden oder es durch eine Markov Lokalisierung zu erweitern.

Im Einzelnen entstand in dieser Arbeit eine hierarchische Datenstruktur für die Repräsentation der Karten auf Basis von Mittelachsen. Des Weiteren wurden Algorithmen für den Aufbau dieser Karten mit Schwerpunkt auf der Fusionierung einzelner Teilkarten entwickelt, sowie Algorithmen für den Vergleich dieser Karten basierend auf dem von Mitchell 1978 entwickeltem Candidate Elimination Algorithmus.

8 Quellen

[1] Borenstein J., Everett H. R., Feng L. *„Where am I“ Sensors and Methods for Mobile Robot Positioning*, Dept. Of Mechanical Engineering and Applied Mechanics, University of Michigan

[2] Delingette H.,. Hebert M, Ikeuchi. K., *Deformable surfaces: A free-form shape representation.*, Proc. SPIE Vol 1570: Geometric Methods in Computer Vision, pages 21-30, 1991

[3] Blum HJ. *A transformation for extracting new descriptors of shape*. In: Wathen-Dunn W, editor. Symposium on Models for the Perception of Speech and Visual Form. Cambridge, MA: MIT Press, 1967b

[4] Klein P. N., Dept. Of Computer Science, Brown University, Providence, research supported by NSF Grant CCR-9700146; Sebastian T. B., Div. Of Engineering, Brown University, Providence; Kimia B. B., Div. Of Engineering, Brown University, Providence, *Shape matching using edit-distance: an implementation*

[5] Larrosa J., Valiente G. *Graph Pattern Matching using Constraint Satisfaction*. Department of Software, Technical University of Catalonia, Barcelona, Spain

[6] Aurenhammer F., *Voronoi diagrams – A survey of a fundamental geometric structure.*, ACM Comput. Surv., vol.23, pp. 345-405, 1991

[7] Jungnickel D., *Graphen, Netzwerke und Algorithmen*, 3. Aufl., BI-Wiss.-Verl., 1994

[8] Mitchell T. M., *Version spaces: An approach to concept learning.* Doctoral dissertation, Department of Electrical Engineering, Stanford University, Palo Alto, CA, 1978

Anhang – Kommandos für die SINAS Simulationsumgebung

Hier finden sich alle mit dem Kartenaufbau und –vergleich in Zusammenhang stehenden Kommandos für die SINAS Simulationsumgebung.

Kommando	Beschreibung
y g a <wert>	Setzt die Pausezeit in msec. Wenn der Wert 0 ist, dann ist die Pause ausgeschaltet.
y g b <1/0>	Globale Karte wird gebaut <start/stopp>.
y g c v <1/2/...>	Setzt die Knotenvergleichsmethodik fest auf: 1 – vollständige Suche 2 – die Heuristik der kleinsten Kantendistanzen
y g d	Löscht die globale Karte.
y g e	Löscht die Vergleichskarte.
y g f <1/0>	Zeigt die Freifläche (Positivbild) der Umgebung <an/nicht an>.
y g g c <id1> <id2> <id3> <id4>	Wählt je zwei benachbarte Knoten aus der letzten (id1, id2) und der aktuellen (id3, id4) Umgebungskarte, um die dazwischen liegenden Kanten zu vergleichen. Achtung: Das ist erst nach Aktivierung des Debug Modus mit „y g g d 1“ möglich.
y g g d <1/0>	Schaltet den Debug Modus <an/aus>.
y g g e <1/0>	Wenn 1, dann werden zwei als gleich befundene Kanten angezeigt, bei denen der Benutzer mit „y g g r <1/0>“ der Gleichheit <zustimmen/ablehnen> muss. Achtung: Das ist erst nach Aktivierung des Debug Modus mit „y g g d 1“ möglich.
y g g m <mouse handler funktion>	Wählt einen Mousehandler, der während des Debug Modus benutzt werden kann. Folgende Möglichkeiten stehen dem Benutzer offen: 0 – deaktiviert alle Mousehandler. 1 – ein Knoten mit der Maus per Doppelklick wählbar, der dann mit sich selbst verglichen wird. 2 – zwei verschiedene Knoten – einer per Doppelklick, der

	Zweite per CTRL und Doppelklick – wählbar, die dann miteinander verglichen werden. 3 – ein Knoten per Doppelklick wählbar, der mit allen anderen Knoten verglichen wird. Das Ergebnis ist anhand von unterschiedlich schwarz markierten Knoten sichtbar – je dunkler die Markierung desto besser passen die Knoten zusammen.
y g g n <id1> <id2>	Wählt einen Knoten der aktuellen (id1) und der letzten Umgebungskarte (id2), um sie miteinander zu vergleichen. Achtung: Das ist erst nach Aktivierung des Debug Modus mit „y g g d 1“ möglich.
y g g p <parameter> <wert>	Verändert die Parameter in der Struktur <m.param>. Der Parameter wird mit der Nummer <parameter> bestimmt und auf den neuen Wert <wert> gesetzt.
y g g r <1/0>	<Bestätigung/Ablehnung> eines Ergebnisses; beispielsweise bei der Kantengleichheitsbestimmung.
y g g v <1/0>	Wenn 1, dann werden zwei als gleich befundene Knoten angezeigt, bei denen der Benutzer mit „y g g r <1/0>“ der Gleichheit <zustimmen/ablehnen> muss. Achtung: Das ist erst nach Aktivierung des Debug Modus mit „y g g d 1“ möglich.
y g i <0/1/2/...>	Zeigt verschiedene Knotennummern nach Benutzerwunsch an: 0 – keine Nummern. 1 – normale (alle) Nummer (standard). 2 – Nummer ist der Platz in der globalen Knotenliste der Karte für alle Knoten. 3 – Nummer ist der Platz in der globalen Knotenliste der Karte nur für Kreuzungsknoten. 4 – Nummer ist der Platz in der globalen Knotenliste der Karte nur für Kreuzungs- und Endknoten.
y g l <1/0>	<Startet/Stopt> den Bau einer Vergleichskarte.
y g m <1/0>	<Started/Stopt> den Kartenvergleich.

y g p <1/0>	Schaltet die Anzeige der Kantenelemente <an/aus>.
y g r <name>	Liest eine globale Karte aus der Datei „name“ ein.
y g s <1/0>	Schaltet die Anzeige der Karte <an/aus>.
y g t <1/0>	Schaltet die Anzeige der Vergleichskarte <an/aus>.
y g w <name>	Schreibt die globale Karte in die Datei „name“.
y g x	Berechnet die Statistikmatrix für die Knoten der globalen Karte und schreibt sie in die Datei „graph_stat.sta“.