Autonome Generierung kognitiver Karten zur landmarkenbasierten Navigation

Diplomarbeit vorgelegt von
Oliver Harwart

Betreuer: Jörg Bruske

Lehrstuhl für Kognitive Systeme
des
Instituts für Informatik
der
Christian-Albrechts-Universität zu Kiel

10. Dezember 1997

Inhaltsverzeichnis

KAPITEL 1 EINFÜHRUNG **1**

KAPITEL 2 NAVIGATION BIOLOGISCHER LEBEWESEN **5**

2.1 RATTEN 5
2.2 AMEISEN 8
2.3 BIENEN 8

KAPITEL 3 BISHER EINGESETZTE VERFAHREN **10**

3.1 NAVIGATION MIT BESETZTHEITSGITTERN 10
3.1.1 Navigation mit Landmarken *10*
3.1.1.1 Jonathan H. Connell, Sridhar Mahadevan, OBELIX 10
3.1.2 Navigation ohne Landmarken *12*
3.1.2.1 Sebastian B. Thrun: COLUMBUS 12
3.2 NAVIGATION MIT TOPOLOGISCHEN KARTEN BZW. GRAPHEN 14
3.2.1 Navigation mit Landmarken *14*
3.2.1.1 Min Meng und A.C.Kak, NEURO-NAV 14
3.2.1.2 Ivan A. Bachfelder und Allen M. Waxman: MAVIN 15
3.2.1.3 Maja J. Mataric: TOTO 17
3.2.1.4 David S. Touretzky, Hank S. Wan, A. David Redish 18
3.2.2 Navigation ohne Landmarken *20*
3.2.2.1 Sjur J. Vestli, Nadine Tschichold-Gürman, Henrik Andersson 20
3.2.2.2 Schölkopf und Mallot 21
3.2.2.3 Buhmann, Burgard, Cremers, Thrun: The Mobile Robot Rhino 21
3.2.2.4 A. Kurz: ALEF 22
3.3 VERHALTENSBASIERTE ANSÄTZE OHNE KARTENGENERIERUNG UND OHNE LANDMARKEN 24
3.3.1 Thomas Röfer (Controlling a Robot With Image-based Homing) *24*

KAPITEL 4 ANFORDERUNGSKATALOG **26**

KAPITEL 5 DAS KONZEPT FÜR DEN ROBOTER **27**

KAPITEL 6 KOMPONENTEN DES MODELLS **30**

6.1 SCHEMA VON KILROY 30
6.2 DIE EINGABEDATEN FÜR DAS NAVIGATIONSSYSTEM 31
6.2.1 Pfadintegrierer (Hippocampus) *31*
6.2.2 Ultraschallsensoren *31*
6.2.3 Kameras *31*
6.2.4 Kopfbewegung *32*
6.2.5 Lokale Ansicht *32*
6.2.6 Motorbefehle *32*
6.2.7 Zielvorgabe des Bedieners bzw. Erkundungsalgorithmus *32*
6.2.8 Referenzpunkte für Koordinatenangaben *32*
6.3 DIE AUFGABEN DER EINZELNEN NAVIGATIONSMODULE 33
6.3.1 Place-Units *33*
6.3.2 Kartengenerierung zu den ermittelten Place-Unit-Koordinaten *35*
6.3.3 Wegplanung *35*

6.3.4 Fahrmodul ... *36*
6.3.5 Kameraausrichtung ... *36*
6.3.6 Hindernisvermeidung ... *36*
6.3.7 Vorrang Hindernisvermeidung ... *36*
6.3.8 Erkundungsmodul ... *36*
6.4 DESORIENTIERUNG DES ROBOTERSYSTEMS ... 37

KAPITEL 7 MOTIVATION DER METHODEN ... 38

KAPITEL 8 UMSETZUNG DES MODELLS ... 40

8.1 OBJEKT-KOMPONENTEN DES MODELLS ... 40
8.1.1 Landmarken ... *40*
8.1.1.1 Ununterscheidbare Landmarken und daraus resultierende Probleme ... 41
8.1.1.2 Eindeutig unterscheidbare Landmarken und Ihre Vorteile ... 41
8.1.1.3 Kompromiß: verschiedene Typen von Landmarken. ... 41
8.1.1.3.1 Landmarkentypen in der Wirklichkeit ... 42
8.1.1.3.2 Landmarkentypen in der Simulation ... 42
8.1.2 Die lokale Ansicht ... *43*
8.1.2.1 Geeignete Sensoren für die Aufnahme einer lokalen Ansicht ... 43
8.1.2.1.1 Stereo-Kamerasystem ... 43
8.1.2.1.2 Laser-Entfernungsmesser ... 44
8.1.2.2 Datenstrukturen ... 44
8.1.2.3 Algorithmen ... 45
8.1.3 Das Place-Unit-Netz ... *46*
8.1.3.1 Merkmale der Place-Units ... 46
8.1.3.2 Netzaufbau ... 47
8.1.3.3 Bestimmung der aktuellen Position und Lernverfahren ... 47
8.1.4 Die topologische Karte ... *52*
8.1.4.1 Datenstrukturen ... 53
8.1.4.2 Generierung der Karte ... 53
8.1.5 Die Befehlsqueue ... *55*
8.1.6 Das Erkundungssystem ... *55*
8.1.7 Die Umgebung und RPI-Anbindung ... *56*
8.2 C++-KLASSENBIBLIOTHEK ... 56
8.3 DIE PROGRAMMODULE UND DIE ENTHALTENEN C++-KLASSEN ... 57
8.3.1 kilroystd ... *57*
8.3.2 environment ... *58*
8.3.3 kilroySIM ... *59*
8.3.4 object ... *59*
8.3.5 order ... *60*
8.3.6 placeunits ... *61*
8.3.7 placemap ... *63*
8.3.8 roboter ... *63*

KAPITEL 9 AUSWERTUNG ... 66

9.1 EXPERIMENTE ... 66
9.1.1 3 unterscheidbare Landmarken ... *67*
9.1.1.1 Versuchsbeschreibung ... 67
9.1.1.2 Versuchsergebnis ... 70
9.1.2 Hufeisenanordnung ... *73*

9.1.2.1 Versuchsbeschreibung 73
9.1.2.2 Versuchsergebnis 75
9.1.3 2 gleiche Räume, gleiche Landmarken 76
9.1.3.1 Versuchsbeschreibung 76
9.1.3.2 Versuchsergebnis 78
9.1.4 2 gleiche Räume, verschiedene Anordnung 83
9.1.4.1 Versuchsbeschreibung 83
9.1.4.2 Versuchsergebnis 84
9.1.5 2 gleiche Räume, unterschiedliche Landmarken, simulierte Fehler 86
9.1.5.1 Versuchsbeschreibung 86
9.1.5.2 Versuchsergebnis 88
9.1.6 3 ununterscheidbare Landmarken 90
9.1.6.1 Versuchsbeschreibung 90
9.1.6.2 Versuchsergebnis 91
9.1.7 3 Landmarken, simulierte Fehler 95
9.1.7.1 Versuchsbeschreibung 95
9.1.7.2 Versuchsergebnis 96

KAPITEL 10 DISKUSSION 98

KAPITEL 11 AUSBLICK 100

KAPITEL 12 ABBILDUNGSVERZEICHNIS 102

KAPITEL 13 LITERATURVERZEICHNIS 104

Kapitel 1 Einführung

Autonome Robotersysteme sind dazu gedacht, dem Menschen gefährliche, anstrengende oder eintönige Arbeit abzunehmen. Um dies zu leisten, sollen sie ihre Aufgaben mit nur minimaler Einwirkung des Menschen, möglichst auf einfachen Befehl hin, bewältigen. Dazu ist es nötig, daß die Roboter sich in ihrer Umgebung selbständig fortbewegen können, ohne Hindernisse anzufahren und sich zu verirren. Unbekanntes Terrain soll vom Roboter automatisch erkundet und nicht vom Systembetreuer vorgegeben werden.

Um in erkundetem Gebiet ohne Neuerkundung navigieren zu können, muß Information über die Umgebung gespeichert werden. Die von Landkarten her bekannte metrische Erfassung scheint, trotz einfacher Zugänglichkeit seitens des Menschen, nicht das beste Verfahren für heutige Robotersysteme zu sein, die in ihrer Rechen- und Speicherkapazität stark beschränkt sind. Begrenzend sind dabei Abmessungen und zu bewegendes Gewicht des Roboters, der nur begrenzte Anzahl an Gerätschaften und Energiequellen transportieren kann.

Wie so häufig sucht man in der Natur ein Vorbild, das sich in der Evolution als robust und, da von kleinen Lebewesen anwendbar, einfach herausgestellt hat. So hofft man auf vereinfachte Methoden, die auch von dem betrachteten Robotersystem einsetzbar sind.

Die meisten Navigationssysteme verlassen sich derzeit auf sogenanntes *dead rekkoning* (abgeleitet vom Begriff „deduced reckoning“). Dahinter verbirgt sich ein einfaches Verfahren, um den aktuellen Standort zu bestimmen, das eine bekannte Position basierend auf bekannter Richtungs- und Geschwindigkeitsinformation für eine gegebene Zeitdauer aktualisiert.

Odometrie basiert auf der Annahme, daß Radumdrehungen in eine lineare Verschiebung relativ zum Untergrund übersetzt werden können. Diese Annahme hat nur eingeschränkte Gültigkeit, da aufgrund durchdrehender Räder leicht Fehler entstehen können.

Genauer gibt es verschiedene Fehlerquellen. Systematische Fehler entstehen beispielsweise durch ungleiche Raddurchmesser, begrenzte Radsensorauflösung und begrenzte Samplingrate des Radsensors. Hinzu kommen nicht-systematische

Fehler, wie unebener Untergrund, rutschiger Boden, zu starke Beschleunigung des Robotersystems, zu schnelles Durchfahren von Kurven (schleudern) oder die Interaktion mit Hindernissen.
Zusätzliche Sensoren werden eingesetzt, um den Roboter trotzdem erfolgreich durch seine Umgebung zu steuern.
Unser Navigationssystem KiLRoY lehnt sich an ein Verfahren an, das von Touretzky et al. vorgestellt wurde. Es orientiert sich an dem Navigationsverhalten von Ratten. Touretzky et al. haben Versuche, in denen Ratten verstecktes Futter auffinden mußten, in einer Simulation mit ihrem Verfahren nachgestellt und erhielten Ergebnisse, die mit denen von echten Ratten vergleichbar sind.
Ratten orientieren sich anhand von Landmarken in Ihrer Umgebung. Landmarken können beliebige Umgebungsmerkmale sein. Ratten orientieren sich dabei allerdings an der Geometrie, nicht jedoch an der Oberflächenbeschaffenheit ihrer Umgebung. Im Hippocampus des Rattengehirns feuern verschiedene „place cells" für unterschiedliche Positionen, an denen sich die Ratte befindet. Ratten erkennen dabei Landmarkenanordnungen wieder und orientieren sich nicht an einzelnen Landmarken. Im Dunkeln können Ratten einen Weg zuende führen, den sie zuvor geplant haben. Das bedeutet, daß sich Ratten nicht nur auf visuelle Hinweise verlassen, sondern auch eine Form von Pfadintegrierung (dead reckoning) benutzen. Diese Aspekte der Rattennavigation sind Teil dies vorgestellten Navigationssystems KiLRoY.
Einige häufig verwendete Begriffe wollen wir kurz erläutern. Ein *Ort* in der Umgebung des Roboters beschreibt einen kleinen Bereich der Umgebung. Eine *Stelle* bezeichne eine exakte Position innerhalb der Umgebung. Ein *Platz* sei eine Stelle in der Umgebung, die bereits im Place-Unit-Netz des Navigationssystems erfaßt ist.
Eine Place-Unit ist nicht direkt mit einer „place cell" von Ratten zu vergleichen. Vielmehr sollte eine Place-Unit als Ansammlung von „place cells" betrachtet werden. Sie stellt eine Stelle der Karte relativ zu sichtbaren Landmarken dar. Dazu nötige Angaben können *allozentrisch* und *egozentrisch* erfolgen. Eine allozentrische Angabe erfolgt relativ zur Umgebung, eine egozentrische relativ zum Roboter.

Im einzelnen gehen wir wie folgt vor:
Im Kapitel 2 werden wir kurz vorstellen, auf welche Art und Weise sich einige unterschiedliche Gruppen biologischer Lebewesen in ihrer Umgebung zurechtfinden. Unter anderem wird hier die Abhängigkeit der Wahrnehmung von der verfolgten Aufgabe und dem Lebensraum des Lebewesens deutlich.
Das Kapitel 3 zeigt bisher von anderen Forschungsgruppen eingesetzte Verfahren zur Roboternavigation auf. Hierbei offenbaren sich zum Teil sehr unterschiedliche Ansätze.
Die Anforderungen an das Navigationssystem behandeln wir in Kapitel 4 . Hier werden die der Arbeit zugrundeliegenden Ziele aufgezeigt.
In Kapitel 5 entwickeln wir ein Konzept für den Roboter, das sich an den gestellten Anforderungen orientiert. Dabei greifen wir auf eine der zuvor präsentierten Arbeiten zur Roboternavigation zurück.
Kapitel 6 beleuchtet die Komponenten des Modells detaillierter. Hier werden die in Kapitel 5 umrissenen Funktionsgruppen im einzelnen beschrieben. Wir zeigen die Aufgaben der einzelnen Teile auf und charakterisieren die eingesetzten Verfahren zur Bewältigung der Aufgaben.
Die Motivation für das Navigationsverfahren beschreiben wir in Kapitel 7 .
Das Kapitel 8 beschreibt die Umsetzung der Funktionsgruppen in ein Programm. Dabei wird der im objektorientierten Design übliche Prozeß der Klassenmodellierung nachvollzogen: Die Funktionsgruppen werden bezüglich der anzuwendenden Verfahren präzisiert. Dabei werden Teile des Algorithmus in symbolischem Code beschrieben und mit graphischen Darstellungen erläutert. Desweiteren werden wir die resultierende Klassenstruktur darstellen, und die Beziehungen der Klassen untereinander aufzeigen.
In Kapitel 9 beschreiben wir die Durchführung von Experimenten mit der Navigationssimulation. Ferner werden wir die Ergebnisse dieser Versuche und damit die Fähigkeiten des Systems aufzeigen.
Eine kritische Beleuchtung der Resultate erfolgt in Kapitel 10 . Wir werden hier bewerten, ob das Navigationsverfahren den in Kapitel 4 formulierten Anforderungen gerecht wird.

Abschließend geben wir (in Kapitel 11) in einer Übersicht über das Navigationsverfahren alle bisherigen Resultate wieder und führen einige offen gebliebene Fragen auf.

Kapitel 2 Navigation biologischer Lebewesen

Die Natur ist gerade im Bereich der Robotik häufig ein Vorbild bei der Entwicklung, da selbst einfachste Lebewesen oft erstaunliche Fähigkeiten in der Interaktion mit ihrer Umgebung beweisen. Einige Möglichkeiten, die von der Natur hervorgebracht wurden, sollen hier kurz beleuchtet werden.

2.1 Ratten

Ratten setzen verschiedene räumliche Repräsentationen ein, nämlich zum einen auf Ansichten von Landmarken beruhende „Platz“-Definitionen, zum anderen „place cells“, die feuern, wenn sich die Ratte in einer bestimmten Region befindet (empfindlich für visuelle Hinweise).

Es wurde beobachtet, daß das Ortsgespür der Ratten synchron mit der Rotation von Landmarken rotiert. Es liegt daher nahe, daß es nicht auf einzelnen Landmarken beruht, sondern auf Anordnungen mehrerer Landmarken. Orte können selbst dann wiedererkannt werden, wenn aus der Landmarkenanordnung einige Marken entfernt werden, solange genug visuelle Hinweise verbleiben, aus denen eine Zuordnung möglich ist (hinreichend viele „place cells“ müssen also korrekt feuern). Lokale sichtabhängige Ortsbeschreibungen können anhand einzelner Merkmale wieder vervollständigt werden.

Die Ortsspeicherung im Hippocampus beruht auf einer topologischen Darstellung der Umgebung. Diese Repräsentation enthält eine Menge verschiedener Plätze und Informationen über deren Nachbarschaft. Entgegen in der Literatur verbreiteten „place cell“-Modellen als feedforward-Netze jedoch werden tatsächlich rekurrente Verbindungen eingesetzt.

Die Navigation der Nager orientiert sich aber nicht allein an visuellen Stimuli. Die „place cells“ feuern ebenfalls, wenn sich das Tier in der Dunkelheit fortbewegt. Folglich sind die Informationen über die Umgebung zum Teil auch metrische Informationen. Dies wurde in einem Experiment von T.S. Collet gezeigt, in dem ein Nager auf eine leuchtende LED zuläuft, die abgeschaltet wird, bevor das Tier sie erreicht. Das Tier beendet den zuvor bestimmten Pfad in der Dunkelheit. Scheinbar werden also sowohl Entfernung als auch Richtung zum Ziel festgelegt.

Nicht bekannt ist, ob dabei ein polares oder ein kartesisches Koordinatensystem zugrundeliegt. Zwar erscheint ein polares Koordinatensystem als natürliche Wahl, Gallistel jedoch gab an, daß bei inkrementellen Positionsaktualisierungen ein kartesisches Koordinatensystem numerisch stabiler sei.
Weitere Hinweise auf metrische Informationen bei der Ratte sind zum einen die Tatsache, daß das Tier durch unbekanntes Gebiet zum Ziel navigieren kann (Auffinden von Abkürzungen), und daß Nagetiere im Dunkeln zu ihrem Ausgangspunkt auf einer geraden Strecke vom Zielpunkt zurückkehren können, nachdem sie einem verwinkelten Weg zum Ziel gefolgt sind (Suchaufgabe).
Experimente, in denen das Nagetier während einer Suchaufgabe zu einer anderen Stelle versetzt wurde, zeigen, daß der gleiche räumliche Versatz bei der Ankunft im Ziel erhalten blieb.
Ein Hinweis auf Anwendung metrischer Repräsentationen ist die Erklärung der Tatsache, daß „place cells“ ca. 120 msec feuern bevor die Ratte die entsprechende Stelle erreicht hat. Es wird vermutet, daß die Ratte Ihre Position anhand der beabsichtigten Bewegung vorhersagt [TouWan94].
Die erstaunliche Leistungsfähigkeit dieser Navigationsmethode ist nicht leicht auf ein Robotersystem zu übertragen.
Ratten können sich beim Lernen an unveränderlichen Umgebungsmerkmalen orientieren. Roboter hingegen haben derart begrenzte Wahrnehmungsfähigkeiten, daß es schwierig ist, stabile von instabilen Landmarken zu unterscheiden. So ist in einer Büroumgebung beispielsweise die Position eines Stuhls oder Papierkorbs kein stabiler Hinweis. Schon Unterschiede in der Beleuchtung können das Wiedererkennen einer Landmarke für die Maschine unmöglich machen. Besondere, für den Roboter montierte, Landmarken können diese Problematik allerdings stark vereinfachen, was wiederum den Einsatz in beliebigen Umgebungen verhindert.
Während ein Roboter mit veränderten Positionen von Landmarken nur schwer zurechtkommen würde, ist die Fähigkeit der Nager im Auffinden plausibler Alternativmöglichkeiten gut entwickelt. Dies könnte unter anderem damit zusammenhängen, daß die Ratten sich Winkel und Entfernung zu verschiedenen Landmarken merken und nicht deren Positionen in einer Belegungskarte eintragen.

Eine Ortsdefinition besteht dann aus Entfernung und Winkel zu verschiedenen Landmarken.
Ratten verlassen sich bei der Zielfindung sehr stark auf die Geometrie ihrer Umgebung. So benutzen die Tiere, auch wenn andere Merkmale der Umgebung bessere Informationen zur Verfügung stellen könnten, geometrische Hinweise. Überprüft wurde dies, indem Ratten in einem achsensymmetrischen Raum an einer bestimmten Stelle Futter präsentiert wurde und danach die Ratte in einen relativ dazu gedrehten Raum gleichen Aufbaus gesetzt wurde. Dabei ergab sich, daß sich die Ratte in Richtung der Raumhälfte bewegte, die den kleineren Winkelversatz gegenüber dem vorher besuchten Raum aufwies.
Wichtig bei der Nachahmung von Tieren ist die Erweiterung der Wahrnehmungsfähigkeiten. So erreichen Farbkameras deutlich größere Reichweiten als beispielsweise Sonarsensoren, jedoch nicht das 300° umfassende Gesichtsfeld einer Ratte. Ohne die größere Reichweite und Informationsvielfalt, die das Bild einer Farbkamera liefert, ist eine Simulation der Rattennavigation mit ähnlicher Toleranz gegenüber Umgebnugseinflüssen nicht möglich.

2.2 *Ameisen*

Wüstenameisen benutzen visuelle Stimuli zur Zielfindung.
Als Bezugslandmarke benutzen sie das Polarisationsschema des Himmels, das sich während der vergleichsweise kurzen Ausflüge nicht ändert. Die Funktion ist prinzipiell die eines Kompaß. Es werden keine Pheromonspuren gelegt, denen sie zurück zum Nest folgen könnten.
Sobald die Ameisen in die Nähe des Nestes zurückkehren, folgen sie einem spiralförmig verlaufenden Suchmuster zum Ziel. Dabei bringen die Ameisen gespeicherte Bilder mit aktuellen Retina-Bildern in Übereinstimmung (Winkelübereinstimmung, also größenunabhängig). Wald-Ameisen hingegen benutzen große Felsen bzw. Steine als lokale Landmarken, um ihr Nest aufzufinden.

