GMD-Studien Nr. 293

Hartmut Surmann, Jörg Huser,
Jens Wehking

Topologische Karten für fuzzy-gesteuerte mobile Roboter I

Juni 1996

GMD-FORSCHUNGSZENTRUM
INFORMATIONSTECHNIK GMBH

Anschrift der Verfasser / Address of the authors:

Hartmut Surmann
Jörg Huser
Jens Wehking
Institut für Systementwurfstechnik
GMD — Forschungszentrum
Informationstechnik GmbH
D-53754 Sankt Augustin

Die Deutsche Bibliothek - CIP-Einheitsaufnahme

Surmann, Hartmut:
Topologische Karten für fuzzy-gesteuerte mobile Roboter I / Hartmut Surmann ; Jörg Huser ; Jens Wehking. GMD-Forschungszentrum Informationstechnik GmbH. [Verantw. für diesen Bd.: Liliane Peters]. - Sankt Augustin : GMD-Forschungszentrum Informationstechnik. 1996
(GMD-Studien ; Nr. 293)
ISBN 3-88457-293-8
NE: Huser, Jörg:; Wehking, Jens:; GMD-Forschungszentrum Informationstechnik <Sankt Augustin>: GMD-Studien

ISSN 0170-8120
ISBN 3-88457-293-8

GMD — Forschungszentrum
Informationstechnik GmbH

D-53754 Sankt Augustin

Telefon (02241)14-0
Telex 8 89 469 gmd d
Telefax (02241) 142618
http://www.gmd.de

Abstract

Autonome mobile Roboter gewinnen im Dienstleistungsbereich zunehmend an Bedeutung. Entscheidend für den Einsatz in Indoor-Umgebungen (Büro, Krankenhaus) ist, daß solche Systeme preiswert und echtzeitfähig sind. Im Rahmen dieser Arbeit wird ein Planungs- und Kartierungsmodul entwickelt, das mittels linguistischer Kommandos eine globale Steuerung des autonomen mobilen Roboters MORIA ermöglicht.

Wesentliche Bestandteile des Planers sind die Überführung topologischer Informationen wie erkannte Abzweige oder Sackgassen in eine Graphenstruktur während der Kartierung und die Generierung von abzuarbeitenden Kommandolisten, die den Roboter zu seiner Zielposition führen. Ein weiterer wichtiger Bestandteil ist die Aktualisierung einer einmal erstellten Umgebungskarte, falls z.B. neue Abzweige (Transportwege) auftreten oder wichtige Transportwege auf Dauer versperrt sind. Außerdem wird die Güte eines generierten Weges durch eine anschließende Wegoptimierung verbessert, um u.a. die Anzahl zu befahrender Abzweige zu reduzieren.

Diese Arbeit basiert auf einem fahrerlosen Transportfahrzeug, ausgestattet mit grundlegender interner und externer Sensorik sowie einem auf Fuzzyregeln basierenden Navigators. Die Implementierung der Steuerung auf dem mobilen Roboter erfolgt mittels Low-cost-Hardware (PC 486/33MHz, digitale I/O-Karte).

Anwendungsfälle des vorgestellten Systems finden sich in den folgenden Bereichen denkbar:

- Übernahme von Transportaufgaben in Werkhallen oder Bürogebäuden (z.B. Postverteilung o.ä.)
- Kranken- und Behindertenfahrstühle
- Reinigungsarbeiten in Gebäuden außerhalb der normalen Arbeitszeiten

Schlagworte: Topologische Karten, Wegeplanung, autonome mobile Transportsysteme, Fuzzy Navigator, Fuzzy Zustände, Kartengenerierung, Graphensuche, FTF, Positionsbestimmung, Orientierungsfahrt

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Fledermäuse orientieren sich in einer völlig abgedunkelten Umgebung anhand von ausgesendeten Ultraschallsignalen, die eine Frequenz von 30 - 120 KHz haben. Dieses Echolot ermöglicht ihnen sowohl die Kollisionsvermeidung mit auftretenden Hindernissen als auch die Ortung ihrer Beute. Es wurde sogar schon beobachtet, daß eine Große Hasenmaul-Fledermaus[1] einen Fisch durch seine Rückenflosse oder anhand von auftretenden Oberflächenwellen orten und ergreifen konnte [1]. Diesem lokalen Orientierungsvermögen innerhalb des gerade sichtbaren Bereiches ist ein globales überlagert, durch das die Fledermaus z.B. ihren Weg zurück zu ihrer Behausung findet.

Ziel der vorliegenden Arbeit ist es, dieses globale Orientierungsvermögen für einen autonomen mobilen Transportroboter zu realisieren. Dabei wird eine topologische Repräsentation der Roboterumgebung verwendet, die durch eine Orientierungsfahrt generiert werden kann. Weiterhin muß der Roboter MORIA[2] dazu in der Lage sein, sich in dieser Umgebung zu bewegen und eventuell auftretende Änderungen der Umgebung gegenüber seiner internen Karte zu registrieren.

Grundlage für das Planungsmodul ist eine lokale Steuerungskomponente in Form eines Navigators, der auf einer Fuzzysteuerung[3] basiert [2] [3] [4] [5] [6, 7, 8, 9, 10, 11, 12] [13]. Die lokale Fuzzysteuerungskomponente geht dabei von einer strukturellen Beschreibung der Umgebung aus. Die bisher eingesetzten autonomen Innenraumfahrzeuge basieren im Gegensatz dazu auf einer genauen geometrischen Beschreibung der Roboterumgebung und/oder gesetzten Landmarken in Verbindung mit einem Koppelnavigationssystem [14] [15]. Die Trajektorieberechnung und die Überwachung der Befahrung eines berechneten Weges ist bei einer solchen Verfahrensweise jedoch sehr zeitaufwendig und speicherintensiv [16] [17] [18] [19] [20]. Desweiteren sind solche Systeme auf die Kenntnis der genauen Position und Orientierung innerhalb der zu befahrenden Umgebung angeweisen. Durch auftretende Fehler bei der Wegmessung kommt es dabei zu Inkonsistenzen zwischen berechneter und tatsächlicher Position dieser hochgenauen Koppelna-

[1] lat. Noctilio leporinus

[2] Der Name ist aus dem Buch „Herr der Ringe (The Lord of the Rings)“ von John Ronald Reuel Tolkien abgeleitet. MORIA ist ein Labyrinth in einem Gebirge, das von Zwergen zum Mythrilabbau geschaffen wurde. Später wird MORIA von bösen Kräften wie „Orcs“ und „Balrogs“ besetzt, ähnlich wie Hindernisse in einer dynamischen Umgebung. Die Zwerge werden vertrieben und die guten Gefährten müssen das Gebirge überqueren oder den Weg durch MORIA finden. In dieser Arbeit ist MORIA kein Labyrinth, sondern ein Fahrzeug, das sich selbständig in einer unbekannten Umgebung orientieren kann.

[3] fuzzy - fusselig, unscharf

vigationssysteme. Aus diesem Grund müssen bei den meisten dieser Fahrzeuge Referenzpunkte (Landmarken) innerhalb ihrer Umgebung plaziert werden, um einen genauen Positionsabgleich zu ermöglichen. Einfache Systeme beinhalten eine Spurgebundenheit, d.h. es werden feste Routen innerhalb der Umgebung des Roboters definiert und diese durch Verlegung von Spursystemen innerhalb des Bodens fixiert. Solche Systeme sind nicht autonom. Die Installation der benötigten Spursysteme ist naturgemäß sehr kostenaufwendig. Im Gegensatz dazu soll MORIA dazu in der Lage sein, sich vollständig autonom in seiner Umgebung zu bewegen.

Da die Simulationsumgebung inklusive Navigator sowohl auf einem PC (ab 386er aufwärts) als auch auf Unix-Workstations vom Typ SUN-SPARC lauffähig ist und es sich bei dem Steuerungsrechner des Roboters ebenfalls um einen PC mit INTEL-Prozessor (486 DX/2) handelt, muß das implementierte Planungsmodul auf allen genannten Systemen lauffähig sein. Die Implementierung der notwendigen Module erfolgt in der Programmiersprache C, wie sie durch den Entwurf definiert wird, der am 31. Oktober 1988 dem ANSI eingereicht wurde. Dieser Entwurf wurde als ”American National Standard for Information Systems - Programming Language C, X3.159-1989” genehmigt [21]. Außerdem wird auch der K&R-Standard unterstützt.

2 MORIA

Der autonome mobile Roboter MORIA (Bild 2.1) gehört zur Klasse der fahrerlosen Transportfahrzeuge (FTF), das mit Energieversorgung, Antrieb, Bremsen, Sicherheitstechnik und einer internen Sensorik ausgestattet ist. Ein FTF ist ein Fahrzeug, daß mit Hilfe von Sensoren einen vorgegebenen Weg findet. Weiterhin spricht man von einem autonomen mobilen Roboter, wenn ein FTF zusätzlich mit einem mehrachsigen, freiprogrammierbaren Roboterarm ausgestattet ist. Da ein solcher Roboterarm ebenfalls für das vorhandene System vorgesehen ist, wird dieses im folgenden als autonomer mobiler Roboter MORIA bezeichnet.

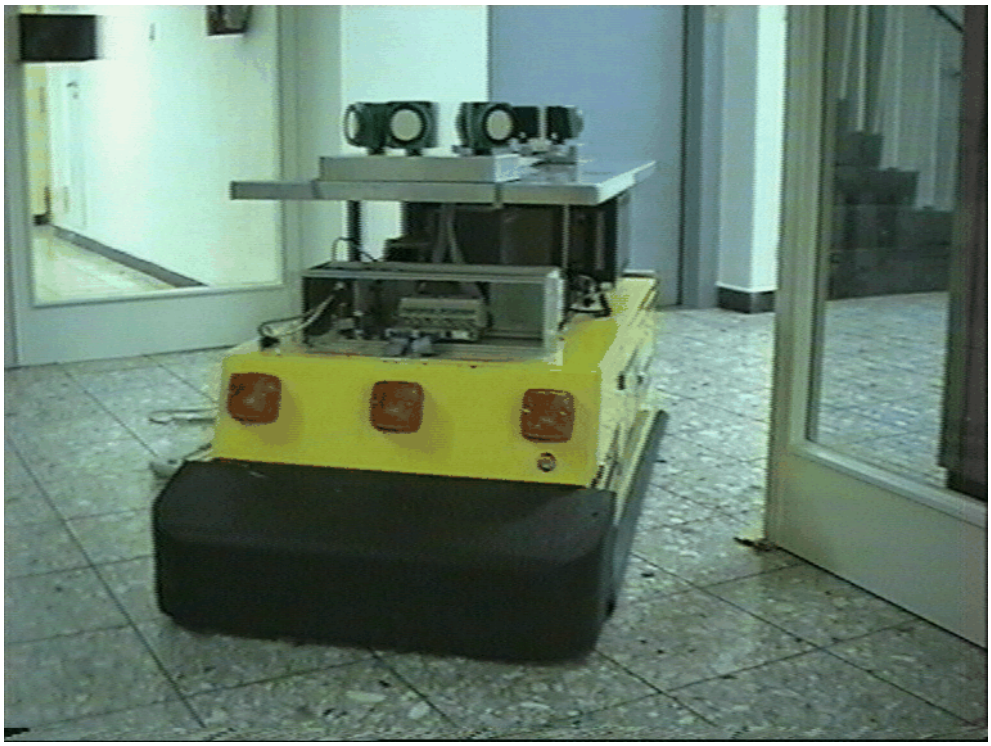


Bild 2.1: Der autonome mobile Roboter MORIA

Bild 2.2 zeigt das Komplettsystem, so wie es sich in seiner voraussichtlich endgültigen Form darstellt. Die Sensorik des Roboters besteht dabei aus 8 Ultraschallsensoren [4]. Bild 2.3 stellt die Lage und Orientierung der betreffenden Sensoren dar. Desweiteren ist eine lokale Fuzzysteuerung vorhanden [4]. Diese lokale Fuzzysteuerung, im folgenden als Navigator bezeichnet, dient zur Bewegungssteuerung des Roboters in seiner lokalen sichtbaren Umgebung. Der Navi-

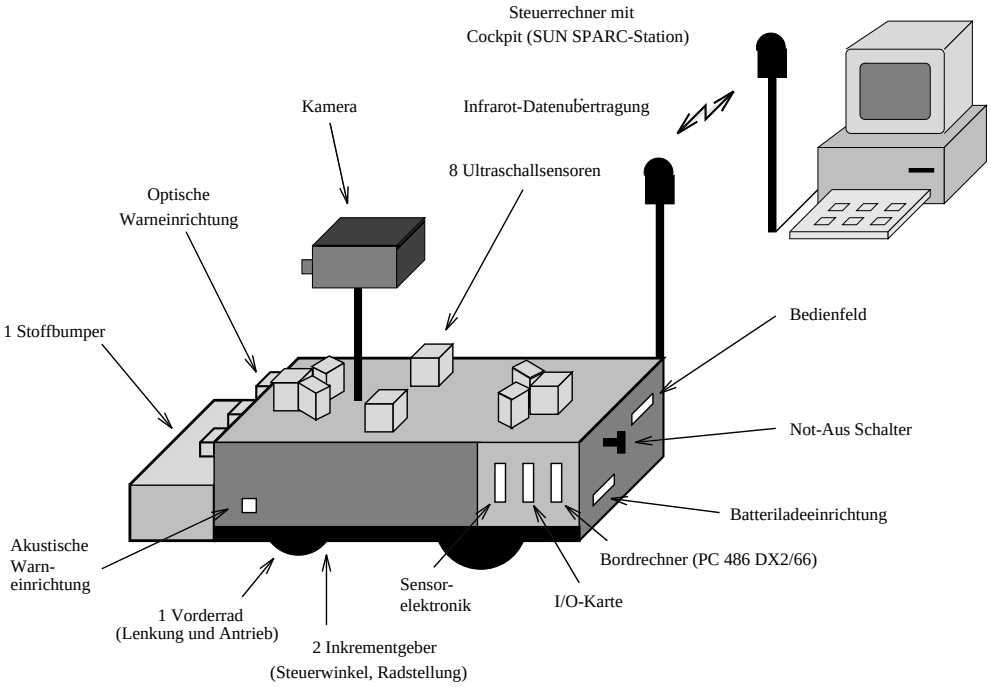


Bild 2.2: Das Komplettsystem

gator übernimmt dabei folgende Aufgaben.

- Kollisionsvermeidung in Echtzeit
- Zielorientiertes Verhalten. Der Navigator ermöglicht dem Planungsmodul, globale Kommandos wie “fahre geradeaus“ oder “fahre nächsten Abzweig links“ zu setzen und mit diesen eine Bewegungssteuerung des Roboters auf einer hohen Abstraktionsebene durchzuführen.
- Erkennung komplexer Umgebungsstrukturen. Aufgrund der gewonnenen Informationen wird eine Abstraktion der betreffenden Strukturen durchgeführt.
- Reaktion auf verschiedene Fahrsituationen

Um die korrekte Ausführung eines abzuarbeitenden Kommandos zu überprüfen, benötigt das Planungsmodul jedoch weitere Informationen. Diese Informationen bestehen zum einen aus den Zustandsvariablen des Fuzzy-Controllers und zum anderen aus weiteren durch den Navigator generierten Werten wie z.B. der zurückgelegten Strecke seit dem letzten erkannten Abzweig oder der aktuellen Orientierung des Roboters in seiner Umgebung. Bild 2.1 zeigt die für das Planungsmodul relevanten Daten.

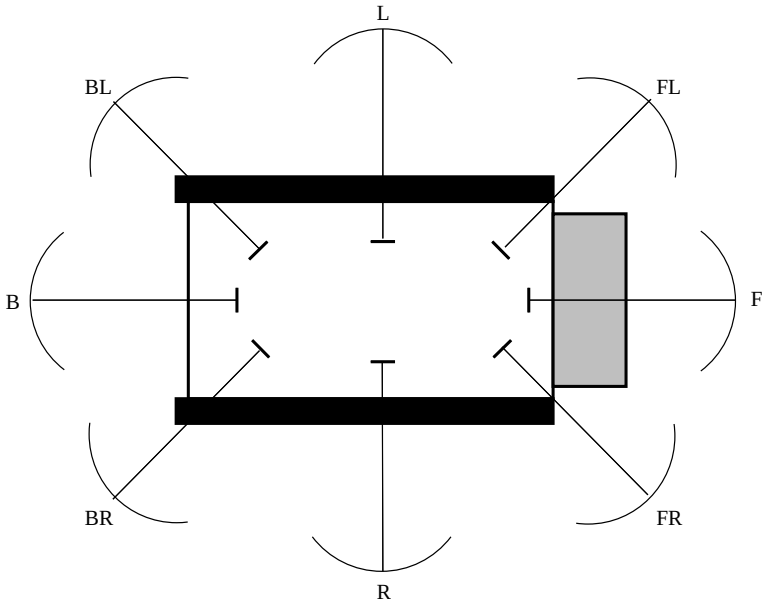


Bild 2.3: Lage und Bezeichnung der Ultraschallsensoren

Diese Daten werden sowohl von dem für das Fahren in bekannter Umgebung als auch von dem für die Kartierung zuständigen Modul unabhängig voneinander verwendet.

Die Abzweigerkennung erfolgt über die Ableitung der gemessenen Abstandswerte bezüglich der beiden vorderen, in einem Winkel von 45 Grad zur Fahrtrichtung liegenden Ultraschallsensoren. Tritt eine Abstandsänderung von über einem Meter auf, so ist die Erkennung eines entsprechenden Abzweigs erfolgt. Auftretende Meßfehler werden durch Mehrfachmessungen weitestgehend erkannt und bei Beibehaltung der alten Meßwerte ignoriert. Die Aufgabenbereiche der verschiedenen Sensoren sind in Tabelle 2.2 dargestellt.

Zur Kollisionsvermeidung werden die drei vorderen Sensoren F, FL und FR benötigt. Der Sensor F bestimmt im wesentlichen die Geschwindigkeit. Die seitlichen Sensoren L und R werden für ein verbessertes Abbiegeverhalten und zur Berücksichtigung des Ausschwenkens und des Überschleifens verwendet. Eine Änderung der Fahrtrichtung hat eine entsprechende Vertauschung der Sensorbezeichnungen zur Folge (F wird zu B, FL wird zu BR usw.). Die Abzweigerkennung soll zukünftig erweitert werden, indem die vorderen Sensoren FL und FR nur einen möglichen Abzweig erkennen. Dieser wird dann bestätigt, wenn die entsprechenden Sensoren L bzw. R diesen möglichen Abzweig während der weiteren Fahrt bestätigen. Die Geschwindigkeitsmessung erfolgt durch einen induktiven Tachogenerator. Aus den gemessenen Werten wird u.a. die Position, die aktuelle Orientierung des Roboters und der zurückgelegte Weg ermittelt.

Tabelle 2.1: Relvante Daten für den Planer

Datum	Beschreibung	Zustandsvariablen
ROT	Rotation	-
SMSV	Umschaltg. Mot.- u. Sensorwerte	Vorwärts, rückwärts
LBR	Linksabzweig	erkannt, nicht erkannt
RBR	Rechtsabzweig	erkannt, nicht erkannt
TURN	Abbiegen	geradeaus, rechts, links
V	Aktuelle Geschwindigkeit	[-1.0,1.0] m/sec
ANGLE	Steuerwinkel	[-75,75] Grad
WIDTH	Aktuelle Korridorbreite	-
LENGTH	Aktuelle Korridorlänge	-
ORI	Aktuelle Orientierung, Kompaß	-
DIR	Aktuelle Fahrtrichtung	Vorwärts, rückwärts
COMMAND	Aktuelles gesetztes Kommando	siehe Tabelle 4.1
VMAX	Maximalwert für Geschwindigkt.	0.0 - 1.0 m/sec
VOVR	Aktuelle Beschleunigung	-
ANMAX	Maximale Steuerwinkeländerung	-
X-KOORD	X-Koordinate des Standortes	-
Y-KOORD	Y-Koordinate des Standortes	-

2.1 Die Simulationsumgebung

Die Testphase einer Steuerung für ein physikalisches System kann u.U. sehr zeitaufwendig sein. Allein der Platzbedarf für eine vom Roboter zu befahrende Testumgebung stellt sicherlich in den meisten Fällen ein unlösbares Problem dar. Weiterhin muß zur Überprüfung der Steuerung eine Vielzahl von Fahrversuchen durchgeführt werden, die ebenfalls einen hohen Zeitaufwand erfordern. Die Überprüfung der anfallenden Daten und das Nachvollziehen einer Testfahrt anhand von u.a. Orientierung, zurückgelegter Strecke und Geschwindigkeit ist auf einem physikalischen System ebenfalls sehr zeitaufwendig und damit kostenintensiv. Daher wurde für die Entwicklungsphase die Simulationsumgebung RoboVis [22] entwickelt, die Komponenten zur Interaktion, Simulation und Visualisierung enthält. Bild 2.4 zeigt das Cockpit, das als Benutzerschnittstelle der Simulationsumgebung dient.

Am unteren Rand des Bildes sind die verschiedenen Möglichkeiten des Benutzers abgebildet, um in den Ablauf der Simulation einzugreifen. Diese umfassen die Komponenten der Kommandosetzung (Handsteuerung und Automatikbetrieb) und globale Steuerungsparamter wie Start- und Stopkommando oder Setzen des Zielpunktes der aktuellen Fahrt. Am linken Rand sind die Steuerungsparame-

Tabelle 2.2: Aufgabenbereiche der Sensoren

	F	FL	L	BL	B	BR	R	FR
Geschwindigkeitsanpassung	X							
Kollisionsvermeidung	X	X						X
Orientierung an rechter Wand								X
Wandabstand		X						X
Abzweigerkennung		X						X
Sackgassenerkennung	X	X						X
Abbiegeverhalten	X	X	X	X		X	X	X
Ausschwenkreduktion				X		X		
Überschleifreduktion			X	X		X	X	
Bez. bei Rückwärtsfahrt	B	BR	R	FR	F	FL	L	BL

ter wie maximale Geschwindigkeit und Beschleunigung beeinflußbar. Am oberen Bildrand sind die wichtigsten Zustandvariablen und aktuellen Meßwerte für Geschwindigkeit, gefahrene Strecke, Orientierung und weitere Parameter erkennbar. Das aktuelle Kommando ist in der unteren rechten Ecke sichtbar.

Die Simulationsumgebung war im Rahmen dieser Arbeit ein wichtiges Hilfsmittel, um die Funktionalität des Planers zu überprüfen. Eine der grundlegenden Funktionen des Planungsmoduls ist die Wegsuche innerhalb einer bekannten Umgebung (siehe Kapitel 4). Das folgende Kapitel soll einen Überblick häufig verwendeter Algorithmen zur Wegsuche vermitteln.

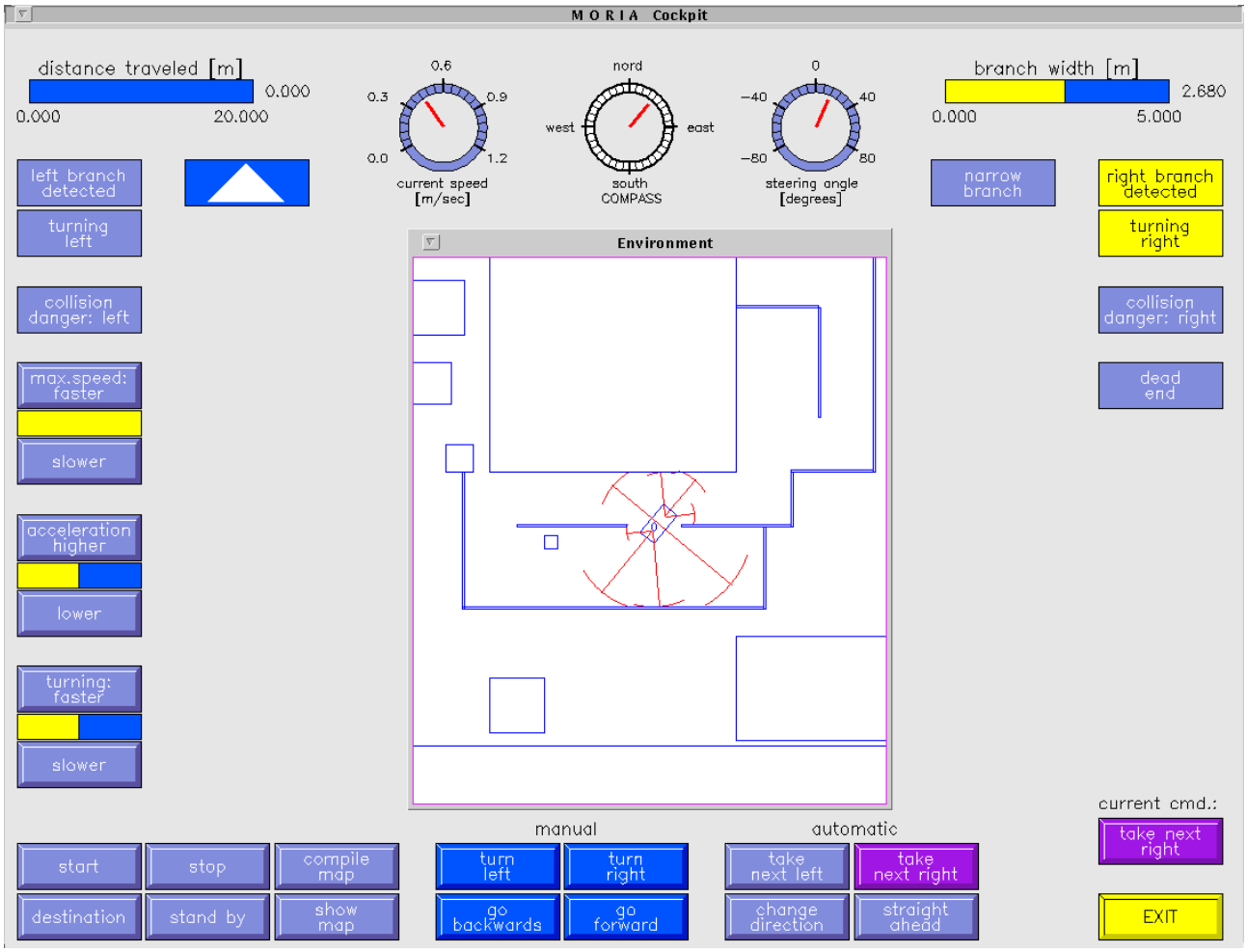


Bild 2.4: Das Cockpit mit Umgebung

3 Suchalgorithmen

In der Graphentheorie existieren viele verschiedene Suchverfahren für die Erkundung von beliebigen Graphen [23, 24, 25, 26]. Bezüglich einer Wegsuche seien hier nur Methoden zur Auffindung von Eulerkreisen oder Hamiltonlinien erwähnt. Eine tiefergehende Erläuterung der verschiedenen Erkundungsstrategien sprengt den Rahmen dieser Arbeit, daher wird an dieser Stelle nur auf die beiden wichtigsten Algorithmengruppen zur Suche von Wegen bzw. schlingenfreien Wegen in ungerichteten bzw. gerichteten Plänen eingegangen. Im folgenden sind zunächst die erforderlichen grundlegenden Definitionen aufgeführt, auf die in den folgenden Kapiteln zurückgegriffen wird.

Definition 1 (Plan).
Ein Plan ist ein 2-Tupel $P = (K, G)$ bestehend aus einer nichtleeren Menge K und einer Menge $G \subseteq K \times K$. Die Elemente der Menge K werden als Kreuzungen und die Elemente G als **G**änge bezeichnet. Für zwei Kreuzungen $x,y \in K$ wird der Gang $g \in G$ von x nach y durch $\overrightarrow{xy}$ oder (x,y) bezeichnet.

Definition 2 (ungerichteter Gang).
Ist für $x,y \in K$ sowohl $\overrightarrow{xy} \in G$ als auch $\overrightarrow{yx} \in G$, so heißt der Gang von x nach y ungerichteter Gang und wird durch $\overleftrightarrow{xy}$ bezeichnet.

Definition 3 (ungerichteter Plan).
Der Plan P heißt ungerichteter Plan, falls alle Gänge G des Plans ungerichtet sind.

Die Menge $G \subseteq K \times K$ entspricht einer Relation G in der nichtleeren Menge K. Im ungerichteten Fall ist G symmetrisch und irreflexive. Diese Relation bezeichnet man dann als *Nachbarschaft*.

Definition 4 (benachbarte Kreuzung, benachbarte Gänge).
Seien $a,b,c,d \in K$ Kreuzungen, dann heißt

1.) die Kreuzung b benachbart[1] zur Kreuzung a, wenn es einen Gang $\overrightarrow{ab} \in G$ gibt.

2.) der Gang $g_i = \overrightarrow{ab}$ benachbart zu $g_j = \overrightarrow{c,d}$, wenn $b = c$ ist.