2.3 *Bienen*

Bienen beweisen bereits erstaunliche Navigationsfähigkeiten, die es ihr ermöglichen, Distanzen von bis zu 10km rund um den Bienenstock zu überbrücken und auf effizienten Wegen zurückzukehren [WalHal93]. Zusätzlich haben Bienen besondere Fähigkeiten für Kurzstreckennavigation, die es Ihnen ermöglichen, gezielt den Eingang zum Bienenstock oder zum futtertragenden Teil von Blüten zu finden.
Über lange Distanzen navigieren Bienen durch Pfadintegrierung. Dazu überwachen sie die Richtung in die sie fliegen mittels eines auf die Richtung der Sonne kalibrierten Kompaß. Diese Werte werden auf einen auf den Bienenstock weisenden Vektor integriert. Da Pfadintegrierung ein ungenauer Prozeß ist, wendet die Biene in der Nähe des Zielpunktes einen visuellen Blidervergleich an. Dazu nimmt die Biene Schnappschüsse wichtiger Regionen in ihrer Umgebung auf. Nähert sich die Biene einer der so erfaßten Regionen, ruft sie relevante Schnappschüsse ab und vergleicht sie mit den Bildern auf der Retina, um entsprechende Bewegungsvektoren zu generieren, die sie beispielsweise genau zum Eingang des Bienenstocks führen.
Versuche von Cartwright und Collett zeigen, daß Bienen nahezu unverarbeitete Bildern speichern. Die Bilder enthalten keine explizite Information über die Entfernung zu den enthaltenen Objekten. Die Biene speichert zusammen mit dem

Bild die Flugrichtung zum Zeitpunkt der Erstellung des Schnappschusses. Ein aktueller Schnappschuß wird nur mit dem gespeicherten Bild verglichen, wenn die dazugehörige Flugrichtung übereinstimmt [WalHal93].

Bienen benutzen zum Auffinden der Heimflugrichtung den Stand der Sonne als Referenz. Angeblich finden aber auch hier zusätzlich Polarisations-Schemata des Himmels Verwendung.

Bienen lernen die winkelabhängige Größe von Landmarken aus Sicht der Futterquelle und benutzen diese zur Heimfindung. Die Objektrichtung auf der Retina ist dabei wichtiger als die tatsächliche Größe.

Kapitel 3 Bisher eingesetzte Verfahren

Unterschiedliche Ziel- und Voraussetzungen bei der Entwicklung von Navigationsverfahren für autonome mobile Roboter brachten sehr verschiedene Lösungsansätze zutage. Die meisten davon generieren eine Umgebungskarte. Die Systeme orientieren sich mit Hilfe von Landmarken oder anderen Umgebungsmerkmalen. Einige Lösungen davon stellen wir nun vor.

3.1 Navigation mit Besetztheitsgittern

Besetztheitsgitter (häufig auch als occupancy grids bezeichnet) sind eine metrische Repräsentation der Umgebung. Über die Umgebung wird dabei ein Gitter gelegt, dessen Zellen eine definierte Kantenlänge besitzen. Für jede Zelle ermittelt das Navigationssystem nun einen Wahrscheinlichkeitswert dafür, daß die Zelle ein Hindernis beinhaltet. Ein Vorteil einer metrischen Umgebungsrepräsentation ist die Tatsache, daß in erkundeten Bereichen Abkürzungen ermittelt werden können, die eine Wegplanung auf der Karte optimieren. Bei großen zu kartographierenden Bereichen wird der Speicheraufwand allerdings sehr groß.

3.1.1 Navigation mit Landmarken

3.1.1.1 Jonathan H. Connell, Sridhar Mahadevan, OBELIX

Connell und Mahadevan trainieren ihre Roboterplattform OBELIX in einer realen Umgebung [ConMah93]. Der Roboter agiert verhaltensbasiert und wird mittels Reinforcement-Lernen trainiert und bildet wahrgenommene Zustände auf gewünschte Aktionen ab. Die Leistungsfähigkeit des Roboters soll optimiert und die Lerngeschwindigkeit erhöht werden, indem durch Reinforcement-Lernen eine Kostenfunktion minimiert wird und dabei häufiger als üblich eine Erfolgs-Mißerfolgsmeldung an den Controller gegeben wird (nicht nur bei Erfolg/Mißerfolg der Gesamtaufgabe). Problematisch ist, daß Aktionen des Roboters, die nur scheinbar zum Ziel führen, ein lokales Kostenminimum darstellen können. Nach Connell et al. beschleunigen dabei komplexe Belohnungsfunktionen den Lernvorgang.

OBELIX hatte als Aufgabe, einen Kasten vor sich herzuschieben.

Die Steuerung wurde mit einer Variation des Q-Lernens trainiert. Sie ist verhaltensbasiert und kennt drei Verhaltenselemente:

- suche potentielle Kästen
- schiebe den Kasten durch einen Raum
- entkomme verfahrenen Situationen.

Jede der drei Verhaltensweisen wird einzeln mittels Q-Lernen trainiert. Es gibt für jede Situation eine dominierende Verhaltensweise.

Während des Betriebs werden Sensordaten in ein quadratisches Besetztheitsgitter eingetragen, das aus 144 Parzellen besteht. Die in den Zellen enthaltenen Werte sagen aus, wie sicher der Roboter ist, daß die entsprechende Parzelle besetzt ist. Um die 144 Werte schnell zu verarbeiten, wurde die Interpretation so ausgeführt, daß jede Zelle ein Votum für eine bestimmte Aktion abgibt, die sie für angebracht hält. Das Besetztheitsgitter wird als Q-map bezeichnet.

Jedes der drei Verhaltensmodule hat eine eigene Q-map. In jedem Verarbeitungsschritt ist nur eines dieser Module aktiv, so daß auch nur die jeweilige Q-map ausgewertet werden muß. Jede Parzelle gibt in der Q-map des aktiven Verhaltensmoduls ein Votum für eine auszuführende Aktion. Eine Aktion gewinnt die Abstimmung und wird ausgeführt. Die Zellen, die richtig gestimmt hatten, werden belohnt. Die Q-maps der anderen Verhaltensmodule werden nicht aktualisiert, damit sie nicht dafür bestraft werden, daß eine andere Verhaltensweise dominiert hat. In Versuchen von Connell et al. hat sich gezeigt, daß das beschriebene Verfahren der Zustands-Clusterung überlegen ist.

Durch Verwendung von Aktions-Modellen, die beschreiben, wie sich die Sensordaten nach Durchführung einer jeweiligen Aktion ändern würden, wird versucht, Wissen auf verschiedene Aufgabenstellungen zu übertragen.

Zur Planung von Fahrtrouten erstellt das System eine Belegungskarte, in die alle entdeckten Landmarken eingetragen werden. Landmarken sind für dieses System Auftauchen und Verschwinden von Wänden aus dem Sensorbereich (durch Unterbrechungen, Abbiegungen). Zusätzlich sind Wege zwischen den Landmarken verzeichnet. Dabei wird, wie bei vielen Ansätzen der Einfachheit halber davon ausgegangen, daß Wände parallel oder rechtwinklig zueinander stehen, Kurven also rechtwinklig sind. Die Durchführung der gewählten Aktionen, wie z.B. das Durchfahren eines Korridors, erfolgt bei OBELIX verhaltensbasiert.

3.1.2 Navigation ohne Landmarken

3.1.2.1 Sebastian B. Thrun: COLUMBUS

Ziel der Roboterplattform COLUMBUS ist die möglichst effiziente Erkundung unbekannter, strukturierter Umgebungen [Thrun93]. Das System lernt anhand von Beispielen und generalisiert die Erlebnisse aus der realen Welt, welche durch Annäherungs- und Sonarsensoren wahrgenommen werden, mittels zweier neuronaler Netze.

Die Umgebung wird mit einem Sensor-Interpretations- und einem Vertrauensnetz modelliert. Ersteres liefert mit Sensordaten als Eingabe eine Schätzung der zu erwartenden Belohnung, das zweite den zu erwartenden Fehler. Die vom ersten Netz erhaltene Sensor-Interpretation wird mit diesem Fehlerwert gewichtet.

Die Netzwerke kodieren spezifische Charakteristika der Sensoren und typischer Roboterumgebungen. Die erfolgreiche Übertragung des Wissens auf andere Umgebungen zeigte sich darin, daß der trainierte Roboter auch dort in der Lage war, Hindernissen auszuweichen [Thrun93].

Nachteilig ist, daß die Netze mit Methoden für überwachtes Lernen trainiert werden müssen, was eine bekannte Umgebung voraussetzt.

Das Hauptziel, das der Roboter verfolgt, ist, auf kürzestem Wege in unerforschtes Gebiet zu gelangen. Die dazu eingesetzte Karte ist als Besetztheitsgitter ausgeführt.

Eine Besonderheit bei COLUMBUS ist, daß besonders darauf geachtet wird, daß die Erforschung unbekannten Terrains effizient erfolgt. Viele Robotersysteme suchen ihre Umgebung auf zufällige Art und Weise ab. Dies ist sehr ineffizient und bewirkt, daß die Trainingszeiten zum Erforschen der Umgebung unverhältnismäßig hoch werden. Häufig wird daher eine Methode benutzt, die dafür sorgt, daß sich der Roboter immer zu der am wenigsten erforschten Stelle begibt.

COLUMBUS soll sich auf kürzestem Weg in unerforschtes Gebiet begeben, um den Wissensgewinn zu maximieren. Um diesen Weg zu finden, vergibt COLUMBUS in seiner Belegungskarte „exploration utilities“, die angeben, wie sinnvoll eine Erkundung erscheint. Je mehr ein Gebiet erkundet wurde, desto niedriger wird der erwartete Erkundungsnutzen. Die Änderungen des Erkundungsnutzens eines Belegungsfeldes werden zu den umliegenden Feldern weiter propagiert und ihr Erkundungsnutzen damit aktualisiert. Der Roboter benutzt einen einfachen Suchalgorithmus, um die Stellen mit höchstem Erkundungsnutzen aufzufinden.

Dieser Wegplaner wird ebenfalls für das Anfahren vom Benutzer spezifizierter Orte eingesetzt, die dabei in der Karte den höchsten Erkundungsnutzen zugeteilt bekommen.

3.2 Navigation mit topologischen Karten bzw. Graphen

Topologische Karten bilden den räumlichen Zusammenhang von Stellen der Umgebung ab. Die Information über die Umgebung wird damit auf unbedingt notwendige Daten eingeschränkt. Dafür ist die Frage nach der Erreichbarkeit einer Stelle von einem Startpunkt aus einfacher zu beantworten als mit einer metrischen Karte. Dies resultiert aus dem Vorteil, eine Graphensuche zur Wegplanung auf die topologische Karte anwenden zu können. Zudem ist der Speicheraufwand vergleichsweise gering, so daß sich eine topologische Karte auch zur Repräsentation größerer Umgebungen eignet.

3.2.1 Navigation mit Landmarken

3.2.1.1 Min Meng und A.C.Kak, NEURO-NAV

Das Robotersystem von Meng et al. [MenKak93] hat zwei menschenähnliche Eigenschaften: zum einen benutzt es ein nichtmetrisches Modell der Umgebung, andererseits kann es typisch-menschlichen Kommandos folgen, wie: „Folge dem Korridor und biege an der zweiten T-Kreuzung rechts ab.“
Einige neuronale Netzwerke werden darauf trainiert, die visuelle Information zu interpretieren und Navigationsaufgaben, wie Korridordurchquerung und Landmarkendetektion, zu vollbringen.
Zur Steuerung der Roboterplattform kommt ein 68030-Rechner zum Einsatz. NEURO-NAV selbst läuft auf einer 16MIPS leistenden Plattform und leistet die Auswertung der Kameradaten bis zur Ausgabe von Steuerbefehlen in 2 Sekunden. In Innenräumen kann der Roboter mit Geschwindigkeiten von ungefähr 8 m/min navigieren
Die Steuerung des Roboters erfolgt durch ein regelbasiertes Steuerungssystem, das komplexe Navigationsaufgaben koordiniert. Einfachere Teilaufgaben der Steuerung fallen den folgend beschriebenen neuronalen Netzen zu.
Der *Korridor-Durchquerer* besteht aus einer Reihe neuronaler Netze mit folgenden Aufgaben: Korridor-Durchquerer, Linksabbieger, Rechtsabbieger. Dem Linksabbieger wird beispielsweise die Kontrolle übertragen, wenn der Roboter einer Abbiegung nach links folgen soll. Er besteht aus zwei neuronalen Netzen,

von denen eines die linke, das andere die rechte Ecke des Fußbodens überwacht. Es werden beide Ecken überwacht, damit der Roboter sich nicht zu weit um die Kurve bewegt. All diese Netze bestehen aus 3 Schichten und werden mittels Backpropagation trainiert. Die Module sorgen auch dafür, daß vom bisherigen Pfad abgewichen wird, wenn ein Hindernis den direkten Weg versperrt.
Der *Landmarken-Detektor* besteht aus einem neuronalen Netz, das Kreuzungen und Sackgassen detektieren kann und eine grobe Schätzung der Entfernung dazu gibt („weit“, „nahe“, „erreicht“).
Das zuvor erwähnte regelbasierte Steuerungssystem kontrolliert die Aktivierung und Deaktivierung der verschiedenen neuronalen Netze. Soll der Roboter ein bestimmtes Ziel anfahren, so legt der Wegplaner den Pfad fest. Detektieren die Sonarsensoren nun ein Hindernis, so unterbricht das Steuerungssystem die Bildverarbeitung und übergibt die Steuerung an das Hindernisvermeidungsmodul. Hat dies seine Aufgabe beendet, setzt der Roboter wieder seine ursprüngliche Aufgabe fort.
Die von NEURO-NAV erstellte Karte wird vom Korridor-Modellierer angefertigt und stellt einen attributierten Graphen dar, dessen Graphenknoten Korridorkreuzungen und Sackgassen repräsentieren. Die Knoten enthalten zusätzliche Informationen, wie Verweise auf die Korridorlänge und Objekte, die links und rechts an einem Korridor liegen.

3.2.1.2 Ivan A. Bachfelder und Allen M. Waxman: MAVIN

MAVIN führt sichtbasierte Navigation mittels neuronaler Netze durch und bildet das von Ratten eingesetzte Erlernen von Plätzen nach [BacWax94]. Das System erstellt Schnappschüsse von der Umgebung. Ein solcher Schnappschuß (*hippocampal place*) wird durch sichtbare Landmarken im lokalen Sensorbereich definiert. Ein Platz definiert sich als Liste von Landmarken, die nach aufsteigendem absolutem Azimuth geordnet sind.
Die „*place cells*“ von Ratten sind empfindlich für Art, Lage und räumliche Verteilung von Landmarken in einem lokalen Sensorbereich der Umgebung. Dagegen beziehen sie nicht Farben, Texturierung und Größe von Objekten mit ein.

Orte werden danach definiert, welche Landmarken in welcher ungefähren Pose sichtbar sind, oder nach ihrer Anordnung vom Sichtpunkt aus. Damit läßt sich eine zweidimensionale Karte in „place regions“ einteilen. Für jede der „place regions“ feuern unterschiedliche *place cells*.
Plätze definieren sich durch ein „Was“ und ein „Wo“.
Das „Was“ definiert einen Platz aus der Art einer Landmarke und aus der Richtung aus der sie gesehen wird. Ist eine Landmarke in einem Bereich durchgängig sichtbar, so kann die Position des Roboters in diesem Bereich allein daraus bestimmt werden, daß eine gewisse Ansicht der Landmarke zu sehen ist (sofern die Betrachtungsrichtungen eine solche Unterscheidung zulassen, wie z.B. dadurch daß eine Kugel als Landmarke Licht-/Schattenregionen aufweist).
Das „Wo“ definiert einen Platz durch eine Panoramaansicht aller Landmarken rundherum (somit unabhängig von der absoluten Drehrichtung des Roboters).
Die „Was“- und „Wo“-Komponenten werden kombiniert, bevor eine Klassifizierung durchgeführt wird.
Dazu wird das Was in *identity coding nodes* und das Wo in *spatial coding nodes* kodiert. Deren Aktivierungen werden paarweise multiplikativ von einer Anzahl Zwischenneuronen (*token nodes*) kombiniert.
Ein Fuzzy-ART-Netzwerk [BacWax94] vollzieht unüberwachtes Lernen und Erkennung von *token node*-Aktivitäts-Mustern, sobald das Panorama des aktuellen Standorts vom Roboter hinreichend aufgenommen wurde.
Eine Umgebung wird dadurch selbstorganisierend in charakteristische Ansichten (Plätze) unterteilt.
Die Eigenschaft von ART-Netzen, auch Muster wiederzuerkennen, die nicht vollständig sind, erweist sich als hilfreich. So können Plätze auch wiedererkannt werden, wenn Landmarken entfernt bzw. hinzugefügt wurden.
Im Ergebnis kartographiert der Roboter die Umgebung, indem er sie in „place regions“ unterteilt. Dies geschieht durch unüberwachtes Lernen. In dem entsprechenden Bereich feuert die zugehörige „place cell“ am stärksten. Da sich die Bereiche, in denen „place cells“ feuern, überlappen, reichen wenige davon aus, um die Umgebung zu kodieren.

Die Autoren geben an, das vorgestellte Modell sei biologisch konsistent mit der heutigen Vorstellung von Rattennavigation.

3.2.1.3 Maja J. Mataric: TOTO

Mataric setzt einen verhaltensbasierten Roboter namens TOTO ein [Matari92]. Zur Orientierung benutzt er sonarbasierte Landmarken und Wegplanung. Als Sensor kommen Sonarsensoren und ein Kompaß zum Einsatz. Letzterer zeigt mit einem 4-bit-Wert die Ausrichtung des Roboters an. Aufgrund seiner Ungenauigkeit werden jedoch nur 2 bits der Darstellung benutzt. Bei den Sonarsensoren werden nur die Kurzstrecken-Echos ausgewertet, die akkuratere Ergebnisse als Langstreckenechos liefern. Die Steuerung ist verhaltensbasiert ausgeführt. Das System besteht aus drei wesentlichen Modulen: grundlegende Navigation (wie Hindernisvermeidung und Entlangfahren an einer Wand), Landmarkendetektion und kartenbasierte Berechnungen (wie Wegplanung, Kartenkonstruktion und -aktualisierung). Der Roboter navigiert mit bis zu 12 m/min [Matari92].
Die Hindernisvermeidung teilt die nähere Umgebung in sichere und gefährliche Zonen ein. Gerät ein Objekt in eine gefährliche Zone, so wird es als Hindernis behandelt.
Landmarken sind für TOTO Wände zur rechten und zur linken, sowie Korridore. Um Fehler in den Sensorsignalen zu minimieren, beobachtet der Detektor nicht einzelne Sensordaten, sondern den Datenstrom über die Zeit.
Der Kartenalgorithmus schließlich bildet die Karte aufgrund räumlicher Zusammenhänge zwischen den Landmarken. Es kommt keine detaillierte Karte zum Einsatz, da eine geringere Anzahl an Kartendetails, die jedoch fundamentalere Eigenschaften der Umgebung abbilden, weniger anfällig für eine sich ändernde Umgebung ist. Die Karte ist ein topologischer Graph, dessen Knoten die Landmarken repräsentieren. Verbindungen zu Nachbarknoten zeigen physikalische Nähe zu anderen Landmarken an. Der erzeugte Graph ist isomorph zur Topologie der Umgebung. Die Zahl der Verbindungen eines Graphenknotens ist beschränkt, zumal der Roboter sich an Hindernissen entlang bewegt und somit eine Stelle nur aus wenigen verschiedenen Richtungen kommend besuchen kann. Implementiert wurde der Graph mit maximal 8 Verbindungen für einen Graphenknoten.

Die Wegplanung funktioniert folgendermaßen: Das gewünschte Ziel sendet ein Signal über den Graphen, das irgendwann die aktuelle Position des Roboters im Graphen erreicht. Die Autorin gibt an, diese Methode sei im Aufwand äquivalent zu einer parallelen Suche im Graphen und terminiere im worst case in linearer Zeit O(n). Signalverluste des ausgesendeten Signals ermöglichen es, den kürzesten Weg im Graphen zu bestimmen. Ein Greedy-Algorithmus sorgt für die Wahl des kürzesten Weges. Während der Roboter dem ermittelten Pfad folgt, werden Kollisionen durch das Navigationsverhalten vermieden. Die vorgestellte Methode ist biologisch durchführbar und erinnert an einige Eigenschaften des räumlichen Mappings von Ratten.
Einerseits begrenzt die topologische Darstellung zwar die Wegoptimierung auf bereits durchquerte Wege, anderen ist die Kartendarstellung für den Bediener intuitiv verständlich. Eine mögliche Erweiterung könnte ein Modul sein, das Abkürzungen durch Auswertung geometrischer Schlußfolgerungen auffindet.