[1] adjazent

Ist $\vec{ab} \in G$, aber $\vec{ba} \notin G$, dann wird $\vec{ab} \in G$ als Einbahnstraße bezeichnet. In einer Einbahnstraße ist nur b zu a benachbart (erreichbar), aber nicht umgekehrt.

In der strukturellen Repräsentation der Karte wirkt sich die Deklaration von Einbahnstraßen in der Weise aus, daß alle der zugelassenen Fahrtrichtung entgegen abzweigenden Gänge aus der Datenstruktur des betreffenden Ganges entfernt werden. Ob eine Einbahnstraße entgegen der zugelassenen Fahrtrichtung bei der Weggenerierung durchfahren werden muß, wird mittels einer Überprüfung der zurückliegenden Abzweige durchgeführt. Dazu wird in jedem Rekursionsschritt nach Auswahl des nächsten zu fahrenden Ganges überprüft, ob der Gang, von dem aus abgezweigt wurde ebenfalls in entgegengesetzter Fahrtrichtung erreichbar ist. Ist dies nicht der Fall, so wurde die definierte Einbahnstraße in nicht zulässiger Fahrtrichtung durchquert. Andernfalls ist der gewählte Abzweig korrekt und wird in den betreffenden Weg aufgenommen.

Definition 5 (gerichteter Plan).
Der Plan P heißt gerichteter Plan, falls mindistens ein Gang des Plans gerichtet ist.

Definition 6 (endlicher Plan).
Ein Plan heißt endlicher Plan, wenn K und damit auch G endlich sind.

Definition 7 (Grad, Valenz).

1.) Sei $P = (K,G)$ ein ungerichteter Plan und $x \in K$. Dann heißt
$d(x) := |\{ \vec{\vec{xy}} \mid \vec{\vec{xy}} \in G \wedge x \in \vec{\vec{xy}}\}|$
der Grad (oder die Valenz) des Kreuzungspunktes x. Im Falle $d(x) = 0$ heißt x isolierte Kreuzung und im Falle $d(x) = 1$ der Gang $\vec{\vec{xy}} \in G$ Sackgasse.

2.) Ist $P = (K,G)$ ein gerichteter Plan und $x \in K$, so ist der Ausgangsrad $d_+(x)$ [bzw. der Eingangsgrad $d_-(x)$] von x definiert als Anzahl der Gänge mit x als Startecke [Zielecke]. Für einen ungerichteten Plan gilt: $d(x) = d_+(x) + d_-(x)$.

Definition 8 (Gangfolge, Weg, Schlingenfreier Weg).

Sei $P = (K,G)$ ein Plan und $x_i \in K$. Eine Gangfolge $g_i \in G$ mit dem Startgang g_0 und dem Zielgang g_m ist eine Sequenz von Gängen der Form $g_1 = \vec{x_0x_1}, g_2 = \vec{x_1x_2}, \ldots, g_m = \vec{x_{m-1}x_m}$.
Eine Weg ist ein Gangfolge aus lauter verschiedenen Gängen.
Ein Schlingenfreier Weg ist ein Weg mit jeweils paarweise verschiedenen Kreuzungen, abgesehen von möglicherweise $x_0 = x_m$.

Definition 9 (zyklisch, azyklisch).
Ein Plan $P = (K, G)$ heißt azyklisch, wenn in ihm ausschließlich kreisfreie Wege existieren. Andernfalls heißt er zyklisch.

Man beachte das die Startkreuzung eines Ganges g_i jeweils die Zielkreuzung des Ganges g_{i-1} ist und die Wege / Schlingenfreien Wege dementsprechend gerichtete Wege / Schlingenfreie Wege sind. Nach der Definition grundlegender Begriffe werden im folgenden wichtige Erkundungsstrategien aus der Graphentheorie anhand von Beispielen erläutert.

Sei P ein ungerichteter verbundener Plan gemäß obiger Definition, von dem alle Gänge aufgesucht werden sollen. Dieses Problem stellt im Vergleich zur Wegsuche von einem beliebigen Startgang aus zu einem beliebigen Zielgang innerhalb eines ungerichteten Plans eine Verallgemeinerung dar. Weiterhin sei die Möglichkeit gegeben, jeden bereits aufgesuchten Gang zu markieren um anzuzeigen, daß er bereits besucht worden ist. Die beiden häufigst verwendeten Erkundungstechniken sind die

- Tiefenerstsuche und die
- Breitenerstsuche.

Im folgenden Kapitel sind diese Erkundungsstrategien näher erläutert.

3.1 Tiefenerstsuche

Um eine komplette Tiefenerst-Durchquerung des ungerichteten Plans durchzuführen, ist es notwendig, einen beliebigen Gang innerhalb des Plans als Anfangspunkt zu wählen und diesen zu markieren um anzuzeigen, daß er bereits besucht worden ist. Existiert ein zum Startgang benachbarter Gang, so wird dieser als neuer Startgang gewählt und derselbe Schritt erneut durchgeführt. An dieser Stelle ist bereits die Rekursivität der Tiefenerstuche erkennbar. Die Strategie hierbei ist es also, möglichst viele Rekursionsebenen zu generieren und zwar solange, bis kein weiterer unmarkierter Gang zu dem aktuellen Startgang benachbart ist. In diesem Fall ist die Rekursion blockiert, und es wird auf einer höheren Ebene mit der Erkundung des Plans fortgefahren. Eine höhere Ebene bezeichnet dabei eine Rekursionsebene, die zu einem früheren Zeitpunkt erreicht wird. Der Algorithmus wird dann abgebrochen, wenn alle Gänge innerhalb des ungerichteten Plans markiert sind. Schließt man die Bildung von Kreisen aus, so wird an dieser Stelle deutlich, daß der Graph bei einer vollständigen Durchquerung in einen k-nären Baum zerlegt wird, wobei k die maximale Anzahl von Kindgängen ist, die in bezug auf alle Vatergänge im Plan vorkommt. Um die Wirkungsweise dieses Algorithmus zu verdeutlichen, erfolgt an dieser Stelle die Erläuterung anhand eines Beispiels:

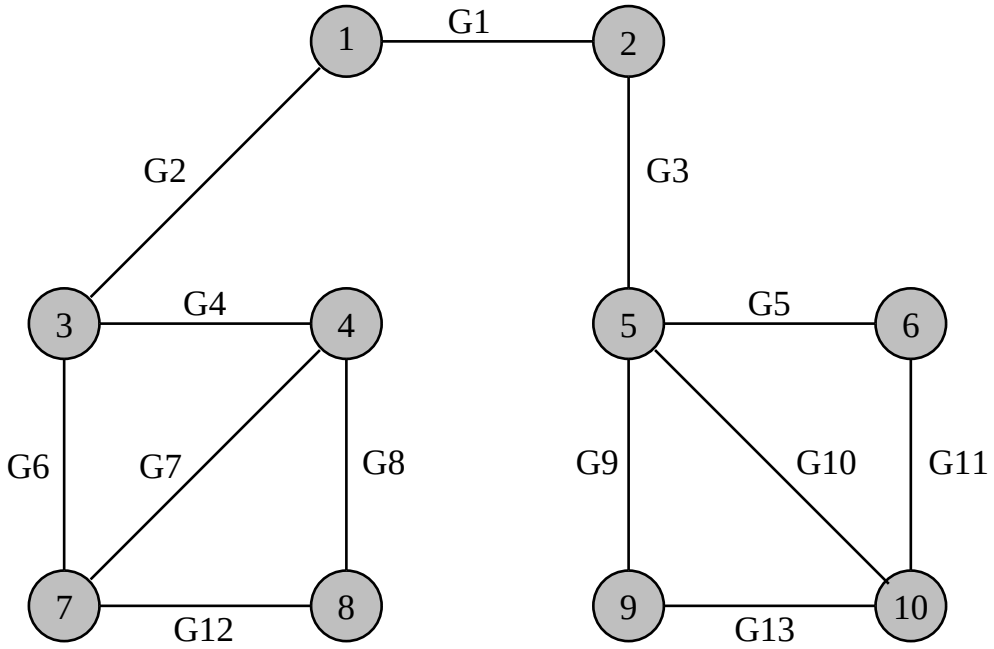


Bild 3.1: Tiefenerstsuche in ungerichteten Plänen

Geht man davon aus, daß die Abzweige eines gegebenen Ganges in numerisch aufsteigender Reihenfolge untersucht werden und Gang 1 als Startgang deklariert wird, so verläuft eine Tiefenerstsuche in folgender Art und Weise ab, wobei jeder rekursive Aufruf mit TES(G_i) mit i als Index des aktuellen Ganges bezeichnet wird:

1.) TES(G_1) Erster Aufruf.

2.) TES(G_2) Rekursiver Aufruf.

3.) TES(G_4) Rekursiver Aufruf.

4.) TES(G_7) Rekursiver Aufruf.

5.) TES(G_6) Rekursiver Aufruf. Weitere Suche blockiert.

6.) TES(G_{12}) Abzweig von G_7 wurde noch nicht aufgesucht.

7.) TES(G_8) Rekursiver Aufruf. Weitere Suche blockiert.

8.) TES(G_3) Rekursiver Aufruf.

9.) TES(G_5) Rekursiver Aufruf.

10.) TES(G_{11}) Rekursiver Aufruf.

11.) TES(G_{10}) Rekursiver Aufruf.

12.) TES(G_9) Rekursiver Aufruf.

13.) TES(G_{13}) Rekursiver Aufruf. Alle Gänge abgearbeitet.

In obigem Beispiel werden sämtliche Gänge des ungerichteten Plans durchlaufen, da die Bildung von Kreisen nicht ausgeschlossen ist. Hält man sich die Wirkungsweise des Algorithmus vor Augen, so wird klar, daß es sich bei dem bei der Tiefenerstsuche entstehenden Graphen um einen Baum handelt. Schließt man die Bildung von Kreisen jedoch aus, d.h. der dabei entstehende Graph ist azyklisch, so ordnet die Durchquerung eines verbundenen Plans mittels Tiefenerstsuche diesem einen Spannbaum zu. Gängen, die zur Bildung eines Kreises führen, werden im entstehenden Baum keine entsprechenden Kanten zugeordnet. Der Startgang wird bei der Erkundung zur Wurzel des Baumes.

Ist der ungerichtete Plan hingegen nicht verbunden, so ordnet ihm die Tiefenerstsuche nicht nur einen, sondern mehrere Bäume zu, nämlich jeweils einen Baum für jede verbundene Komponente des Plans.

3.2 Gelenkgänge

Gelenkgänge, in der Graphentheorie als Gelenkpunkte bezeichnet, stellen für die Optimierung eines einmal berechneten Weges, wie in späteren Kapiteln noch gezeigt wird, ein besonders wichtiges Kriterium dar. Die Definition eines Gelenkganges bezogen auf einen ungerichteten Gang $\overrightarrow{\leftarrow}xy \in G$ lautet, abgeleitet aus der Graphentheorie, folgendermaßen:

Definition 10 (Gelenkgang).
Ein ungerichteter bzw. gerichteter Gang $\overrightarrow{\leftarrow}xy \in G$ ist ein Gelenkgang, wenn der Subgraph, den man erhält, wenn $\overrightarrow{\leftarrow}xy \in G$ aus dem Graphen gelöscht wird, nicht mehr verbunden ist.

Dies wird deutlich, wenn man den Beispielgraphen der Tiefenerstsuche aus Bild 3.1 betrachtet. Löscht man den Gang 1 aus dem Graphen, so entstehen zwei verbundene Teilgraphen (Komponenten), bestehend aus
$\{G_2,G_4,G_6,G_7,G_8,G_{12}\}$ und $\{G_3,G_5,G_9,G_{10},G_{11},G_{13}\}$.

Um noch einmal herauszustellen, welche besondere Aufmerksamkeit eventuell in einer gegebenen Umgebung vorhandenen Gelenkgängen bei der Wegoptimierung gewidmet werden muß, soll an dieser Stelle bereits eine Testumgebung für die in dieser Arbeit implementierte Wegesuche als Beispiel angeführt werden. Bild 3.2 zeigt das entsprechende Gangsystem.

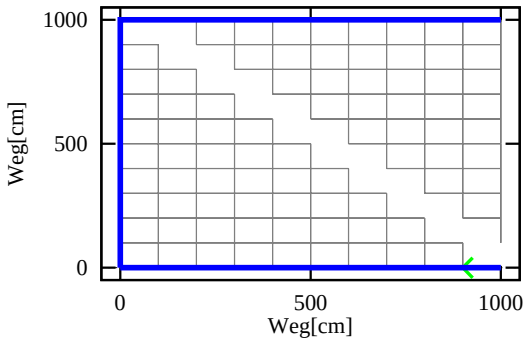


Bild 3.2: Kenntnis von Gelenkgängen

Man erkennt deutlich, daß bei der Entfernung von einem der beiden Gänge in der oberen linken Ecke der Umgebung diese in zwei zusammenhängende Teilumgebungen zerfällt. Der in Bild 3.2 gezeigte Weg muß über beide erwähnte Gelenkgänge führen. Auf die Wegoptimierung wird im Kapitel 5 näher eingegangen, es sei an dieser Stelle nur soviel erwähnt, daß für die Generierung eines Weges mit möglichst wenigen Abzweigen die Kenntnis der bereits erwähnten Gelenkgänge unumgänglich ist.

3.3 Tiefenerstsuche in gerichteten Plänen

Im Prinzip funktioniert der Algorithmus der Tiefenerstsuche in gerichteten Plänen auf dieselbe Art und Weise wie bei ungerichteten Plänen, sieht man von der Interpretation benachbarter (adjazenter) Gänge gemäß Defintion 4 ab. Beachtet man den Bedeutungswechsel der Definition für benachbart, so läßt sich der oben genannte Algorithmus TES(G_i) ebenfalls auf gerichtete Pläne anwenden. Bei Überführung der ungerichteten Gänge aus dem Beispielplan in gerichtete Gänge in der Art und Weise, daß der im folgenden aufgeführte Plan entsteht, verhält sich der Algorithmus folgendermaßen. Als Startgang wird wiederum Gang 1 gewählt und die Abzweige eines Ganges werden in numerisch aufsteigender Reihenfolge untersucht.

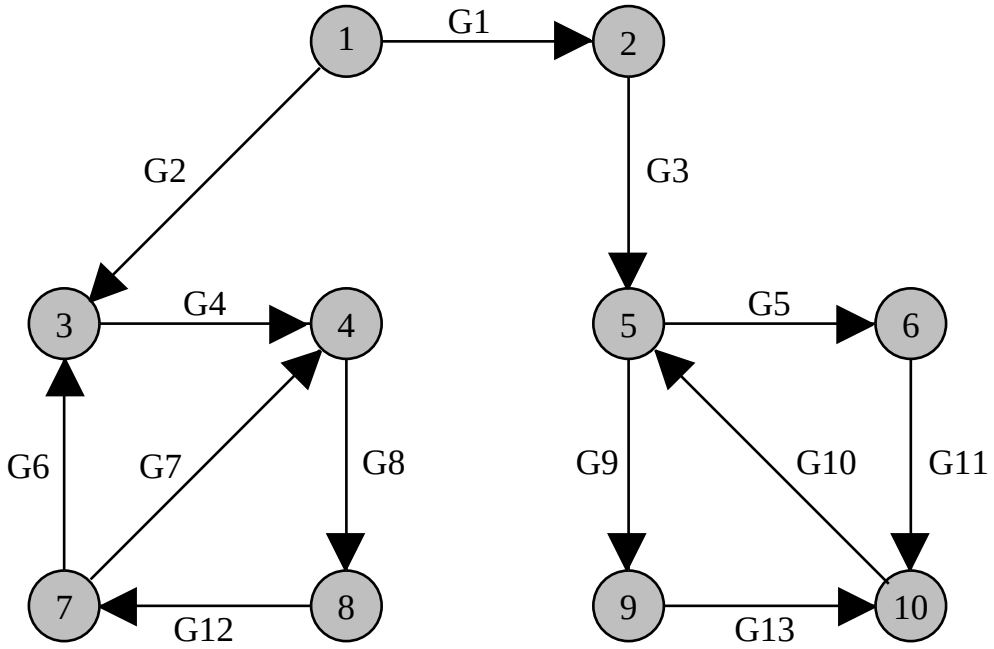


Bild 3.3: Tiefenerstsuche in gerichteten Plänen

1.) TES(G_1) Erster Aufruf.

2.) TES(G_3) Rekursiver Aufruf.

3.) TES(G_5) Rekursiver Aufruf.

4.) TES(G_{11}) Rekursiver Aufruf.

5.) TES(G_{10}) Rekursiver Aufruf.

6.) TES(G_9) Rekursiver Aufruf.

7.) TES(G_{13}) Rekursiver Aufruf. Weitere Suche blockiert.

8.) TES(G_2) Abzweig von G_1 wurde noch nicht aufgesucht.

9.) TES(G_4) Rekursiver Aufruf.

10.) TES(G_8) Rekursiver Aufruf.

11.) TES(G_{12}) Rekursiver Aufruf.

12.) TES(G_6) Rekursiver Aufruf. Weitere Suche blockiert.

13.) TES(G_7) Rekursiver Aufruf. Alle Gänge abgearbeitet.

Weiterhin ist zu beachten, daß ein verbundener, gerichteter Plan durchaus aus mehreren Bäumen bestehen kann. Der in Bild 3.4 gezeigte gerichtete Plan soll dieses verdeutlichen.

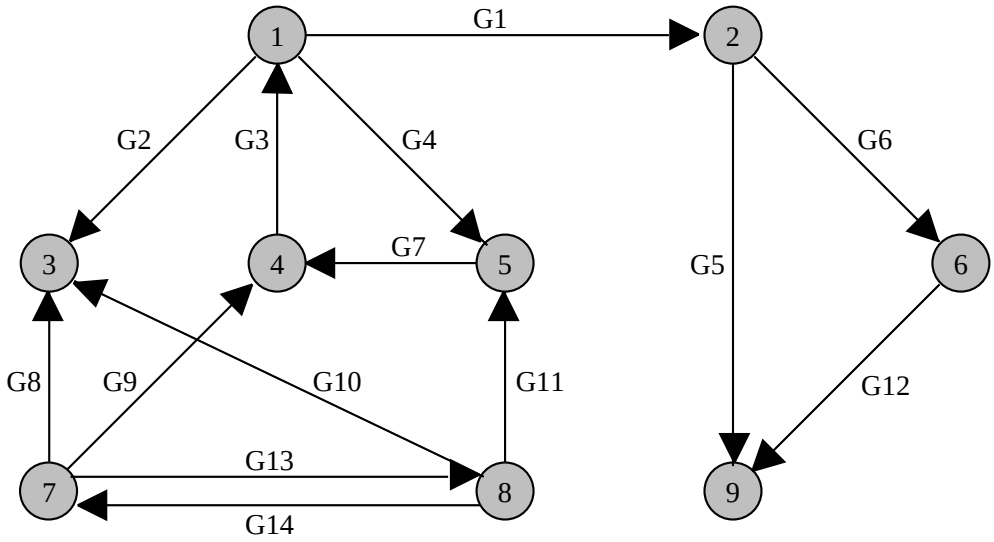


Bild 3.4: Verbundener, gerichteter Plan, bestehend aus mehreren Bäumen

Die Kreuzungen 7 und 8 in Bild 3.4 sind nur zueinander benachbart. Da keine Nachbarschaft zu den anderen Kreuzungen des Plans besteht, generiert sowohl die Tiefenerstsuche als auch die Breitenerstsuche zwei Bäume bestehend aus $\{G_1,G_5,G_6,G_{12},G_2,G_4,G_7,G_3\}$ und $\{G_8,G_9,G_3,G_1,G_5,G_6,G_{12},G_2,G_4,$ $G_7,G_{13},G_{10},G_{11},G_{14}\}$ wenn als Startgänge z.B. G_1 und G_8 gewählt werden.

3.4 Breitenerstsuche

Wie im vorhergehenden Abschnitt gezeigt, versucht eine Tiefenerstsuche in einem gerichteten bzw. ungerichteten Plan, ausgehend von einem Startgang, zunächst dessen nächstmöglichen Abzeig, dann wiederum dessen nächstmöglichen Nachbargang zu besuchen usw. Im Gegensatz dazu geht der Algorithmus der Breitenerstsuche von einem Startgang aus und besucht zunächst sämtliche möglichen Abzweige dieses ersten Ganges. Im zweiten Schritt werden wiederum alle möglichen Nachbargänge der gewählten Abzweige besucht. Diese Verfahrensweise wird solange fortgesetzt, bis alle in dem betreffenden verbundenen gerichteten bzw. ungerichteten (Teil-) Plan vorhandenen Gänge abgearbeitet wurden. Der Ablauf der Breitenerstsuche erinnert somit an den Diffusionsprozeß eines Gases in einer abgeschlossenen Umgebung, verursacht durch die Brown'sche Molekularbewegung. Hält man sich die Wirkungsweise des Algorithmus vor Augen, so ist leicht ersichtlich, daß die Breitenerstsuche wie auch die Tiefenerstsuche aus einem gege-

benen Plan einen Baum generiert. Verwendung findet dieser Algorithmus in der Graphentheorie bei der Erkundung unendlicher Graphen oder beispielsweise, um einen möglichst kurzen Pfad von einem Punkt zu einem anderen zu generieren.

Dieses Verfahren wurde auch in [27] angewendet. Hierbei wird die gegebene Umgebung, in der sich der Roboter bewegen soll, inklusive eventuell vorhandener Hindernisse durch ein Gittermodell dargestellt. Der verwendete Planungsalgorithmus basiert auf der Entfernungsfeldmethode. Prinzipiell kann er folgendermaßen erklärt werden:

- Die Zielposition ist mit einem Startwert (’1’) initialisiert.
- Jede der vier Nachbarzellen erhält einen um 1 inkrementierten Wert, sofern sie nicht den Gitterrand oder ein Hindernis enthält.
- Diese Prozedur wird wiederholt, bis die Zelle, die die Fahrzeugposition enthält, mit einem Wert versehen worden ist.
- Von dieser Zelle aus springt man zur jeweils niederwertigsten der 8 umgebenden Zellen und findet somit den kürzesten Pfad zum Ziel.

Bild 3.5 soll dieses Verfahren illustrieren.

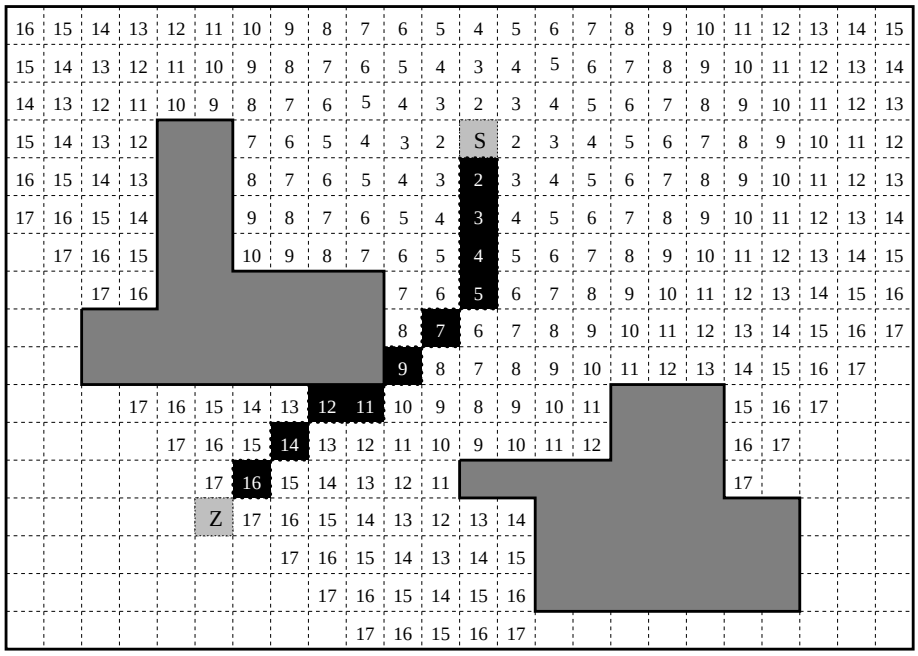


Bild 3.5: Gradientensuche

Nach der Erläuterung wichtiger Suchalgorithmen, ist im folgenden Kapitel der vom Planungsmodul verwendete Algorithmus zur Wegsuche eingehend beschrieben.

4 Die Wegsuche

Um eine geeignete Auswahl des zu verwendenden Suchalgorithmus für die Wegesuche des Planungssystems zu treffen, ist es zunächst notwendig, die erforderlichen Kriterien festzulegen. Wie im vorhergehenden Kapitel gezeigt, ermöglicht die Breitenerstsuche zwar eine Generierung des kürzesten Weges in einer gegebenen Umgebung, es ist jedoch fraglich, ob dieser kürzeste Weg auch der optimale ist. Um diese Frage zu klären, soll an dieser Stelle eine gegebene Umgebung betrachtet werden, die aus einem Gangsystem aus zehn mal zehn Gängen besteht.

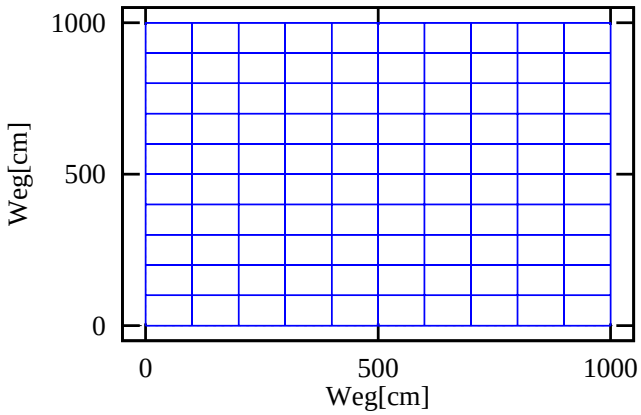


Bild 4.1: Eine einfache Testumgebung

Numeriert man die Gänge in Bild 4.1 zeilenweise von unten links nach oben rechts durch und startet eine Breitenerstsuche vom Startgang in der unteren linken Ecke zum Zielgang in der obereren rechten Ecke, so generiert der Algorithmus der Breitenerstsuche einen Weg, der diagonal vom Startgang zum Zielgang führt. Bild 4.2 verdeutlicht dieses.

Die Länge des berechneten Weges entspricht bei näherem Hinsehen der Strecke, die vom Startgang über eine der Ecken links oben bzw. rechts unten zum Zielgang zurückgelegt werden muß. Die Anzahl der zu fahrenden Abzweige ist bei dem diagonalen Pfad jedoch um viele Male höher als bei einem der Wege über die genannten Ecken des Gangsystems. Dies hat zur Folge, daß die Fahrzeit über den generierten Weg ebenfalls um den gleichen Faktor ansteigt. Eine Optimierung des berechneten Weges hinsichtlich der zu fahrenden Abzweige ist aus den gewonnenen Informationen nicht möglich. Die Breitenerstsuche liefert somit in bezug auf die Optimalität eines ermittelten Weges kein befriedigendes Ergebnis. Um die notwendigen Informationen für die Ermittlung eines optimalen Weges unter Berück-

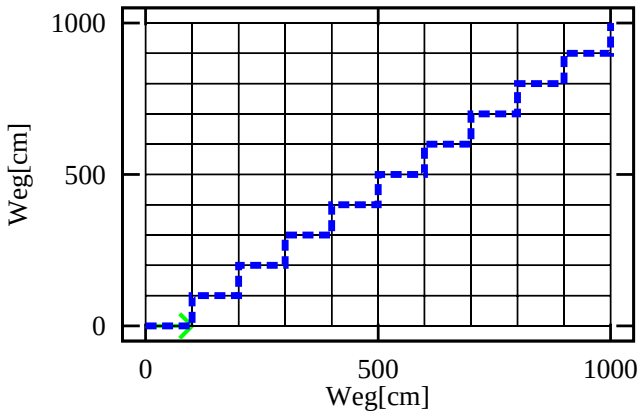


Bild 4.2: Breitenerstsuche

sichtigung der Fahrdynamik des autonomen mobilen Robotersystems MORIA zu erhalten, wurde daher der Algorithmus einer abgewandelten Tiefenerstsuche verwendet.

4.1 Die Karte

Als Beispiel für die Überführung der grafischen Repräsentation einer Karte in die verwendete strukturelle Darstellung des Planers sei hier eine aus vier Gängen bestehende Kreuzung aufgeführt, wobei alle Gänge mit der betreffenden Kreuzung benachbart sind.