3.2.1.4 David S. Touretzky, Hank S. Wan, A. David Redish

Touretzky et al. haben eine Theorie über Rattennavigation entwickelt [TouWan94], die Analogien zum Navigationsverhalten von Ratten aufweist. Sie haben mit Ihrem Modell Versuche mit echten Ratten nachgestellt und vergleichbare Ergebnisse bezüglich des Orientierungsverhaltens erreicht. In Abbildung 1 ist die funktionelle Organisation von Orientierungs- und Wahrnehmungsmechanismen der Ratte dargestellt. Kreise stellen Eingaben dar, quadratische Module verarbeiten sie. Dicke Linien kennzeichnen Hauptinformationswege des Modells.
Es werden redundante Merkmale der Umgebung verwendet. So reichen z.B. allozentrische Winkel zu zwei Landmarken aus, um eine Stelle in der Umgebung festzulegen, solange diese Stelle nicht colinear mit den Landmarken liegt. Die verwendete Winkeldifferenz zwischen Landmarken entspricht dem Winkel auf der Retina beim Tier. Dieser Winkel kann genauer bestimmt werden als die absolute Entfernung zur Landmarke.
Durchgeführt wird eine Fuzzy-Auswertung der Informationen, so daß das System beispielsweise bei Dunkelheit nur Informationen des Pfadintegrierers benutzt, bei fehlenden Pfadintegriererdaten jedoch ausschließlich visuelle Informationen zur

Lokalisation einsetzt. Dadurch wird der Pfadintegrierer reinitialisiert. Dies zeigt sich beim Vorbild in Versuchen von Collett et al., in denen gezeigt wurde, daß Ratten, die in einem abgedunkelten Raum auf eine leuchtende LED zuliefen, auch dann ihren Weg beenden konnten, wenn die LED abgeschaltet wurde.

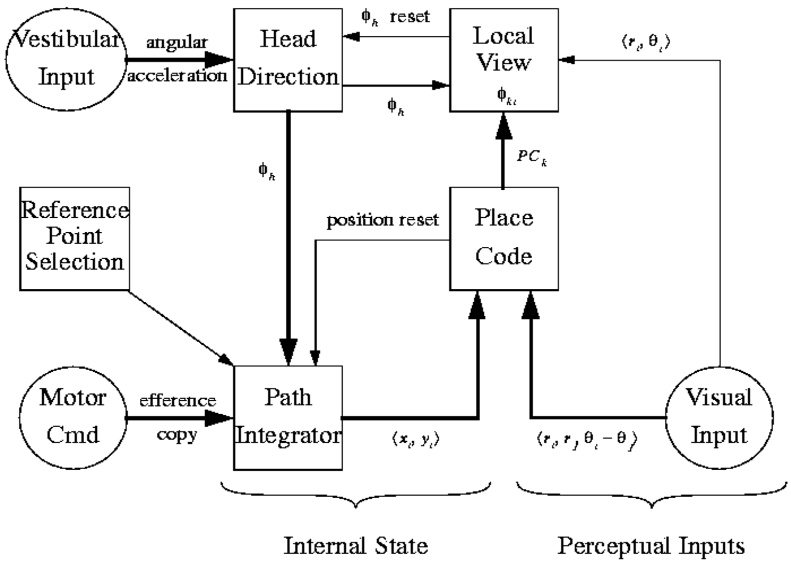


Abbildung 1 ***Funktionelle Organisation von Orientierungs- und Wahrnehmungsmechanismen der Ratte (aus: [WanTou94])***

Für die durchgeführten Versuche wurde die Umgebung nicht erkundet, sondern Place-Units wurden durch ein Lernverfahren gezielt erzeugt. Die simulierte Ratte wird in jedem Lernschritt an einer zufällig gewählten Stelle der Umgebung positioniert. Dabei werden an jeder Stelle, an der nicht mindestens 20 Place-Units feuern, weitere angelegt. Der Trainingsvorgang endet, wenn bei 20 aufeinanderfolgenden zufälligen Positionierungen keine neuen Place-Units angelegt werden müssen. Damit wird schnell eine flächendeckende Erkundung erreicht.
Touretzky et al. bilden nicht tatsächliche Bewegung oder Futtersuche nach, sondern positionieren die simulierte Ratte an 100 zufällig gewählten Stellen der Umgebung. An jeder dieser Stellen wird eine Positionsbestimmung vorgenommen. Die dabei entstehenden Positionsergebnisse werden auf ihre Verteilung hin unter-

sucht, um festzustellen, ob bei Landmarkenanordnungen, auf die Ratten unterschiedlich reagieren, im Modell vergleichbare Orientierungsergebnisse auftreten. Teile des Modells wurden auf einem Roboter (Xavier) implementiert [RedTou97].

3.2.2 Navigation ohne Landmarken

3.2.2.1 Sjur J. Vestli, Nadine Tschichold-Gürman, Henrik Andersson

Vestli et al. entwickeln eine Roboterplattform, die in einem Bürogebäude als Postverteiler eingesetzt werden kann [VesTsc93]. Das System ist mit einem Laser-Entfernungsmesser ausgestattet. Hauptproblematik bei der Anwendung dieses Sensors sind die Bestimmung des Störsignalpegels und die Signalfilterung.
Bei der Eigenlokalisation bestimmt der Roboter zuerst eine grobe Schätzung, dann seine genaue Position. Zur Situationsidentifikation benutzt der Roboter dabei die Anzahlen von Diskontinuitäten, Ecken, Wänden und Türen. Die Anzahlen von Ecken werden dabei durch Kohonen-Netze ermittelt.
Seinen Standort ermittelt das System, indem Ecken durch ein von der Hough-Transformation abgeleitetes Verfahren aus den Sensordaten extrahiert werden. Anhand der Position von Ecken kann der Roboter seinen eigenen Standort bestimmen [VesTsc93].
Um Odometriefehler auszugleichen, führt der Roboter eine Rekalibirierung an Positionen von bekannten Ecken durch. Ecken werden aus den Daten des Laser-Entfernungsmessers mittels eines auf dem Hough-Algorithmus basierenden Verfahrens extrahiert.
Aus den Sensordaten fertigt der Roboter eine topologische Karte an. Notwendige Verhaltensweisen für verschiedene Situationen werden zusammen mit den Pfadsegmenten gespeichert. Die Navigation ist situationsorientiert, die Steuerung als „RuleNet“ ausgeführt. Dieses neuronale Feed-Forward-Netz erhält als Eingabe aufbereitete Sensordaten (Situationen) und wird darauf trainiert, die dazu passende Aktion zu ermitteln (winner-takes-all). Vorteile des RuleNet sollen das einfache Einbringen von Vorwissen mittels IF-THEN-ELSE-Klauseln und ebenso einfache Extraktion erlernter Klauseln sein. Das Netz leistet eine Klassifikation von Situationen im Merkmalsraum mit Hyperwürfeln.

Das Netz erreichte bei der Klassifizierung von Situationen eine Trefferrate von 89,6% in unbekanntem Terrain. Ein mit 12 Beispielen trainiertes RuleNet (9 versteckte Neuronen) konnte in Versuchen in einer Büroumgebung mit Hindernisvermeidung fahren.

3.2.2.2 Schölkopf und Mallot

Schölkopf et al. lassen eine Kognitive Karte aus einer Sequenz von Ansichten und Bewegungsentscheidungen lernen [MalSch95]. Sie wollen in der Hauptsache aufzeigen, daß ein „view-graph“ (Ansichtsgraph) genügend Information enthält, um die Topologie- und Richtungsstruktur eines Labyrinths zu rekonstruieren [SchöMal94].
Durch Odometriefehler passiert es, daß ein Roboter nur ungenaue Kenntnis über seine Fahrtrichtung hat. Diese Problematik wird hier vermieden, indem ausgenutzt wird, daß Gänge im Versuchs-Labyrinth immer rechtwinklig zueinander sind. Drehungswinkel können also nur ein Vielfaches von 90° betragen.
Ein neuronales Netzwerk lernt den Ansichtsgraphen mit einem unüberwachten Lernverfahren. Zeitliche Ansichtssequenzen werden dabei in Verbundenheit im Netzwerk übersetzt. Das Netzwerk nutzt dann sein Wissen der topologischen und direktionalen Struktur des Labyrinths, um Erwartungen über die zukünftig wahrgenommenen Ansichten zu generieren und so die Wahrnehmungsleistung zu verbessern.

3.2.2.3 Buhmann, Burgard, Cremers, Thrun: The Mobile Robot Rhino

Der Roboter von Buhmann et al. [BuhBur95] soll Navigation und Manipulation in Innenräumen erlernen. Auch hier kommen neuronale Netze zur Interpretation der Sonarsensor-Information zum Einsatz. Die Steuerung ist reaktiv.
Ausgestattet ist der Roboter mit 24 Sonarsensoren, einem Stereo-Farbkamera-Kopf und zwei i486-Rechnern. Konstruiert wurde Rhino mit Hinblick auf den AAAI-Wettbewerb für mobile Roboter 1994.
Das Robotersystem arbeitet mit einem Besetztheitsgitter und der Annahme, daß alle Korridore untereinander Winkel aufweisen, die ein Vielfaches von 90° darstellen.

Fehler in der Odometrie werden kompensiert, indem Daten der Sonarsensoren mit denen aus dem Besetztheitsgitter verglichen werden, so daß eine Rekalibirierung der Radsensoren möglich ist.
Das System stellt zusätzlich eine aktuelle topologische Karte zur Verfügung, die ständig aus dem aktuellen Stand des lokalen Besetztheitsgitters generiert wird.
Zur Unterstützung der sonarbasierten Hindernisvermeidung können ein Kamerasystem und Bildverarbeitung mit einfachen Filtern eingesetzt werden.
Das Vision-System erfüllt im wesentlichen zwei Aufgaben. Es erkennt zum einen wichtige Objekte einer Büroumgebung.
Zum anderen unterstützt es Hindernisvermeidung und Navigation des Roboters durch lokale Besetztheitsgitter der Umgebung und relative Positionen von Objekten zum Roboter.
Die Bilder der Kameras werden Tiefpaß-gefiltert und auf eine niedrigere Bildauflösung heruntergerechnet. Diese Aufgabe erledigt einer der beiden onboard-Rechner. Im Folgenden wird eine Segmentierung des Bildes durchgeführt.

3.2.2.4 A. Kurz: ALEF

Der Roboter ALEF von A. Kurz benutzt 24 in einem Ring angeordnete Sonar- und Radsensoren zum Erfassen seiner Umgebung [Kurz95].
Die Informationen der Sensoren werden auf zwei Ebenen verarbeitet. Auf der niedrigeren senso-motorischen Ebene werden Grundfertigkeiten, wie Hindernisvermeidung, erlernt, die es dem Roboter ermöglichen, sich in seiner Umgebung kollisionsfrei zu bewegen. Die höhere, symbolische Ebene dient der Erstellung von Karten und der Ausführung von Plänen.
Die senso-motorische Ebene der Steuerung besteht aus drei hierarchisch angeordneten Untersystemen.
Das erste ist ein einfaches Notsystem, das den Roboter stoppt, wenn eine Kollision mit einem Hindernis unmittelbar bevorsteht.
Die anderen beiden Systeme werden dem Roboter antrainiert. Davon ist eines für zielorientiertes Fahren (d.h. beispielsweise gerades Zufahren auf einen Zielpunkt) zuständig, das andere für Hindernisvermeidung, damit das Notsystem zur Kollisionsvermeidung nicht aktiv werden muß. Hierfür wurden jeweils die Daten von 13

Sonarsensoren eingesetzt. Trainiert wurde die Klassifikation des 13-dimensionalen Vektors in nur wenige „Systemzustände" mit Hilfe unüberwachten Lernens, dem Restricted Coulomb Energy Classifier (RCE).
Die höhere symbolische Ebene beinhaltet diverse Untersysteme, die um eine Kartenstruktur herum gruppiert sind. Eines der Untersysteme erzeugt die Karte aufgrund symbolischer Daten, die von einem anderen Untersystem aus den Sensordaten extrahiert werden. Ein weiteres ermittelt die aktuelle Position des Roboters. Das Wegplanungssystem empfängt Befehle, interpretiert sie und generiert mit Hilfe der Karte eine Reihe von Unterzielen. Liegen keine Befehle vor, übernimmt ein Untersystem die Erkundung der Umgebung.
Die Karte besteht aus einem Graphen, dessen Knoten freie Bereiche der Karte repräsentieren. Der freie Raum wird in Situationsbereiche unterteilt, die sich von anderen durch mangelnde Ähnlichkeiten in den Sensordaten unterscheiden lassen. Jeder dieser Situationsbereiche wird durch einen Knoten im Graphen dargestellt, der zusätzlich eine Positionsangabe enthält. Um verschiedene Situationen zu unterscheiden, also Klassen von Sensordaten zu bilden, werden Kohonen-Netze oder RCE-Klassifizierer (Restricted Coulomb Energy Classifier) benutzt. Die Fehler in der Odometrie des Roboters werden dadurch ausgeglichen, daß an bekannten Ekken die Positionsangaben des Roboters neu angeglichen werden und ein erweitertes Kalman-Filter auf die gestörten Meßdaten angewendet wird. Dieses EKF berechnet in jedem Zeitschritt eine Korrekturmatrix, die einen Kompromiß zwischen dem vorhergesagten dead-reckoning und der Position in der Situationskarte findet. Ein weiteres Problem entsteht, wenn noch keine Situationskarte vorhanden ist. Die Positionsangaben der Knoten können dann nur aus der fehlerhaften Odometrie des Roboters bestimmt werden, Positionskorrekturen aber wiederum nur aus der Situationskarte. In der Praxis hat sich herausgestellt, daß die Positionsschätzung des EKF exakt genug ist, solange ein korrektes Matching hergestellt werden kann. Nachteilig ist, daß das System nur bei Karten bis ca. $200m^2$ zuverlässig arbeitet, da ansonsten die Odometrieabweichungen zu groß werden und kein einwandfreies Matching erzielt wird.
Bei der Navigation ermittelt der Roboter mit Hilfe einer Graphensuche einen möglichen Weg durch den Graphen und verwendet die einzelnen Knoten als Un-

terziele zur Wegfindung. Es hat sich gezeigt, daß die Bewegungen des Roboters weniger eckig sind, wenn nur jeder zweite Unterknoten als Unterziel benutzt wird.

3.3 Verhaltensbasierte Ansätze ohne Kartengenerierung und ohne Landmarken

Robotersteuerungen ohne Kartengenerierung eignen sich nur für den Einsatz in räumlich extrem begrenzten Umgebungen. Einige Systeme arbeiten komplett verhaltensbasiert und bewegen sich auf diese Weise zwar kollisionsfrei auch durch eine fremde Umgebung, können jedoch nicht gezielt weiter entfernte Orte der Umgebung ansteuern oder auch nur den Fahrtweg dorthin zurückverfolgen.

3.3.1 Thomas Röfer (Controlling a Robot With Image-based Homing)

T. Röfer generiert mit seinem Roboter eindimensionale 360°-Ansichten der Umgebung [Röfer93]. Es steht nur ein Helligkeitssensor zur Verfügung, der um 360° geschwenkt werden kann.

Das Modell ist vom biologischen Modell der Biene inspiriert und erlaubt es, den Weg von der momentanen Position zum Ziel nur anhand der eindimensionalen 360°-Ansichten zu berechnen. Diese 360°-Ansichten bestehen aus 64 Meßwerten, die in unterschiedlichen Winkeln aufgenommen werden.

Das Verfahren bringt immer zwei Bilder in Überdeckung und vergleicht kleine, normalisierte Fenster der Intensitätswerte zwischen beiden Bildern. Zusätzlich finden bei diesem Verfahren die erste und zweite Ableitung der Helligkeitsfunktion und die Topologie der Pixel (Annahme: Nachbarschaften von Helligkeitsschwankungen bleiben im Zielbild erhalten) Verwendung. Um diese Übereinstimmungen aufzufinden kommt ein modifiziertes Kohonen-Netz zum Einsatz. Die Modifikation betrifft die zyklische Behandlung der 360°-Ansicht und die Wahrung der Bild-Topologie.

Die Richtung zur Zielposition wird aus den Gewichtsvektoren des trainierten Kohonen-Netzes errechnet. Das modifizierte Kohonen-Netz wird mit dem Zielbild zu initialisiert und mit dem aktuellen Bild zu trainiert. Nach erfolgreichem Training haben sich die Vektoren des Zielbildes zu den Vektoren des aktuellen Bildes verändert.

Abbildung 2 zeigt, wie die Verschiebung von Pixeln der 360°-Ansicht die Bewegung des Roboters beschreibt.

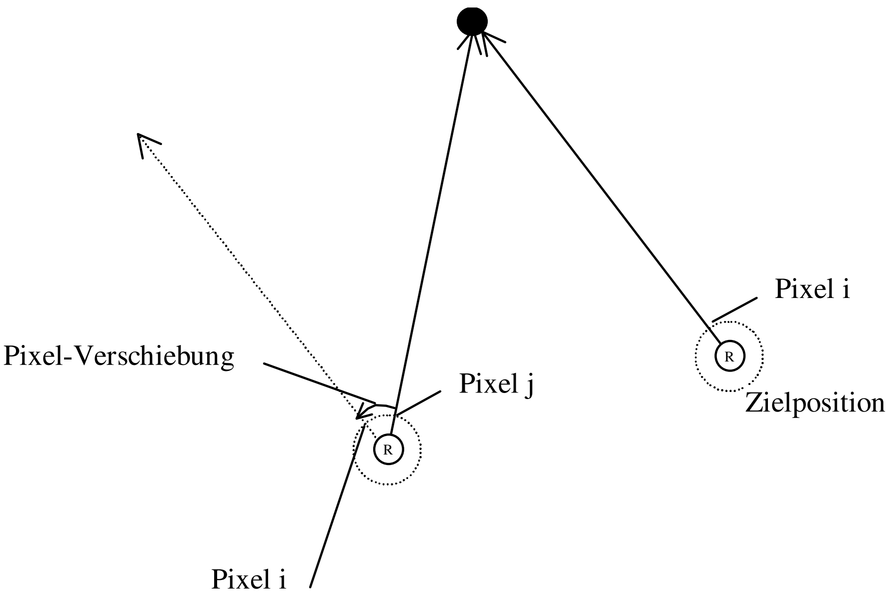


Abbildung 2 Pixelverschiebung durch Bewegung des Roboters

Um je zwei Bilder vergleichbar zu halten, kann der Roboter sich fortbewegen, ohne sich zu drehen. Wegen Ungenauigkeiten in der Fortbewegung wird trotzdem bestimmt, um welchen Winkel zwei Bilder gegeneinander verdreht sind. Durch Pixel-Verschiebung wird dieser Fehler korrigiert. Problematisch ist dabei, daß pro Ansicht nur 64 Meßwerte genommen werden, so daß die Korrekturmöglichkeit auf ein vierundsechzigstel der Gesamtumdrehung begrenzt ist.

Die Steuerungsarchitektur des Roboters ist hierarchisch. Sie besteht aus einer Robotersteuerung, einfacher Hindernisvermeidung und einem Zielbildauffindungssystem.

Der Roboter kann mit diesem Verfahren nur kürzeste Strecken zurücklegen. Ansonsten stellen plötzliche Licht-/Schattenwechsel den Roboter ebenso vor Probleme wie eine zu gleichförmige Beleuchtung.

Kapitel 4 Anforderungskatalog

Die Zielsetzung für eine Entwicklung beeinflußt die Wahl der eingesetzten Verfahren. Großen Einfluß auf die Konzeption eines Systems hat beispielsweise die Vorgabe, ob eine Umgebung, in der sich das Robotersystem bewegen soll vorgegeben wird, oder selbst zu erkunden ist. Die Wahl von Sensoren wiederum beeinflußt die Möglichkeiten des Systems.

Unser System wurde für die folgenden Anforderungen konzipiert:

Das Navigationssystem soll autonom agieren, also nicht den steuernden Einfluß eines Bedieners benötigen. Das bedeutet, daß der Roboter unbekanntes Terrain eigenständig untersucht. Bei dieser Untersuchung soll eine Karte angelegt werden, die es dem System ermöglicht, in erkundetem Gebiet auch bei vollkommener Desorientierung den eigenen Aufenthaltsort zu bestimmen und bereits erkundete Stellen gezielt anzusteuern.

Zunächst wird eine perfekte Erkennung von Landmarken vorausgesetzt. Das System muß robust auf das Fehlen oder erfolglose Erkennen einzelner Landmarken und Landmarkentypen reagieren.

In einmal erkundetem Gebiet sollten Odometriefehler ausgeglichen werden, die sich bei sehr langem Betrieb stark akkumulieren. Die Fortbewegung des Roboters soll kollisionsfrei verlaufen.

Kapitel 5 Das Konzept für den Roboter

Das Robotersystem besteht aus drei wesentlichen Komponenten.
Diese sind in Abbildung 3 dargestellt. RPI steht für Robot Programming Interface und bezeichnet die Befehls/Informationsschnittstellen, die durch den Aufbau eines Robotersystems vorgegeben sind.
Die erste Komponente besteht aus Systemen zur Datenerfassung und –vorverarbeitung. Dazu gehören die Sensoren des Roboters sowie die nötige Vorverarbeitung der erhaltenen Daten. Diese besteht beispielsweise darin, aus den Sensordaten Entfernungen für verschiedene Richtungen bis zu Hindernissen zu extrahieren und Landmarkenmarkierungen zu erkennen. Diese Markierungen müssen in Ihrer Beschaffenheit den Fähigkeiten der eingesetzten Sensoren angepaßt sein.
Die zweite Komponente, das Navigationssystem, wendet die aus dem ersten Bereich erhältlichen Daten an, um Steuerungsaufgaben für den Roboter wahrzunehmen. Dazu gehören neben gezielter Abfrage bestimmter Sensordaten die Erkundung der Umgebung nebst Generierung einer entsprechenden Karte, das Vermeiden von Kollisionen und eine Planungsmöglichkeit zum Anfahren ausgewählter Ziele im erkundeten Bereich der Umgebung.
Die dritte Komponente, das Steuerungssystem, nimmt einfache Befehle aus dem zweiten Bereich entgegen und besteht bei einem üblichen Robotersystem zum größten Teil aus Elektronik und Bewegungselektrik.
Einzelne Module der Komponenten sind in Abbildung 4 dargestellt.
Die Entwicklung des Navigationssystems KiLRoY (**Ki**eler **L**andmarken **Ro**boter) ist zentrales Ziel dieser Arbeit. Dazu werden die Komponenten zur Erfassung von Daten und Steuerung des Roboters simuliert.

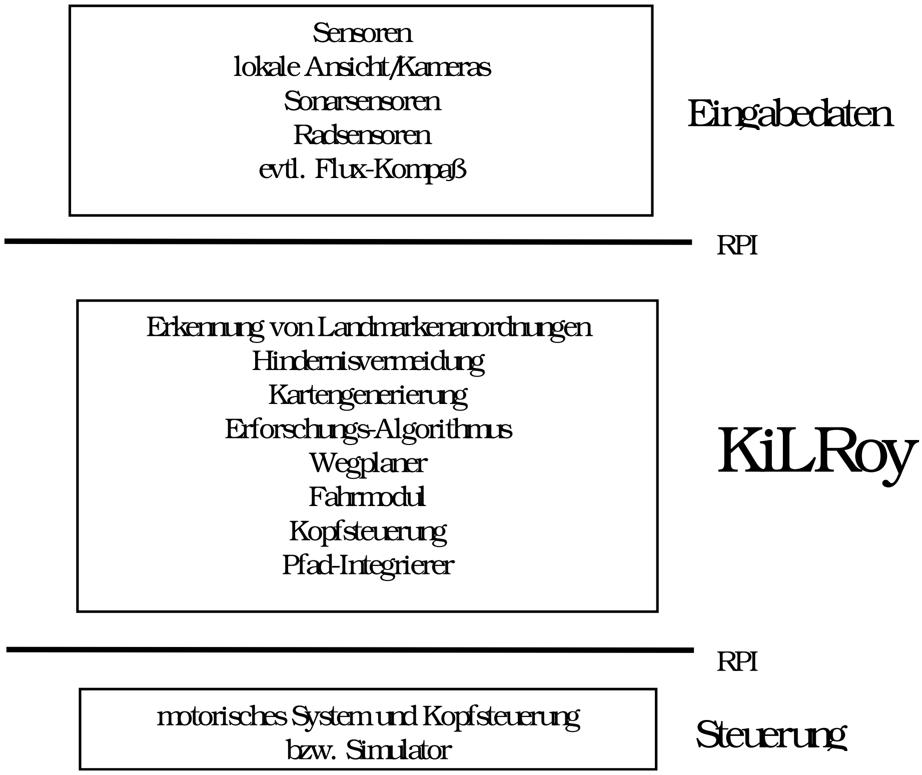


Abbildung 3 ***Die drei Komponenten eines Roboternavigationssystems.***

Vorbild für das Navigationsverfahren ist das Orientierungsverhalten von Ratten, da sie diese Aufgabe sehr erfolgreich bewältigen und das Funktionsprinzip relativ gut erforscht ist.

Das generelle Funktionsprinzip der Positionsidentifizierung haben wir dem Modell von Touretzky et al. entnommen [WanTou94], da es sich stark am biologischen Vorbild orientiert und die durchgeführten Experimente zeigen, daß sich von Ratten bekannte Phänomene darin widerspiegeln. Dabei wurde das Modell den Erfordernissen einer Implementierung für ein tatsächliches Robotersystem angepaßt. Ferner wurde das Prinzip der Fuzzy-Auswertung derart erweitert, daß auch die erfolglose Erkennung von Landmarkentypen berücksichtigt wird.

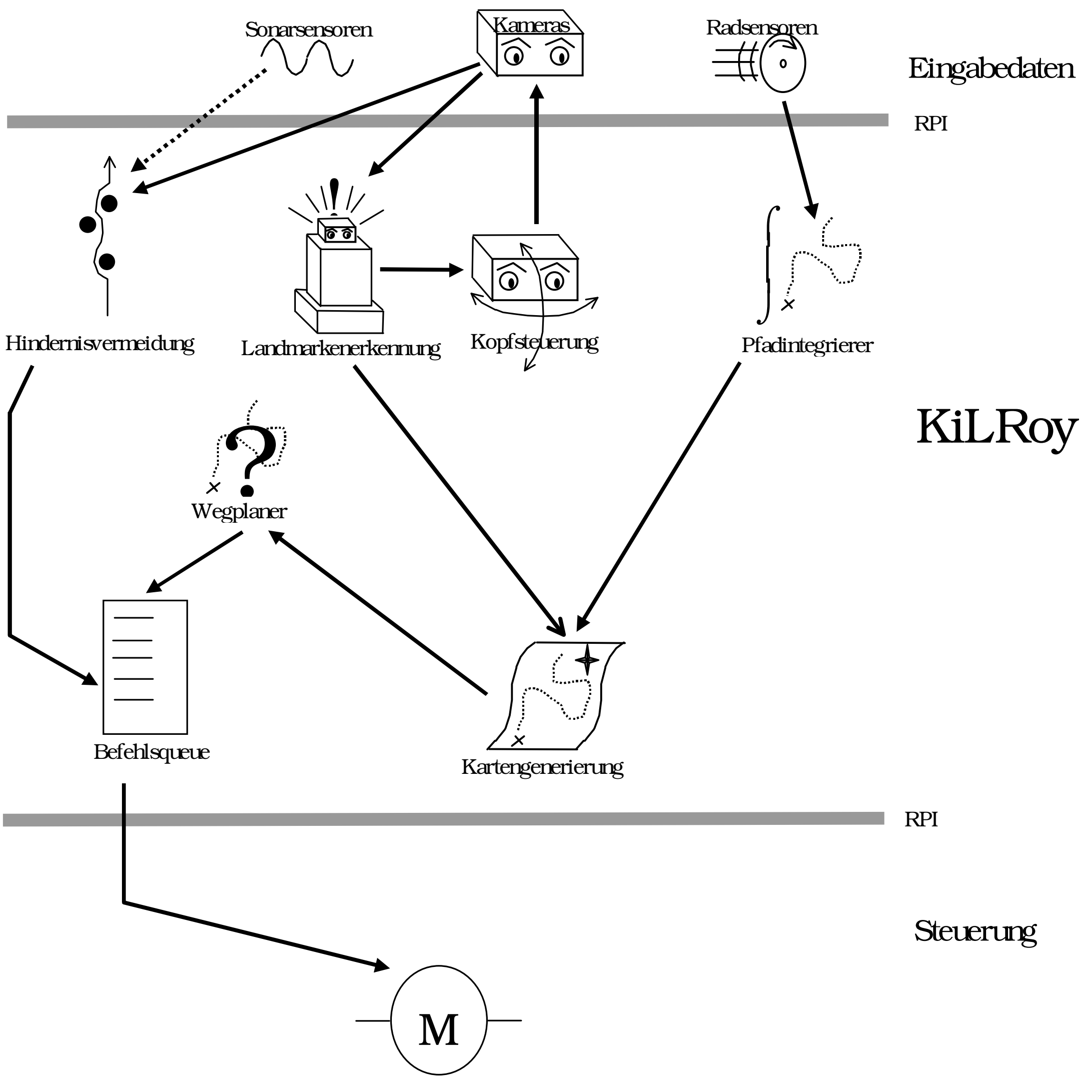


Abbildung 4 ***Module der drei Roboterkomponenten***

Während der Erkundung wird eine topologische Karte erstellt, die eine Wegplanung im erkundeten Gebiet ermöglicht. Um möglichst kurze Wege wählen zu können, werden anhand metrischer Informationen Kreuzungspunkte zwischen bereits befahrenen Strecken automatisch in die topologische Karte eingetragen, um für das erkundete Gebiet ein Kartennetz zu erhalten.

Kapitel 6 Komponenten des Modells

Das eben erarbeitete schematische Zusammenwirken der Modellkomponenten wird nun präzisiert.

6.1 Schema von KiLRoY

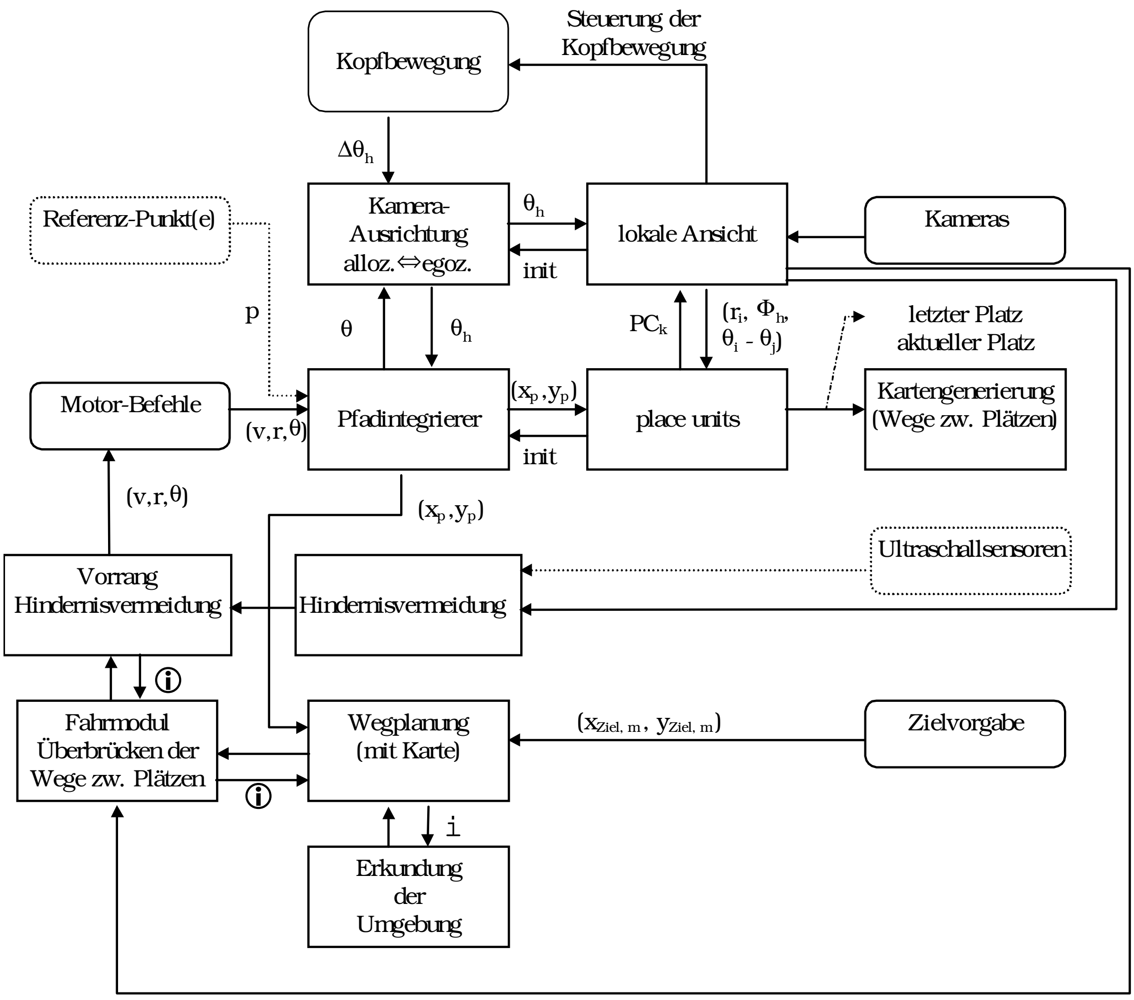


Abbildung 5 ***Detailliertes Schema des Modells***

6.2 Die Eingabedaten für das Navigationssystem

Die in Abbildung 5 durch abgerundete Rechtecke dargestellten Komponenten stellen dem System Eingabedaten zur Verfügung. Ihre Funktion erläutern wir nun im einzelnen.

6.2.1 Pfadintegrierer (Hippocampus)

Über den Pfadintegrierer (Radsensoren) gewinnt das Robotersystem bei Erkundung einer unbekannten Umgebung Informationen über die ungefähr zurückgelegte Strecke und Fahrtrichtung. Bei Dunkelheit ist der Pfadintegrierer für das Robotersystem die einzige Informationsquelle für die Positionsbestimmung.
Der Pfadintegrierer wird rekalibriert, wenn eine bekannte Landmarke erkannt wird.
Zur Positionsdarstellung kommt ein kartesisches Koordinatensystem zum Einsatz, da dessen numerische Stabilität bei dieser Aufgabe besser ist als bei der Darstellung in Polarkoordinaten [TouWan94].

6.2.2 Ultraschallsensoren

Ultraschallsensoren sind für die Navigation nach diesem Verfahren im Prinzip ungeeignet, da Ihr Öffnungswinkel zu groß und Ihre Reichweite für eine sinnvolle Umgebungswahrnehmung nicht ausreichend ist. Ferner erlauben Ultraschallsensoren nicht die Unterscheidung von Landmarken und Hindernissen bzw. Erkennung verschiedener Landmarkentypen. Wie in anderen Navigationssystemen auch, eignen sie sich jedoch als zusätzliche Datenquelle zur Vermeidung von Kollisionen.

6.2.3 Kameras

Beide Farb-CCD-Kameras sind bei unserem physikalischen System auf einem BiSight Kamerakopf montiert. Die Sichtrichtung der Kamerasysteme wird vom Kopfbewegungsmodul gesteuert. Über die Kameras sollen Landmarken in der Umgebung des Roboters erkannt werden. Zusätzlich arbeitet das Fahrmodul mit Informationen, die aus den Kamerabildern gewonnen werden. Die Kameras liefern damit Informationen zur Hindernisvermeidung. Unser Simulator versorgt das Navigationssystem mit vorverarbeiteten Kameradaten, die sichtbare Landmarken und Entfernungen zu Hindernissen enthalten.

6.2.4 Kopfbewegung

Das Kopfbewegungsmodul wird vom Modul „lokale Ansicht" gesteuert. Es bedient den simulierten Kamerakopf bzw. enthält die Möglichkeit die Steuerung für einen tatsächlichen Kamerakopf anzubinden. Das Modul gibt die Änderung der Sichtrichtung an das Kameraausrichtungsmodul weiter, das eine Umrechnung von allozentrischen in egozentrische Koordinaten (und umgekehrt) ermöglicht.

6.2.5 Lokale Ansicht

Zur Bestimmung der Position werden sichtbare Landmarken benutzt, nicht jedoch Oberflächentexturen von Gegenständen o.ä. Das Modul versucht, laufend Landmarken zu erkennen, und gibt Entfernung und Richtung aller aktuell erkannten Landmarken an das Place-Unit-Modul weiter. Um die direkte Umgebung zu erfassen, werden Steuerungsbefehle an das Kopfbewegungsmodul gegeben.

6.2.6 Motorbefehle

Motorbefehle sollten zur Kontrolle der Plausibilität der Pfadintegriererwerte herangezogen werden. Fährt der Roboter geradeaus in eine bestimmte Richtung, so kann mittels der bekannten Geschwindigkeit die Zielkoordinate bei optimaler Haftung der Räder ermittelt werden. Der errechnete Kurs sollte dann auch tatsächlich geradlinig in Richtung der bekannten Ausrichtung des Roboters ausgeführt worden sein. In der Simulation werden die Pfadintegriererwerte basierend auf den Motorbefehlen errechnet.

6.2.7 Zielvorgabe des Bedieners bzw. Erkundungsalgorithmus

Der Benutzer kann dem Robotersystem ein Ziel vorgeben, das angefahren werden soll. Liegen für den Roboter keine Kommandos vor, so wird das Erkundungsmodul aktiviert, das durch weitere Erkundung der Umgebung eine Ausarbeitung der internen topologischen Karte ermöglicht.

6.2.8 Referenzpunkte für Koordinatenangaben

In einer Ausbaustufe können wie von Touretzky et al. vorgeschlagene Referenzpunkte für Koordinatenangaben benutzt werden, um Positionsdaten für mehrere, eventuell voneinander unabhängige, Umgebungen zu verwalten. Arbeitet das Ro-

botersystem beispielsweise abwechselnd in verschiedenen Gebäuden, legt das System dann für jede der beiden Umgebungen eigene Referenzpunkte an. Alle Koordinatenangaben, die innerhalb der jeweiligen Umgebung benutzt werden, erfolgen dann relativ zum Referenzpunkt für diese Umgebung.
Ferner läßt sich durch wiederholtes Anlegen neuer Referenzpunkte beim Erkunden einer unbekannten Umgebung der absolute Koordinatenfehler der von den Place-Units gelernten Koordinaten verringern, da mit Anlegen eines neuen Referenzpunktes die relativen Abweichungen bezüglich des neuen Referenzpunktes entfallen. Erfolgen alle Koordinatenspeicherungen während einer Umgebungserkundung anhand eines einzigen Referenzpunktes und akkumulieren sich währenddessen Odometriefehler immer weiter auf, so ist gegen Ende der Erkundung der zu erwartende Fehler neu angelegter Koordinatenangaben bezüglich dieses einzelnen Referenzpunktes relativ groß. Werden während der Erkundung neue Referenzpunkte angelegt, ist der Fehler von Koordinatenangaben bezüglich dieser neuen Referenz kleiner, da Odometriefehler sich seit Anlegen dieses weiteren Referenzpunktes weniger akkumulieren konnten.

6.3 Die Aufgaben der einzelnen Navigationsmodule

6.3.1 Place-Units

In unserem Modell bilden wir durch Place-Units „place cells“ von Ratten nach. Eine Place-Units entsprechen dabei nicht zwangsweise einer „place cell“. Vielmehr sollte eine Place-Unit als Ansammlung von Zellen betrachtet werden, die in ihrer Gesamtheit die Funktionalität einer Place-Unit erzielen [TouWan94].
Jede Place-Unit von KiLRoY feuert für den Aufenthalt an einer bestimmten Position oder das Vorhandensein von Landmarken an entsprechenden Betrachtungswinkeln.
Zu jeder Place-Unit werden für zwei sichtbare Landmarken i und j die allozentrischen Ortskoordinaten, die Entfernungen (r_i, r_j) zum Roboter und allozentrischen Betrachtungswinkel (θ_i, θ_j), deren Typ, der Betrachtungswinkel zwischen zwei

beliebigen weiteren Landmarken (θ_k-θ_l) und die Typen dieser beiden Landmarken gespeichert.
Zur Kodierung einer typischen Büroumgebung werden ca. 10^5 Place-Units benötigt. Für Testumgebungen mit mehreren einfachen Räumen ist mit ca. 10^4 Place-Units zu rechnen. Für eine zuverlässige Positionierung ist zudem eine flächendekkend erfolgte Erkundung des Gebietsbereiches nötig.
Die Koordinatenangaben werden relativ zu einem dem Roboter bekannten Referenzpunkt angegeben. Die Feuerrate einer Place-Unit gibt Aufschluß über den Grad der Übereinstimmung mit der aktuellen Position.
Nicht vorhandene Terme zur Aktivierung der Place-Units werden nicht berücksichtigt (fuzzy evaluation, im weiteren als Fuzzy-Auswertung bezeichnet). So wird bei Dunkelheit nicht die „lokale Ansicht“ berücksichtigt, wohl aber die vom Pfadintegrierer gelieferten Positionskoordinaten. Sind wegen Desorientierung die Koordinaten des Pfadintegrierers ungültig, so wird nur mit visueller Eingabe gearbeitet.
Zur Bestimmung des Standorts erstellt der Roboter einen Merkmalsvektor für seinen aktuelle Position, der im Aufbau einer Place-Unit gleicht. Dieser Vektor enthält ebenfalls die Radsensorposition und zwei zufällig gewählte Landmarkenpaare mit den dazugehörigen Werten. Dieser Vektor dient als Eingabe für das Place-Unit-Netz. Ähnlichkeit zwischen diesem Merkmalsvektor und einer Place-Unit resultiert in größerer Aktivierung der Place-Unit, die, mit ihrer Aktivierung gewichtet, für ihren Standort als Positionsschätzung stimmt. Die gewichtete Vektorsumme der sich so ergebenden Verschiebungen der Positionsschätzung resultieren nach einigen Iterationen in der endgültigen Positionsschätzung als Ausgabe des Netzes.
Ist ein Ort der Umgebung noch nicht durch Place-Units erfaßt worden, werden für mögliche Landmarkenkombinationen Place-Units neu erzeugt. Einen Ort der Umgebung, der durch Place-Units erfaßt wurde, bezeichnen wir als „Platz“. Werden Landmarken an einem Platz entfernt, gibt es zumeist noch genügend viele Paarungen, die ausreichen, um den Platz wiederzuerkennen. Bei zu wenig sichtbaren Landmarken können bei Verfügbarkeit Pfadintegriererkoordinaten hinzugezogen werden.

6.3.2 Kartengenerierung zu den ermittelten Place-Unit-Koordinaten

Das Kartengenerierungsmodul merkt sich jede von den Place-Units erkannte Position (Platz). Für jeden Platz werden für die Karte zusätzliche Informationen gespeichert. So wird als wichtigste Information zwischen dem zuletzt erkannten Platz und dem aktuellen eine Verbindung geschaffen, die anzeigt, daß zwischen den beiden erkannten Plätzen eine direkte Verbindung besteht. Zuvor eingetragene Verbindungen werden bei Überkreuzung miteinander verbunden, um ein Verbindungsnetz zu schaffen, das eine effiziente Navigation an gewünschte Ziele ermöglicht. Die Pfadintegriererkoordinaten der Punkte sind bereits in den Place-Units gespeichert.