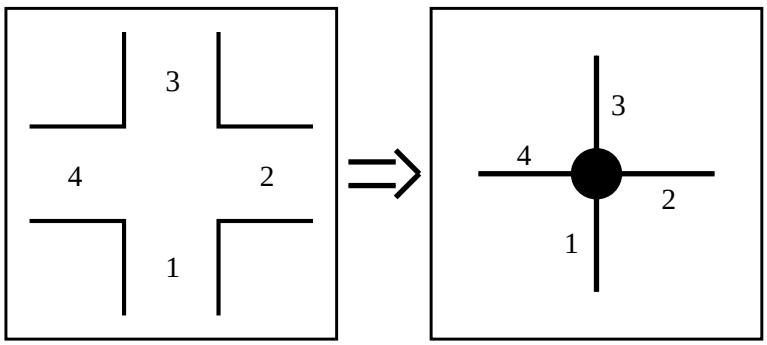


Bild 4.3: Überführung der graphischen Repräsentation einer Karte in die planerinterne strukturelle Darstellung

Die strukturellen Informationen der in Bild 4.3 links gezeigten Kreuzung werden so erfaßt, daß für jeden Gang seine benachbarten Abzweige mit den folgenden Parametern gespeichert werden. Zur Erläuterung der verwendeten Orientierungen ist in Bild 4.4 der Kompaß des Navigators abgebildet.

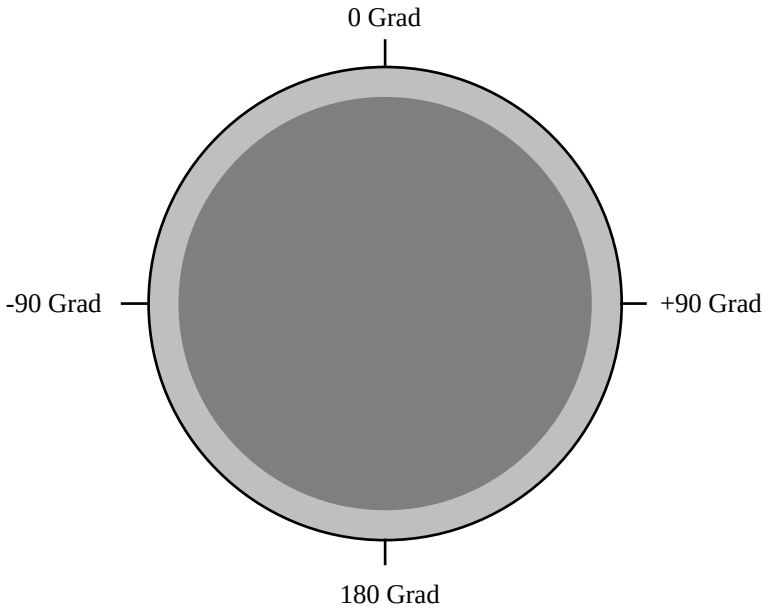


Bild 4.4: Der Kompaß des Navigators

Die strukturellen Informationen sind:

- Gangnummer des Abzweigs. Als Beispiel Gang 1 aus Bild 4.3.
- Orientierung des Gangendes, an welchem der Abzweig in bezug auf den jeweiligen Gang liegt. Gang 1 hat die drei Abzweige 2, 3 und 4. Die betreffende Orientierung für Gang 2 ist 0 Grad.
- Orientierung, in der sich der Roboter weiterbewegt, falls der betreffende Abzweig befahren wird. Dieser Parameter ist für Gang 2 90 Grad.
- Als weiteres Kriterium wird für jeden Gang sowohl die ermittelte Ganglänge als auch die Gangbreite angegeben.

4.2 Der verwendete Suchalgorithmus

Zur Lösung des Problems der Wegegenerierung bietet es sich an, eine koordinatenabhängige Suche durchzuführen. Hierzu ist es notwendig, die Koordinaten jedes Gangendes in x- und y-Richtung abzuspeichern. Startet man die Wegesuche vom Startgang aus, unter Berücksichtigung der Vorgabe, daß die x-Koordinate zwischen dem Gang der aktuellen Rekursionsebene und dem Zielgang abgeglichen werden soll, so erhält man den in dem Bild 4.5 gezeigten Weg. Er führt

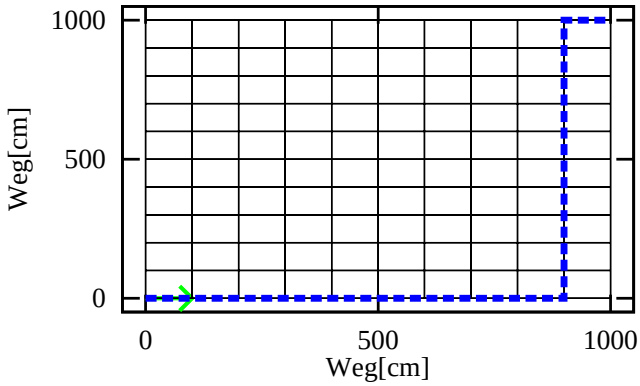


Bild 4.5: Abgleich der x-Koordinate

vom Startgang über die untere rechte Ecke des Gangsystems zum Zielgang in der oberen rechten Ecke.

Hinsichtlich der Anzahl der zu fahrenden Abzweige ist dieser sicherlich einer der beiden optimalen Wege. Desweiteren ist in diesem Fall die gleiche Strecke zurückzulegen wie über den diagonalen Weg. Im umgekehrten Fall, d.h. bei Abgleich der y-Koordinate, wird ein Weg ermittelt, der vom Startgang über die obere linke Ecke zum Zielgang führt. Bild 4.6 zeigt den generierten Weg.

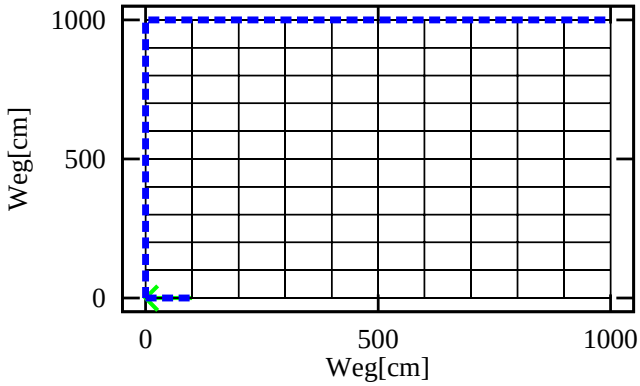


Bild 4.6: Abgleich der y-Koordinate

In der Testphase dieser ersten Stufe einer Tiefenerstsuche zeigte sich jedoch, daß die Wegesuche mit nur einem Parameter (Abgleich einer der beiden Koordinaten bei einseitiger Suche vom Startgang aus) nicht der Weisheit letzter Schluß ist. Die weitere Verfeinerung des Algorithmus wird an folgendem Beispiel erläutert.

Bild 4.7 zeigt das Ergebnis einer Weggenerierung bei einseitiger Suche vom Startgang aus, wobei ein Abgleich der x-Koordinate zwischen dem Gang der aktuellen Rekursionsebene und dem Zielgang durchgeführt wurde. Man erkennt, daß

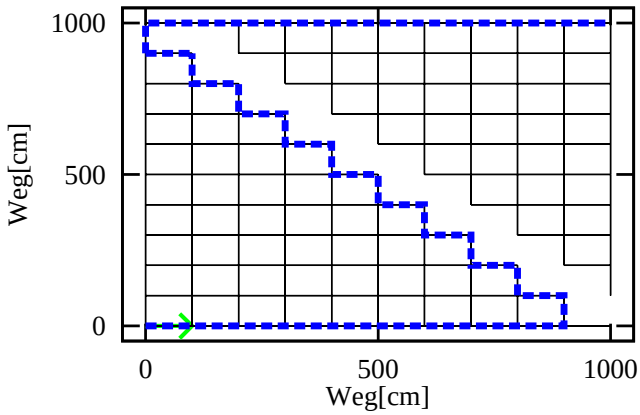


Bild 4.7: Unidirektionale Suche bei Abgleich der X-Koordinate

die Suche zunächst in das untere Dreieck der Umgebung läuft und schließlich über den Gelenkgang in der oberen linken Ecke in das obere Dreieck gelangt. Einmal am Gelenkgang angekommen, ist die Suche in bezug auf später zu befahrende Abzweige unkritisch, da der Abgleich der x-Koordinate direkt über die obere waagerechte Gangreihe erfolgen kann.
Bezüglich der zu fahrenden Abzweige und der Länge ist jedoch der generierte Weg bis zu dem erwähnten Gelenkgang nicht optimal.

Um dieses Problem zu umgehen, war eine Erweiterung der Funktionalität des verwendeten Algorithmus notwendig. Diese Erweiterung beinhaltet zum einen eine Ausdehnung der einseitigen (unidirektionalen) Suche, die nur vom Startgang aus zum Zielgang erfolgt, auf eine beidseitige (bidirektionale) Suche. Dies bedeutet, daß der rekursive Algorithmus der Wegsuche im ersten Rekursionsschritt eine Suche vom Startgang zum Zielgang und im zweiten eine umgekehrte Suche vom Zielgang zum Startgang durchführt. Im dritten Rekursionsschritt wiederholt sich dieser Vorgang. Der Vorteil der beidseitigen gegenüber einer einseitigen Suche besteht darin, daß nicht nur der Abgleich einer Koordinaten, sondern sowohl der x- als auch der y-Koordinaten durchgeführt werden kann. Die Festlegung, welche der beiden Wegsuchen vom Start- bzw. Zielgang aus dabei versucht, welche Koordinate des jeweiligen Zielgangs abzugleichen, ist dabei zunächst willkürlich . Es muß jedoch gewährleistet sein, daß die jeweils abzugleichende Koordinate für jede der beiden Suchen zunächst konstant bleibt.

Als weiterer Vorteil, vor allen Dingen bezüglich der Laufzeit des Algorithmus, hat sich herausgestellt, den Koordinatenabgleich so durchzuführen, daß nicht die jeweilige Koordinate des Ziel- bzw. Startganges, sondern die des gewählten Ganges der aktuellen Rekursionsebene abgeglichen wird. Das hat zur Folge, daß die beiden Spitzen der bis zu einem bestimmten Zeitpunkt generierten Teilwe-

ge vom Start- bzw. Zielgang aus sich im Laufe der weiteren Wegsuche einander zu nähern versuchen. Der Koordinatenabgleich findet dabei solange statt, bis die jeweilige abzugleichende Koordinate der des Ganges der aktuellen Rekursionsebene der Gegensuche entspricht. Sind die beiden Wegsuchen zu diesem Zeitpunkt noch nicht aufeinander getroffen, so erfolgt der Abgleich der jeweiligen noch nicht gewählten Koordinate. An dieser Stelle soll nur auf die Funktionsweise des Suchalgorithmus eingegangen werden, daher finden sich nähere Informationen bezüglich des Koordinatenabgleichs im Abschnitt 4.4.

Die Abbruchkriterien für eine Weggenerierung, die in obiger Form durchgeführt wird, lassen sich somit folgendermaßen festlegen:

Ein Abbruch der Wegsuche erfolgt dann, wenn

- kein regulärer Weg vorhanden ist, d.h. alle möglichen Abzweige wurden vom Algorithmus abgearbeitet,
- die Wegsuche vom Startgang aus auf den Zielgang getroffen ist,
- die Wegsuche vom Zielgang aus auf den Startgang getroffen ist oder
- beide Gänge der aktuellen Rekursionsebenen vom Start- bzw. vom Zielgang aus identisch sind, d.h. die beiden Wegsuchen sind aufeinander getroffen.

In bezug auf Punkt 2, 3 und 4 ist es nach der Wegsuche notwendig, eine allgemein verwendbare Funktion zu implementieren, die die Kommandogenerierung für jeden der drei Fälle übernimmt. Näheres hierzu findet sich im Abschnitt 4.5. Bei Erweiterung der Funktionalität des Suchalgorithmus wird bei dem zuletzt genannten Beispiel der in Bild 4.8 generiert.

Wie im Bild 4.8 leicht zu erkennen ist, wird als Abgleichsparameter bei der Suche vom Startgang aus die x-Koordinate, bei der vom Zielgang aus die y-Koordinate verwendet. Der generierte Weg erreicht jedoch bezüglich der zu fahrenden Abbiegungen nicht die Güte in dem Maße, wie es bei einseitiger Suche und Abgleich nur einer, nämlich der x-Koordinate, der Fall war.

Um diesen Nachteil auszugleichen, liegt es nahe, die Wegsuche einfach noch einmal durchzuführen, allerdings werden diesmal die auszugleichenden Gangkoordinaten einfach vertauscht. Wurde bei der ersten Berechnung vom Startgang aus die Koordinate in x-Richtung abgeglichen, so erfolgt nun der Abgleich der y-Koordinate. Entsprechendes gilt für die Wegsuche vom Zielgang aus. Aus den beiden Berechnungen wird der jeweils optimale Weg bezüglich der zu fahrenden Abbiegungen gewählt.

Führt man die Berechnung in der vorgenannten Art und Weise durch, so erhält man folgendes Bild:

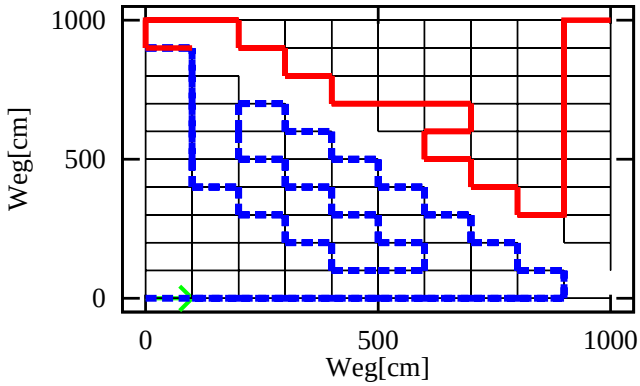


Bild 4.8: Beidseitige Suche bei Abgleich der X-Koordinate vom Startgang und der Y-Koordinate vom Zielgang aus

Betrachtet man die Länge des Weges und die Anzahl der zu fahrenden Abbiegungen, so läßt sich dieser im zweiten Schritt berechnete Weg sicherlich als der optimale bezeichnen. Wie aus Bild 4.9 ersichtlich ist, handelt es sich bei dem zuvor genannten Beispiel jedoch nur um eine, auf die bisherige Funktionalität des Suchalgorithmus optimale Problemstellung.

In der in Bild 4.10 gezeigten Umgebung ist, wie in den Beispielen zuvor, ebenfalls ein Gelenkgang vorhanden. Es handelt sich dabei um den Gang, der die von der Struktur her dreieckigen Gangsysteme im oberen und unteren Teil der Umgebung in der Mitte miteinander verbindet.

Das Bild 4.10 zeigt, daß ein einfacher Koordinatenabgleich bei der Wegsuche, auch wenn er mit jeweils beiden Koordinaten durchgeführt wird, bezüglich der zu fahrenden Abbiegungen keine optimale Lösung erbringt. Es ist also eine zusätzliche Optimierung des im ersten bzw. zweiten Schritt berechneten Weges notwendig. Weitere genaue Informationen zu diesem Problem finden sich im Kapitel 5. Im folgenden ist die Funktionsweise des verwendeten Suchalgorithmus mit anschließendem Beweis in mathematischer Form gemäß den Definitionen aus Kapitel 3 dargestellt.

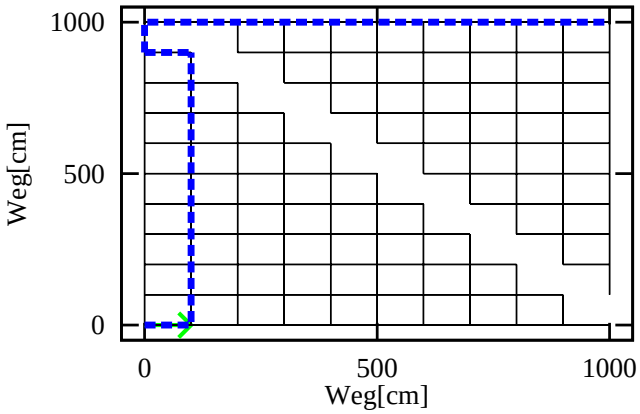


Bild 4.9: Bidirektionale Suche bei Abgleich der Y-Koordinate vom Startgang und der X-Koordinate vom Zielgang aus

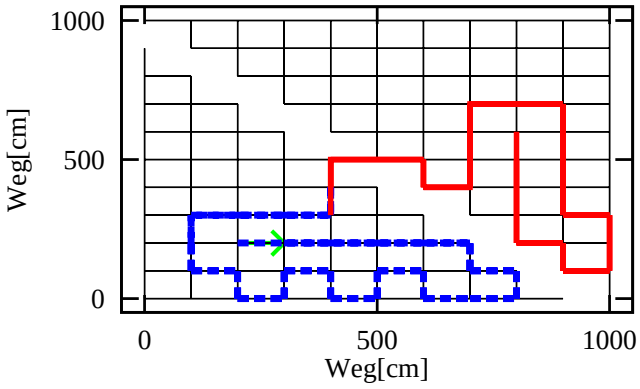


Bild 4.10: Bidirektionale Suche bei Abgleich der X-Koordinate vom Startgang und der Y-Koordinate vom Zielgang aus

Algorithmus 1 (Wegsuche).
Sei $P = (K, G)$ ein endlicher Plan und $g_0 = \overrightarrow{x_0 x_1}, g_m = \overrightarrow{x_{m-1} x_m} \in G$ zwei Gänge mit der Startkreuzung x_0 und der Endkreuzung x_m. Sei SL ein Weg vom Startgang und ZL ein Weg vom Zielgang ausgehend. Weiterhin sei $MA_m \in \{-2,-1,0,1,2\}$ eine Liste von Markierungen für jeden Gang $g \in G$.

Initialisierung Zu Beginn seien der Weg vom Startgang $SL = g_0, i = 0$ und vom Zielgang $ZL = g_m, j = m$, d.h. $g_i \in SL$ ist der aktuelle Gang in dem Weg vom Start und $g_j \in ZL$ der aktuelle Gang in dem Weg vom Ziel aus. Die Markierung $MA_k = 0, k = 1 \ldots m-1$ ist mit 0 markiert. Desweiteren wird in der Markierungsliste $MA_0 = 1$ und $MA_m = -1$ markiert. [1]

1.) Falls $g_i = g_j$, d.h. der aktuelle Gang g_i im Weg vom Startgang entspricht dem aktuelle Gang im Weg vom Zielgang, dann ist $L = SL \cup ZL$ und L ist der gesuchte Weg.

2.) Falls für den aktuellen Gang i in der Markierungsliste $MA_i = -1$, so ist $L = SL \cup ZL_i$, wobei ZL_i der Weg vom Gang g_i bis zum Zielgang ist.

3.) Falls für den aktuellen Gang j in der Markierungsliste $MA_j = 1$, so ist $L = SL_j \cup ZL$, wobei SL_j der Weg vom Gang g_i bis zum Startgang ist.

4.) Falls für den aktuellen Gang i in der Markierungsliste $MA_i = -2$, d.h. der Gang g_i ist bereits schon mal in einem Weg vom Zielgang vorgekommen, so ist $L = SL \cup L_{neu}$ mit L_{neu} = Finde Weg von g_i nach g_m.

5.) Falls für den aktuellen Gang j in der Markierungsliste $MA_j = 2$, d.h. der Gang g_j ist bereits schon mal in einem Weg vom Startgang vorgekommen, so ist $L = L_{neu} \cup ZL$ mit L_{neu} = Finde Weg von g_j nach g_0.

6.) Sei $g_{i,j} = \overrightarrow{x_a x_b}$. Berechne $k = d_+(x_a) = |\, NB_{g_{i,j}} \,|$ mit $NB_{g_{i,j}} := \{\overrightarrow{x_a x_c} \,|\, g_{i,j} = \overrightarrow{x_a x_b} \wedge g_{i+1,j-1} = \overrightarrow{x_a x_c} \wedge x_c$ ist benachbart $x_a,\ x_a, x_b, x_c \in G\}\ |$.

   (a) Falls $k = 0$ markiere $MA_{i,j} = 2, -2$ und entferne $g_{i,j}$ aus dem Weg vom Startgang bzw. Zielgang und beende den Aufruf. Für $k > 0$ sortiere die Menge $NB_{g_{i,j}}$ der möglichen Nachbargänge.

   (b) Wähle aus $NB_{g_{i,j}}$ den nächsten Gang g_{i+1} bzw. g_{j-1}.

   (c) Falls $MA_{i+1,j-1} = 1, -1$ ist, so wird der Gang ignoriert d.h. $NBg_{i,j} = NBg_{i,j} \setminus g_{i+1,j-1}, k = k-1$ und es wird bei (a) fortgefahren.

[1] $MA_i, i = 1 \ldots m$ ist mit 1 bzw. -1 markiert, wenn g_i von dem Weg vom Startgang bzw. Zielgang ausgewählt wurde. Zusätzlich wird zur Verbesserung des Laufzeitverhaltens der Gang $g_i = 2$ bzw. $g_i = -2$ markiert, falls der Gang besucht wurde, aber nicht im aktuellen Weg vom Start- oder Ziel vorkommt.

(d) Ergänze den Start- bzw. Zielweg um den Gang g_{i+1}, g_{j-1} d.h. $SL = SL \cup g_{i+1}$ bzw. $ZL = g_{j-1} \cup ZL$ und markiere $MA_{i+1} = 1$ bzw. $MA_{j-1} = -1$.

(e) Finde Weg von g_{i+1} nach g_j bzw. von g_i nach g_{j-1}.

(f) Falls L ein Weg ist, so wird der Aufruf beendet. Andernfalls ist $NBg_{i,j} = NBg_{i,j} \setminus g_{i+1,j-1}, k = k-1$ und es wird bei (a) fortgefahren.

Bemerkung:
Die Wegsuche kann direkt zu einer Suche Schlingenfreier Wege ergänzt werden, wenn nähmlich 6.(c) ersetzt wird durch:

> Falls in $NB_{g_{i,j}}$ $\exists MA_{i+1,j-1} = 1, -1,\ i, j = 1 \ldots k$, so ist $NBg_{i,j} = \emptyset, k = 0$ und es wird bei (a) fortgefahren.

Satz 1
Sei $P = (K, G)$ ein endlicher Plan und $g_0, g_m \in G$ zwei Gänge. Der Algorithmus 1 findet genau dann einen Weg zwischen g_0, g_m wenn ein Schlingenfreier Weg zwischen g_0, g_m existiert.

Beweis:
„$\Rightarrow$“ Falls der Algorithmus einen Weg gefunden hat, so erhält man einen Schlingenfreien Weg, indem alle Gänge zwischen zwei gleichen Kreuzungen aus dem Weg entfernt werden.
„$\Leftarrow$“ Es existiert ein Schlingenfreier Weg zwischen g_0, g_m. Der Beweis erfolgt nun durch Induktion über n.

Für $n = 0$: Da $g_0 = g_0$ und die Markierungen MA richtig initialisiert werden, terminiert der Algorithmus nach Bedingung 1.

Für $n \to n+1$: OBdA. wird der Weg vom Startgang aus betrachtet.

Fall 1: $g_{n+1} = g_j$ der Algorithmus terminiert nach 1.

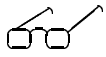
Fall 2: Ist $MA_{n+1} = -1$, d.h. der neue Gang $n+1$ ist bereits im Weg vom Ziel enthalten, so terminiert der Algorithmus nach 2.

Fall 3: Ist $MA_{n+1} = -2$, d.h. der neue Gang $n+1$ war bereits einmal als Gang im Weg vom Zielgang, so existiert dementsprechend ein Weg L vom Startgang zum Zielgang.[2]

Fall 4: In diesem Fall ist g_{n+1} ein neuer Gang in Richtung zum Ziel und es muß gezeigt werden, daß sich die Komplexität reduziert, die Markierungen richtig gesetzt sind (Invariante) und die neue Gangfolge $SL = SL \cup g_{n+1}$ ein Weg ist.

[2] Der Weg L wird nach 4 des Algorithmus dann durch einen weiteren Aufruf der Algorithmus mit dem Startgang g_i und dem Zielgang g_m berechnet

Da P laut Vorraussetzungen ein endlicher Plan ist, so ist G und damit auch NB_{g_n} endlich. Bei der Auswahl des nächsten Gangs wird NB_{g_n} durch 6.(c) und 6.(f) jeweils reduziert. Somit terminiert der Alorithmus. Die Invariante ist gegeben durch die Markierung von MA_{n+1} in 6.(d) bei der Auswahl des Gangs $n+1$. Laut Definition 8 ist ein Weg eine Gangfolge aus lauter verschiedenen Gängen. Da laut Vorrausetzung *SL* für n einen Weg ist, gilt dies ebenso fur $n+1$, da durch 6.(c) sichergestellt wird, daß kein Gang ausgewählt wird der schon in *SL* enthalten ist.

 Satz 1

Die aufzuwendende Zeit für die Wegsuche nach Algorithmus 1 ist von der Ordnung O(n), da alle besuchten Gänge g_i bzw. g_j markiert werden. Existiert ein Weg vom Startgang g_0 zum Zielgang g_m, so wird dieser zwar gefunden (siehe Beweis), es ist aber noch nicht geklärt, ob dieser Weg auch die optimale Lösung darstellt. Daher wird im Anschluß an die Wegsuche eine Wegoptimierung durchgeführt. Näheres dazu findet sich im Kapitel 5.

4.3 Weitere Parameter der Wegsuche

Da es sich bei MORIA um ein Transportfahrzeug handelt, ist es notwendig, die Kriterien der Wegsuche zu erweitern. Die Befahrbarkeit eines generierten Weges wird durch die Ladung des Roboters entscheidend beeinflußt. Wie in Kapitel 5 noch eingehend erläutert wird, hat die Anzahl der zu befahrenden Abbiegungen entscheidenden Einfluß auf die Güte eines generierten Weges. Einerseits nimmt die Fahrzeit mit steigender Anzahl der Abbiegungen zu, andererseits ist beim Transport von beweglicher Ladung wie z.B. gefüllten Champagnerflaschen das Befahren von vielen Abbiegungen ungünstig. Denkbar ist aber auch, daß die Außenmaße der Ladung die des Roboters bei weitem übertreffen und daher nicht jede Kreuzung bzw. jeder Gang befahren werden kann. Desweiteren können sich auch mehrere Roboter in ein und derselben Umgebung bewegen. Sind größere Bereiche der Umgebung durch wenige oder gar nur einen Gang verbunden, so kann es zu Engpässen beim Befahren der entsprechenden Gänge kommen. Aus diesen Gründen wird die Wegsuche durch zwei weitere Parameter beeinflußt.
Diese sind:

- Defintion einer minimalen Gangbreite, die ein Gang haben muß um befahren werden zu können
- Definition von Einbahnstraßen

Bild 4.11 stellt die Modularisierung des Planers zusammenfassend dar.

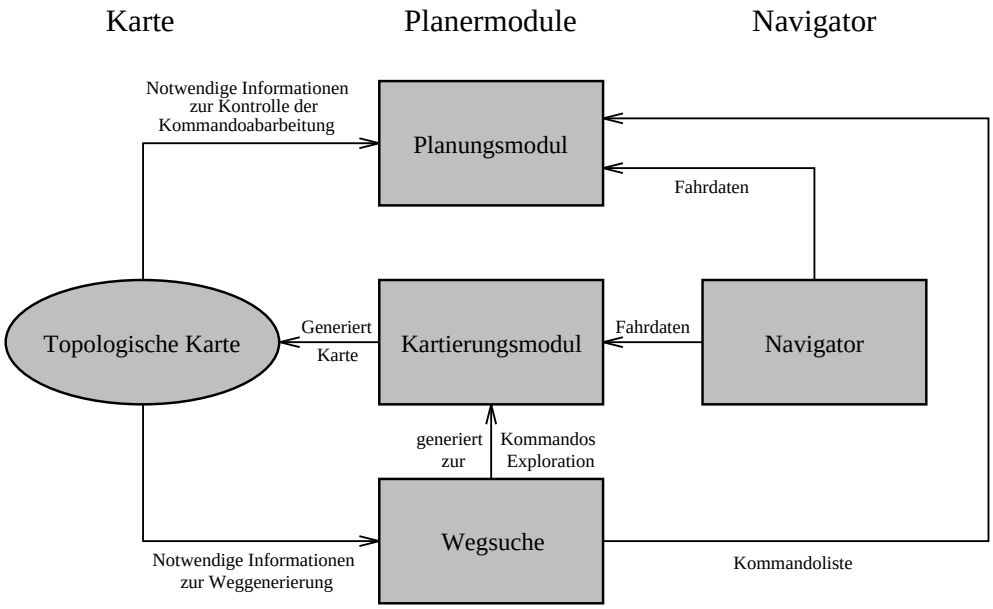


Bild 4.11: Die Module des Planers

4.4 Koordinatenabgleich und Auswahl des nächsten Abzweigs

Wie bereits im vorhergehenden Abschnitt gezeigt, stellt der Koordinatenabgleich zwischen dem Gang der aktuellen Rekursionsebene und dem aktuellen Zielgang der Wegsuche ein besonders wichtiges Kriterium dar. Ziel des angewendeten Verfahrens ist es, für den Gang der aktuellen Rekursionsebene einen Auswahlvektor zu bestimmen, der es ermöglicht, aus allen vom aktuellen Gang abzweigenden Gängen den optimalen auszuwählen. Die Bildung diese Auswahlvektors soll anhand Bild 4.12 erläutert werden.