Als Besonderheit werden jeweils geradlinige Streckensegmente angelegt. Eine lange Kurve wird in eine Reihe geradliniger Strecken zerlegt. Jeweils an den Streckenübergängen legen wir einen neuen Platz auf der Karte mit entsprechenden Place-Units an. An diesen Plätzen ist es für den Roboter je nach Wegverlauf notwendig, eine Drehung durchzuführen.

Wird der Roboter zwischen der Erkennung zweier Orte desorientiert, erstellt das System keine Wegverbindung zwischen zuletzt und gerade erkanntem Ort, da nicht notwendigerweise ein direkter Pfad zwischen beiden existiert.

6.3.3 Wegplanung

Die Wegplanung ist nicht implementiert, in den Datenstrukturen der Simulation jedoch vorgesehen. Folgend geben wir eine Übersicht der angestrebten Funktionsweise.

Die Wegplanung baut auf der topologischen Karte auf. Ist der Zielpunkt bereits in kartographiertem Gebiet, so kann mit Hilfe dieser Information z.B. durch Graphensuche oder das Entsenden eines „Signals“ vom Zielpunkt aus (das erste am Startpunkt eintreffende Signal beschreibt den Weg der Wahl) ein Weg zum Ziel ermittelt werden. Das Wegplanungsmodul gibt für das als nächstes zu durchfahrende Segment Befehle an das Fahrmodul weiter, das dann die entsprechende Wegstrecke zurücklegt.

Ist der gewünschte Weg blockiert oder greift die Hindernisvermeidung ein, wird das Streckensegment in der topologischen Karte als zeitweise blockiert markiert, die anliegenden Fahrbefehle verworfen und das Wegplanungsmodul ermittelt einen neuen Weg, sofern alternative Zugänge zum Ziel existieren. Ist dies nicht der Fall, werden alle als blockiert markierten Streckensegmente in der Karte freigegeben und ein weiterer Versuch unternommen, das Ziel zu erreichen.

6.3.4 Fahrmodul

Das Fahrmodul führt die vorliegenden Befehle aus, die anhand der Wegplanung (topologische Karte) oder durch das Erkundungsmodul ermittelt wurden. Die Hindernisvermeidung ist Teil des Fahrmoduls und detektiert mögliche Kollisionen anhand der lokalen Ansicht

6.3.5 Kameraausrichtung

Die Daten vom Kopfbewegungsmodul, die Änderungen der Blickrichtung anzeigen, werden hier zur Umrechnung von allozentrischen in egozentrische Koordinaten und zurück benutzt. Durch Informationen des Pfadintegrieres wird auch die momentane Ausrichtung des Roboters berücksichtigt.

6.3.6 Hindernisvermeidung

Die Hindernisvermeidung überprüft für auszuführende Steuerbefehle anhand der vorliegenden lokalen Ansicht, ob eine Ausführung des Befehls Aussicht auf Erfolg hat (also kein Hinderniskontakt stattfindet). Ist dies nicht der Fall, so werden dem Fahrmodul anliegende Befehle nicht ausgeführt und ein evtl. geplanter Weg muß erneut ab der letzten Position vorgegeben werden.

6.3.7 Vorrang Hindernisvermeidung

Dieses Modul ist Teil der Hindernisvermeidung und sorgt dafür, daß geplante Aktionen verworfen werden, wenn eine mögliche Kollision festgestellt wird.
Das Wegplanungsmodul wird verständigt und ermittelt einen neuen Pfad zum Zielpunkt.

6.3.8 Erkundungsmodul

Das Erkundungsmodul wird vom Fahrmodul informiert, sobald keine Fahrbefehle mehr auszuführen sind. Daraufhin beginnt das Erkundungsmodul damit, den Roboter durch die Umgebung zu bewegen und bei Erreichen eines Hindernisses in

eine neue Fahrtrichung mit hinreichendem Bewegungsspielraum auszuweichen. Die neue Richtung (links/rechts) wird dafür zufällig bestimmt, der Winkel der Drehung jedoch wird anhand der lokalen Ansicht ermittelt, um eine Richtung zu wählen, die eine sinnvolle Bewegung des Roboters ermöglicht.
Die Befehle des Erkundungsmoduls werden jeweils einzeln in die Befehlsqueue gestellt. Erst nach Abarbeitung eines Befehls wird anhand der aktuellen lokalen Ansicht entschieden, wie weiter erkundet wird.
Neue Place-Units werden automatisch im Place-Unit-Netz angelegt und in der Karte berücksichtigt.

6.4 Desorientierung des Robotersystems

Bei Desorientierung des Robotersystems (fehlerhafte Positions- und Ausrichtungsdaten) wird anhand von Entfernungen, Winkeldifferenzen und Typen gerade sichtbarer Landmarken die Position des Roboters in der Umgebung bestimmt.
Jede Place-Unit hat zugehörige Pfadintegriererkoordinaten gespeichert. Wird eine größere Anzahl von Units aktiv, so ergibt die Schätzung der Roboter-Position Inkonsistenzen. Die Verteilung der nach Aktivität gewichteten Koordinaten wird gemittelt. Diese Positionsschätzung wird dann wieder den Place-Units zugeführt, um die Schätzung zu präzisieren. Von der Schätzung extrem weit entfernte Koordinaten läßt das System aus der aktiven Menge der Place-Units herausfallen. Dieser Prozeß wird wiederholt, bis die Koordinatenschätzung konvergiert.
Anhand von gelernten Winkelinformationen einer Place-Unit und den Richtungen zu gerade sichtbaren Landmarken kann dann die Abweichung der Ausrichtungsinformation korrigiert werden.
Eine Reinitialisierung auf korrektere Werte für Position und Roboterausrichtung ist nur in bereits erkundetem Gebiet möglich. In unbekanntem Gebiet werden entsprechend neue Place-Units gebildet. Zudem wird der topologische Zusammenhang zwischen Plätzen unabhängig von der Koordinatenposition hergestellt.

Kapitel 7 Motivation der Methoden

Ein biologisch motiviertes Navigationsverfahren verspricht eine Leistungsfähigkeit, die sich grob an der des Vorbilds orientiert. Gewisse diesbezügliche Abstriche sind natürlich zu erwarten, da die Wahrnehmungsfähigkeiten eines Roboters mit denen einer Ratte noch nicht entfernt vergleichbar sind. Daher verwenden wir als simulierte Landmarken speziell gekennzeichnete Umgebungsmerkmale.

Wie sich zeigen wird, ist das Verfahren robust gegenüber Entfernen und Hinzufügen von Landmarken. Lernen neuer Plätze und Landmarken verursacht kein Vergessen früher gelernten Wissens.

Ermöglichen Umgebung und Wahrnehmungsfähigkeiten des Systems eine Unterscheidung verschiedener Landmarkentypen oder sogar einzelner Landmarken steigt die Fähigkeit zur Unterscheidung geometrisch ähnlicher Landmarkenanordnungen. Außerdem läßt sich der Einzugsradius einzelner Place-Units vergrößern und damit die Kompensation selbst starker Fehlpositionierungen erreichen. Der Einzugsradius sollte dann klein gewählt werden, wenn die Landmarken einer Umgebung nur schlecht oder überhaupt nicht unterscheidbar sind. In dem Falle sorgt ein kleiner Radius dafür, daß Plätze, die weit von der vermuteten Position entfernt sind, bei der Bestimmung des Standortes benachteiligt werden. Dabei jedoch wird davon ausgegangen, daß die Pfadintegriererkoordinaten in der Nähe der tatsächlichen Roboterposition liegen. Ist dies nicht der Fall, scheitert eine Standortbestimmung oder liefert falsche Ergebnisse.

Unser Navigationssystem kombiniert topologische und metrische Informationen, um bei der Kartengenerierung Kreuzungspunkte zwischen kartographierten Strekken zu ermitteln und in der topologischen Karte Kosten (Streckenlängen) für die Durchquerung der einzelnen Strecken angeben zu können, die eine optimale Wegplanung innerhalb erforschter Gebiete ermöglichen. Zudem können so gezielte Steuerungsbefehle für den Roboter angegeben werden.

Die lokale Ansicht bildet die direkte Umgebung des Roboters ab und stellt die Hauptinformationsquelle für die Positionsbestimmung dar. Die Konzeption läßt den Einsatz verschiedener Sensortypen zu. Die Positionsbestimmung kann die Arbeit selbst dann fortsetzen, falls widrige Umstände die Gewinnung von Daten des Hauptsensors (z.B. Kamerasystem) oder aber des Pfadintegrierers unmöglich machen. Selbst auf eingeschränkte Wahrnehmung des Hauptsensors kann reagiert werden, da eine entsprechende Nichtbeachtung des Landmarkentyps bei der Positionsbestimmung vorgesehen ist.

Kapitel 8 Umsetzung des Modells

Wir vollziehen in diesem Kapitel den Schritt vom Modell zur Objektklassenstruktur und Implementation. Die Vorgehensweise wird am Modell motiviert.

8.1 Objekt-Komponenten des Modells

Die Aufgabenteilung und Abhängigkeitsstruktur des Modells muß sich im Programmdesign wiederfinden. Wir beschreiben nun besondere Eigenschaften eingesetzter Datenobjekte und zeigen Anforderungen auf, die besonders im Design berücksichtigt wurden. Eingesetzte Datenstrukturen und Algorithmen werden hier charakterisiert.

8.1.1 Landmarken

Landmarken sind Merkmale der Umgebung, die von Lebewesen oder Robotern zur Orientierung herangezogen werden. Lebewesen haben Robotern gegenüber dabei diverse Vorteile. Unter anderem sind bestimmte Lebewesen dazu in der Lage, Besonderheiten der Umgebung eine gewisse Aufgabe zuzuordnen. Ein Mensch beispielsweise kann einschätzen, welche Objekte der Umgebung sich dazu eignen, zur Orientierung zu dienen. So ist beispielsweise bei einem Straßenschild davon auszugehen, daß es seine Position in der nächsten Zeit nicht verändert. Ein geparktes Fahrzeug hingegen ist kurze Zeit später eventuell nicht mehr auf dem Parkplatz anzutreffen. Roboter können noch nicht einwandfrei beliebige Elemente der Umgebung identifizieren. Diese Fähigkeit wird KiLRoY in der Simulation dadurch zuteil, daß die Umgebung besondere Markierungen enthält, die durch den verwendeten Sensortyp leicht als Landmarken (sogar eines gewissen Typs) zu identifizieren sind. Bei einem Kamerasystem beispielsweise könnten die Landmarken mit großen Zahlenaufschriften versehen sein die durch Segmentierung des Bildes extrahiert werden und von einem neuronalen Klassifizierer erkannt werden. In der Simulation sind Landmarken ein Sonderfall einer Klasse, die Objekte der Umgebung als Polygone beschreibt.

8.1.1.1 Ununterscheidbare Landmarken und daraus resultierende Probleme

Ist der Roboter mit nur ungenau arbeitenden Sensoren, wie z.B. Ultraschallsensoren, ausgestattet, ist eine Erkennung von Landmarken schwierig. Eine Unterscheidung mehrerer Landmarkentypen wird eventuell sogar unmöglich. Es können also nur Richtung und Entfernung zu verschiedenen Landmarken zur Erkennung herangezogen werden. Treten nun ähnliche Anordnungen von Landmarken mehrfach in der Umgebung des Roboters auf, ist über diese Information allein eine korrekte Positionsbestimmung nicht möglich. Es wird damit nötig, auch die vorangegangenen Positionen zu betrachten. Also wird dann zur Positionsbestimmung der Pfadintegrierer benötigt. Einfach läßt sich dies dadurch bewerkstelligen, daß nur die Place-Units in einem bestimmten Umkreis der (nach Pfadintegrierer) vermuteten Position bei der Bestimmung von Aktivierungen in das Ergebnis eingehen können. Damit wird es unwahrscheinlicher, im durch den Radius vorgegebenen Bereich auf weitere vergleichbare Landmarkenanordnungen zu treffen. Die Wahrscheinlichkeit einer Verwechslung sinkt mit kleiner gewähltem Radius. Bei kleinen Radien ist jedoch die Kompensation eines Positionierungsfehlers nicht mehr möglich: Die durch die Radsensoren ermittelte Position ist wenig verläßlich. Ist diese Abweichung zu groß für den gewählten Radius, kann die aktuelle Position durch das Place-Unit-Netz nicht mehr bestimmt werden.

8.1.1.2 Eindeutig unterscheidbare Landmarken und Ihre Vorteile

Sind die Landmarken der Umgebung eindeutig unterscheidbar, wird die Odometrie für den Positionsbestimmungsprozeß vollkommen bedeutungslos, da Verwechslungen mit anderen Landmarkenanordnungen nicht vorkommen können. Daher können alle Place-Units des Netzes bei der Standortbestimmung benutzt werden. Dadurch sind auch große Positionierungsfehler kompensierbar.

8.1.1.3 Kompromiß: verschiedene Typen von Landmarken.

In der Praxis hat sich gezeigt, daß einige verschiedene Typen von Landmarken benutzt werden. Damit werden Verwechslungen von Landmarkenanordnungen hinreichend ausgeschlossen und die Nähe zum natürlichen Vorbild ist gegeben. Touretzky et al. verwenden nach einer aktuellen Veröffentlichung (siehe [RedTou97]) ebenfalls Landmarkentypisierungen.

8.1.1.3.1 Landmarkentypen in der Wirklichkeit

In der Wirklichkeit sind Landmarken besondere Merkmale der Umgebung. Dies können sowohl besondere Gegenstände oder Pflanzen, aber auch Wandbemalungen oder Bodenbeschaffenheiten sein.
In der Natur dienen Bäume, Gewässer, Futter- und Schlafplätze usw. als Landmarken
In einer künstlichen (Büro-)Umgebung hingegen können Landmarken aus Gegenständen wie Papierkörben, Schreibtischen, Stühlen, Türen, Fenstern oder aber speziellen Wandmarkierungen für den Roboter bestehen

8.1.1.3.2 Landmarkentypen in der Simulation

In der Simulation werden Landmarken durch speziell gekennzeichnete Objekte repräsentiert. Zusätzlich ist eine Typeninformation gespeichert, die eine Unterscheidung verschiedener Landmarkentypen zuläßt. Hat jede Landmarke einen eigenen Typ, so sind die Landmarken der Umgebung eindeutig unterscheidbar. Haben alle Landmarken denselben Typ, liegt der Sonderfall ununterscheidbarer Landmarken vor. In der Simulation wird von einer perfekten Erkennung des Landmarkentyps ausgegangen. In der lokalen Ansicht werden der Robotersteuerung die erkannten Landmarken zugänglich gemacht.

8.1.2 Die lokale Ansicht

Das Modul „lokale Ansicht“ steuert die Bewegung des Kamerakopfes, um sich an der aktuellen Roboterposition umzusehen. Dabei werden für jede Richtung die Entfernung zum nächsten Hindernis, als auch eventuell erkannte Landmarken in die lokale Ansicht eingezeichnet. Sowohl Steuerbefehle als auch erhaltene Daten werden simuliert, lassen jedoch prinzipiell durch Austausch einer Objektklasse den Zugriff auf ein echtes Robotersystem zu. Dadurch besteht grundsätzlich die Möglichkeit, verschiedenste Sensortypen für dieses Navigationsverfahren einzusetzen. Anhand der lokalen Ansicht trifft das Navigationssystem Entscheidungen für die Erkundung der Umgebung. Ferner werden mit dieser Information neue Place-Units im Netz angelegt und Hindernisse im Weg des Roboters erkannt.

8.1.2.1 Geeignete Sensoren für die Aufnahme einer lokalen Ansicht

Sowohl alle Steuerbefehle an den Roboter als auch alle Datenströme von den Sensoren zur Simulation durchlaufen eine Klasse, die alle Schnittstellen zwischen dem Navigationssystem auf der einen Seite und Umgebung und Roboterkomponenten auf der anderen Seite enthält. Bei Simulationsstart kann die entsprechende Klasse mitgeteilt werden, so daß auf einfache Weise verschiedene Robotersysteme als auch der simulierte Roboter angesteuert werden können. Daher benutzen wir für die lokale Ansicht eine Datenrepräsentation, die sich mit Hilfe verschiedenartiger Sensoren erzeugen läßt.

8.1.2.1.1 Stereo-Kamerasystem

Das Stereo-Kamerasystem auf dem BiSight-Kamerakopf des Instituts bietet das größte Potential, um in der Umgebung des Robotersystems Landmarken zu erkennen. Bei entsprechender Vorverarbeitung der Daten können prinzipiell auch nicht speziell montierte Landmarken zur Positionsbestimmung eingesetzt werden. Bei speziellen Markierungen ließe sich über das Kamerasystem eine Kennung (z.B. eine Nummer) durch Bildverarbeitung erkennen.

8.1.2.1.2 Laser-Entfernungsmesser

Der Laser-Entfernungsmesser hat gegenüber dem Stereo-Kamerasystem den Vorteil, daß die Bestimmung einer Entfernung zum nächsten Hindernis genauer als bei einem Kamerasystem erfolgen kann. Es wird damit allerdings schwieriger (jedoch nicht unmöglich), verschiedenste Landmarkentypen mit Hilfe der Sensordaten zu identifizieren.

Die Kombination eines Laser-Entfernungsmessers mit einem Kamerasystem wäre vielversprechend, da so die genauere Entfernungsbestimmung des Laser-Entfernungsmessers mit dem größeren Potential des Kamerasystems zur Landmarkenerkennung gekoppelt würde.

8.1.2.2 Datenstrukturen

Die lokale Ansicht verwaltet eine Liste von Objekten. Jedes verwendete Objekt ist eigentlich ein Polygon, da sich damit beliebige Objektformen verwalten lassen. Für jede Blickrichtung wird die Entfernung bis zum nächsten Hindernis ermittelt. Daraus ergibt sich ein Punkt, an dem das Hindernis beginnt. Dieser Punkt (kein Polygon!) wird durch ein umschließendes Dreieck mit definierbarer Kantenlänge dargestellt, so daß sich für nebeneinanderliegende Punkte eine zusammenhängende Struktur ergibt.

8.1.2.3 Algorithmen

In der Simulation wird die lokale Ansicht aus den Daten der Simulationsumgebung auf folgende Art ermittelt:

```
BEGIN
      errechne_absolute_tatsaechliche_Blickrichtung;
      (* Der Simulator kennt die tatsächliche *)
      (* Ausrichtung des Roboters *)
      Sichtweg=Streckensegment(Blickposition, Blickrichtung,
maxSichtweite);

      FOR ALL (kartenobjekt aus Simulationsumgebung) DO BEGIN
            IF (Schnitt(kartenobjekt, Sichtweg) nicht-leer) THEN
            BEGIN
                  ermittle_dichtesten_Schnittpunkt_vom_Roboter_aus;
                  IF bisher_am_dichtesten THEN
                        gesehenes_objekt=kartenobjekt;
            END;
      END;

      IF (gesehenes_objekt ist Landmarke
        AND gesehenes_objekt_nicht_in_Karte_eingetragen) THEN
      BEGIN
            markiere_als_eingetragen(gesehenes_objekt);
      END;

      gesehenes_objekt.trage_Richtung_vom_Roboter_aus_ein;
      gesehenes_objekt.trage_ Entfernung_zum_Roboter_ein;

      fuege_Dreieck_Markierung_für_gesehenen_Punkt_ein;
END;
```

Die graphische Darstellung der Objekte ist in display- und redraw-Methoden zusammengefaßt. Grundsätzlich können sich alle Programmelemente selbst in einem übergebenen Fenster darstellen, sofern dies für das Element sinnvoll erscheint.

Die graphische Darstellung der Karte ist für sich also sehr einfach:

```
BEGIN
      FOR ALL (kartenobjekt aus Simulationsumgebung) DO
            kartenobjekt.zeichnen(Bildschirmfenster);
END;
```

8.1.3 Das Place-Unit-Netz

Eine Place-Unit soll feuern, wenn ihre Daten eine gewisse Ähnlichkeit zu einem aktuellen Merkmalsvektor haben. Bei abnehmender Ähnlichkeit soll die Aktivierung der Place-Unit sinken. Diese Abnahme der Aktivierung folgt einer Gauß-Funktion. Das Place-Unit-Netz ist entsprechend als radiales Basisfunktionsnetz ausgeführt.

8.1.3.1 Merkmale der Place-Units

Eine Place-Unit soll feuern, wenn sich der Roboter an einer bestimmten Stelle in der Umgebung befindet. Die Stelle wird bestimmt durch:

- Die Position, die mittels Radsensoren bestimmt wurde.
- Entfernung, Richtung und Typ einer sichtbaren Landmarke
- Entfernung, Richtung und Typ einer weiteren sichtbaren Landmarke
- Typen und Winkeldifferenzen zweier sichtbarer Landmarken.

Der Merkmalsraum ist somit 11-dimensional.
P_x und P_y bezeichnen x- und y-Komponente einer Positionsangabe.
Für Landmarken $i,j \in \mathbb{N}_{\leq n}$ bezeichne r_i den Abstand zum Roboter, ω_i den absoluten Winkel zur Landmarke i vom Roboter aus und t_i ihren Typ. ω_{ij} bezeichne die zwischen den egozentrischen Betrachtungswinkeln der Landmarken i und j bestehende Winkeldifferenz. Dann gilt für den Merkmalsvektor:
$\vec{x} = \left(P_x, P_y, r_i, \omega_i, t_i, r_j, \omega_j, t_j, \omega_{kl}, t_k, t_l\right)^T$ für Landmarken i, j, k und l aus n Stück mit: $i, j, k, l \in \mathbb{N}_{\leq n}$ und $i \neq j$, $k \neq l$.

Aus der Zusammenstellung der Informationen ist leicht ersichtlich, daß für eine sinnvolle Positionierung immer mindestens zwei Landmarken sichtbar sein sollten, da sonst keine Winkeldifferenz zwischen verschiedenen Landmarken bestimmt werden kann.

8.1.3.2 Netzaufbau

Man kann sich die Place-Units in einer zweidimensionalen Anordnung vorstellen, die durch die ersten beiden Merkmalsvektorkomponenten vorgegeben sind. Dabei liegen diverse Place-Units an gleicher Position, da für eine Stelle der Umgebung immer diverse Landmarken für verschiedene Landmarkenkombinationen angelegt werden.
Für die topologische Karte werden Wegverbindungen zwischen diesen Place-Unit-Positionen verwaltet.

8.1.3.3 Bestimmung der aktuellen Position und Lernverfahren

Die aktuelle Position des Roboters wird durch das Place-Unit-Netz bestimmt. Dazu ermittelt der Roboter einen Merkmalsvektor für seine aktuelle Position. Dieser enthält neben der Schätzposition (laut Radsensoren) Winkel und Entfernungen zu verschiedenen Landmarken, sein Aufbau entspricht dem einer Place-Unit. Bei Ausfall verschiedener Informationsquellen, wie den Radsensoren oder wegen Desorientierung fehlender Information über die Ausrichtung des Roboters, fließen die entsprechenden Vektorkomponenten nicht mit in die Berechnung ein. Bei Ausfall der Radsensorik und fehlender Information über die Ausrichtung des Roboters könnten noch folgende Komponenten zur Positionsbestimmung dienen: Die Entfernungen zu zwei verschiedenen Landmarken und die Winkeldifferenz zwischen zwei weiteren Landmarken (nicht notwendigerweise unterschiedliche) sowie deren Typ.
Wie sich zeigt ist insbesondere bei nur wenigen verschiedenen Landmarkentypen damit trotzdem eine präzise Positionsbestimmung möglich.
P_x und P_y bezeichnen x- und y-Komponente einer Positionsangabe.
Für Landmarken $i,j \in \mathbb{N}_{\leq n}$ bezeichne r_i den Abstand zum Roboter, ω_i den absoluten Winkel zur Landmarke i vom Roboter aus und t_i ihren Typ. ω_{ij} bezeichne die zwischen den egozentrischen Betrachtungswinkeln der Landmarken i und j bestehende Winkeldifferenz.

Die Anzahl aktuell sichtbarer Landmarken sei n mit: $n \in \mathbb{N}$.

Definiere eine Verfügbarkeitsfunktion $\tilde{x} = \begin{cases} 1, \text{falls x verfügb.} \\ 0, \text{sonst} \end{cases}$

Um einzelne Vektorkomponenten auszublenden, wird eine Verfügbarkeitsmatrix V ($V \in \mathbb{R}^{11\times 11}$) benutzt:

$$V = \begin{pmatrix} \tilde{P}_x & & & & & & & & & & 0 \\ & \tilde{P}_y & & & & & & & & & \\ & & \tilde{r}_i & & & & & & & & \\ & & & \tilde{\omega}_i & & & & & & & \\ & & & & \tilde{t}_i & & & & & & \\ & & & & & \tilde{r}_j & & & & & \\ & & & & & & \tilde{\omega}_j & & & & \\ & & & & & & & \tilde{t}_j & & & \\ & & & & & & & & \tilde{\omega}_{kl} & & \\ & & & & & & & & & \tilde{t}_k & \\ 0 & & & & & & & & & & \tilde{t}_l \end{pmatrix}$$

für Landmarken i, j, k und l mit: $i, j, k, l \in \mathbb{N}_{\leq n}$ und $i \neq j$, $k \neq l$.

Sind alle Komponenten des Merkmalsvektors verfügbar, ergibt sich: $V = id_{11}$

Aktivierung a_i für die Place-Unit i ($i \in \mathbb{N}_{\geq 0}$):

$$a_i = e^{-\frac{(x-p_i)^T \cdot V^T \cdot V \cdot (x-p_i)}{r^2}}$$

Die Bestimmung der Aktivierung und die entsprechende Veränderung der momentanen Positionsschätzung *Mittelpunkt* in symbolischem Code lautet:

```
IF ( innerhalb_des_Ausschluss_Radius(placeunit) ) THEN
BEGIN
      Differenzvektor= x - placeunit;

      ZwErg = (Differenzvektor^T * V^T) * (V * Differenzvektor);
      Aktivierung = e^(-ZwErg / (r * r));

      IF ( Aktivierung < Schwellwert ) THEN
            placeunit.nichtmehrimRennen();
      ELSE BEGIN
            Verschiebung = placeunitPosition - Mittelpunkt;
            Mittelpunkt=Mittelpunkt + Aktivierung * Verschiebung;
      END;
END;
```

Die Positionsbestimmung verläuft in symbolischem Code so:

```
update_Availability_Matrix(Radsensor_avail);

IF (genug Landmarken sichtbar) THEN BEGIN

      Konstruiere_x_Vektor(x, LandmarkenListe, Schaetzposition);
      (* Der Vektor x stellt die Eingabe für das Place-Unit-Netz *)
      (* dar. Er enthält, wie eine Place-Unit, einen zufällig    *)
      (* zusammengestellten Merkmalsvektor.                      *)

      itstep=0;
      Mittel=Schaetzposition;

      REPEAT
            itstep ++;
            letzter_Mittel=Mittel;
            number_of_active_units=0;
            r=max_dist(itstep); (* r = maxSichtweite / itstep *)
            (* maxSichtweite, da davon die Schrittgröße der *)
            (* Roboterbewegung abhängt.                     *)

            FOR all units DO BEGIN
```

```
                Aktivierung_und_entspr_Mittelpunkt_errechnen;
          END;

          IF (itstep>70) THEN BEGIN
                (* Iteration kommt zu keinem Ergebnis, also *)
                (* neu beginnen. *)
                Alle_PlaceUnits_wieder_im_Rennen;
                Konstruiere_x_Vektor(x, LandmarkenListe,
                                        Schaetzposition);
          END;

          IF (zu_wenig_units_aktiv AND (itstep<=min_itstep)
                AND Radsensor_available) THEN BEGIN
                REPEAT
                      temporaere_Unit_anlegen;
                      mit_anderen_units_vergleichen;
                UNTIL (zu_viele_Versuche OR bis-
          her_nicht_vorhanden);

               IF (bisher_nicht_vorhanden) THEN unit_übernehmen;
          END
          ELSE BEGIN
                zu_weit_entfernt_liegende_units_markieren;
                (* markierte Place-Units werden im folgenden *)
                (* Iterationsschritt nicht berücksichtigt. *)
          END;

    UNTIL( ermittelte_Position_konvergiert );

    topologische_Karte_aktualisieren;

   return ermittelte_Position;

END;
(* ELSE Position nicht ermittelbar *)
```

Begonnen wird bei der Positionssuche als Mittelpunkt mit der Position, die durch die Radsensoren dem Weg des Roboters gefolgt ist. Für jede aktive Place-Unit wird nun ein Vektor gebildet, der die Verschiebung vom Mittelpunkt zur Place-Unit-Position darstellt. Der Mittelpunkt wird daraufhin um den mit der Aktivierung ($a_i \in [0, 1]$) multiplizierten Vektorbetrag in Richtung der Place-Unit-Position verschoben.

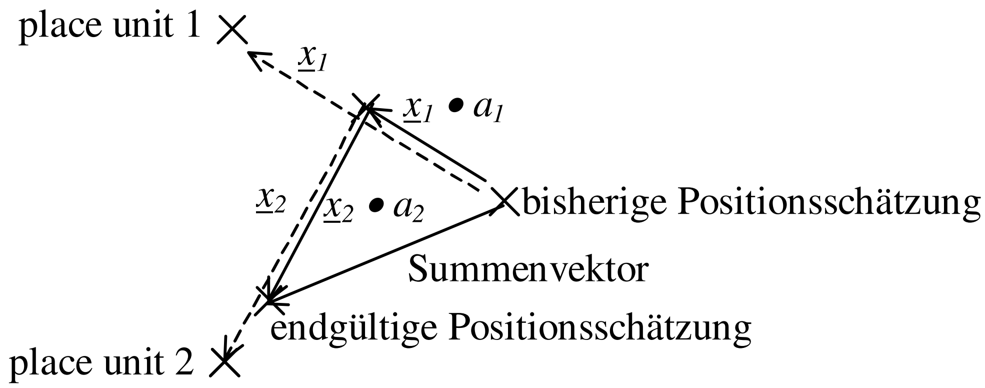


Abbildung 6 ***Positionsschätzung mit Bildung eines gewichteten Positionsmittels***

In den folgenden Iterationen wird der Radius für die gültigen Place-Units um die neue Positionsschätzung herum verkleinert. Damit konvergiert die Positionsschätzung gegen einen stabilen Wert.

Die im aktuellen Positionsbestimmungsvorgang nicht mehr gültigen Place-Units (zu große Entfernung von der Schätzposition) werden markiert, damit sie im folgenden Iterationsschritt nicht berücksichtigt werden.

Um diese Markierung später nicht für jede einzelne Place-Unit aufheben zu müssen, merkt sich das Programm einen Zeitstempel als Referenzwert, der bei Markierung einer Place-Unit in selbiger eingetragen wird. Alle Place-Units die eine dem Referenzwert entsprechenden Wert aufweisen gelten als markiert.

Für den nächsten Durchgang muß daher keine Gültigkeitsmarkierung zurückgesetzt werden. Durch Wahl eines neuen aktuellen Zeitstempels als Referenzwert werden alle Place-Units damit automatisch wieder gültig.

Werden zu wenig Place-Units aktiv müssen weitere angelegt werden, um eine Positionsbestimmung zu ermöglichen. Dazu werden auf Basis der aktuellen loka-

len Ansicht jeweils neue Place-Units angelegt, für die je nach Einstellung geprüft wird, ob sie bereits im Netz vorhanden sind. Die zugehörigen Landmarkenpositionen, -winkel und –typen werden der lokalen Ansicht entnommen. Für die Place-Unit-Position werden die aktuellen Pfadintegriererkoordinaten eingesetzt.
Um die aktuelle Position zu bestimmen sollten mindestens zwei Landmarken sichtbar sein, da sonst kein sinnvoller Merkmalsvektor gebildet werden kann. Befindet sich der Roboter in einer Umgebung mit nur wenigen Landmarken, in der insbesondere zeitweise nur einzelne sichtbar sind, ist es sinnvoll, auch Place-Units für einzelne sichtbare Landmarken anzulegen. Die enthaltene Information ist dann jedoch sehr redundant, da Entfernung, Richtung und Typ zweifach in der Place-Unit abgelegt werden und die Komponente für die Winkeldifferenz zwischen zwei Landmarken einen 0°-Winkel ausweist. Für diese Fälle ist die Prüfung auf Wahl jeweils zweier unterschiedlicher Landmarken im Programm abschaltbar.

8.1.4 Die topologische Karte

Die Wegplanung kann mit einer üblichen Graphensuche implementiert werden. Sowohl Start- wie auch Zielpunkt sind Knoten des bidirektionalen Graphen. Momentan zu vermeidende Streckenabschnitte (die Kanten des Graphen) können temporär markiert werden, falls sie nicht passierbar sind. Damit kann bei einer Wegplanung dieser Streckenabschnitt umgangen werden. Außerdem enthalten die Streckenabschnitte Informationen über die ungefähre Länge, so daß auch die Kosten für die Strecke bekannt sind. Vom Startpunkt wird ein Signal durch den Graphen geschickt, das mit der Zahl der Streckenabschnitte und/oder der insgesamt zurückgelegten Wegstrecke schwächer wird. Dazu wird die in den Kanten vermerkte Streckenlänge (Kosten) benutzt. Von allen im Zielpunkt eintreffenden Signalen wird die Strecke mit dem stärksten Signal für die Durchquerung der Karte benutzt. Auf gesperrte Streckenabschnitte treffende Signale verfolgen gesperrte Richtungen nicht weiter.
Aus den ermittelten Streckenabschnitten lassen sich nun leicht Dreh- und Fahrbefehle erzeugen, die in eine Befehlsqueue eingereiht werden. Diese kann ganze Befehlssequenzen entgegennehmen und einzelne Anweisungen zur Ausführung

zur Verfügung stellen. Tritt die Hindernisvermeidung in Aktion wird die aktuelle Position erneut bestimmt und der Weg von dort erneut geplant.

8.1.4.1 Datenstrukturen

Die topologische Karte ist ein ungerichteter Graph, dessen Knoten besuchte Plätze der Umgebung darstellen. Die Kanten (PlaceConnection) enthalten Informationen über Länge und Sperrung der Strecken.

8.1.4.2 Generierung der Karte

Während die Umgebung erkundet und entsprechend neue Place-Units angelegt werden, wird die Karte erweitert.
Da im Place-Unit-Netz für eine Stelle viele Place-Units angelegt werden, wäre es ineffizient, alle untereinander erreichbaren Place-Units miteinander zu verbinden. Daher wird eine eigene topologische Karte verwaltet, die extrem dicht beieinanderliegende Stellen als einen Platz verwaltet.

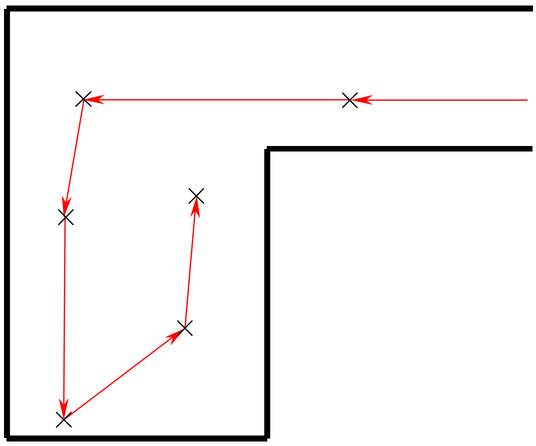


Abbildung 7 ***Der Roboter erkundet eine Sackgasse.***

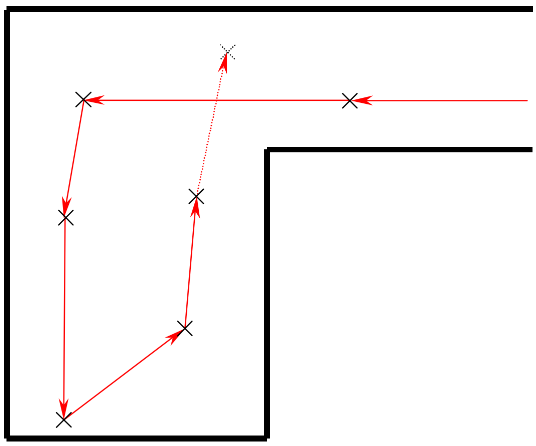


Abbildung 8 ***Kreuzung eines bereits befahrenen Streckenabschnitts***

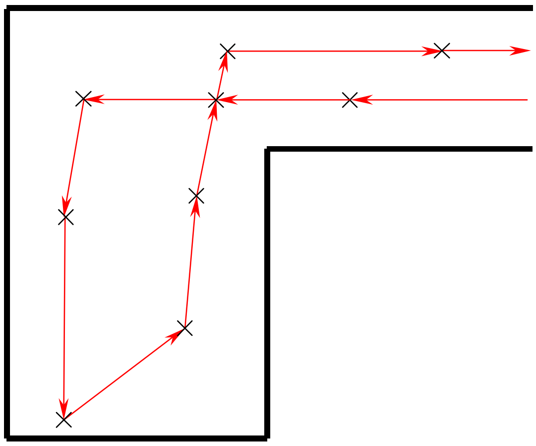


Abbildung 9 ***Anlegen eines neuen Kreuzungspunktes***

Für bisher unbekannte Stellen der Umgebung werden neue Graphenknoten erstellt und mit zuvor besuchten Stellen über eine Graphenkante verbunden. Kreuzen sich dabei Graphenkanten, was mit Hilfe der vorhandenen metrischen Information festgestellt wird, so werden die Kanten am Kreuzungspunkt mit einem neuen Graphenknoten verbunden. Abbildung 7 bis Abbildung 9 zeigen die Erkundung einer Sackgasse. Die schwarzen Kreuze kennzeichnen das Ende eines Erkundungsschritts, die Pfeile Verbindungen in der topologischen Karte. Abbildung 8 stellt gestrichelt den nächsten Erkundungsschritt dar, der einen bereits befahrenen Streckenabschnitt kreuzt. In Abbildung 9 ist zu sehen, wie die Strecke des zweiten Erkundungsschritts aufgetrennt wird. Die an der Kreuzung neu angelegte Verbindung wurde mit dem neuen Kreuzungspunkt verbunden.

Durch dieses Vorgehen bildet sich ein Kartennetz aus, das es erlaubt, eine Wegplanung zu implementieren, die im erkundeten Kartenbereich eine optimale Route wählen kann.

8.1.5 Die Befehlsqueue

Die Befehlsqueue nimmt alle Befehle für das Roboterfahrwerk entgegen.
Bei der Queue handelt es sich um einen FIFO-Puffer (first in first out). Die in die Queue gestellten Befehle werden der Reihe nach für die Verarbeitung zur Verfügung gestellt. Mit Hilfe der lokalen Ansicht wird überprüft, ob die Befehlsausführung zu einer Kollision führen würde. Ist dies der Fall, wird der Inhalt der Befehlsqueue verworfen und die Wegplanung muß eine neue Route zum Ziel entwerfen. Ansonsten wird der Fahrbefehl umgesetzt.

8.1.6 Das Erkundungssystem

Das Erkundungssystem tritt in Aktion, sobald keine Anordnungen mehr in der Befehlsqueue vorliegen.
Die Fähigkeiten eingesetzter Sensoren geben eine gewisse Maximalsichtweite vor, mit der ein Einsatz sinnvoll ist. Es ist beispielsweise nicht sinnvoll für einen Sonarsensor eine Sichtweite von 20m anzugeben, da in dieser Entfernung Sensordaten überhaupt nicht oder in nur unzureichender Qualität zur Verfügung stehen.
Sofern kollisionsfrei möglich, fährt der Roboter pro Erkundungsschritt maximal eine halbe Sichtweitenlänge geradeaus. Kann ein Erkundungsbefehl wegen eines Hindernisses nicht ausgeführt werden, wird eine neue Fahrtrichtung ermittelt. Dazu dreht der Roboter sich zufällig nach links oder rechts. Dabei erhält diejenige Richtung mit 75% Wahrscheinlichkeit den Vorzug, die bei der vorangehenden Drehung eingesetzt wurde. Damit wird aufeinanderfolgendes hin- und herdrehen weitestgehend vermieden. Der Winkel der Drehung wird mit Hilfe der lokalen Ansicht so bestimmt, daß in der neuen Richtung eine Mindestentfernung zurückgelegt werden kann. Kann diese Anforderung in keiner Richtung erfüllt werden, wird die Richtung mit maximaler Fahrtstrecke gewählt.
Während der Erkundung werden vorhandene Place-Units dazu benutzt, die geschätzte Position anhand der Landmarken zu korrigieren, falls die Radsensoren zu ungenau funktionieren. Feuern zu wenige der im Netz vorhandenen Place-Units

für die aktuelle Position des Roboters, werden neue angelegt. Nach jedem Erkundungsschritt werden die erreichten Stellen in der topologischen Karte miteinander verbunden. Da jeder Erkundungsschritt aus einer geraden Fortbewegung besteht, ist ein spezielles Fahrtmodul für gebogene Gänge o.ä. nicht erforderlich. Ist eine gerade Streckenfortführung nicht möglich, dreht der Roboter einfach etwas und fährt erneut weiter geradeaus. Die Zielrichtung ω von Drehungen wird zufällig mit einer zufälligen Abweichung $\Delta \in [-5°, 5°[$ versehen. Damit wird vermieden, daß der Roboter in einem geraden langen Gang immer genau hin- und herfährt und nur einzelne Bereiche von Räumen erreicht. Die hiermit erreichte Erkundung überdeckt ganze Flächenbereiche, so daß sich ein Kartennetz ergibt, welches auch größere Freiräume der Umgebung abdeckt.

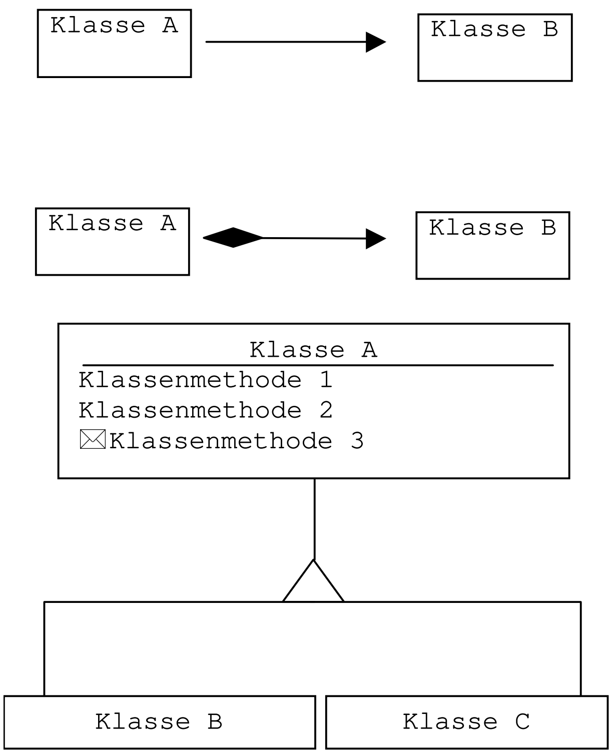
8.1.7 Die Umgebung und RPI-Anbindung

8.2 C++-Klassenbibliothek

LEDA (Library of Efficient Data Types and Algorithms) ist eine Bibliothek von Datentypen und Algorithmen. So sind einfache Datentypen für rationale Zahlen mit beliebiger Genauigkeit, strings, Vektoren vorhanden. Für geometrische Anwendungen gibt es u.a. Punkte, Streckensegmente, Kreise, Polygone. Ebenso sind Datentypen für typisierte und untypiserte Graphen verfügbar, die gerichtet oder ungerichtet sein können. Zur Verarbeitung dieser Datentypen stehen auch effiziente Algorithmen zur Verfügung, die häufig benötigte Aufgaben abdecken.
LEDA ist als C++ Klassenbibliothek implementiert und kommt in dieser Arbeit in der Version 3.5.1 zum Einsatz. Für Forschungsanwendungen ist die Anwendung der Bibliothek kostenfrei.