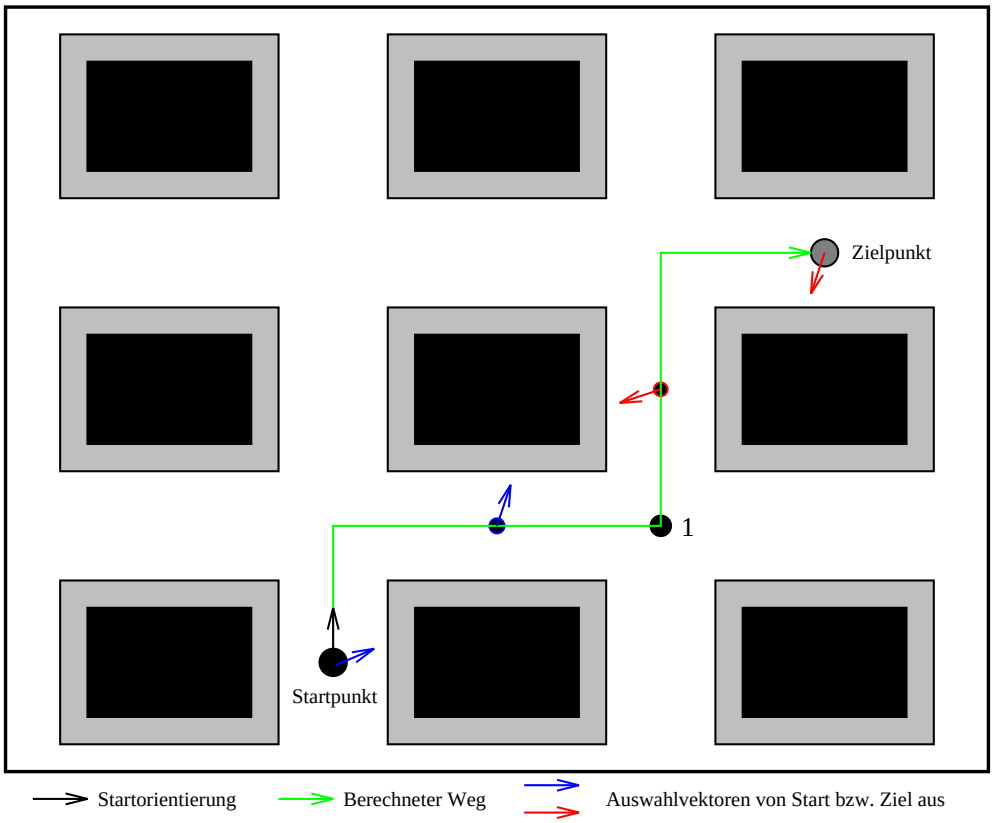


Bild 4.12: Auswahl des nächsten Abzweigs

Die bidirektionale Suche erfolgt vom Startpunkt aus bei Abgleich der x-Koordinaten und vom Zielgang aus bei Abgleich der y-Koordinaten zwischen den gewählten Gängen der beiden aktuellen Rekursionsebenen. Um vom Startpunkt aus einen Abgleich der x-Koordinaten zwischen Start- und Zielgang (erster Rekursionsschritt) zu erreichen, ist die Auswahl eines Ganges, der in der Orientierung +90 Grad (d.h. in Bild 4.12 nach rechts) vom Startgang abzweigt, die optimale Lösung. Da ein solcher Abzweig jedoch sowohl in Richtung der Startorientierung als auch

in entgegengesetzter Orientierung existent sein kann, werden als zweiter Parameter die y-Koordinaten der beiden Gänge herangezogen. Hierbei ist festzustellen, daß der Betrag der y-Koordinate des Startganges einen geringeren Wert als der Betrag der y-Koordinate des Zielganges darstellt. Um die Auswahl zwischen parallel liegenden Abzweigen eines Ganges zu ermöglichen, wird vom optimalen Wert bezüglich des Abgleichs der x-Koordinate in diesem Fall ein Wert von +10 Grad abgezogen. Da die Information, an welchem Gangende bzw. in welche Richtung (Orientierung) ein Gang vom aktuellen Gang abzweigt, in der strukturellen Repräsentation der Umgebungskarte enthalten ist, läßt sich somit ermitteln, welcher der vorhandenen Abzweige der lokal gesehen beste ist. Dies geschieht nahand der Berechnung der absoluten Differenz zwischen Sollorientierung des Auswahlvektors und Orientierung des Gangendes und der absoluten Differenz zwischen Sollorientierung des Auswahlvektors und Orientierung in der der Abzweig vom aktuellen Gang abzweigt.

Desweiteren wird dadurch vermieden, daß ein Abzweig gewählt wird, der entgegengesetzt zur Startorientierung bzw. zur Orientierung, in der ein Gang durchfahren werden soll liegt. Wird ein solcher Abzweig ausgewählt, so muß entweder ein zeitaufwendiges Wendemanöver im betreffenden Gang oder eine Umschaltung der Fahrtrichtung und somit eine rückwärtige Durchquerung des betreffenden Ganges durchgeführt werden. Eine Rückwärtsfahrt des autonomen mobilen Roboters MORIA ist deshalb nicht von Vorteil, weil

- sich eventuell auszuführende Änderungen des Steuerwinkels bei Kurvenfahrten schlechter einstellen lassen, da sich die Lenkachse im vorderen Drittel des Fahrzeugs befindet und sich damit eine Änderung des Steuerwinkels um bspw. 20 Grad viel stärker auswirkt als bei Vorwärtsfahrt und
- nur die Frontseite des Fahrzeugs gegen eventuell auftretende Kollisionen durch den angebrachten Bumper Schutz vor Verletzung von Personen bzw. Beschädigungen des Fahrzeugs oder Gegenständen in seiner Umgebung bietet.

Die einzigen Situationen, in der sich eine Rückwärtsfahrt nicht vermeiden lassen, sind:

- Durch Auftreten eines Hindernisses ist die weitere Durchfahrt des aktuellen Ganges versperrt und ein Wendemanöver ist nicht möglich.
- Bei dem Zielgang der letzten durchgeführten Fahrt handelt es sich um eine Sackgasse und ein Wendemanöver ist nicht möglich.

Im zweiten Rekursionsschritt erfolgt die Auswahl des ersten Abzweigs vom Zielgang aus. Die abzugleichenden Koordinaten sind hierbei die y-Koordinaten zwischen Ziel- und Startgang. Da die y-Koordinate des Zielpunktes einen höheren Wert als die des Startpunktes hat, ergibt sich als Richtung für den ermittelten Auswahlvektor eine Orientierung von 180 Grad. Im zweiten Schritt ermittelt

die Auswahlfunktion die Lage des Zielpunktes zum Startpunkt bezüglich der x-Koordinate. Da das Ergebnis das gleiche Resultat wie für die y-Koordinaten liefert, wird die Orientierung des Auswahlvektors um 10 Grad vermindert. Seine endgültige Orientierung wird somit auf einen Wert von -170 Grad festgelegt.

Im dritten Rekursionsschritt wird auf die gleiche Art und Weise der von der Suche vom Zielgang aus im zweiten Schritt gewählte Gang in den generierten Weg vom Startgang aus aufgenommen. Dasselbe geschieht wiederum im vierten Rekursionsschritt bezüglich der Suche vom Zielgang aus. Die Wegesuche wird im fünften Rekursionsschritt beendet, da der von der Suche vom Zielgang zuerst gewählte Abzweig mit dem Gang der aktuellen Rekursionsebene der Such vom Startgang aus übereinstimmt. Bei der Auswahl des Abzweigs der Suche vom Startgang aus wird die Gewichtung der Koordinaten geändert, da die x-Koordinate des zuerst gewählten Abzweigs in Punkt 1 mit der des zuerst gewählten Abzweigs von seiten der Wegsuche vom Zielgang aus übereinstimmen. Für beide Wegsuchen wird bei Erreichen der an Punkt 1 angrenzenden Gänge die Gewichtung der x- und y-Koordinaten vertauscht. Alle weiteren Berechnungen fallen in das Gebiet der Wegoptimierung und werden daher im Kapitel 5 näher erläutert.

4.5 Generierung der Kommandolisten

Eine Umgebung, in der sich der autonome mobile Roboter MORIA bewegen soll, läßt sich bis zu einem bestimmten Grad in ein System aus Gängen überführen. Sowohl die strukturelle Repräsentation der Umgebungsdaten in der internen Karte als auch die Generierung von abzuarbeitenden Fahrkommandos basiert auf einer Überführung des betreffenden Gangsystems in einen verbundenen gerichteten oder ungerichteten Graphen. Diese wurde bezüglich der Umgebungsdaten bereits im Kapitel 4.1 erläutert. Ziel der Weggenerierung ist es weiterhin, für die Übergänge von einem zum nächsten zu befahrenden Gang der entsprechenden Rekursionsebenen jeweils ein auszuführendes Kommando zu berechnen und dieses in eine Liste zu übernehmen. Welche globalen Kommandos zur Ausführung an den lokalen Fuzzycontroller übergeben werden können, wird aus Tabelle 4.1 ersichtlich.

Tabelle 4.1: Ausführbare globale Kommandos

Kommando	Beschreibung
0	Stop
1	Geradeaus
2	Nächster Abzweig links
3	Nächster Abzweig rechts
4	Rückwärts
5	90 Grad links (Handsteuerung)
6	90 Grad rechts (Handsteuerung)
7	Vorwärts
8	Umschaltung Motor- und Sensorwerte (Fahrtrichtung)
9	Letzter Abzweig links
10	Letzter Abzweig rechts

5 Wegoptimierung

Die Wegsuche mittels Koordinatenabgleich der x- bzw. y-Koordinate führt, wie im vorhergehenden Kapitel deutlich wurde, zwar in vielen jedoch nicht in allen Fällen zur Generierung eines optimalen Weges. Die Anzahl der zu fahrenden Abbiegungen innerhalb des generierten Weges nimmt entscheidenden Einfluß auf die Fahrzeit, die für das Zurücklegen eines berechneten Weges aufgewendet werden muß. Handelt es sich bei dem nächsten zu fahrenden Abzweig beispielsweise um eine Rechtsabbiegung, die im Winkel von +90 Grad zur aktuellen Fahrtrichtung liegt, und fuhr der Roboter im aktuellen Gang mit Höchstgeschwindigkeit, so muß bei einem entsprechend schmalen Abzeig die Geschwindigkeit analog zum Autofahren vom Navigator zurückgenommen werden.
Hierzu ein Beispiel:

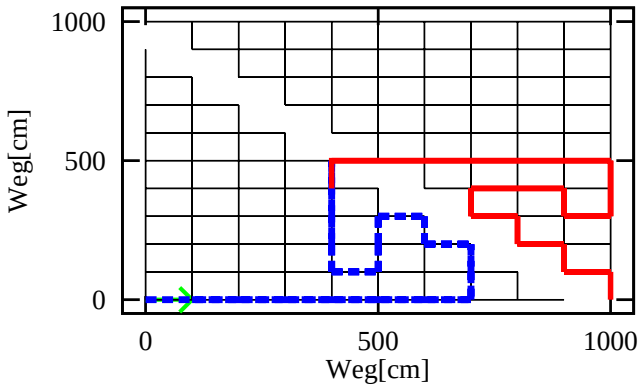


Bild 5.1: Generierter Weg über Koordinatenabgleich

Bei dem generierten Weg fällt auf, daß sowohl eine längere Strecke als auch mehr Abbiegungen als eigentlich notwendig befahren werden müssen. Den optimalen zu fahrenden Weg zeigt Bild 5.2.

Die aufzuwendende Fahrzeit wird bei diesem Weg sicherlich geringer sein als im ersten Fall. Um eine Optimierung des ersten generierten Weges durchzuführen, ist es notwendig, bestimmte mögliche Gelenkgänge festzulegen, die bei der Optimierung auf jeden Fall durchlaufen werden müssen. Da die bei der Kartierung ermittelte Umgebung in ein System von Gängen aufgeteilt wird, werden diese Stützpunkte im folgenden als Stützgänge bezeichnet. Betrachtet man die gezeigte Umgebung, so fällt auf, daß der Verbindungsgang zwischen der unteren und

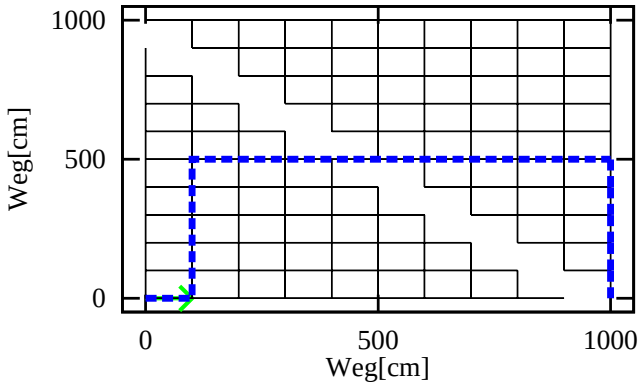


Bild 5.2: Optimierter Weg

oberen dreieckförmigen Hälfte des Gangsystems ein besonders zu beachtender Gang bezüglich der Wegoptimierung darstellt. Erfolgt eine Wegsuche vom Startgang aus über diesen Verbindungsgang zum Zielgang hin, und erfolgt diese Suche mittels des im vorhergehenden Kapitel beschriebenen Suchalgorithmus, so wird schnell deutlich, daß man den in Bild 5.2 gezeigten Weg erhält. Die Bestimmung des erwähnten möglichen Gelenkganges ist jedoch während des ersten Durchlaufs der Wegsuche nicht möglich, da die Entscheidung, welcher Abzweig vom Gang der aktuellen Rekursionsebene als nächstes in den zu generierenden Weg aufgenommen werden soll, nur eine Ebene im voraus getroffen werden kann. Aus diesem Grund wird eine post-mortem-Analyse nach Berechnung des ersten Weges angewendet. Um bei dieser post-mortem-Analyse die geforderten Stützgänge zu ermitteln, ist es zunächst notwendig, die dafür erforderlichen Kriterien festzulegen.
Diese Kriterien für die Wegoptimierung sind:

- Änderung der Winkelsumme im berechneten Weg vom Startgang bzw. Zielgang aus und
- Änderung des quadratischen Abstands im berechneten Weg vom Gang der aktuellen Rekursionsebene zum Startgang bzw. Zielgang

5.1 Winkelsumme

Die strukturelle Repräsentation einer bekannten Umgebung umfaßt u.a. die Information, in welcher Orientierung die Abzweige eines Ganges liegen. Bildet man nun den Betrag der Differenz zwischen der Orientierung im Gang der aktuellen Rekursionsebene (hierbei handelt es sich um die Orientierung, in der der Gang später durchfahren werden soll) und der zu erreichenden Orientierung nach

Abbiegen in den folgenden Abzweig, so erhält man eine Winkeldifferenz. Diese Differenz wird während der Wegsuche für jeden Gang innerhalb des generierten Weges gebildet. Diese Orientierungsänderungen werden aufsummiert und die aktuelle Winkelsumme dem entsprechenden Gang zugeordnet. Im folgenden wird für das zuletzt genannte Beispiel aus Bild 5.2 das Kriterium Winkelsummenänderung erläutert. Bild 5.3 zeigt die Winkelsumme für den nicht optimierten Weg als Funktion der aktuellen Gängen vom Startgang der Wegsuche aus.

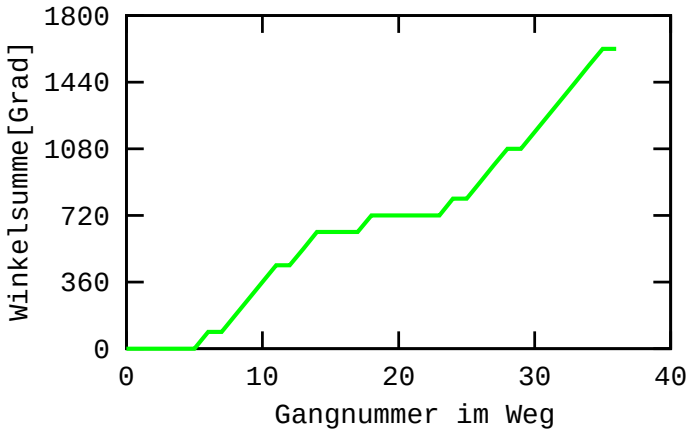


Bild 5.3: Die Winkelsumme für den Weg aus Bild 5.1

Bei näherem Betrachten fällt auf, daß es sich hierbei um eine Art Treppenfunktion handelt. Die interessanten Punkte für die Wegoptimierung sind die Gänge, bei denen die Ableitung der dargestellten Funktion gleich oder annähernd null ist. Dies bedeutet, daß innerhalb des generierten Weges gerade Strecken vorhanden sind, bei denen sich die gebildete Winkelsumme nicht oder nur in geringem Maße ändert. Existieren solche geraden Strecken innerhalb des errechneten Weges , so werden der Gang vor Erreichen der betreffenden Strecke und der Gang direkt danach als Stützgang in die Wegoptimierung miteinbezogen. Durch diese Maßnahme lassen sich insbesondere Gänge ermitteln, von denen aus der Zielgang über direkte weitgehend gerade Strecken erreicht werden kann.

5.2 Quadratischer Abstand

Als zweiter Parameter für die Optimierung des ersten generierten Weges wird der quadratische Abstand zwischen Startgang bzw. Zielgang und den restlichen Gängen des errechneten Weges vom Ziel- bzw. Startgang aus für jeden einzelnen Gang ermittelt. Als Referenzpunkte dienen hierbei die Gangmittelpunkte, die sich über die x- bzw. y-Koordinaten der jeweiligen Gangendpunkte berechnen lassen. Der quadratische Abstand läßt sich somit nach folgender Formel ermitteln:

$$QA(\vec{x},\vec{y}) = \sqrt[2]{(x_s - x_z)^2 + (y_s - y_z)^2} \tag{5.1}$$

mit

$$x_s = \frac{|x_{1s}| + |x_{2s}|}{2} \tag{5.2}$$

$$y_s = \frac{|y_{1s}| + |y_{2s}|}{2} \tag{5.3}$$

$$x_z = \frac{|x_{1z}| + |x_{2z}|}{2} \tag{5.4}$$

$$y_z = \frac{|y_{1z}| + |y_{2z}|}{2} \tag{5.5}$$

x_{1s}, x_{2s}, y_{1s} und y_{2s} bezeichnen dabei die jeweiligen Koordinaten der Gangendpunkte eines Ganges im Start- bzw. Endweg.

Die Stützpunkte werden durch die Bildung der Ableitung der Funktion für den quadratischen Abstand ermittelt. Entscheidend sind hierbei die relativen Extrempunkte, d.h. die Ableitung der Funktion nimmt den Wert Null an. Ein typisches Beispiel für einen solchen Extrempunkt bezüglich des quadratischen Abstands zwischen dem aktuellen Gang innerhalb des Weges und dem Startgang ist der Gelenkgang aus der letztgenannten Beispielumgebung. Der quadratische Abstand nimmt zunächst linear zu, bis die Wegsuche in der unten rechts liegenden Sackgasse mündet. Bis zu dem erwähnten Gelenkgang nimmt er ab, hinter dem Gelenkgang wiederum zu. Als Stützpunkte werden wiederum der Gang vor bzw. nach dem betreffenden Gang gewählt, bei dem das relative Extremum festgestellt wurde.

5.3 Verwendung der Stützgänge und möglichen Gelenkgänge

Sämtliche nach den obigen Kriterien ermittelten Stützgänge werden ausgehend vom Startgang in aufsteigender Reihenfolge in einer einfach verketteten Liste gespeichert. Hierbei ist gewährleistet, daß bei Ermittlung eines Stützganges durch das Kriterium Winkelsumme dieser nicht erneut in die Liste aufgenommen wird, falls das Kriterium Quadratischer Abstand ebenfalls für diesen Gang greift. Start- und Zielgang werden jeweils als erster bzw. letzter in die Liste aufgenommen.

Ziel der Wegoptimierung ist es, einen optimierten Weg aus Teilwegen zusammenzusetzen. Diese Teilwege werden berechnet, indem als Zielgang für die

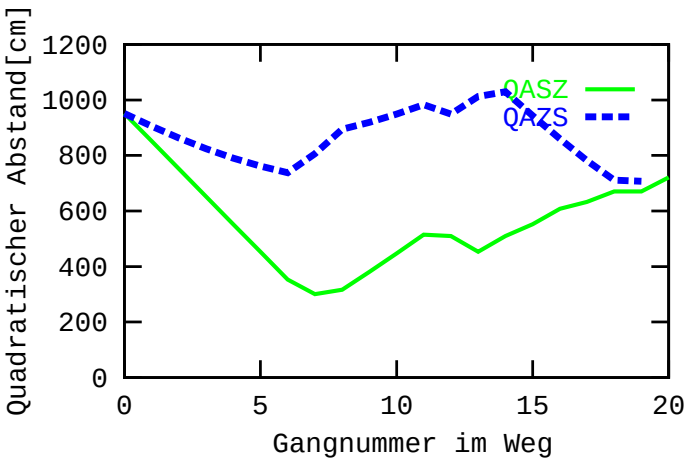


Bild 5.4: Der quadratische Abstand für den Weg aus Bild 5.1

Weggenerierung die Stützpunkte vom Ende der Liste an in rückwärtiger Richtung gesetzt werden. Die Liste wird nun von hinten nach vorner durchlaufen und in jedem Schritt ein Teilweg mittels des im Kapitel 4 erläuterten Verfahrens berechnet. Diese Berechnung erfolgt solange, bis ein bestimmter Grenzwert der Winkelsumme des aktuellen Weges unterschritten wird oder der dem aktuellen Startgang in der Liste nachstehende Stützgang erreicht wird. Der Grenzwert der Winkelsumme ist auf einen Wert von 90 Grad festgelegt. Ist der Teilweg generiert, so wird der Stützpunkt, welcher als aktueller Zielgang gesetzt ist, als neuer Startgang für die Berechnung des nächsten Teilweges gewählt. Nun erfolgt wiederum die Abarbeitung der Stützpunktliste sukzessive von hinten nach vorne, indem der aktuelle Stützgang als Zielgang gesetzt wird. Diese Abarbeitung wird solange wiederholt, bis entweder die Winkelsumme eines Teilweges vom aktuellen Startgang zum eigentlichen Zielgang den geforderten Grenzwert unterschreitet, oder zwischen dem aktuellen Startgang und dem eigentlichen Zielgang kein weiterer Stützgang in der Liste vorhanden ist.

5.4 Fahren nach Kommandoliste

Im Rahmen dieser Arbeit erfolgt die strukturelle Repräsentation einer Umgebung anhand einer Überführung in ein System aus Gängen. (siehe auch Abschnitt 4.1 bzw. Abschnitt 4.5). Diese Überführung läßt sich durchführen, insofern keine größeren leeren Räume in der Umgebung vorhanden sind, deren Außenmaße den sichtbaren Bereich des Roboters übersteigen. Im folgenden wird vorausgesetzt, daß eine solche strukturelle Repräsentation einer Umgebung dirch ein System aus Gängen möglich ist. Ziel der Wegsuche bzw. Wegoptimierung ist es, einen möglichst optimalen Weg vom definierten Start- zum Endgang zu finden und aus

den Daten der Karte eine Kommandoliste zu generieren. Diese Kommandoliste muß anhand der Informationen des Navigators korrekt zur Ausführung gebracht werden. Inkonsistenzen zwischen der aktuellen Position des Roboters und der internen Position des Planers müssen unter allen Umständen vermieden werden, da die weitere Ausführung der Kommandoliste den Roboter sonst an eine falsche Position dirigiert. Ein besonderes Problem stellt dabei die Variabilität einer einmal kartierten Umgebung dar. Ein befahrbarer Gang kann bei einer erneuten Durchfahrt durch einen Gegenstand verstellt sein und wird daher als Sackgasse registriert. Solche versperrten Gänge werden nicht in die interne Karte aufgenommen. In der aktuellen Version des Planers wird davon ausgegangen, daß bei der ersten Kartierungsfahrt alle wichtigen Transportwege befahrbar gehalten werden und diese somit in der Karte aufgeführt sind. Daher wird eine unplanmäßig auftretende Sackgasse auch nicht dementsprechend in der Karte vermerkt, sondern es erfolgt an dieser Stelle nur eine erneute Wegberechnung vom aktuellen Standpunkt aus zum Zielgang. Existiert kein solcher Weg, d.h. die unvermutet aufgetretene Sackgasse muß für das Erreichen des Zielganges befahren werden, so stoppt der Roboter an der aktuellen Position. Grundsätzlich geht der Planer also von der Befahrbarkeit der in der internen Karte repräsentierten Gänge aus.

Um eine korrekte Abarbeitung der für einen Weg berechneten Kommandoliste zu gewährleisten, werden die im folgenden, vom Navigator gestellten Daten verwendet.

- Erkannter Rechts- und/oder Linksabzweig
- Aktuelle Orientierung
- Gefahrene Strecke seit Erkennung des letzten Abzweigs

Die Kriterien sind, eine gegenüber der Kartierungsfahrt unveränderte Umgebung vorausgesetzt, in der Reihenfolge ihrer Gewichtung aufgeführt. Oberste Priorität für die Abarbeitung des nächsten Kommandos hat das Erkennen eines Abzweigs. Der nächste Schritt besteht in einer Plausibilitätsüberprüfung des betreffenden Abzweigs. Besteht die zu befahrende Umgebung z.B. aus einem 100 Meter langen Gang, der von der Startposition des Roboters aus gesehen zunächst einen Abzweig nach rechts und nach weiteren 20 Metern einen weiteren Abzweig nach rechts hat, so wird diese Umgebung durch ein System aus fünf Gängen dargestellt (siehe Abschnitt 4.1). Soll der Roboter in den zweitenen Rechtsabzweig einfahren, so berechnet der Planer eine Kommandoliste, die aus den Kommandos 1 für ”Geradeaus”, 3 für ”Nächster Abzweig rechts” und 0 für ”STOP” besteht. Wird vor der ersten Kreuzung jedoch ein neuer Rechtsabzweig erkannt, so ist dieser nicht in der aktuellen Karte vermerkt. Das aktuelle Kommando ”Geradeaus” ist nach dem Rücksetzen der Zustandsvariable RBR (Rechtsabzweig) abgearbeitet. Wird keine Überprüfung auf Plausibilität durchgeführt, so wird das folgende

Kommando ”Nächster Abzweig rechts” gesetzt. Da der nächste in Fahrtrichtung liegende Rechtsabzweig jedoch um der ersten der beiden Kreuzungen liegt, biegt der Roboter an dieser Stelle einen Abzweig zu früh nach rechts ein und erreicht somit eine falsche Zielposition. Eine erneute Wegsuche geht jedoch von der eigentlichen Zielposition aus und muß den Roboter zwangsläufig zu einer anderen als der vorgesehenen Position führen.

Als zweites Kriterium wird die Orientierung des Abzweigs herangezogen. Dabei handelt es sich um die Orientierung, in der sich der Roboter nach Einbiegen in den Abzweig weiterbewegt. Stimmt die Orientierung des ersten in der Karte vorhandenen Rechtsabzweigs mit der des zweiten überein, so ist jedoch noch nicht zu erkennen, daß sich der Roboter in einem falschen Abzweig befindet. Aus diesem Grund wird an dieser Stelle das dritte Kriterium herangezogen. Stimmt die befahrene Strecke vom Startpunkt bis zur Erkennung des ersten Rechtsabzweigs nicht mit der in der Karte vermerkten Ganglänge des ersten Ganges überein, so wird das aktuelle Kommando erneut gesetzt. An der betreffenden Stelle muß die Karte aktualisiert werden. Dabei wird der erste befahrene Gang in zwei Gänge mit den entsprechend zu berechnenden Parametern Ganglänge und -breite bzw. den vorhandenen Abzweigen aufgeteilt. Die erneute Ausführung des aktuellen Kommandos ist jedoch nur durchführbar, da es sich dabei um das Kommando ”Geradeaus” handelt. Ist jedoch das Kommando ”Nächster Abzweig rechts” gesetzt, so muß der Roboter zunächst in diesen Abzweig einfahren, um dann durch Setzen des Kommandos ”Letzter Abzweig links” erneut in den ursprünglich zu befahrenden Gang einzufahren. Die befahrene Strecke seit dem letzten erkannten Abzweig ist mit Fehlern behaftet, da sie sich z.B. bei Auftreten von zu umfahrenden Hindernissen vergrößern kann. Daher wird im Rahmen dieser Arbeit das dritte Kriterium nur unscharf betrachtet, d.h. die befahrene Strecke seit Erkennen des letzten Abzweigs kann in einem Bereich von $\pm 1,5$ Metern schwanken. Bewegt sich der Parameter in dem genannten Bereich, so wird die Plausibilitätsüberprüfung mit einem positiven Ergebnis abgeschlossen. Das bedeutet, daß das entsprechende Kommando korrekt abgearbeitet wurde. Die Installation einer Kamera ermöglicht an dieser Stelle einen weitaus besseren Positionsabgleich und kann daher auch auftretende Fehler bei der Wegmessung besser abgleichen. Abschließend zur Wegoptimierung wird an dieser Stelle die generierte Kommandoliste für den optimierten Weg aus Bild 5.2 gezeigt. Sie beinhaltet die Kommandos ”fahre nächsten Abzweig links”, 4 mal ”fahre geradeaus”, ”fahre nächsten Abzweig rechts”, 8 mal ”fahre geradeaus”, ”fahre nächsten Abzweig rechts” und 4 mal ”fahre geradeaus”.

6 Der Kartierungsalgorithmus

Ein weiterer Teil dieser Arbeit besteht in der Aufgabe, eine Karte der Umgebung mit allen notwendigen topologischen Informationen zu erstellen. Die Ausgangssituation, in der sich das autonome mobile Robotersystem MORIA normalerweise befindet, beinhaltet die Unkenntnis sämtlicher Umgebungsdaten. Diese Ausgangssituation erfordert zunächst eine Analyse der Frage, wie die zu erstellende Karte im Planungsmodul repräsentiert werden soll. Wie schon in der Einleitung und im Kapitel 2 angesprochen, basieren die traditionell erhältlichen Planungsalgorithmen auf einer präzisen geometrischen Umgebungskarte mit ihren bereits angesprochenen Nachteilen. Im Abschnitt 4.2 wird ersichtlich, daß die Kartierung des in dieser Arbeit implementierten Planungsmoduls auf einer strukturellen Repräsentation der Umgebungsdaten in Anlehnung an die Graphentheorie basiert. Diese Tatsache ist notwendigerweise auch als Parameter für die Entwicklung eines Kartierungsverfahrens von entscheidender Bedeutung. Ziel einer Kartierungsfahrt ist es somit, anhand der vom Navigator gestellten Daten, wie z.B. erkannten Abzweigen, befahrener Ganglänge seit letztem Abzweig und mittlere Gangbreite, die zu befahrende Umgebung in die Struktur eines Graphen zu überführen. Im Bild 6.1 sind einige der wichtigsten typischen Problemstellungen der Kartierung aufgeführt.

Das Bild zeigt einige wichtige Fahrsituationen, die mehrere zu behandelnde Kriterien der Kartierung beinhalten. Diese sind u.a.

- Ermittlung der Koordinaten beider Gangendpunkte. Hierbei ist von besonderem Interesse, wo dieser Gangendpunkt festgelegt wird. An Kreuzungen ist es beispielsweise sinnvoll, diesen Endpunkt in die Mitte der Kreuzung zu legen.

- Ermittlung der Ganglänge und -breite des gerade befahrenen Ganges. Hierbei ist zu beachten, daß die Ganglänge mit nicht zu vernachlässigenden Fehlern behaftet ist. Dies liegt hauptsächlich daran, daß der Navigator einen Gang nicht unbedingt auf geradem Weg durchfährt, da z.B. auftretende Hindernisse umfahren werden müssen. Die Ganglänge kann daher bei zwei aufeinander folgenden Durchfahrten eines Ganges schwanken. Für die Gangbreite ist es sinnvoll einen Mittelwert zu errechnen. Bei größeren Schwankungen wird dann der aktuelle Gang beendet und ein neuer in die Karte eingefügt.

- Ermittlung des Gangendes und der Orientierung, an bzw. in der ein Abzweig

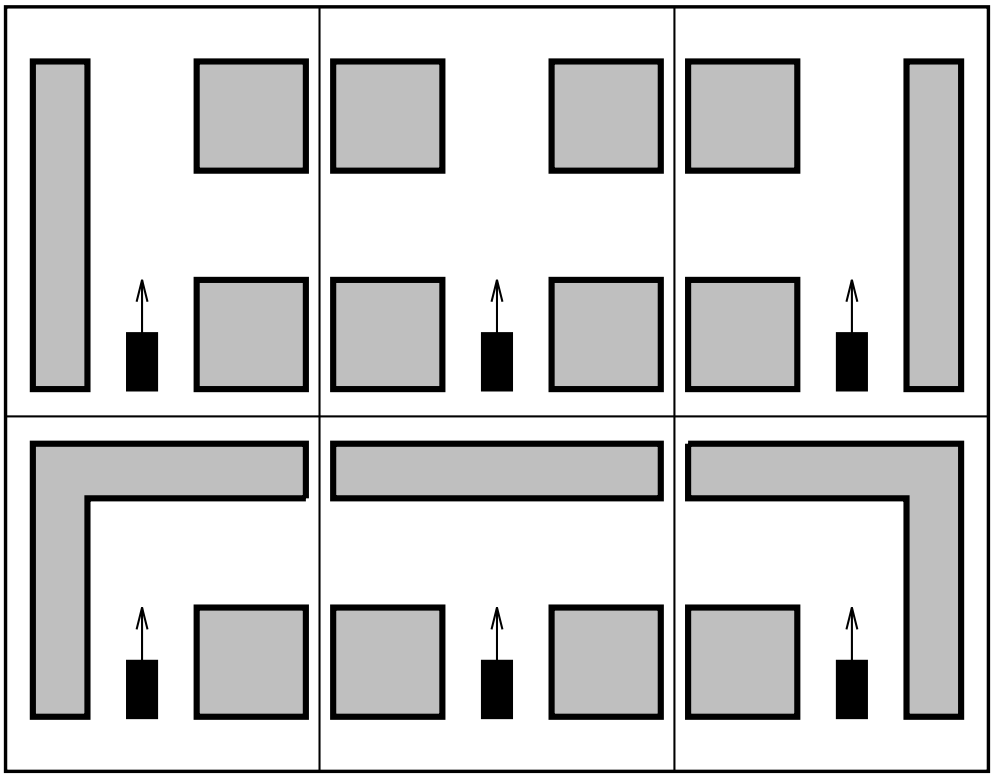


Bild 6.1: Typische Problemstellungen bei der Kartierung

des aktuellen befahrenen Ganges liegt. Diese kann einfach ermittelt werden, indem die Orientierung über die letzten aufgetretenen Werte gemittelt wird.

- Ermittlung der Orientierung, die sich ergibt, wenn ein Abzweig befahren wird. Um diese Orientierung zu ermitteln, ist es notwendig, den betreffenden Abzweig solange zu befahren, bis sich für die gemittelte aktuelle Orientierung ein stabiler Wert ergibt.

Das Bild 6.2 zeigt die Funktionsweise des verwendeten Kartierungsalgorithmus. Standardmäßig wird bei der Kartierung das Kommando geradeaus bei Fahrtrichtung vorwärts gesetzt. Wird ein Rechts- und/oder Linksabzweig erkannt, so erfolgt die Generierung einer für die Erkundung nicht kartierter Abzweige spezifischen Kommandoliste. Die Abarbeitung dieser Kommandoliste soll es ermöglichen, die Orientierung zu ermitteln, in der der Roboter in dem bzw. den betreffenden Abzweig(en) vom aktuellen Gang aus gesehen weiterfährt. Um z.B. eine Kreuzung gemäß Bild 6.1 Mitte oben zu erkunden, ist es notwendig, die Orientierungen der nach rechts, links und oben führenden Abzweige zu ermitteln. Eine Strategie hierbei wäre, direkt nach der Erkennung der Abzweige nach links bzw. rechts das entsprechende Kommando zu setzen, um in den Abzweig einzubiegen. Da die Maximalgeschwindigkeit des Roboters 1.0 m/sec beträgt, ist jedoch ein defensi-

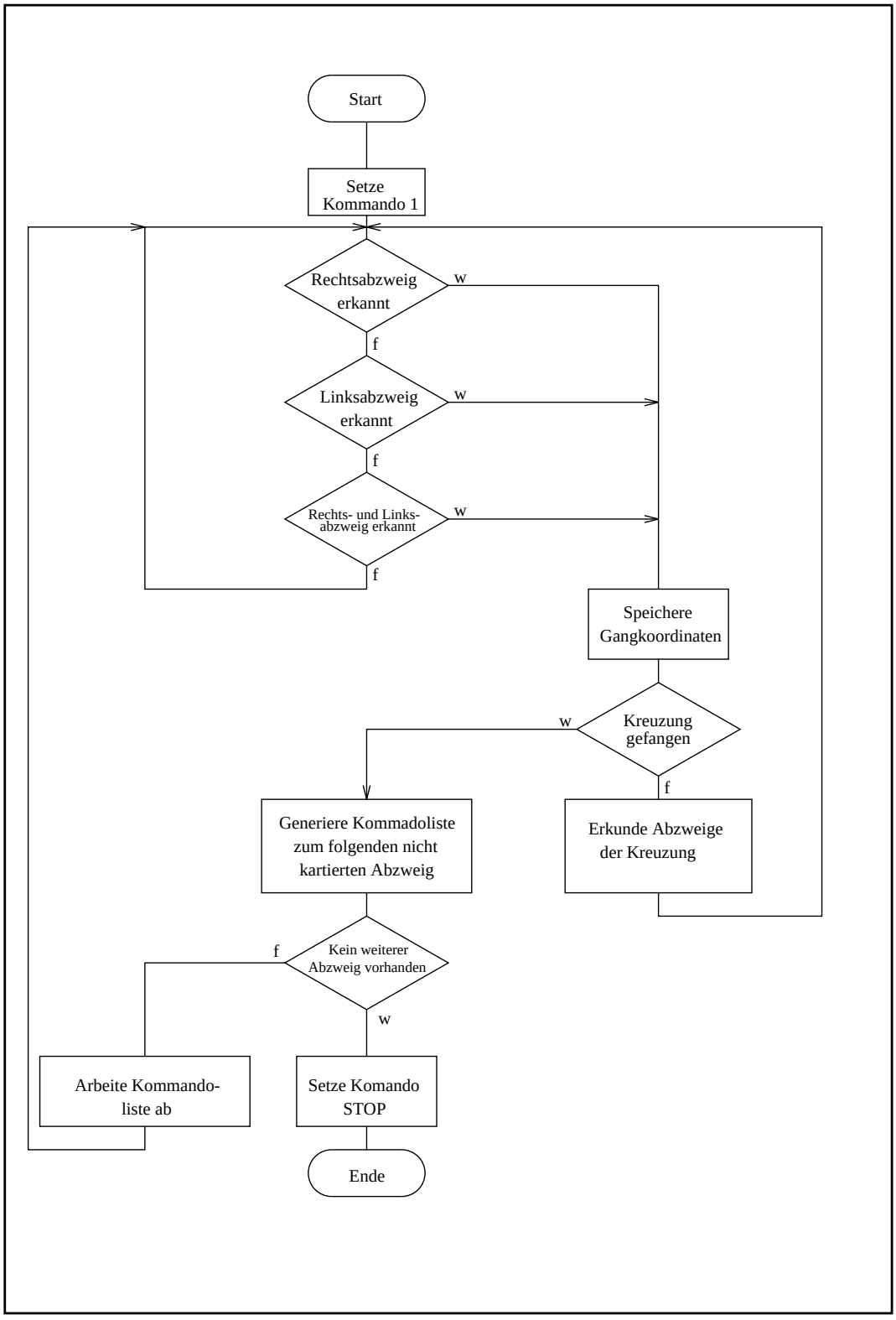


Bild 6.2: Der Kartierungsalgorithmus

veres Fahrverhalten notwendig. Die Begründung dafür liegt in der Schonung der Batterie und der transportierten Gegenstände. Das abrupte Abbremsen belastet die Batterie aufgrund der Trägheit und Masse des Roboters und die transportierten Gegenstände verrutschen u.U. auf der Ladefläche. Aus diesem Grund ist die Ausführung der Kommandos ”nächster Abzweig rechts” bzw. ”nächster Abzweig links” nach der Erkennung eines entsprechenden Abzweigs von seiten des Navigators blockiert. Um es dennoch zu ermöglichen einen erkannten Abzweig zu befahren wurde der Navigator um die beiden globalen Kommandos ”fahre letzten Abzweig links” bzw. ”fahre letzten Abzweig rechts” erweitert. Die Exploration der oben erwähnten Kreuzung erfordert somit eine Kommandoliste, die die nacheinander abzuarbeitenden Kommandos 10,9,9,10 enthält. Der Ablauf der Exploration wird in Bild 6.3,6.4,6.5 und 6.6 verdeutlicht.

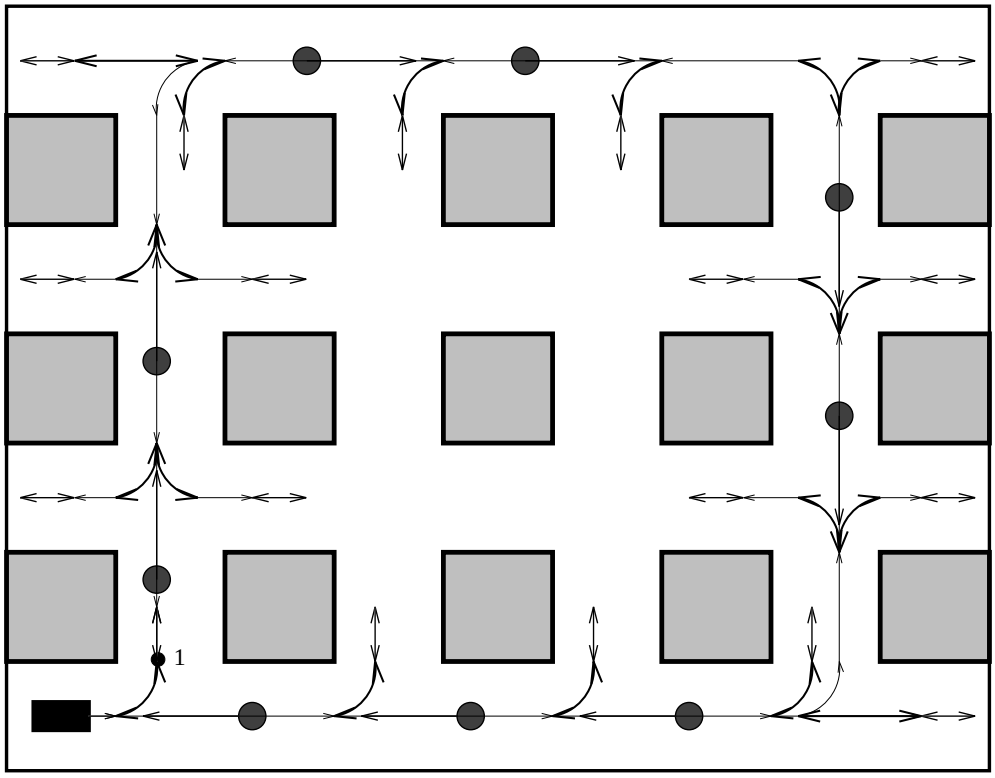


Bild 6.3: Ablauf des Kartierungsalgorithmus 1

Das Bild 6.3 zeigt den Ablauf der Orientierungsfahrt bis zu dem Zeitpunkt, zu dem die in Punkt 1 markierte Stelle der Umgebung erreicht ist. Das Startkommando ist 1 (geradeaus) mit der Fahrtrichtung vorwärts. Es wird zunächst der erste Linksabzweig erkannt. Nachdem die dafür vorgesehene Zustandsvariable zurückgesetzt ist, d.h. der Roboter ist an dem Abzweig vorbeigefahren, wird eine Kommandoliste mit den Kommandos ”fahre letzten Abzweig links” und ”fahre letzten Abzweig rechts” generiert. Die schwarzen Punkte kennzeichnen je-

weils die Stellen, an denen die Abarbeitung einer generierten Kommandoliste erfolgt. Am ersten Punkt wird somit das Kommando ”fahre letzten Abzweig links” zur Ausführung gebracht. Der Navigator setzt nach der erneuten Vorbeifahrt bei rückwärtiger Fahrtrichtung nach erneuter Umschaltung auf vorwärts das Kommando ”fahre nächsten Abzweig links”. Ist das Kommando und die entsprechenden Zustandsvariable TURN für eine Kurvenfahrt nach links gesetzt, und ist desweiteren die Zustandsvariable LBR für einen erkannten Abzweig zurückgesetzt, so erfolgt, sofern sich die erreichte Orientierung stabilisiert hat, die Abarbeitung des nächsten Kommandos. Dieses hat zur Folge, daß der Roboter den letzten rechten Abzweig befährt und sich in seiner ursprünglichen Orientierung weiterbewegt. Dieser Vorgang wiederholt sich solange, bis die untere rechte Ecke des Gangsystems erreicht ist. Wie aus Bild 6.3 ersichtlich ist, fährt MORIA an dieser Stelle in eine Sackgasse ein. Am rechten Ende der Sackgasse wird vom Navigator die Zustandsvariable BLCORRI gesetzt, d.h. es wurde eine Sackgasse registriert. Der Navigator fährt bei Erkennung eines versperrten Ganges bis zum letzten Abzweig zurück und biegt in diesen ein. Befindet sich an dieser Stelle jedoch eine Kreuzung, so wird der erste erkannte Abzweig befahren. Dies ist jedoch nicht unbedingt der korrekte Abzweig in bezug auf die Kartierungsfahrt. Aus diesem Grund greift der Kartierungsalgorithmus an dieser Stelle ein und setzt das entsprechende Kommando (”fahre letzten Abzweig links” bzw. ”fahre letzten Abzweig rechts”), welches bei der weiteren Kartierung das vorteilshafteste ist. Im vorliegenden Fall wird das Kommando ”fahre letzten Abzweig links” gesetzt. Dadurch wird der Roboter in die rechte nach oben führende Gangreihe gesteuert. Die weitere Abarbeitung erfolgt solange, bis eine bereits zuvor befahrene Kreuzung erreicht wird. Bis zu diesem Zeitpunkt wird somit ein Folge von Gängen generiert, wobei jeder Abzweig mit seiner Abbiegeorientierung und seiner Orientierung im jeweiligen Gang vermerkt ist.

Im Punkt 1 wird nun die zuerst befahrene Kreuzung erreicht. Um eine solche Stelle zu registrieren, wurde im Rahmen dieser Arbeit eine koordinatenabhängige Fangfunktion implementiert. Diese Funktion stellt fest, ob die aktuellen Koordinaten in einem Umkreis von drei Metern von einem der bereits erkundeten Gangenden liegen. Bild 6.4 verdeutlicht die weitere Vorgehensweise.

Das Bild zeigt den bereits erkundeten, grau unterlegten Teil der Umgebung. Die aktuelle Position ist durch Punkt 1 beschrieben. Da die Fangfunktion ein positives Resultat geliefert hat, erfolgt an dieser Stelle die Generierung einer Kommandoliste. Diese Kommandoliste besteht aus den Kommandos, die den Roboter zu einem noch nicht erkundeten Abzweig führen sollen. Diese Abzweige sind im Bild durch kleine Doppelpfeile markiert. Von ihnen ist zu diesem Zeitpunkt nur bekannt, an welchem Gangende sie liegen und in welcher Orientierung sie vom jeweiligen Gang abzweigen.

Ein besonderes Problem stellt die Generierung des ersten bzw. der ersten beiden Kommandos dar. Die Fangfunktion erkennt nur, ob die aktuelle Kreuzung

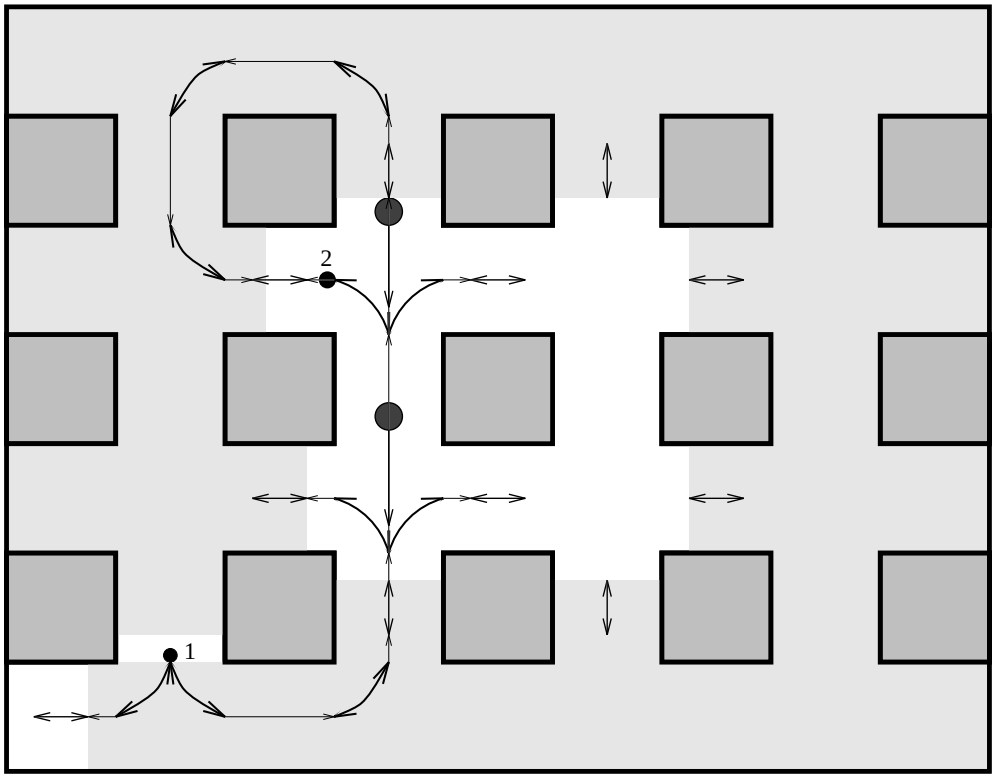


Bild 6.4: Ablauf des Kartierungsalgorithmus 2

an einem Ende eines bereits kartierten Gangs liegt. Die Information, um welches Ende es sich handelt ist jedoch nicht ohne weitere Untersuchungen zu gewinnen.

Um dieses Problem zu lösen, wird jedem kartierten Gang eine Orientierung zugeordnet, die Auskunft über die Richtung gibt, in der er durchfahren wurde. Diese wird im folgenden als Durchfahrtorientierung bezeichnet. Desweiteren werden die Kreuzungen an den Gangenden in der Reihenfolge ihres Auftretens in der Struktur der Karte abgelegt. Liefert die Fangfunktion ein positives Resultat, so lässen sich folgende vier Fälle unterscheiden:

- Die erste in der Struktur der Karte vorhandene Kreuzung wurde erkannt.

  1.) Um den nächsten nicht kartierten Gang zu erreichen, muß der ”gefangene” Gang durchfahren werden. Der Roboter muß daher in der Durchfahrtorientierung des betreffenden Ganges weiterfahren. Da die aktuelle Orientierung des Roboters bekannt ist, kann ohne weiteren Aufwand anhand der beiden Orientierung eine Kommandoliste erzeugt werden, die ihn durch den Gang hindurch führt.

  2.) Um den nächsten nicht kartierten Gang zu erreichen, muß der Roboter von der aktuellen Kreuzung abbiegen. Eine korrekte Abarbeitung der Kommandoliste erfordert nur die Neuberechnung des ersten Kom-

mandos. Dies ist erforderlich, weil als Startposition für die Generierung der Kommandoliste der ”gefangene” Gang deklariert ist. Bei der Berechnung wird ebenfalls der Durchfahrtorientierung verwendet.

- Die zweite in der Struktur der Karte vorhandene Kreuzung wurde erkannt.

  1.) Um den nächsten nicht kartierten Gang zu erreichen, muß der ”gefangene” Gang durchfahren werden. Die Kommandoberechnung erfolgt gemäß 1 mit dem Unterschied, daß nicht entgegengesetzte Wert der Durchfahrtorientierung verwendet wird.

  2.) Um den nächsten nicht kartierten Gang zu erreichen, muß der Roboter von der aktuellen Kreuzung abbiegen. Die Kommandoberechnung erfolgt gemäß 2 mit dem Unterschied, daß zur Berechnung des ersten Kommandos der entgegengesetzte Wert der Durchfahrtorientierung verwendet wird.

Die Ausführung der generierten Kommandoliste führt zunächst zur Exploration des in Fahrtrichtung rechts liegenden Abzweigs. Hierbei handelt es sich um den Startgang, dessen Ganglänge bisher nicht bekannt ist. Danach kehrt der Roboter zu Punkt 1 zurück. Die Fangfunktion liefert erneut ein positives Resultat, und es wird eine Kommandoliste generiert, die den Roboter in Fahrtrichtung links zum nächsten noch nicht kartierten Abzweig führt. Nach der Exploration der beiden nächsten Kreuzungen greift die Fangfunktion am oberen Ende der Umgebung erneut . Die generierte Kommandoliste führt den Roboter zu Punkt 2 in Bild 6.4. Die Kartierung erfolgt nun in der bereits erläuterten Art und Weise, wie es Bild 6.5 und 6.6 noch einmal graphisch darstellt.

Nachdem Punkt 5 erreicht ist, werden noch die letzten offenen Abzweige nach beschriebenem Schema erkundet. Der Algorithmus terminiert, sobald keine weiteren unkartierten Abzweige in der internen Karte vorhanden sind.

Die Karte in Bild 6.7 wurde mit dem Kartierungsmodul erstellt. Man erkennt, daß nur geringe Abweichungen der Gangendkoordinaten auftreten. Innerhalb der Simulationsumgebung treten allerdings nicht so große Fehler in der Wegmessung auf, wi auf dem realen System (Schlupf durch verschiedene Bodenbeläge, etc.).

Die Umgebungskarte in Bild 6.8 wurde durch ein mehrfaches Befahren des gezeigten Gangsystems anhand von Handsteuerkommandos erstellt. Die Wiederholgenauigkeit der ermittelten Gangendkoordinaten ist dabei so gut, daß eine Verwechslung verschiedener Gänge annähernd ausgeschlossen ist. Ist die aktuelle interne Position des Planers in der Umgebung dennoch unsicher, so ist es möglich eine zusatzliche Orientierungsfahrt durchzuführen um die Position abzugleichen.

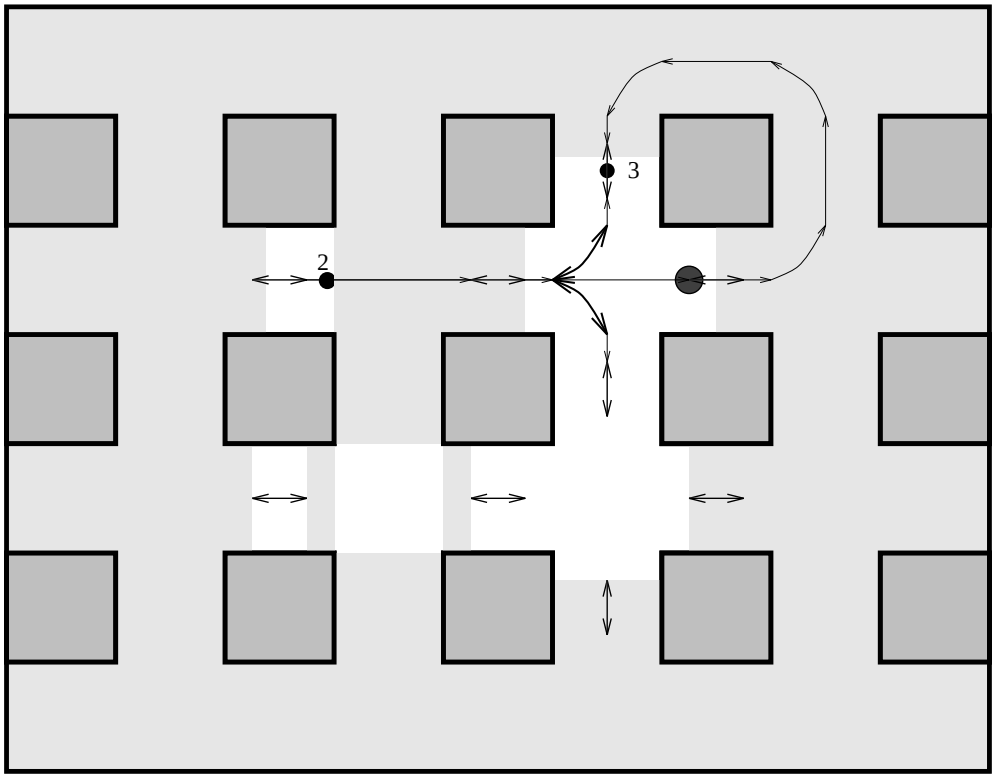


Bild 6.5: Ablauf des Kartierungsalgorithmus 3

6.1 Kartierung von Räumen

Der angewendete Algorithmus läßt sich nicht für Räume anwenden, deren Außenmaße den sichtbaren Bereich des Roboters übersteigen. Eine Erweiterung der Funktionalität des Kartierungsalgorithmus in seiner aktuellen Version war aus Zeitgründen leider nicht mehr möglich. Eine Strategie ist z.B., den Roboter an den Wänden des Raumes entlangfahren zu lassen und diese einzeln zu kartieren. Eine Aufgliederung des Raumes in ein System aus Gängen kann dann nach vollständiger Kartierung der Begrenzungswände vorgenommen werden. Um einen vom Raum abzweigenden Gang korrekt zu erkennen, muß es dann jedoch möglich sein, den Roboter innerhalb des Raumes an der richtigen Stelle zu plazieren. Er muß sich dann an der entsprechenden Wand des Raumes orientieren und an ihr entlangfahren. Der Navigator bietet allerdings in dieser Richtung Erweiterungsmöglichkeiten, die aber in seiner aktuellen Version noch nicht realisiert sind.