8.3 Die Programmodule und die enthaltenen C++-Klassen

Die C++-Module des KiLRoY-Simulators fassen verschiedene zueinandergehörige Klassen zusammen. Diese werden im Folgenden aufgeführt und deren Beziehung anhand von Klassendiagrammen aufgezeigt.
Die Klassendiagramme sind in ihrer Darstellung an die OMT (Object Modelling Technique) angelehnt (siehe Abbildung 10).



Klasse A benutzt Klasse B
(z.B. als Typ bei einer Parameterübergabe)

Klasse A enthält Klasse B als Komponente
(Klasse B ist Attribut von Klasse A)

Die Klassen B und C erben von Klasse A.
In Klasse A wir ferner gezeigt, wie Klassenmethoden aufgeführt werden. Klassen, für die keine Methoden aufgeführt sind, stammen aus einem anderen Modul und sind dort komplett dargestellt. Methoden, die den Namen der Klasse tragen, sind Konstruktoren. Ist eine Methode der Form ~Klassenname enthalten, handelt es sich um den entsprechenden Destruktor der Klasse.
Die Klassenmethode 3 ist eine nicht-öffentliche Methode (☒) .

Abbildung 10 ***Bedeutung der in Klassendiagrammen verwendeten Symbole***

8.3.1 kilroystd

In dieser Headerdatei werden verschiedene Standarddefinitionen durchgeführt und nötige weitere Header der LEDA-Toolbibliothek eingebunden.

8.3.2 environment

Das environment-Modul umfaßt die Klassendefinitionen für die Simulationskarte und die simulierten sowie „echten“ Umgebungen.

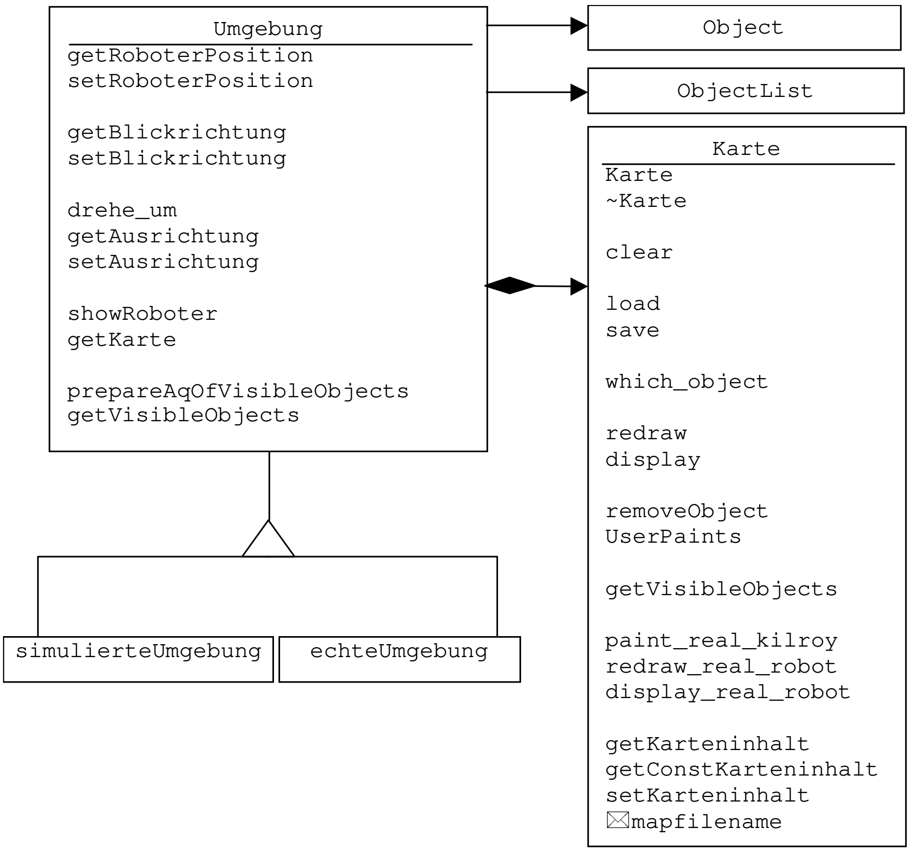


Abbildung 11 ***Klassendiagramm des Environment-Moduls***

Die Simulationskarte enthält die Umgebung, die der Roboter mit seinen simulierten Sensoren wahrnimmt. Auf die Inhalte dieser Karte hat die Robotersimulation keinen Zugriff. Die Klasse „simulierteUmgebung“ schafft die Schnittstelle für die Simulation der Sensoren und die Roboteransteuerung und greift auf die Simulationskarte zu, um daraus die simulierten (vorverarbeiteten) Sensorantworten zu erzeugen. Die Klasse „echteUmgebung“ erlaubt es, eine Schnittstelle zu einem wirklichen Roboter herzustellen. Eine Signalvorverarbeitung je nach Typ der eingesetzten Sensoren ist notwendig.

```
class Karte
class Umgebung
```

```
class simulierteUmgebung: public Umgebung
class echteUmgebung: public Umgebung
```

8.3.3 kilroySIM

Dies ist das Hauptmodul des Programms, das die nötigen Vorkehrungen zur Darstellung des Benutzerinterfaces und der Steuerung des Simulators trifft.

8.3.4 object

Das object-Modul definiert die Umgebungsobjekte, die im Simulator benutzt werden. Ein Objekt ist dabei ein Polygon, so daß beliebige Umrisse dargestellt werden können. Objekte haben dabei weitere Eigenschaften. So sind auch Landmarken Objekte der Umgebung, die von verschiedenem Typ sein können. Ob ein Objekt eine Landmarke darstellt, kann über die Methode *isLandmark* erfragt und über die Methode *makeLandmark* beeinflußt werden.

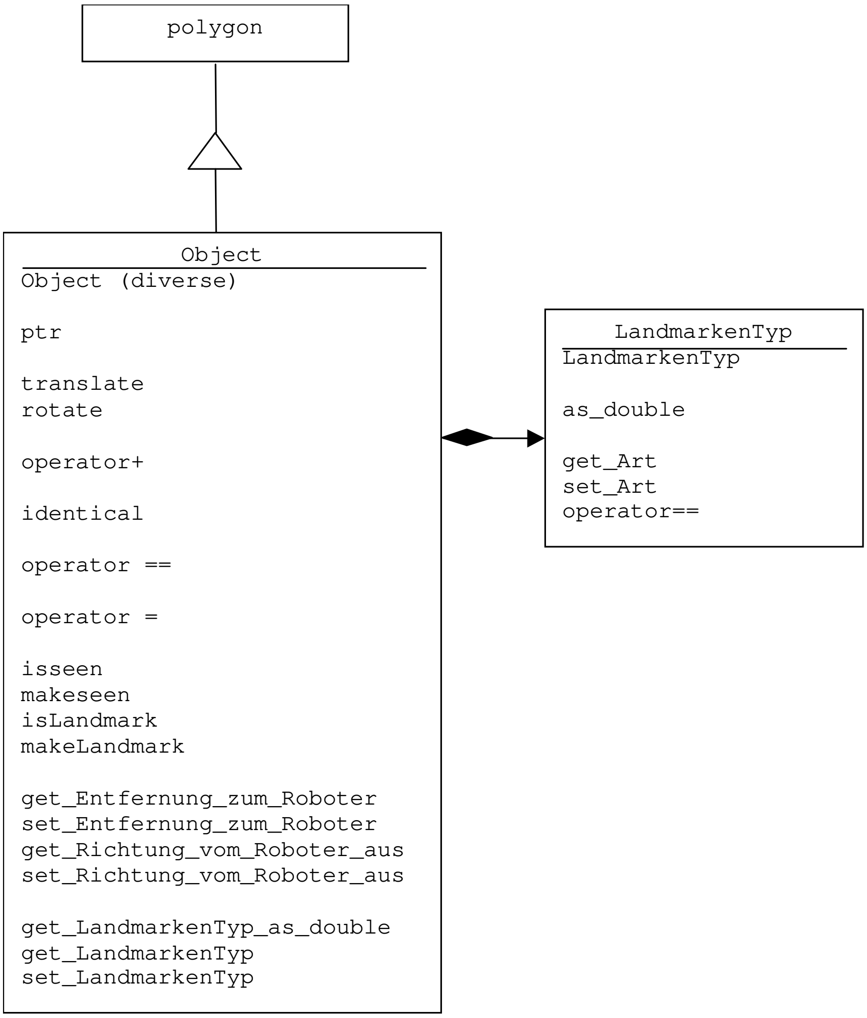


Abbildung 12 ***Klassendiagramm des Object-Moduls***

8.3.5 order

Das order-Modul enthält Klassen zur Darstellung und Aufreihung von Befehlen. Mögliche Befehle sind derzeit Dreh- und Fahrbefehle. Die BefehlsQueue verwaltet Befehlssequenzen und ist Komponente der Klasse Roboter.

```
typedef queue<Befehl> BefehlsQueue;
```

```
Befehl
-----------------------------
Befehl (diverse Konstruktoren)

is_turnNdrive
is_turn
is_drive
is_nop

getParams
```

Abbildung 13 ***Klassendiagramm des Order-Moduls***

8.3.6 placeunits

Das placeunits-Modul definiert die Neuronen des RBF-Netzes. Diese Place-Units enthalten den vorher beschriebenen 11-dimensionalen Merkmalsvektor. Außerdem sind Kennungen enthalten, die das Laufzeitverhalten der Aktivierungsberechnungen für die Neuronen optimieren.

Alle enthaltenen Winkelangaben sind allozentrisch, damit sie von der Drehung des Roboters unabhängig sind.

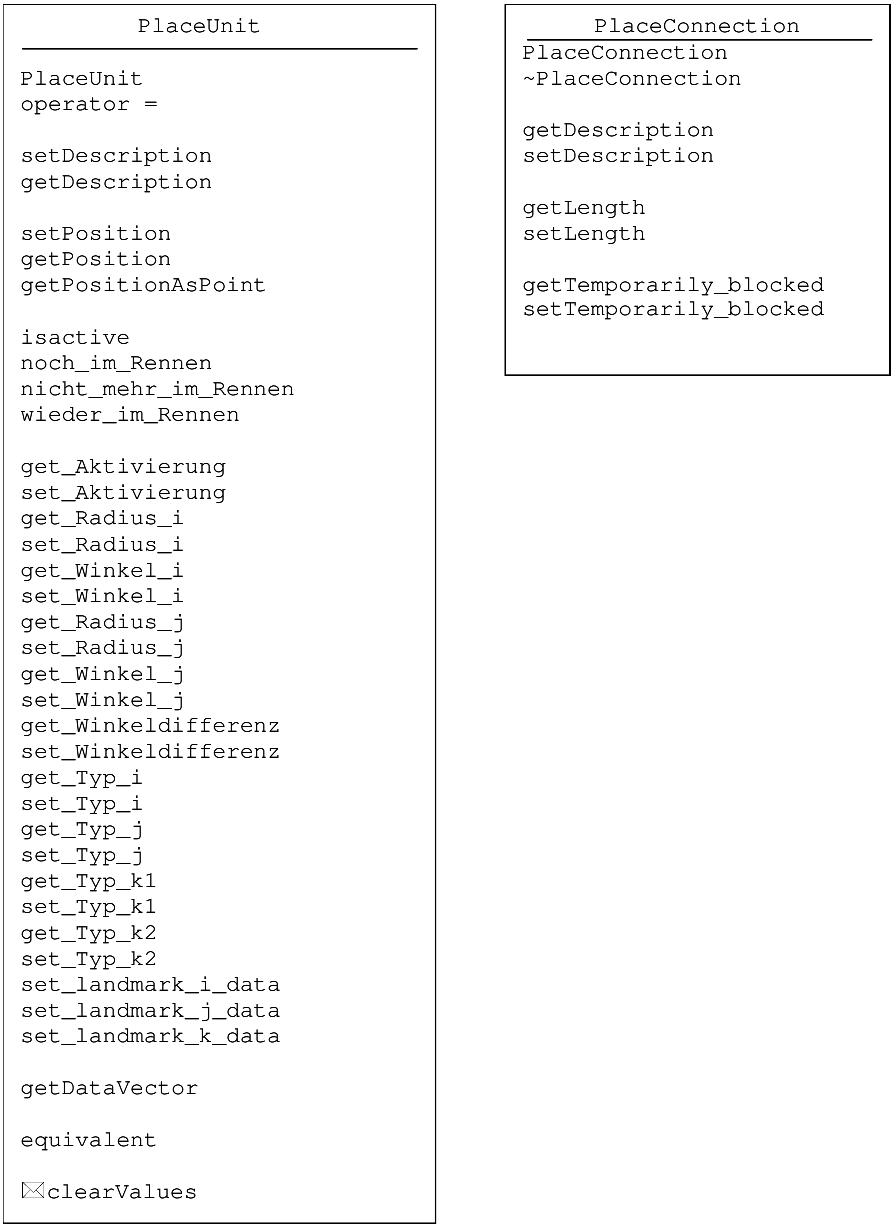


Abbildung 14 ***Klassendiagramm des Placeunits-Moduls***

Weiter sind hier neben Places, die einen Punkt in der Umgebung bezeichnen, die Kartenverbindungen (PlaceConnections) definiert, die Informationen über Länge und Blockierung des Streckenabschnitts enthalten.

8.3.7 placemap

Das placemap-Modul definiert die Graphen, die zur Verwaltung des Place-Unit-Netzes und der topologischen Karte benutzt werden. Die Datentypen für Place-Units, Streckenverbindungen usw. kommen aus dem Modul placeunits.

Fehler! Es ist nicht möglich, durch die Bearbeitung von Feldfunktionen Objekte zu erstellen.

Abbildung 15 ***Klassendiagramm des Placemap-Moduls***

```
typedef UGRAPH<PlaceUnit, PlaceConnection> PlaceUnitGraph;
typedef UGRAPH<Place, PlaceConnection> TopoMap;
```

8.3.8 roboter

Das Modul „roboter" besteht aus einer Reihe von Klassen, die Komponenten eines Robotersystems beschreiben.

Die Klasse Roboter faßt die nötigen Klassen zusammen, die Teile eines Roboters modellieren. So enthält die Klasse Roboter Objekte für Kamerakopf, lokale Ansicht, Fahrwerk, Erkundungsfunktion, und die Befehlsqueue. Ferner wird der Roboter-Klasse bei Instantiierung ein Verweis auf eine Umweltklasse mitgeteilt. Diese ist vom Typ der im Abschnitt 8.3.2 beschriebenen Klassen „simulierteUmgebung" oder „echteUmgebung". So kann bei Anlegen eines Roboter-Objekts auf einfache Art festgelegt werden, ob die Robotersteuerung mit einem echten Roboter oder mit der simulierten Umgebung arbeitet.

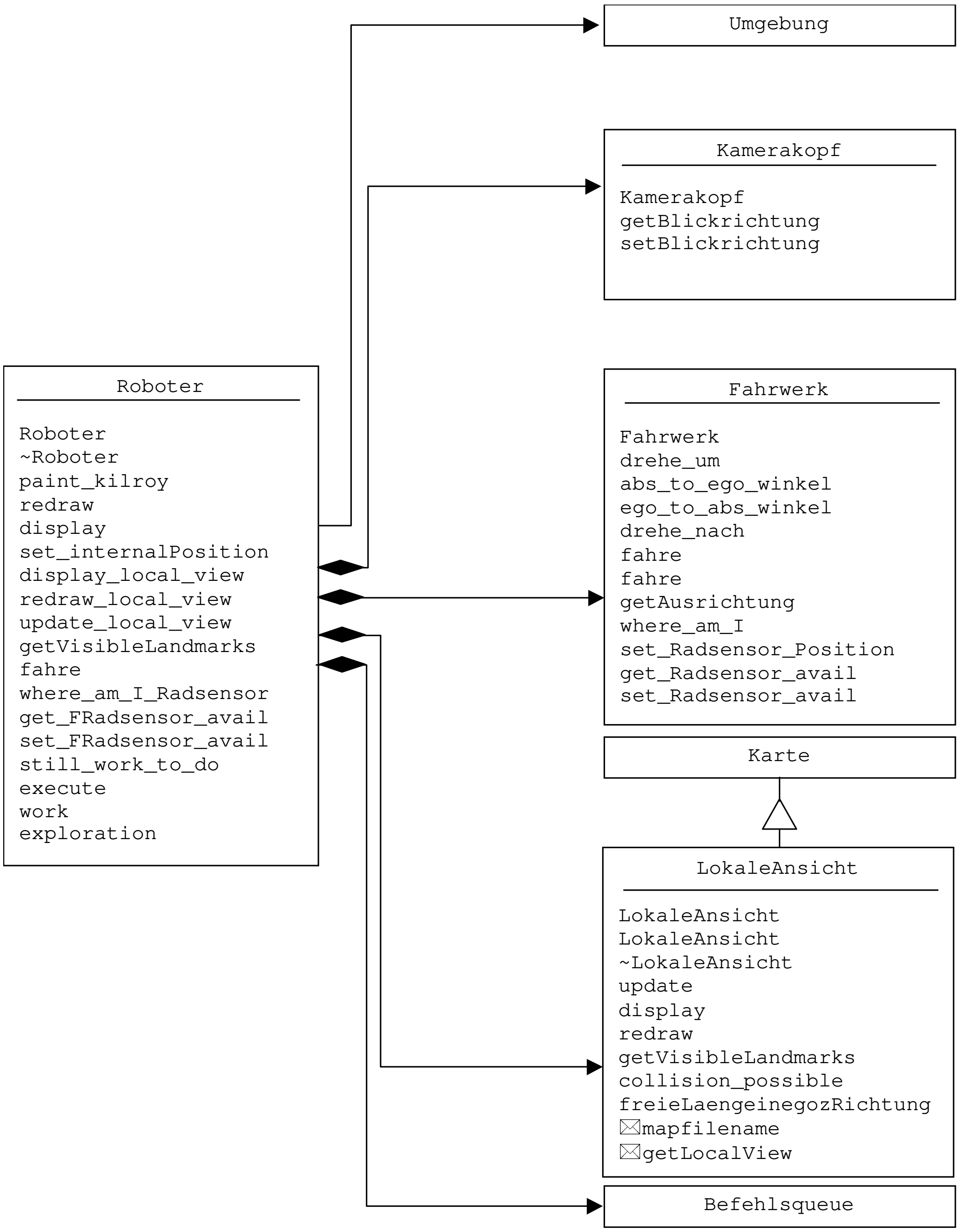


Abbildung 16 ***Klassendiagramm des Roboter-Moduls***

Über die „Kamerakopf“-Klasse kann z.B. ein BiSight-Kamerakopf angesteuert werden. Allgemein ist sie zur Steuerung der Ausrichtung des Hauptsensors gedacht.
Die Klasse „lokale Ansicht“ kapselt die Benutzung des Kamerakopfes und Auswertung der vorverarbeiteten Sensordaten. Das Ergebnis ist eine Karte der direkten Umgebung des Roboters. Über diese Karte erkennt das System Landmarken und vermeidet Kollisionen.
Das Fahrwerk ist für die Ausführung von Steuerbefehlen zuständig. Ferner ermöglicht diese Klasse die Umrechnung zwischen allozentrischen und egozentrischen Winkeln, die von der dem Fahrwerk bekannten momentanen Ausrichtung des Roboters abhängen. Schließlich lassen sich auch die Positionsdaten von den Radsensoren über diese Klasse erfragen.

Kapitel 9 Auswertung

Wir untersuchen nun anhand einiger Experimente, wie sich das Navigationssystem in verschiedenen Umgebungen und bei verschiedenartigen Problemstellungen verhält.
Die Unterteilung in verschiedene Experimente betrifft dabei jeweils unterschiedliche Versuchsumgebungen bzw. Landmarkenanordnungen, bei denen jeweils eine erneute Erkundung durch den Roboter notwendig ist.

9.1 Experimente

Bei den meisten der folgenden Experimente haben wir die Simulation von Odometriefehlern abgeschaltet. Der Roboter bezieht sich beim Anlegen neuer Place-Units nur auf die somit korrekten Angaben des Pfadintegrierers. In den abschließenden Experimenten zeigen wir den Einfluß von Odometriefehlern auf und begründen anhand der Ergebnisse dieses Vorgehen.
Für den Simulator wurden folgende Einstellungen benutzt:

- Eine Prüfung auf doppelt vorhandene Landmarken findet statt.
- Die beiden Landmarkenpaare, die in den Place-Units benutzt werden, bestehen aus je zwei unterschiedlichen Landmarken (dazu müssen jeweils mindestens 2 Landmarken sichtbar sein).