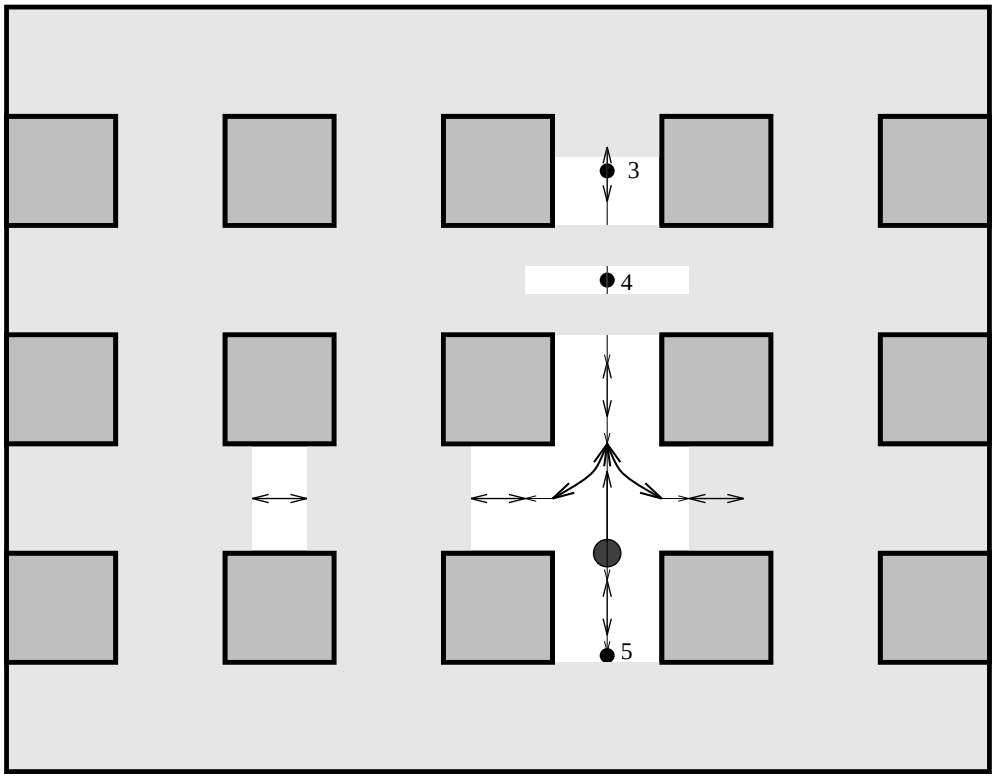


Bild 6.6: Ablauf des Kartierungsalgorithmus 4

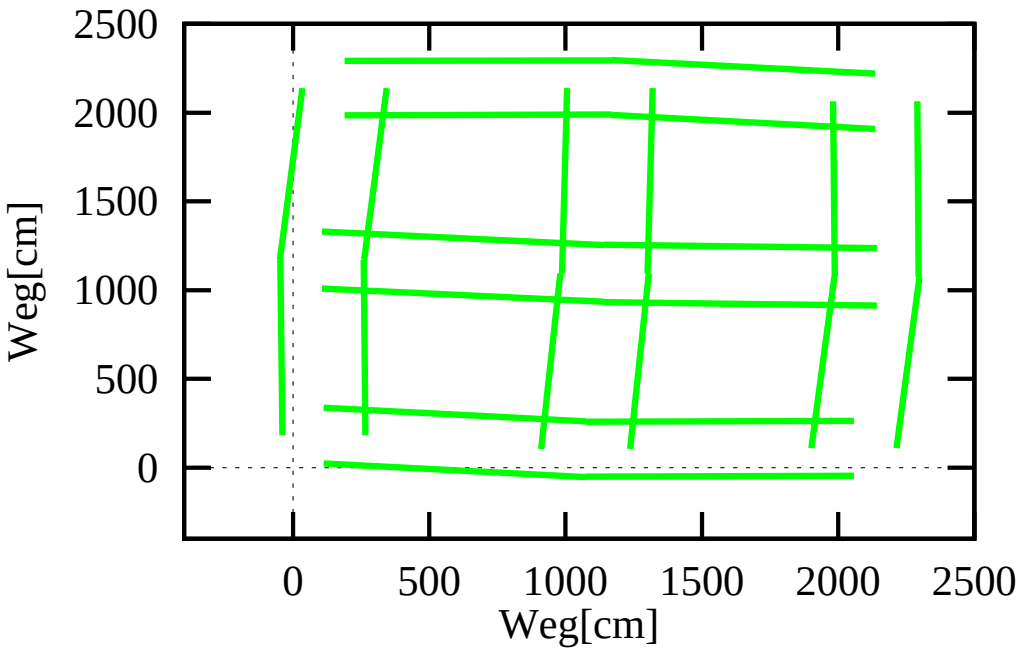


Bild 6.7: Eine generierte Karte bei einfacher Durchfahrt

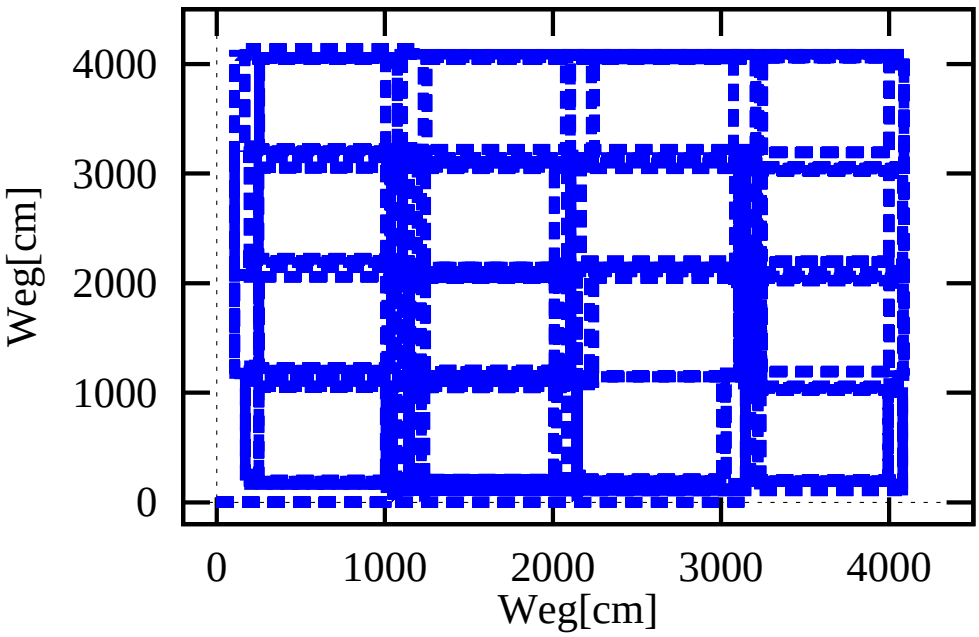


Bild 6.8: Wiederholgenauigkeit bei mehrfacher Durchfahrt

7 Zusammenfassung und Ausblick

Im Rahmen dieser Arbeit wurde ein Planungs- und Kartierungsmodul entwickelt, daß mittels linguistischer Kommandos eine globale Steuerung des autonomen mobilen Roboters MORIA, einem fahrerlosen Transportfahrzeug, ermöglicht. Das Transportfahrzeug ist mit grundlegender interner und externer Sensorik, sowie einem auf Fuzzyregeln basierenden Navigator ausgestattet.

Wesentliche Bestandteile des Planers sind die Überführung topologischer Informationen wie erkannte Abzweige oder Sackgassen in eine Graphenstruktur während der Kartierung und die Generierung von abzuarbeitenden Kommandolisten, die den Roboter zu seiner Zielposition führen. Ein weiterer wichtiger Bestandteil ist die Aktualisierung einer einmal erstellten Umgebungskarte, falls z.B. neue Abzweige (Transportwege) auftreten oder wichtige Transportwege auf Dauer versperrt sind. Diese müssen dann durch eine erneute Kartierungsfahrt aktualisiert werden. Außerdem wird die Güte eines generierten Weges durch eine anschließende Wegoptimierung verbessert, um u.a. die Anzahl zu befahrender Abzweige zu reduzieren.

Erweiterungsmöglichkeiten bieten sich insbesondere im Bereich der Wegsuche an. Es ist denkbar, den Algorithmus um eine lernende Komponente zu erweitern. Hierbei können die in mehreren aufeinanderfolgenden Wegsuchen auftretenden Gänge innerhalb der Umgebung bestimmten Bereichen zugewiesen werden, die durch erkannte Brücken bzw. Gelenkgänge miteinander verbunden sind. Wird bei einer erneuten Wegsuche festgestellt, daß Start- und Zielgang in verschiedenen Bereichen der Umgebung liegen, so kann eine Zuweisung der notwendigen Stützgänge für die Wegoptimierung direkt bei der ersten Weggenerierung vorgenommen und somit ein von vornherein möglichst optimaler Weg berechnet werden. Hierbei spielt allerdings die Größe der zu befahrenden Umgebung eine entscheidende Rolle, ob sich der Aufwand für eine solche Erweiterung überhaupt lohnt und eine normale Wegoptimierung nach beschriebenem Algorithmus nicht schneller zu einem aktzeptablen Ergebnis führt.

Eine Systemerweiterung des Roboters um eine Kamera oder einen Laserscanner und die darauffolgende Kopplung mit dem Planungsmodul bietet die Möglichkeit einer genauen Positionsbestimmung und eines Positionsabgleichs mit der intern geführten Karte des Planers mittels Landmarken. Als Landmarken können dabei z.B. Türschilder oder andere besondere Umgebungsmerkmale verwendet werden. Die Positionsbestimmung alleine durch das Planungsmodul ist nur bis zu einem bestimmten Grad fehlertolerant, daher ist eine entsprechende Erweiterung unumgänglich.

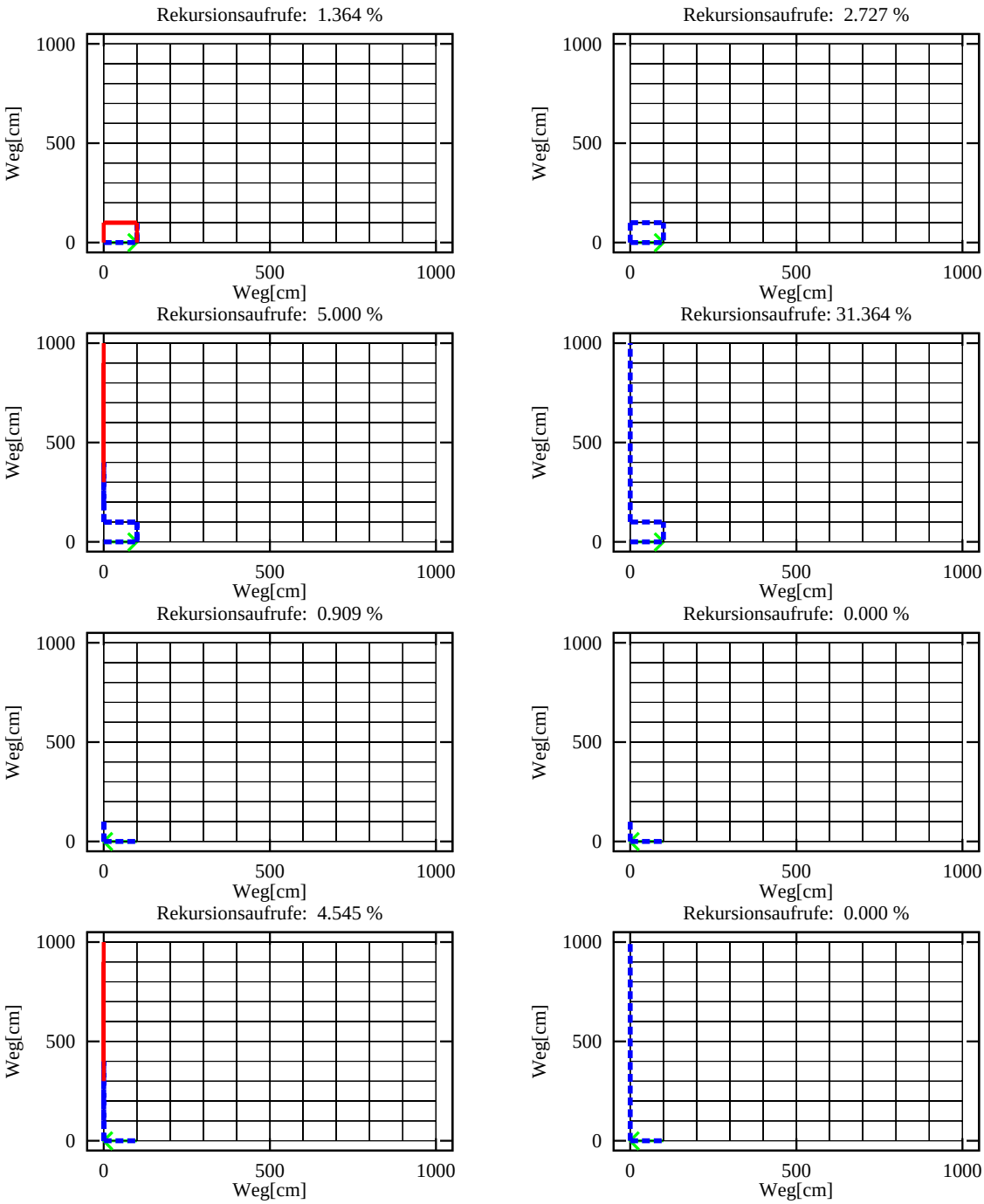
Anwendungsfälle des vorgestellten Systems finden sich in den folgenden Bereichen denkbar:

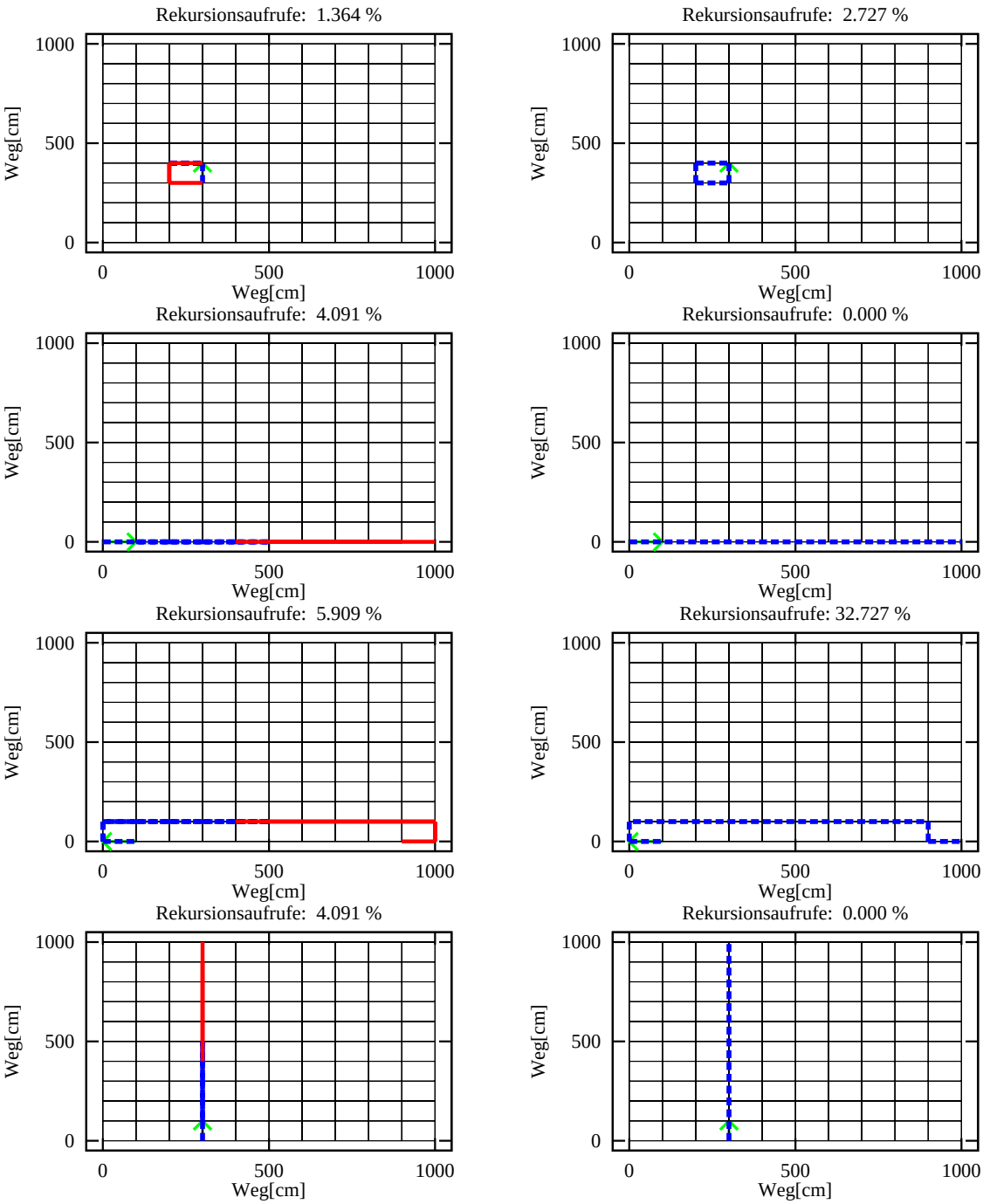
- Übernahme von Transportaufgaben in Werkhallen oder Bürogebäuden (z.B. Postverteilung o.ä.)
- Kranken- und Behindertenfahrstühle
- Reinigungsarbeiten in Gebäuden außerhalb der normalen Arbeitszeiten

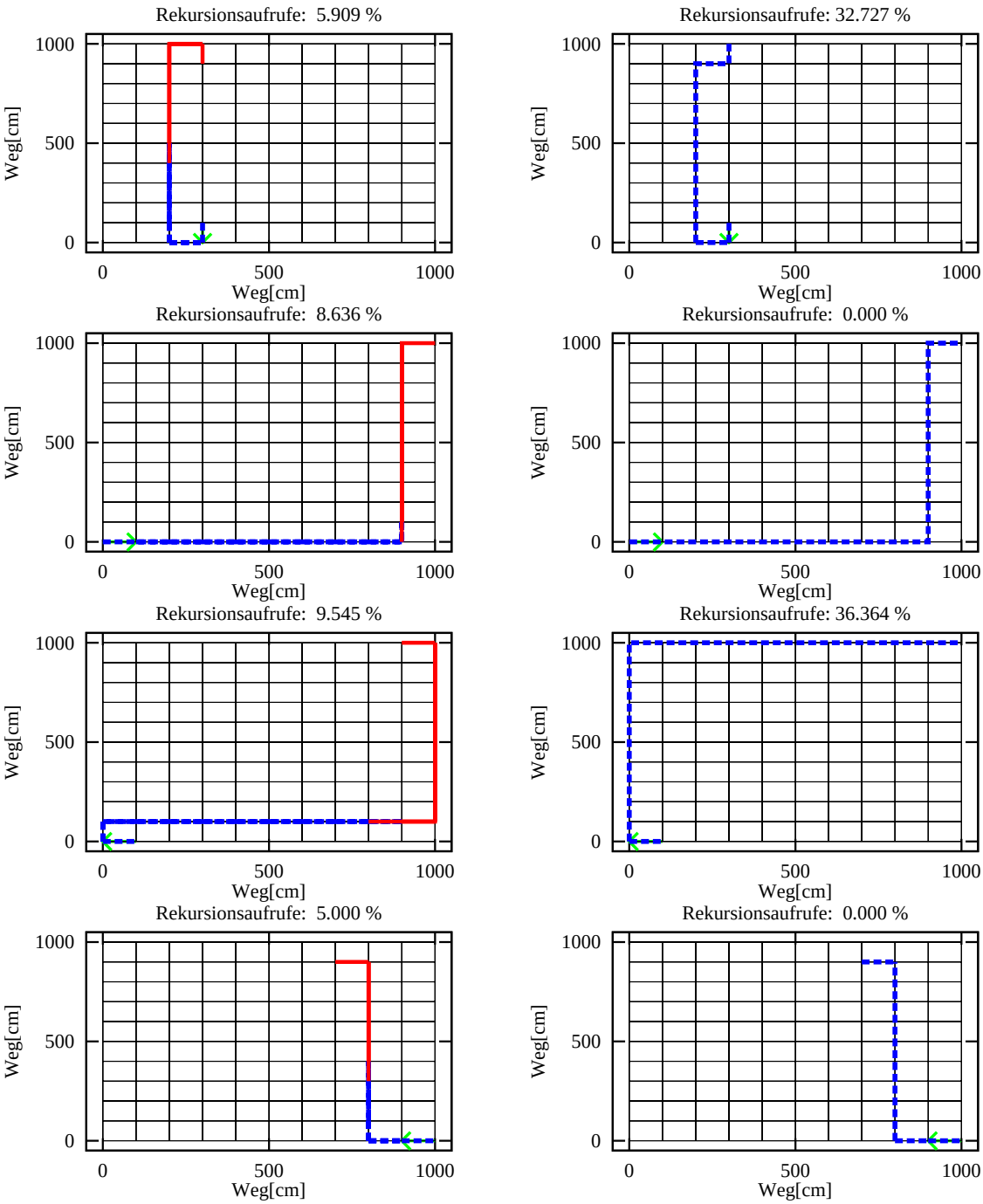
Die Anforderungen an autonome Transportsysteme sind jedoch in bezug auf die Orientierung in einer unbekannten Umgebung sehr hoch. Beispielsweise ist es denkbar, daß sich ein Roboter auch über mehrere Etagen hinweg in seiner Umgebung zurechtfinden und daher Aufzüge bedienen muß. Ob ein solches System in der Zukunft einsetzbar ist, ist sicherlich von der Aktzeptanz am Markt und davon abhängig, wie groß der zusätzlich zu betreibende Aufwand bei Installation eines solchen Systems ist.

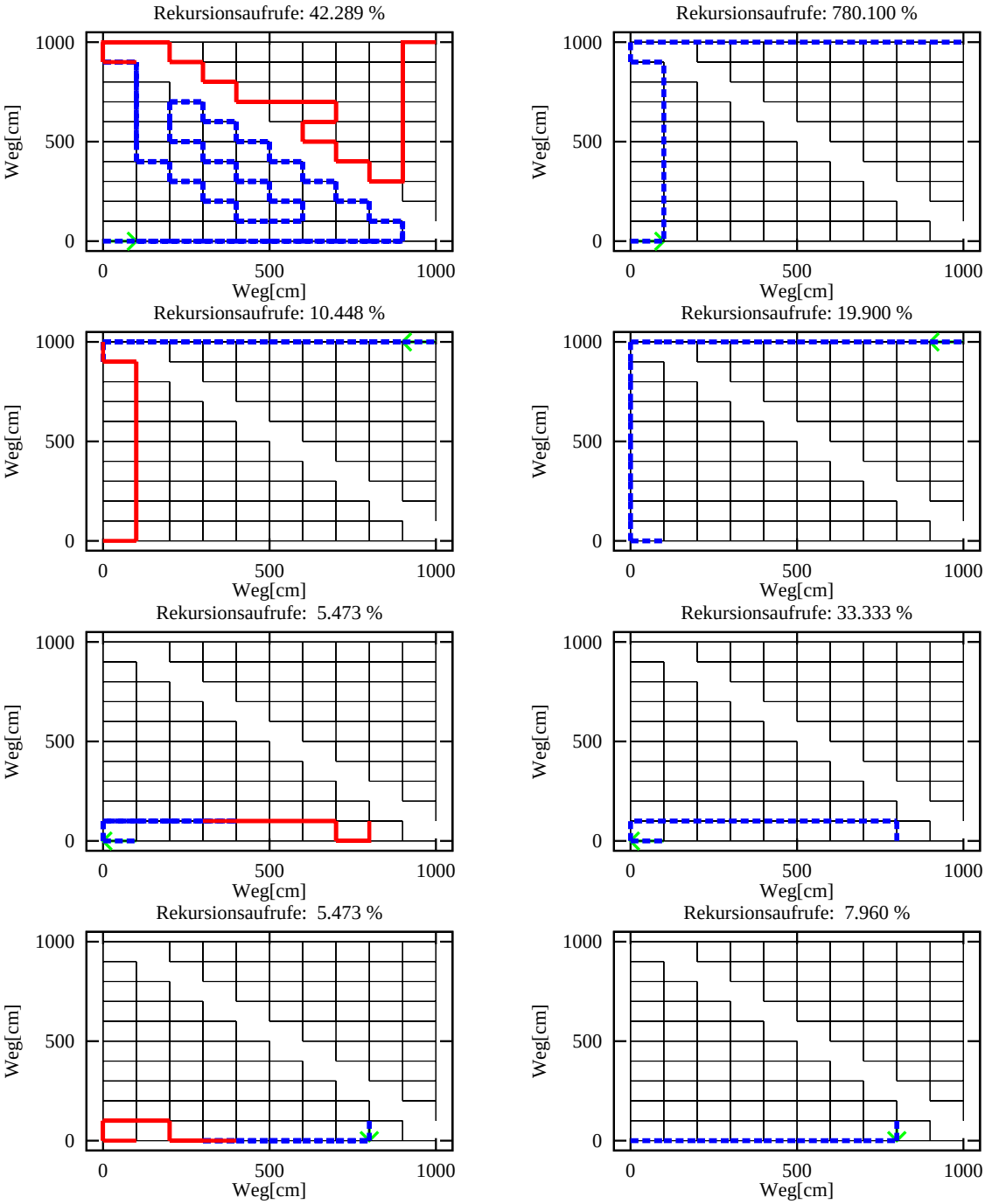
A Beispiele generierter Wege

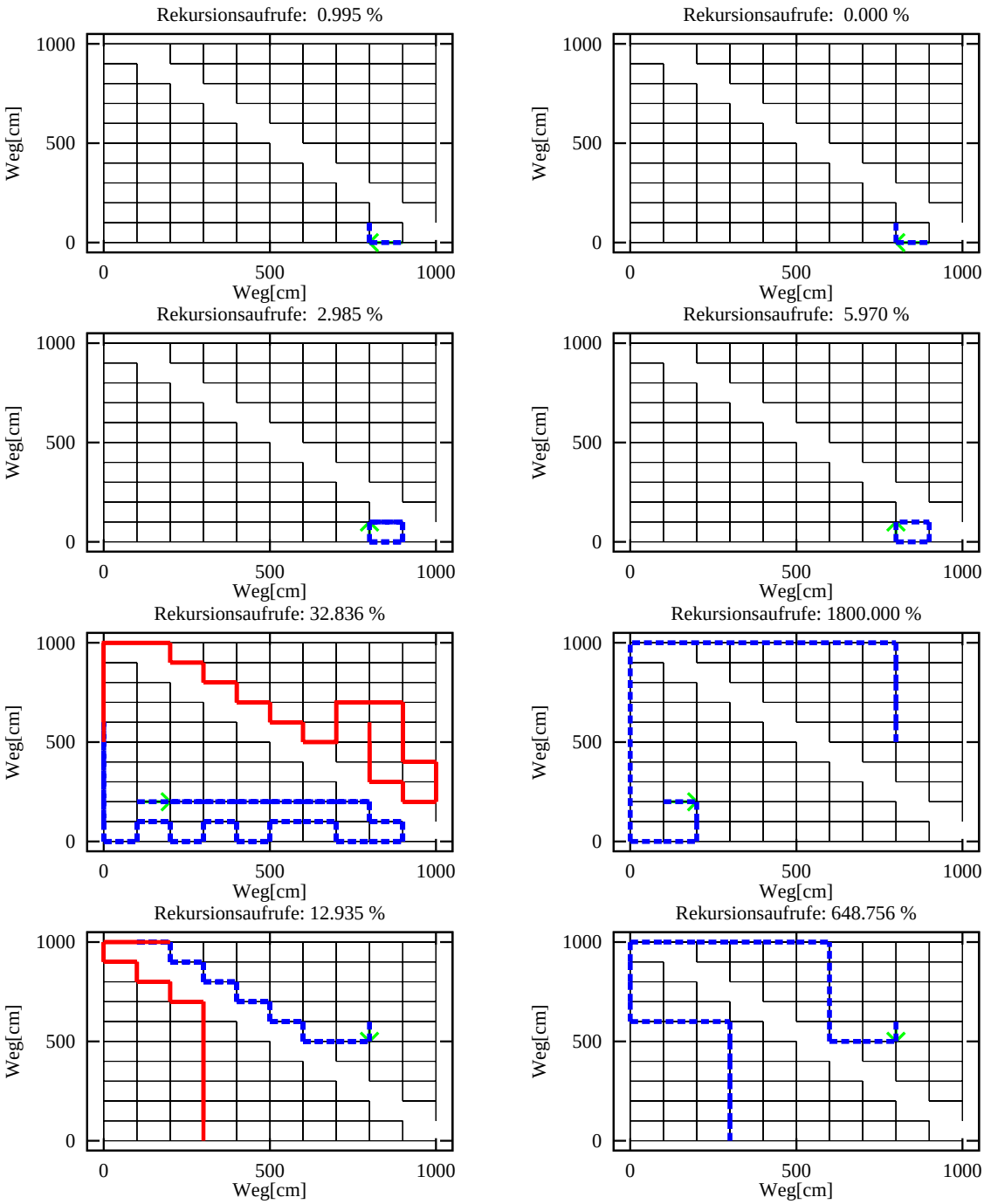
Im folgenden werden generierte Wege aufgeführt, die für fünf verschiedene Umgebungen berechnet wurden. Die Grafiken sind jeweils paarweise zusammengestellt, wobei auf der linken Seite der erste berechnete und auf der rechten der optimierte Weg dargestellt ist. Die Gänge werden durch gestrichelte schwarze Linien dargestellt. Für zusammengesetzte Wege wird der vom Startgang aus generierte Weg in blau, der vom Zielgang aus berechnete in rot eingefärbt. Die Startposition und -orientierung sind durch einen grünen Pfeil gekennzeichnet. Als zusätzliche Information ist über jedem Bild eine Prozentzahl angegeben. Für die nicht optimierten Wege stellt diese Zahl den prozentualen Anteil der durchsuchten an der Gesamtzahl der vorhandenen Gänge dar. Da bei der Optimierung u.U. mehrere Berechnungen von Teilwegen durchgeführt werden, ist der entsprechende kummulierte Wert für die Optimierung ebenfalls oberhalb der jeweiligen Grafik gezeigt.

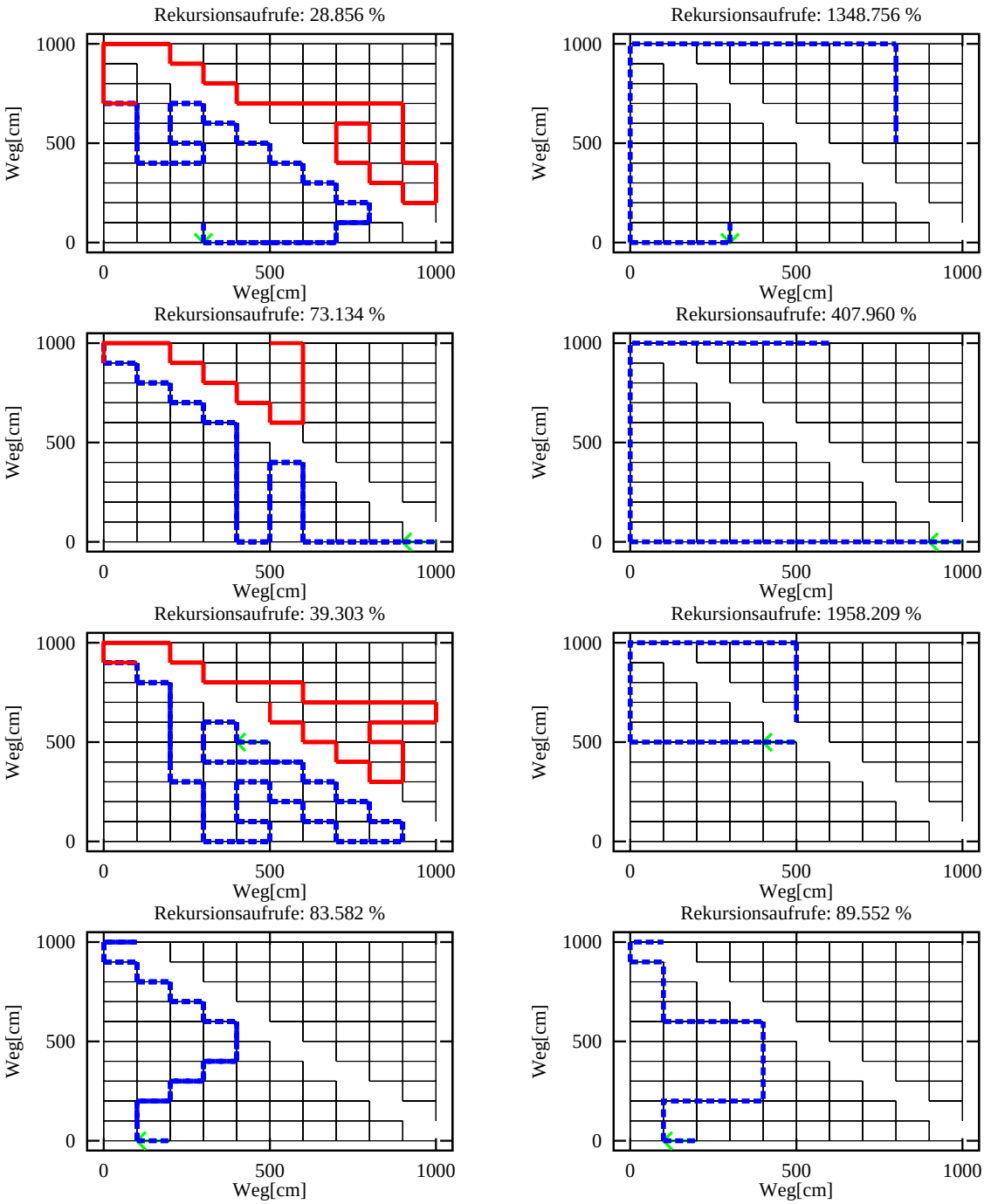

Rekursionsaufrufe: 1.364 %
Rekursionsaufrufe: 2.727 %
Rekursionsaufrufe: 5.000 %
Rekursionsaufrufe: 31.364 %
Rekursionsaufrufe: 0.909 %
Rekursionsaufrufe: 0.000 %
Rekursionsaufrufe: 4.545 %
Rekursionsaufrufe: 0.000 %
Weg[cm]
0
500
1000

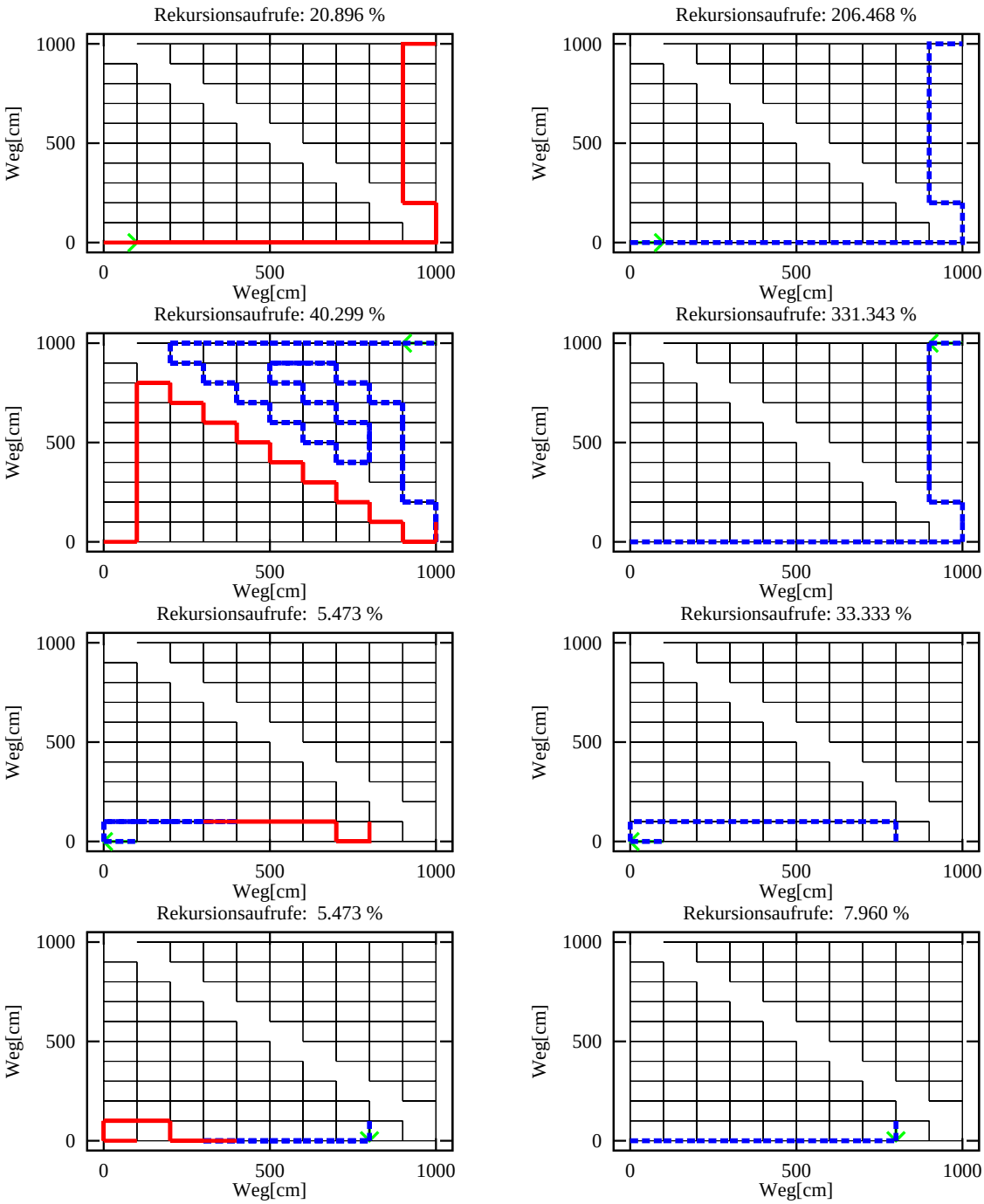

Rekursionsaufrufe: 1.364 %
Rekursionsaufrufe: 2.727 %
Rekursionsaufrufe: 4.091 %
Rekursionsaufrufe: 0.000 %
Rekursionsaufrufe: 5.909 %
Rekursionsaufrufe: 32.727 %
Rekursionsaufrufe: 4.091 %
Rekursionsaufrufe: 0.000 %
Weg[cm]
0
500
1000

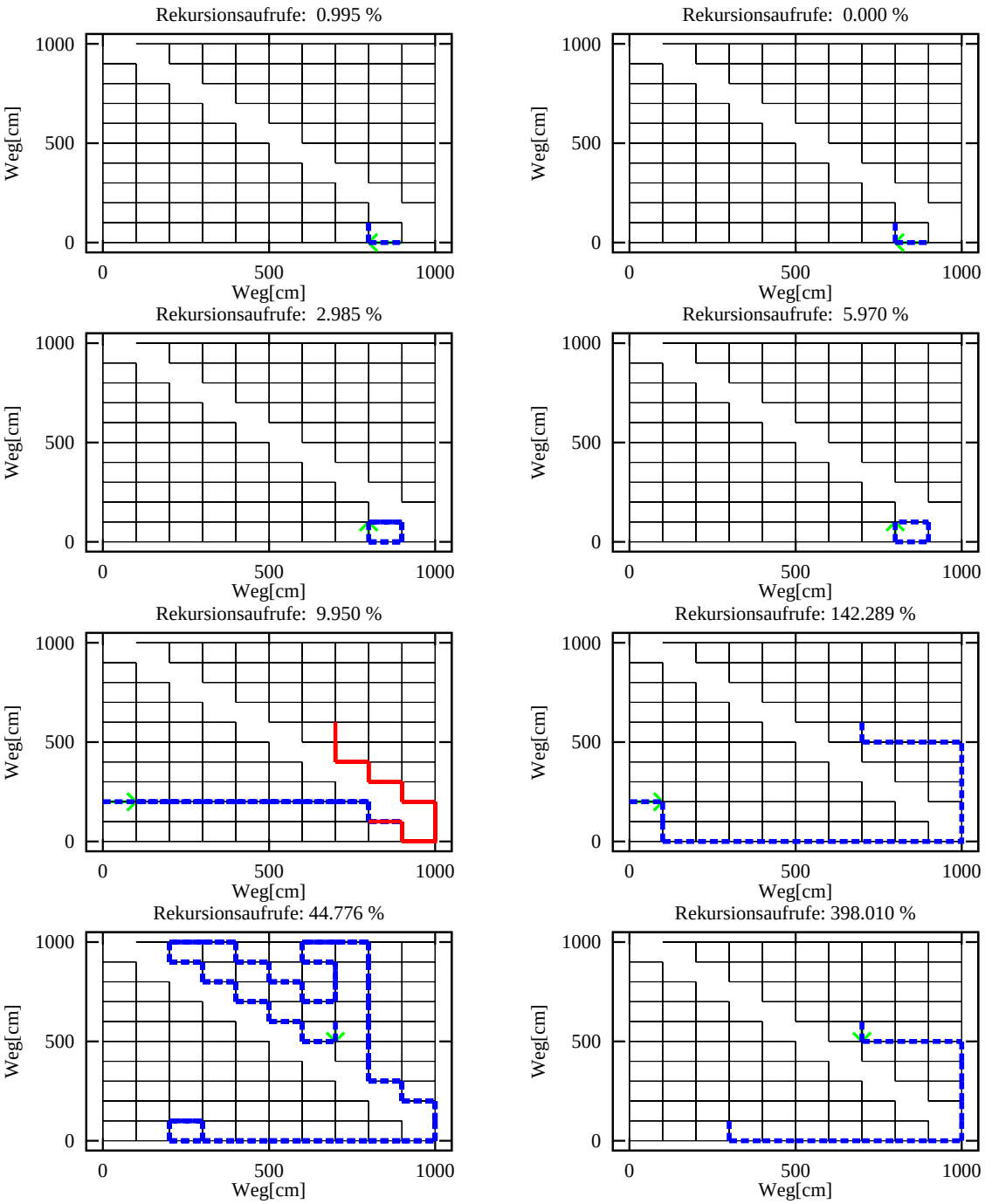

Rekursionsaufrufe: 5.909 %
Rekursionsaufrufe: 32.727 %
Rekursionsaufrufe: 8.636 %
Rekursionsaufrufe: 0.000 %
Rekursionsaufrufe: 9.545 %
Rekursionsaufrufe: 36.364 %
Rekursionsaufrufe: 5.000 %
Rekursionsaufrufe: 0.000 %
Weg[cm]
0
500
1000

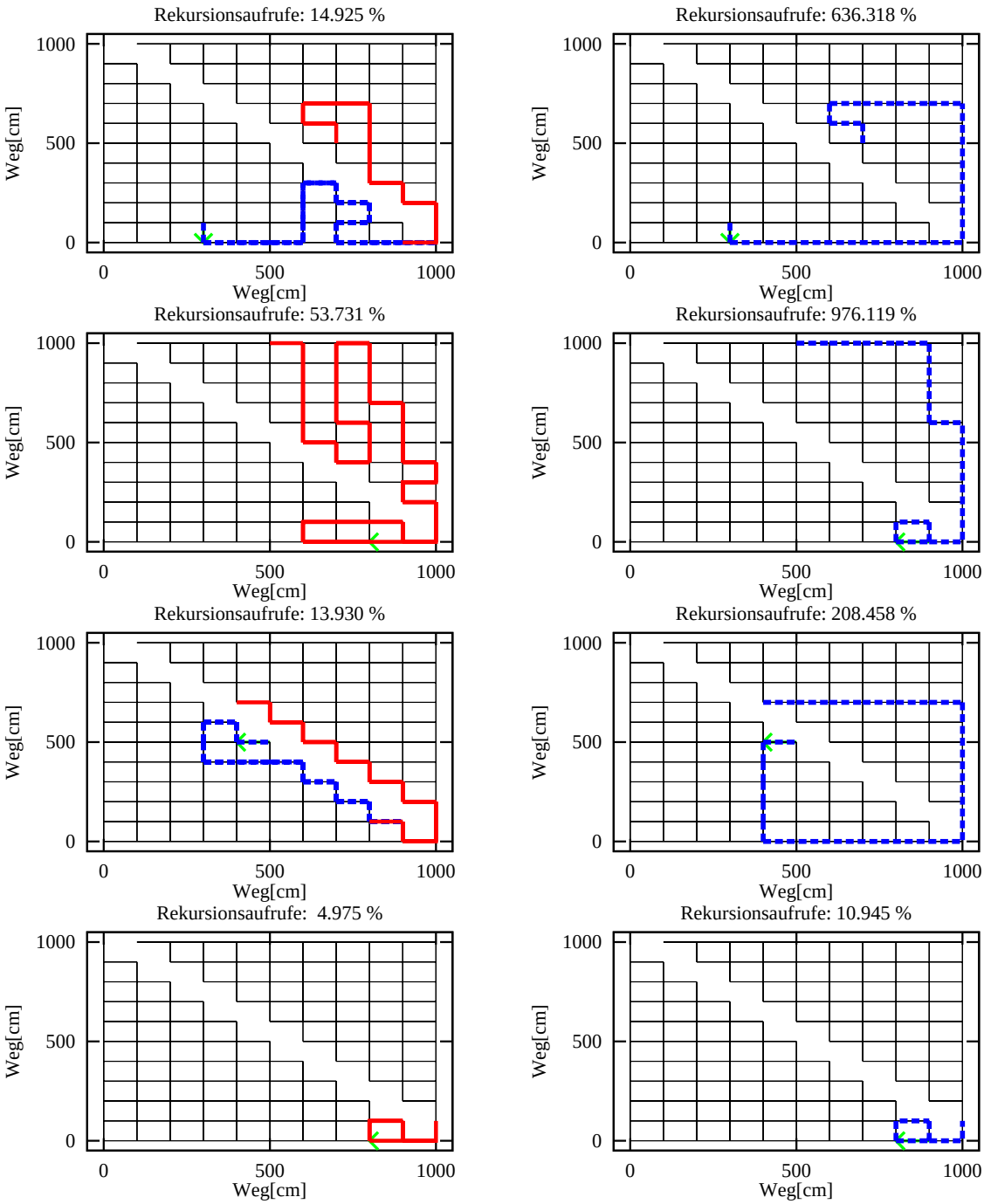

Rekursionsaufrufe: 42.289 %
Rekursionsaufrufe: 780.100 %
Rekursionsaufrufe: 10.448 %
Rekursionsaufrufe: 19.900 %
Rekursionsaufrufe: 5.473 %
Rekursionsaufrufe: 33.333 %
Rekursionsaufrufe: 5.473 %
Rekursionsaufrufe: 7.960 %
Weg[cm]
0
500
1000

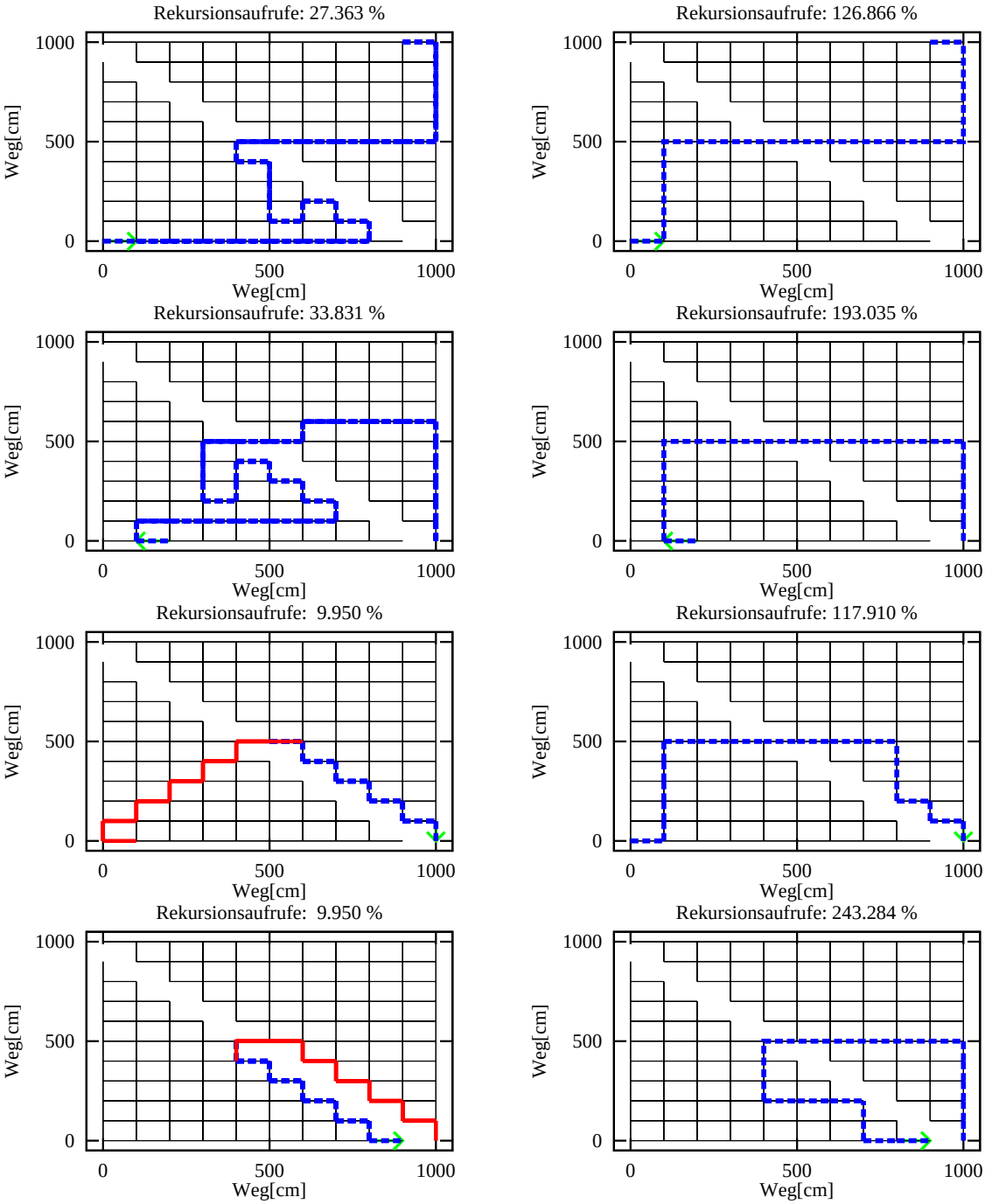

Rekursionsaufrufe: 0.995 %
Rekursionsaufrufe: 0.000 %
Rekursionsaufrufe: 2.985 %
Rekursionsaufrufe: 5.970 %
Rekursionsaufrufe: 32.836 %
Rekursionsaufrufe: 1800.000 %
Rekursionsaufrufe: 12.935 %
Rekursionsaufrufe: 648.756 %
Weg[cm]
0
500
1000

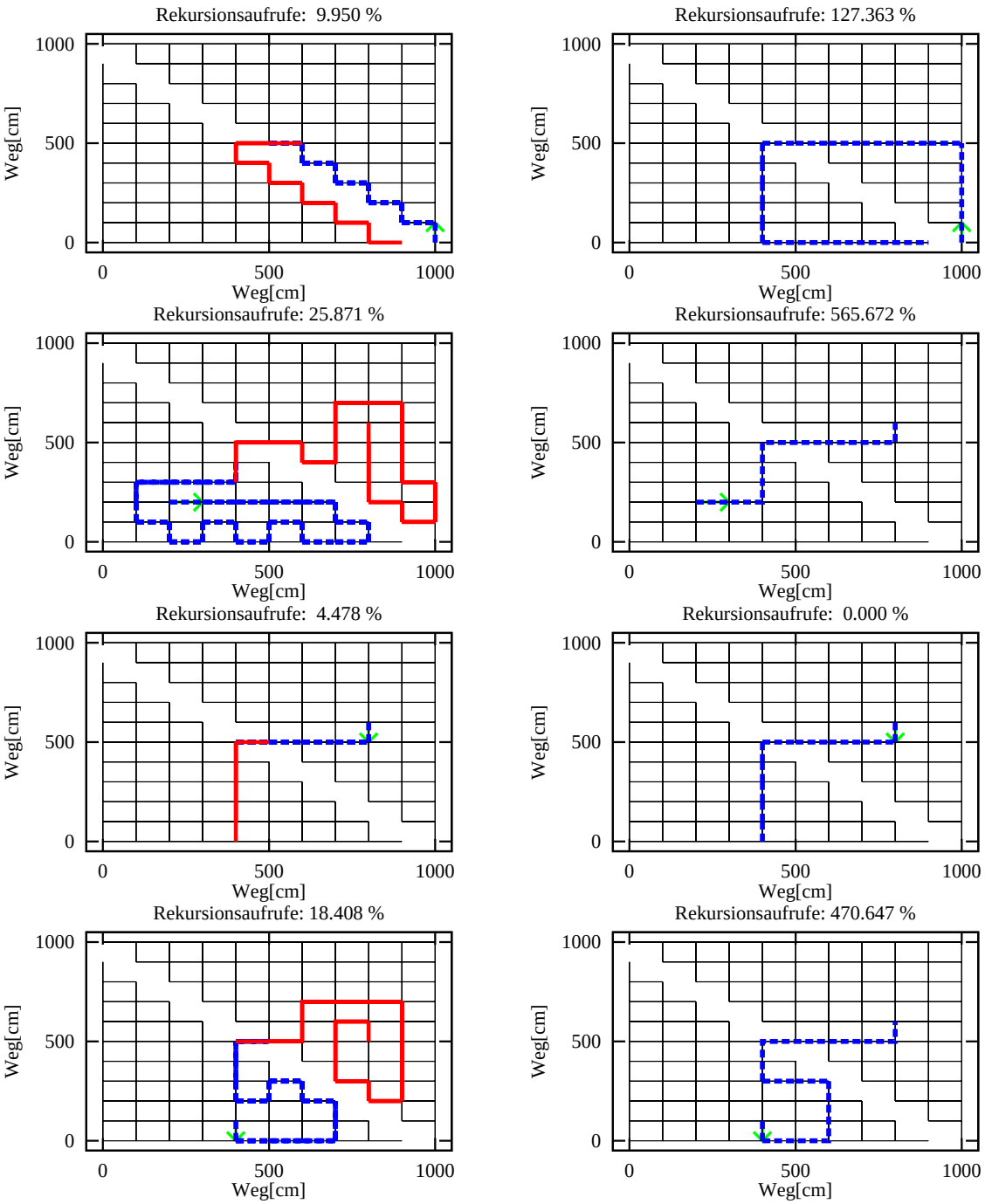

Rekursionsaufrufe: 28.856 %
Rekursionsaufrufe: 1348.756 %
Rekursionsaufrufe: 73.134 %
Rekursionsaufrufe: 407.960 %
Rekursionsaufrufe: 39.303 %
Rekursionsaufrufe: 1958.209 %
Rekursionsaufrufe: 83.582 %
Rekursionsaufrufe: 89.552 %
Weg[cm]
0
500
1000

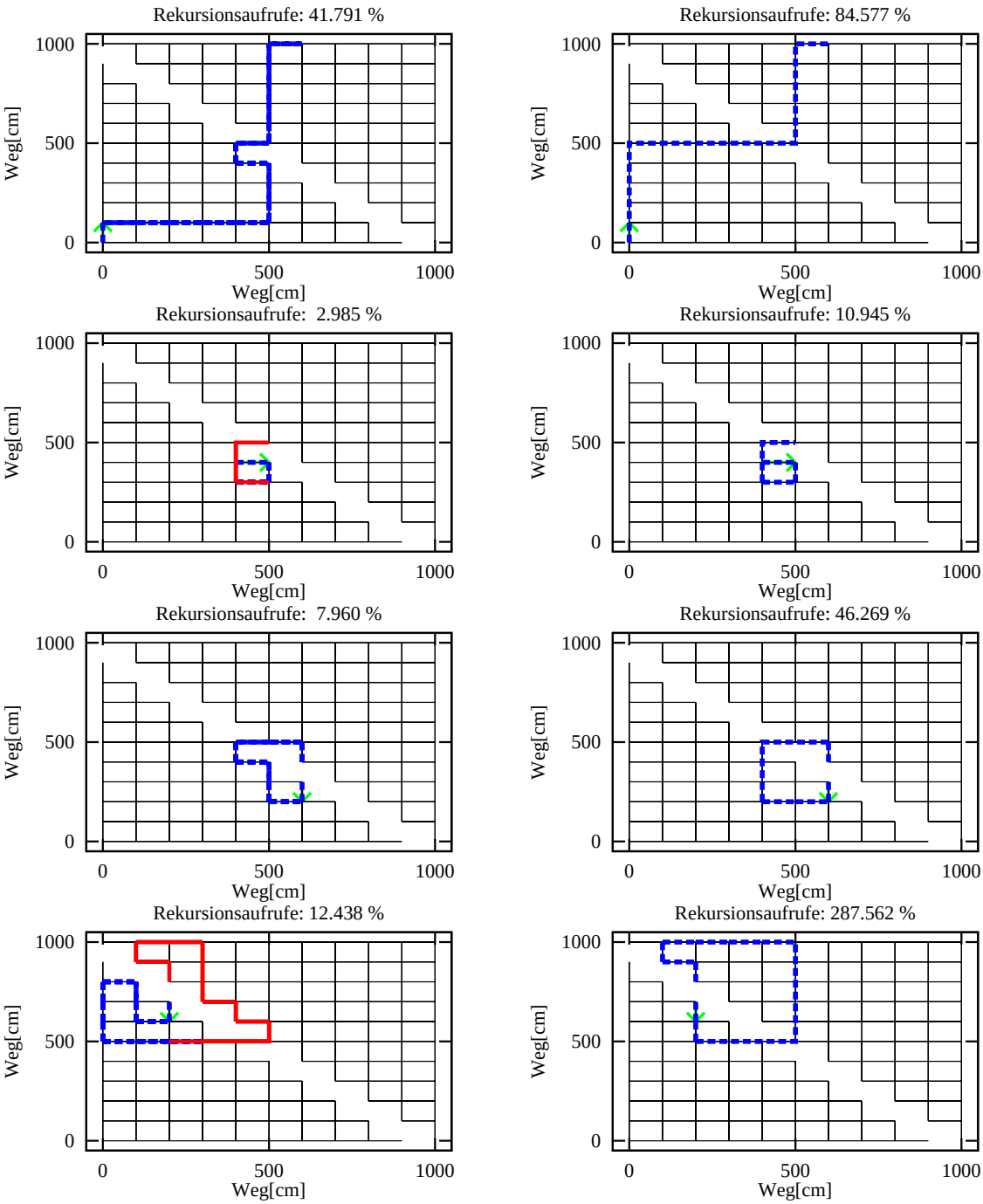

Rekursionsaufrufe: 20.896 %
Rekursionsaufrufe: 206.468 %
Rekursionsaufrufe: 40.299 %
Rekursionsaufrufe: 331.343 %
Rekursionsaufrufe: 5.473 %
Rekursionsaufrufe: 33.333 %
Rekursionsaufrufe: 5.473 %
Rekursionsaufrufe: 7.960 %
Weg[cm]
0
500
1000

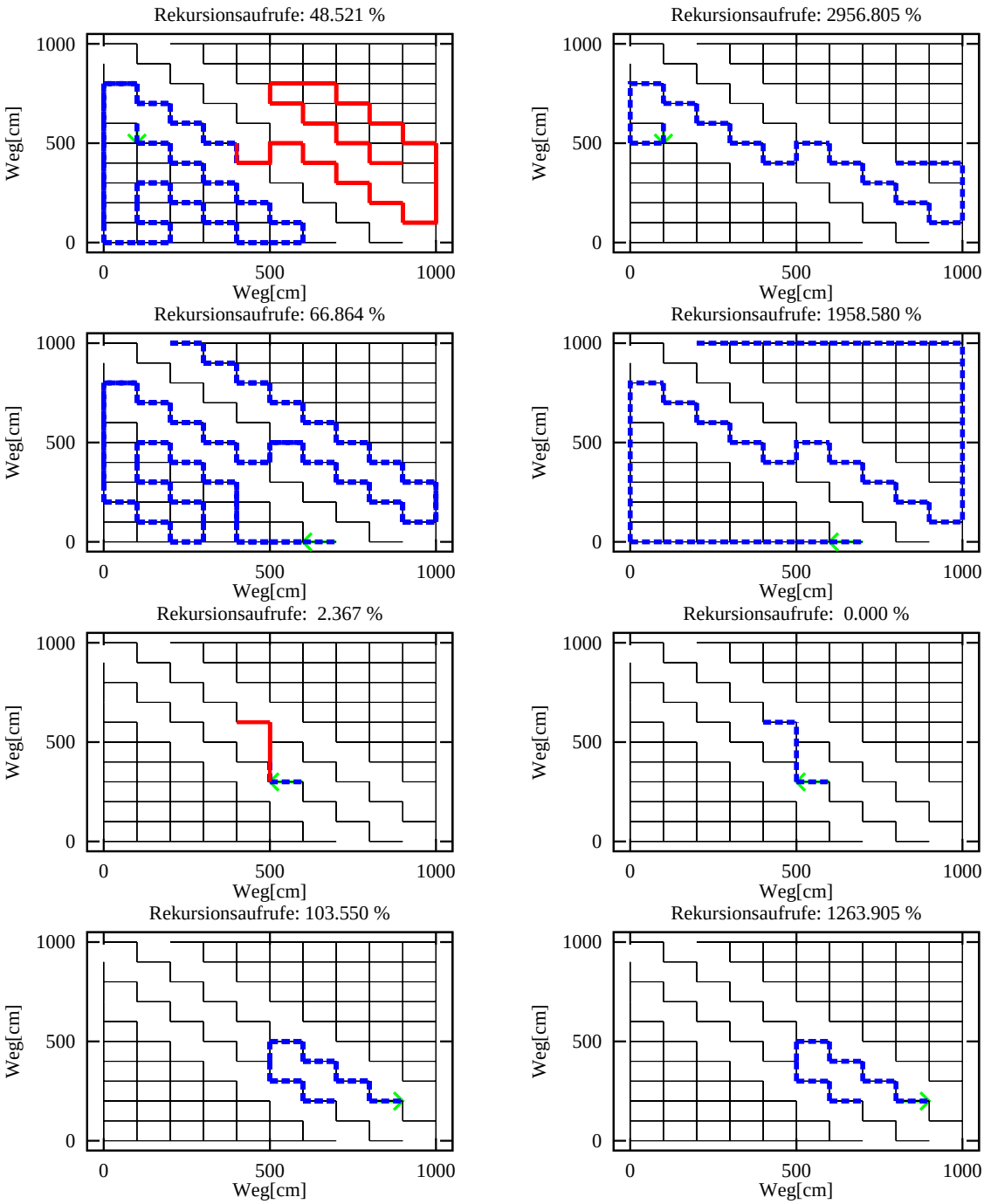

Rekursionsaufrufe: 0.995 %
Rekursionsaufrufe: 0.000 %
Rekursionsaufrufe: 2.985 %
Rekursionsaufrufe: 5.970 %
Rekursionsaufrufe: 9.950 %
Rekursionsaufrufe: 142.289 %
Rekursionsaufrufe: 44.776 %
Rekursionsaufrufe: 398.010 %
Weg[cm]
0
500
1000

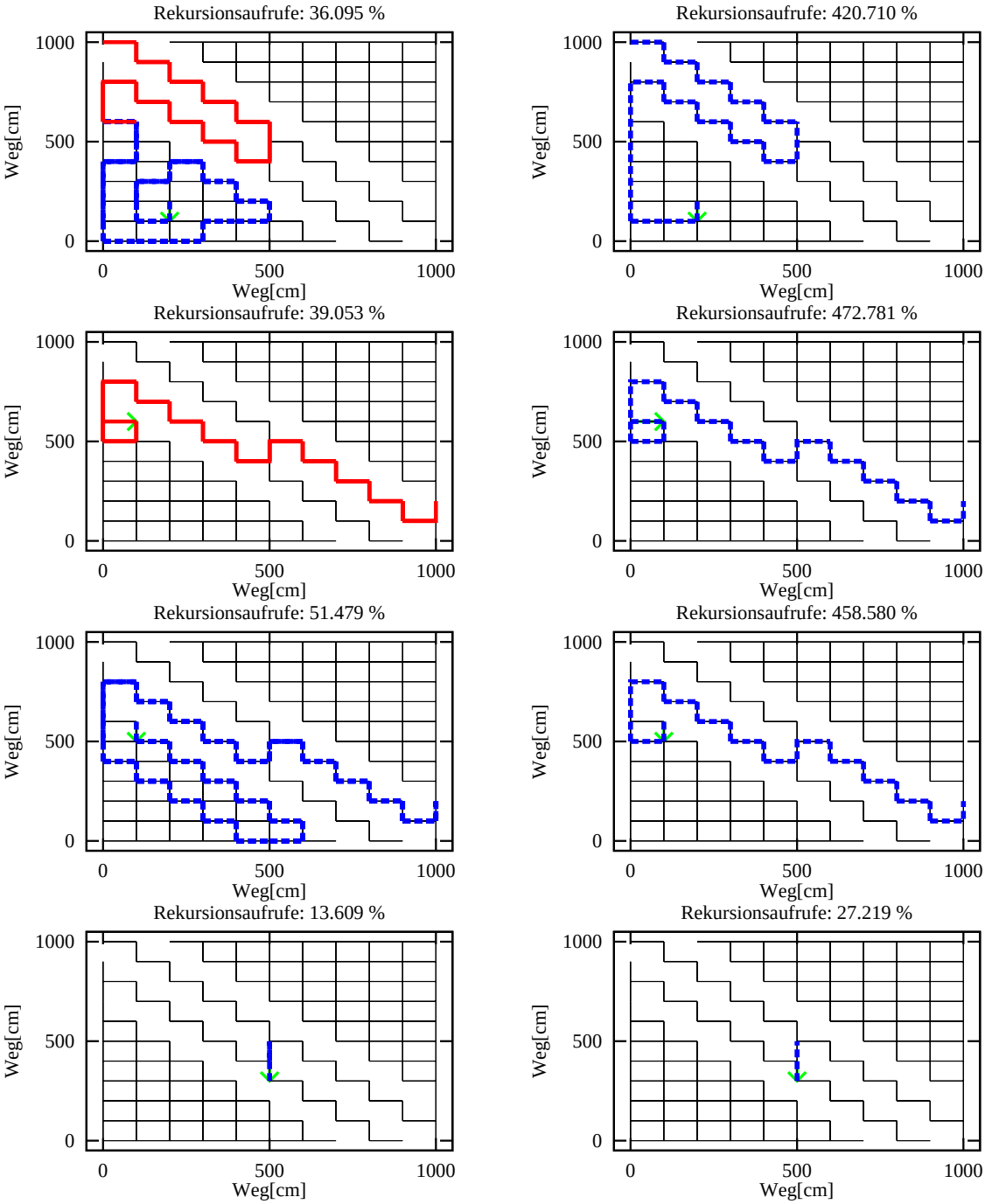

Rekursionsaufrufe: 14.925 %
Rekursionsaufrufe: 636.318 %
Rekursionsaufrufe: 53.731 %
Rekursionsaufrufe: 976.119 %
Rekursionsaufrufe: 13.930 %
Rekursionsaufrufe: 208.458 %
Rekursionsaufrufe: 4.975 %
Rekursionsaufrufe: 10.945 %
Weg[cm]
0
500
1000

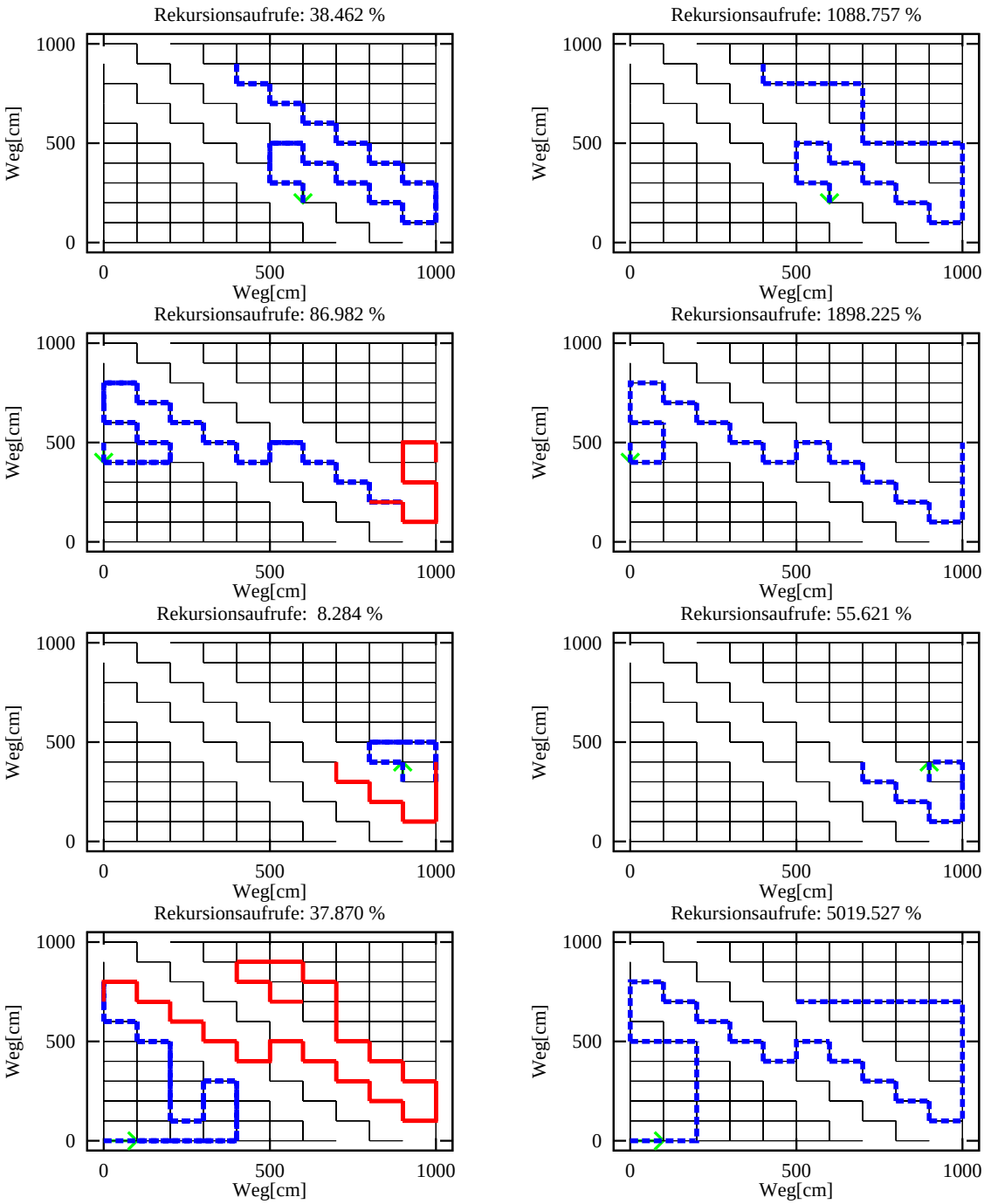

Rekursionsaufrufe: 27.363 %
Rekursionsaufrufe: 126.866 %
Rekursionsaufrufe: 33.831 %
Rekursionsaufrufe: 193.035 %
Rekursionsaufrufe: 9.950 %
Rekursionsaufrufe: 117.910 %
Rekursionsaufrufe: 9.950 %
Rekursionsaufrufe: 243.284 %
Weg[cm]
0
500
1000


Rekursionsaufrufe: 9.950 %
Rekursionsaufrufe: 127.363 %
Rekursionsaufrufe: 25.871 %
Rekursionsaufrufe: 565.672 %
Rekursionsaufrufe: 4.478 %
Rekursionsaufrufe: 0.000 %
Rekursionsaufrufe: 18.408 %
Rekursionsaufrufe: 470.647 %
Weg[cm]
0
500
1000


Rekursionsaufrufe: 41.791 %
Rekursionsaufrufe: 84.577 %
Rekursionsaufrufe: 2.985 %
Rekursionsaufrufe: 10.945 %
Rekursionsaufrufe: 7.960 %
Rekursionsaufrufe: 46.269 %
Rekursionsaufrufe: 12.438 %
Rekursionsaufrufe: 287.562 %
Weg[cm]
0
500
1000


Rekursionsaufrufe: 48.521 %
Rekursionsaufrufe: 2956.805 %
Rekursionsaufrufe: 66.864 %
Rekursionsaufrufe: 1958.580 %
Rekursionsaufrufe: 2.367 %
Rekursionsaufrufe: 0.000 %
Rekursionsaufrufe: 103.550 %
Rekursionsaufrufe: 1263.905 %
Weg[cm]
0
500
1000


Rekursionsaufrufe: 36.095 %
Rekursionsaufrufe: 420.710 %
Rekursionsaufrufe: 39.053 %
Rekursionsaufrufe: 472.781 %
Rekursionsaufrufe: 51.479 %
Rekursionsaufrufe: 458.580 %
Rekursionsaufrufe: 13.609 %
Rekursionsaufrufe: 27.219 %
Weg[cm]
0
500
1000


Rekursionsaufrufe: 38.462 %
Rekursionsaufrufe: 1088.757 %
Rekursionsaufrufe: 86.982 %
Rekursionsaufrufe: 1898.225 %
Rekursionsaufrufe: 8.284 %
Rekursionsaufrufe: 55.621 %
Rekursionsaufrufe: 37.870 %
Rekursionsaufrufe: 5019.527 %
Weg[cm]
0
500
1000

B Zusätzliche Module

Die Testphase des vorliegenden Planungs- und Kartierungsmoduls wurde auf der im Abschnitt 2.1 beschriebenen Simulationsumgebung durchgeführt. Um entsprechende Testumgebungen für relevante Fahrsituationen auf einfache und schnelle Weise zu generieren, gehört zum Umfang dieser Arbeit ein entsprechendes Modul, das diese Funktionen übernimmt. Dieses Modul ermöglicht die Generierung von beliebig aufgebauten Gangsystemen, die allerdings die gleiche Ganglänge und -breite für alle im System vorhandenen Gänge voraussetzt. Diese Parameter können beliebig für alle Gänge gesetzt werden. Alle in Anhang A gezeigten Umgebungen wurden mit Hilfe diese Zusatzmoduls generiert. Außerdem bietet sich dadurch eine enstprechend einfache Überprüfungsmöglichkeit des Algorithmus zur Wegsuche und der darauffolgenden Wegoptimierung. Die für die Simulationsumgebung zwingend erforderlichen Umgebungsdateien werden dabei automatisch generiert.

Um jedoch eine weitergehende Flexibilität hinsichtlich der Testumgebungen zu erreichen, gehört weiterhin eine Schnittstelle zum Umfang der entwickelten Software, die es ermöglicht, mit einem CAD-Programm erstellte Grafikdateien im DXF-Format zu importieren. Durch diese Schnittstelle kann der Benutzer eine aus Linien aufgebaute Umgebung mit dem entsprechenden CAD-Programm zeichnen und sie als Umgebungsdatei für die Simulation importieren. Dadurch ist es möglich, eine dem Planungsmodul unbekannte Umgebung zu erstellen, die dann zur Kartierung verwendet werden kann.

C Programmstart, Compilerflags und Zusatztools

Inhalt :

1.) Start der Simulation

2.) Compilerflags des Planers

3.) Ein- und Ausgabedateien

4.) Tool zur Umgebungsgenerierung

5.) Einlesen von DXF-Dateien

Zu 1. Start der Simulation in /home/sim/CODE/ :

SIMULATION GNU+SIM+CTR Usage: husim <-S>
SIMULATION Wertetest Usage: husim <-PO> <posx> <posy> <ori>
BENCHMARKTEST SIM Usage: husim <-B> <Anzahl der Durchläufe>
BENCHMARKTEST SIM+CTR Usage: husim <-MO> <Anzahl der Durchläufe>
NAGELBRETTEST SIM+GNU Usage: husim <-NA> <Anzahl Positionen>

Das Planungsmodul

Zu 2. Compilerflags in Makefile, einzutragen unter DEFS :

- ANSI : Deklaration der Prototypen nach ANSI. Ist ANSI nicht definiert, so werden die Prototypen nach dem K&R-Standard definiert.

- PLANER : *husim.c*
  Einbinden des Planermoduls inklusive Fahren nach Karte, Kartierung, Definition von Einbahnstrassen und Generierung einer Workspace-Datei aus einem Datenfile im DXF-Format.

- TESTMAIN : *planer.c*
  Ist TESTMAIN definiert, so wird allein die Wegsuche des Planungsmoduls ohne Simulation gestartet.

- WRITEPLOTFLAGS : *planer.c*
  Generierung der Datei plotdata.dat. Hierbei werden alle für den Planer relevanten Daten in die Datei ausgegeben. Anzeige mit 'gnuplot planer.gnu'.
- MSDOS : *planer.c topmap.c mapgen.c*
  Notwendige Passagen für die DOS-Version der Simulation.
- SUN : *planer.c topmap.c*
  Notwendige Passagen für die SUN-Version der Simulation.
- DOCU : *topmap.c*
  Ausgabe der berechneten Wege im Postscript-Format. Ist DOCU nicht definiert, so erfolgt die Ausgabe auf dem Bildschirm.
- GANGDEBUG : *topmap.c*
  Dieses Compilerflag erlaubt es, die Wegsuche schrittweise darzustellen.
- OUTPUT : *topmap.c*
  Es erfolgen Ausgaben auf den Bildschirm, die genaue Informationen der Wegsuche enthalten. Dieses Flag ist nur für Testzwecke gedacht. Die Performance der Wegsuche wird dadurch stark beeinträchtigt.
- GEOMAP : *husim.c init.c*
  Generierung der Sensorkarte.

Zu 3. Generierte Datenfiles des Planungsmoduls

- *umgebung.tm* : Enthält die topologische Beschreibung der generierten Karte. Diese Informationen werden für das Fahren nach Karte benötigt.
- *topmap.dat* : Enthält Daten für die Ausgabe der Umgebung mit Gnuplot.
- *frombeg.dat* : Enthält Daten für die Ausgabe des berechneten Wegs vom Startgang aus. (vor Wegoptimierung)
- *fromend.dat* : Enthält Daten für die Ausgabe des berechneten Wegs vom Zielgang aus. (vor Wegoptimierung)
- *st_zi.dat* : Enthält Daten für die Ausgabe des berechneten Wegs vom Startgang aus (nach Wegoptimierung).
- *zi_st.dat* : Enthält Daten für die Ausgabe des berechneten Wegs vom Zielgang aus (nach Wegoptimierung).
- *QAavSzZP.dat* : Enthält Daten für die Ausgabe des Quadratischen Abstands zwischen aktuellem Gang der Suche vom Startgang aus und dem Zielgang. Wichtig für die Berechnung der Stützgänge mit darauffolgender Wegoptimierung. Ausgabe nur, wenn OUTPUT definiert ist.

- *QAavZzSP.dat* : Enthält Daten für die Ausgabe des Quadratischen Abstands zwischen aktuellem Gang der Suche vom Zielgang aus und dem Startgang. Wichtig für die Berechnung der Stützgänge mit darauffolgender Wegoptimierung. Ausgabe nur, wenn OUTPUT definiert ist.
- *winkelsu.dat* : Enthält Daten für die Ausgabe der Winkelsummen innerhalb des berechneten Weges. Wichtig für die Berechnung der Stützgänge mit darauffolgender Wegoptimierung. Ausgabe nur, wenn OUTPUT definiert ist.
- *oneway.dat* : Enthält Informationen über eventuell definierte Einbahnstrassen in der aktuellen Umgebung.

Zu 4. Tool zur Generierung von Umgebungsdateien für die Simulationsumgebung des Roboters MORIA. Anmerkung: Es können nur Gangsysteme aus rechtwinklig zueinander liegenden Gängen erstellt werden.

Eingabeparameter:

1.) Anzahl der Gänge in x-Richtung

2.) Anzahl der Gänge in y-Richtung

3.) Ganglänge (gilt für alle im System vorhandenen Gänge, Eingabe in Meter $*$ 100)

4.) Gangbreite (gilt für alle im System vorhandenen Gänge, Eingabe in Meter $*$ 100)

Auszufilternde Gänge müssen mit ihrer Gangnummer (eine in jeder Zeile, abschliessend -1) in die Datei filtwege.dat eingetragen werden.
Ausgabe:

1.) Strukturelle Beschreibung für das Planungsmodul in *../umgebung.tm*

2.) Workspace-Datei für die Simulation in *../WSP/workspac.org*. Diese Datei enthält die einzufügenden Sperrwände an den Gangendpunkten. Diese Datei wird beim Starten der Simulation nach *../../WSP/workspac* kopiert.

Zu 5. Einlesen von DXF-Dateien

Der Planer bietet die Möglichkeit, Umgebungsdaten im DXF-Format zu importieren. Die Daten werden aus *../WSP/umgebung.dxf* eingelesen. Bei Start der Simulation und Aufruf des Menuepunktes 5 (Umgebung aus dxf-Datei importieren) wird die Datei *../WSP/workspac.org* generiert. Diese wird bei Start analog zu 4. Tool zur Generierung von Umgebungsdateien für die Simulationsumgebung des Roboters MORIA nach *../WSP/workspac* kopiert.

Literaturverzeichnis

[1] Engeln, “Fischfang im Fluge,” *GEO*, 1/1993.

[2] Goser/Surmann, “Clevere Regler schnell entworfen,” *Elektronik*, Seite 60 – 68, 6/1992.

[3] Surmann/Kanstein/Goser, “Self-Organizing and Genetic Algorithms for an Automatic Design of Fuzzy Control and Decision Systems,” *EUFIT 93, Aachen*, Seite 1097 – 1104, 1993.

[4] J. Huser, *Sensorbasierte implizite Führung eines autonomen mobilen Roboters mittels Fuzzy-Methoden*. Diplomarbeit, Universität Dortmund, 1994.

[5] Kahlert/Frank, *FuzzyLogik und FuzzyControl*. Braunschweig/Wiesbaden: Vieweg, 1994.

[6] L. A. Zadeh, “Fuzzy sets,” *Information and Control*, vol. 8, Seite 338 – 353, 1965.

[7] L. A. Zadeh, “Fuzzy algorithms,” *Information and Control*, vol. 12, Seite 94 – 102, 1968.

[8] L. A. Zadeh, “Outline of a new approach to the analysis of complex systems and decision processes,” *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-3, Seite 28 – 44, 1973.

[9] L. A. Zadeh, “Fuzzy sets as a basis for a theory of possibility,” *Fuzzy Sets and Systems*, vol. 1, Seite 3 – 28, 1978.

[10] L. A. Zadeh, “Making computers think like people,” *IEEE Spectrum*, vol. 8, Seite 26 – 32, 1984.

[11] L. A. Zadeh, “Fuzzy logic,” *IEEE Computer*, vol. 21, Seite 83 – 92, 1988.

[12] L. A. Zadeh, “Foreword of the proccedings of the 2nd int. conf. on f.l. & n.n.,” in *IIZUKA’92*, (Iizuka, Fukuoka, Japan), 1992.

[13] H. Surmann, J. Huser, und L. Peters, “Guiding and controlling mobile robots with a fuzzy controller,” in *Fourth IEEE International Conference on Fuzzy Systems, Yokohama, Japan*, Seite 83 – 88, 20. - 24.3.1995, distinguished with Robot Intelligence Award.

[14] Swietlik, “Geometrische Rekonstruktion von dreidimensionalen Szenen aus Abstandsdaten,” *VDI Fortschrittsberichte Nr. 409*, 1994.

[15] Puttkamer, “Sensorintegration zur Geometrischen Weltmodellierung,” *it + ti*, 36/1994.

[16] Schnare, “Wissensbasierte Trajektoriensuche für autonome Roboter,” *VDI Fortschrittsberichte Nr. 308*, 1992.

[17] Soetadji, “Methode zur Lösung des Routenplanungsproblems für ein Navigationssystem eines autonomen mobilen Roboters,” *VDI Fortschrittsberichte Nr. 70*, 1987.

[18] Spandl, “Lernverfahren zur Unterstützung der Routenplanung für einen mobilen Roboter,” *VDI Fortschrittsberichte Nr. 212*, 1992.

[19] Weiqing, “Ein wissensbasiertes System zur Fahrtplanung und -kontrolle eines autonomen mobilen Roboters,” *VDI Fortschrittsberichte Nr. 110*, 1992.

[20] Xiaozhao/Eng, “Planung kollisionsfreiher Bahnen für autonome Roboter mittels einer Freiraumdarstellung durch sichere Dreiecke,” *VDI Fortschrittsberichte Nr. 73*, 1994.

[21] Kerninghan/Ritchie, *Programmieren in C*. München Wien: Carl Hanser Verlag, 1990.

[22] U. Schnepf, *Entwurf einer Robotersimulationsumgebung für Fuzzy-Evaluation*. Simulationsbeschreibung, Gesellschaft für Mathematik und Datenverarbeitung, 1994.

[23] Busacker, *Endliche Graphen und Netzwerke - Eine Einführung mit Anwendungen*. 1968.

[24] Dörfler, *Mathematik für Informatiker - Finite Elemente und Algebra*. München Wien: Carl Hanser Verlag, 1971.

[25] Simon, *Effiziente Algorithmen für perfekte Graphen*. Teubner Verlag, 1992.

[26] Tontch, *Vorlesungsskript Lineare Algebra und Mathematische Strukturen*. 1991.

[27] Wallner, “Reaktive Bewegungssteuerung der mobilen Roboterplattform PRIAMOS,” *Autonome Mobile Systeme*, 11/1992.