Diese beiden Einstellungen bewirken, daß beim Anlegen neuer Place-Units keine redundante Information erzeugt wird, da das Netz dann nur unterschiedliche Place-Units enthält. Die in den Place-Units berücksichtigten Landmarkenpaare bestehen zudem aus jeweils zwei unterschiedlichen Landmarken, so daß die Entfernungs-/Richtungsinformation nicht doppelt vorhanden ist und die Winkeldifferenz nicht, wie bei Auswahl zweier gleicher Landmarken, 0° ergibt, so daß diese Informationskomponente wertlos wird.
Die Zahl der Versuche, nicht im Netz enthaltene Place-Units anzulegen, werden mit steigender Zahl sichtbarer Landmarken erhöht, da dann auch entsprechend mehr Kombinationsmöglichkeiten für das Anlegen unterschiedlicher Place-Units vorhanden sind.

Damit sich Unterschiede des aktuellen Merkmalsvektors gegenüber den in Place-Units gespeicherten Positions- und Entfernungs-Informationen stärker in sinkender Aktivierung bemerkbar machen, besteht die Möglichkeit, die Vektorkomponenten für Aktivierungsberechnungen zu skalieren. Positionsangaben skalieren wir mit Faktor 4.0, Entfernungsangaben mit dem Wert 2.0 als Faktor. Die Typen der Landmarken sind während der Experimente zuverlässig zu erkennen.

9.1.1 3 unterscheidbare Landmarken

9.1.1.1 Versuchsbeschreibung

In diesem Experiment wird ein abgeschlossener Raum benutzt, in dem 3 Landmarken im Dreieck angeordnet sind. Die 3 Landmarken sind unterschiedlichen Typs, also eindeutig unterscheidbar. Abbildung 17 zeigt den simulierten Raum.

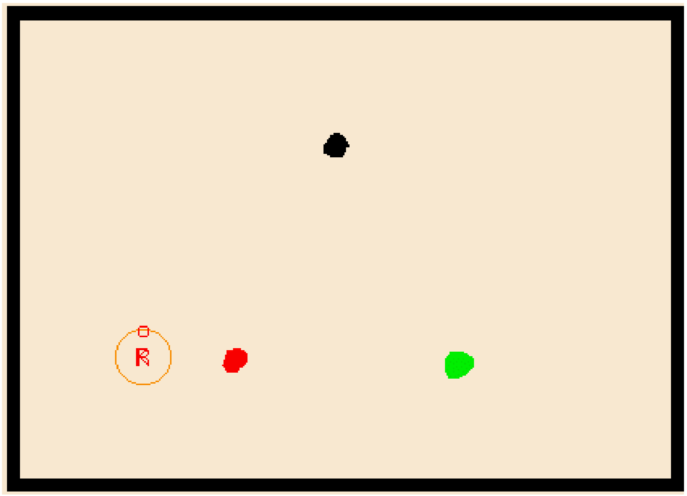


Abbildung 17 ***Die simulierte Umgebung des ersten Versuchs***

Wie wir später zeigen, ist es erst ab einem gewissen Stand der Erkundungen sinnvoll, sich auf die Ergebnisse des Place-Units-Netzes zu verlassen.

Den Roboter lassen wir die Umgebung solange erkunden bis eine flächendeckende Erkundung der Umgebung stattgefunden hat. Dabei wird ebenfalls die topologische Karte der Umgebung aufgebaut.

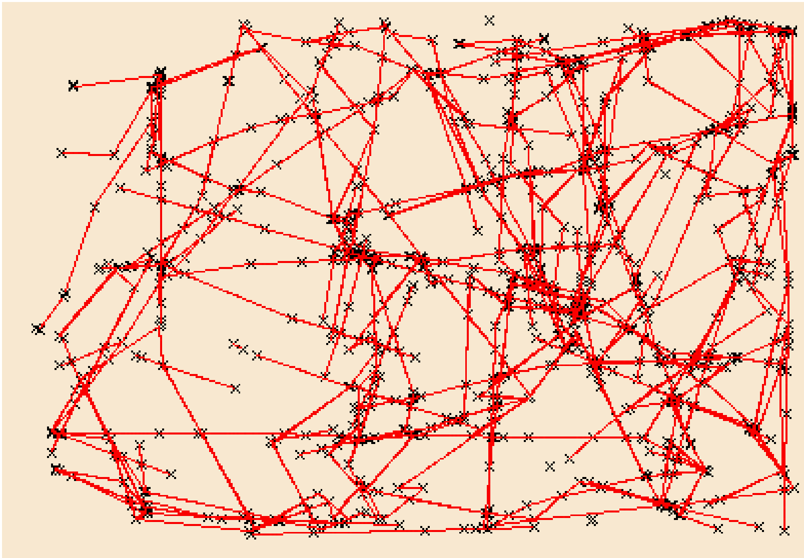


Abbildung 18 ***Die topologische Karte der 3 Landmarken-Aufgabe***

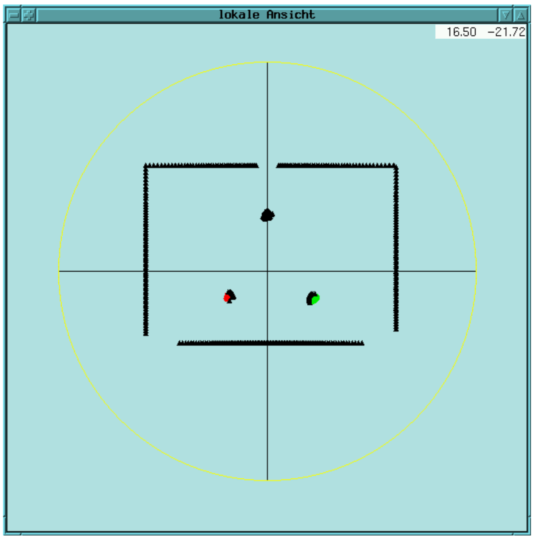


Abbildung 19 L*okale Ansicht der Position, an der wir die Versuchsergebnisse ermitteln*

Der Roboter wird im zuvor beschriebenen Raum ausgesetzt und erkundet ihn unter optimalen Bedingungen. Das bedeutet, daß die Simulation von Odometriefeh-

lern abgeschaltet ist und der Roboter sich beim Anlegen neuer Place-Units nur auf die somit korrekten Angaben des Pfadintegrierers verläßt.
Untersucht werden soll dann, wie genau die Positionierung unter Einsatz des trainierten Netzes arbeitet. Dazu soll der Roboter gezielt innerhalb der Karte positioniert werden. Dem Navigationssystem ist die tatsächliche Position des Roboters nicht bekannt. Mit Hilfe des Place-Unit-Netzes soll dann anhand der sichtbaren Landmarken ermittelt werden, an welcher Position sich der Roboter befindet.
Das Navigationssystem wird in diesem Versuch vier Aufgaben gegenübergestellt. In der ersten muß der Roboter seine Position aus der Mitte des Landmarkendreiecks heraus bestimmen. Wir ermitteln damit, wie genau die Positionsbestimmung in der Mitte der Umgebung arbeitet. Die nächsten beiden Aufgaben wiederholen diesen Versuch unter Entfernen jeweils einer der drei Landmarken (3-1 Landmarken-Aufgabe). Wir beobachten damit, ob die Genauigkeit der Standpunktbestimmung unter dem Fehlen einzelner Landmarken leidet.
Für diese Experimente wurde der Roboter an der Koordinatenposition (50,42) aufgestellt. Die Koordinatenangabe dient der Reproduzierbarkeit des Experiments und ermöglicht Vergleiche mit den Angaben für Mittelwert und Standardabweichung der Lokalisationsergebnisse, die wir für jedes Experiment nennen werden. Das Koordinatensystem für die Simulationsumgebung umfaßt in beiden Richtungen das Intervall [0, 100]. Dieser Bereich wird von den vorgestellten Testumgebungen nicht vollständig ausgenutzt. Der Ursprung des Koordinatensystems ist links unten im Fenster dargestellt. Im Simulator werden im Fenster der Simulationsumgebung die Koordinaten für die Position des Mauszeigers innerhalb der Simulationskarte eingeblendet. Die Abbildungen der experimentellen Ergebnisse zeigen die genannte Roboterposition durch einen mit einem „R“ gekennzeichneten Kreis an.
Für das vierte Experiment wurde der Roboter an der Position (58,42) aufgestellt. Eine weitere Landmarke, die den gleichen Typ wie die an der oberen Dreiecksspitze liegende hat, wurde auf der rechten Seite der Landmarkenanordnung hinzugefügt. Dadurch entsteht ein zweites, auf dem Kopf stehendes Dreiecksmuster. Die dadurch entstehenden Verwechslungsmöglichkeiten sind Gegenstand der Betrachtung.

9.1.1.2 Versuchsergebnis

Das Resultat aus 100 aufeinanderfolgenden Positionierungen an der Stelle (50,42) und anschließender Standortermittlung durch das Place-Unit-Netz ist in Abbildung 20 zu sehen. Der Kreis mit dem Buchstaben „R“ zeigt die tatsächliche Roboterposition. Die schwarzen Kreuze (in diesem Falle innerhalb des Kreises) zeigen die Ergebnisse 100 aufeinanderfolgender Positionsschätzungen des Place-Unit-Netzes. Die hohe Genauigkeit der ermittelten Position zeigt sich auch beim Mittelwert (μ=(50.5218, 41.6555)) und der Standardabweichung (σ=(0.067275, 0.102809)) für die vom Netz ermittelte Position.
Die Ergebnisse der Positionsbestimmungen fallen je nach erzeugtem Eingabevektor für das Place-Unit-Netz unterschiedlich aus, da der Eingabevektor Informationen über zwei Paare jeweils verschiedener Landmarken enthält. Die sich daraus ergebenden, unterschiedlichen, Place-Unit-Aktivierungen können unterschiedliche Ergebnisse für die Standortbestimmung des Roboters bewirken.
Alle Standortschätzungen des Navigationssystems liegen dicht an der tatsächlichen Aufstellposition.

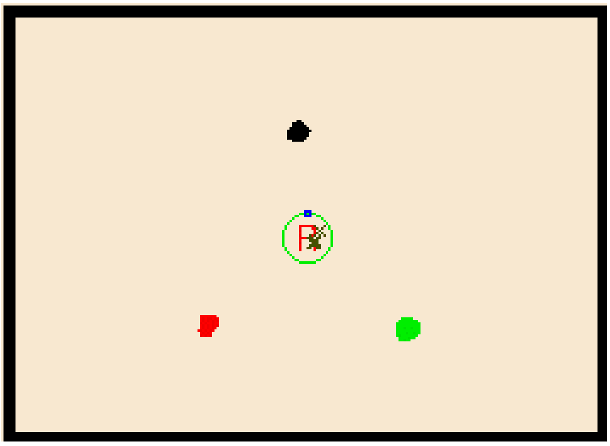


Abbildung 20 ***Versuchsergebnis der Aufgabe mit 3 eindeutig unterscheidbaren Landmarken***

Das nächste Experiment ist ein 3-1 Landmarken-Problem. Nach dem Trainieren in der Umgebung mit 3 Landmarken wird dazu eine Landmarke entfernt. Wir beobachten hierbei, daß die Positionsbestimmung ebenso präzise Ergebnisse liefert wie im vorangehenden Versuch (μ=(50.5643, 41.7122), σ=(0.00921132, 0.00193438)). Durch das Entfernen der Landmarke entsteht hier keine Verwechslungsgefahr, da jede Landmarke einen anderen Typ hat.

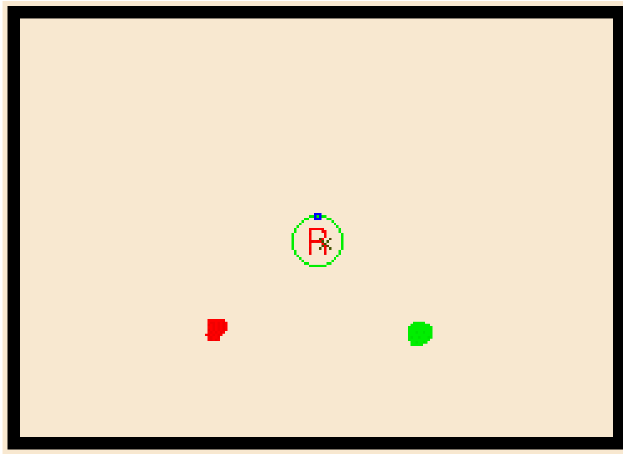


Abbildung 21 ***Lokalisationsergebnis der ersten 3-1 Landmarken-Aufgabe***

Die Wiederholung dieses Versuchs durch Entfernen der rechts unten aufgestellten Landmarke erbringt ein vergleichbares Ergebnis (μ=(50.6575, 41.8444), σ=(0.277406, 0.424805)). Dies erklärt sich daraus, daß hinreichend viele Place-Units des Netzes ihre Position bezüglich der zwei verbleibenden Landmarken gespeichert haben. Dies kann sowohl beim ersten im Merkmalsvektor verwendeten Landmarkenpaar (Richtungen, Entfernungen und Typen), als auch beim zweiten der Fall sein (Winkeldifferenzen und Typen).

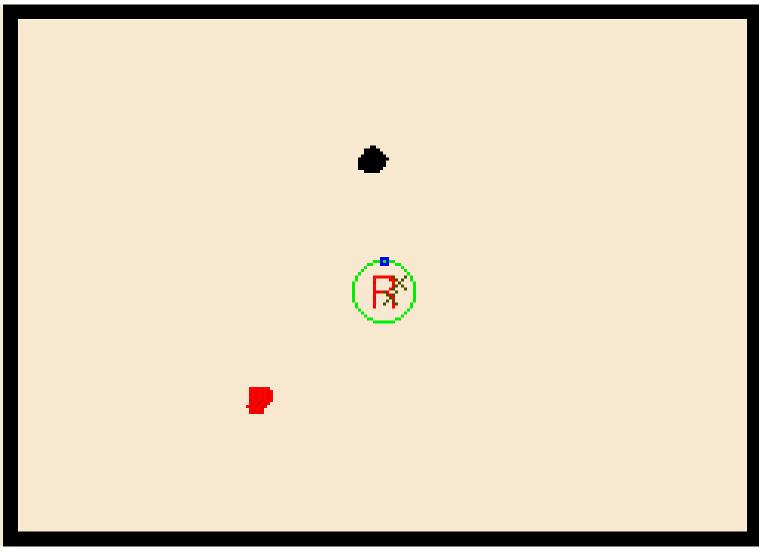


Abbildung 22 ***Lokalisationsergebnis der zweiten 3-1 Landmarken-Aufgabe***

Im folgenden Versuch haben wir eine Landmarke eines bereits zuvor vorhandenen Typs (Typ 1) hinzugefügt (Abbildung 23). Hier antworten jeweils Place-Units, die für Roboterpositionen links oder rechts unterhalb der oberen Landmarke zuständig sind. Da der Roboter sowohl links unterhalb als auch rechts unterhalb einer Typ-1-Landmarke positioniert ist, entsteht zum einen eine Clusterung von Schätzungen an der tatsächlichen Aufstellposition, ferner jedoch eine weniger starke Clusterung links unterhalb der Original-Landmarke vom Typ 1. Dieses Phantombild entsteht, weil der Roboter tatsächlich in die rechte obere Ecke blickend eine Landmarke des Typs 1 erkennen kann. Die Phantomposition ist in Abbildung 23 durch den linken Kreis markiert. Der Kontext für die Phantomschätzung stimmt jedoch nicht, denn dann würde der Roboter knapp oberhalb der Landmarke des Typs 2 (links unten im Bild) stehen. Die entsprechenden Place-Units feuern jedoch nicht, da er ja tatsächlich oberhalb der rechts unten im Bild liegenden Landmarke des Typs 3 steht. Daher ist die Clusterung des Phantombilds viel schwächer. Die Unsicherheit bei der Standortbestimmung zeigt sich auch bei Mittelwert und Standardabweichung der Meßdaten (μ=(51.1428, 38.935), σ=(10.0735, 6.69431)).

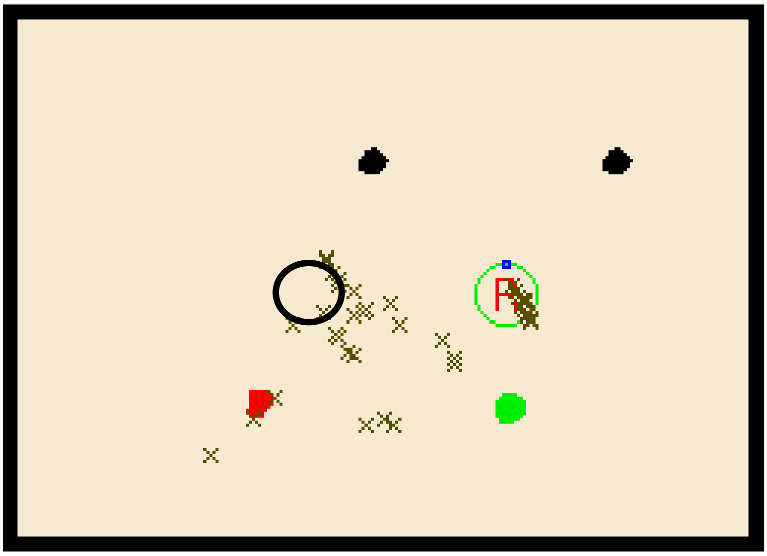


Abbildung 23 ***Lokalisationsergebnis der 3+1 Landmarken-Aufgabe***

9.1.2 Hufeisenanordnung

9.1.2.1 Versuchsbeschreibung

Der simulierte Raum in diesem Versuch hat Hufeisenform (Abbildung 24). Damit wird erreicht, daß nicht von jeder Stelle aus alle Landmarken zu sehen sind. Zudem sind nicht alle Landmarken unterschiedlichen Typs. 6 Landmarken sind in den Ecken des Hufeisens untergebracht. Dabei sind von den Landmarken des Typs 1 und 2 jeweils zwei Stück vorhanden. Insgesamt gibt es also 4 unterschiedliche Typen von Landmarken. Die doppelt vorhandenen Landmarken treten jedoch in unterschiedlichen Anordnungen auf.

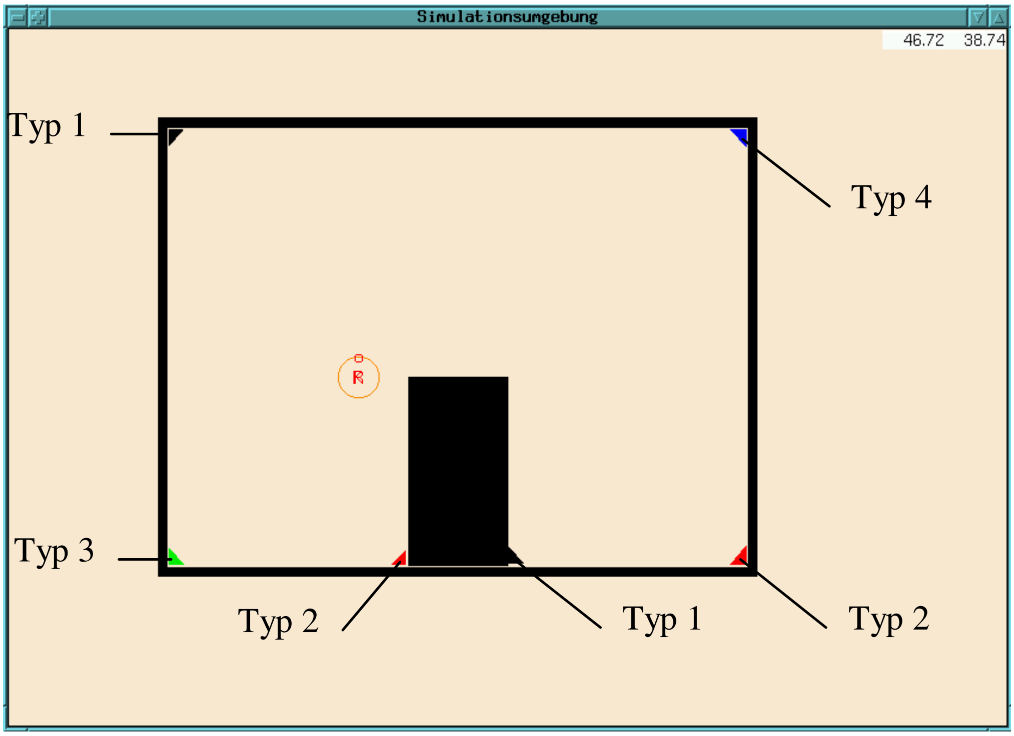


Abbildung 24 ***Die Simulationsumgebung „Hufeisenanordnung“***

Auch diese Umgebung erkundet der Roboter unter optimalen Bedingungen. Seinen Standort soll der Roboter an der Position (27,40) herausfinden.

In diesem Versuch wird überprüft, ob das Vorkommen von Landmarken gleichen Typs an unterschiedlichen Stellen der Umgebung die Positionierung des Roboters ohne den Einsatz von Radsensorinformationen beeinflußt. Die Einstellungen des Simulators entsprechen denen der ersten Versuchsanordnung.

Die aus der Erkundung resultierende topologische Karte ist in Abbildung 25 zu sehen. Der linke Teil der Karte wurde etwas intensiver besucht als der rechte.

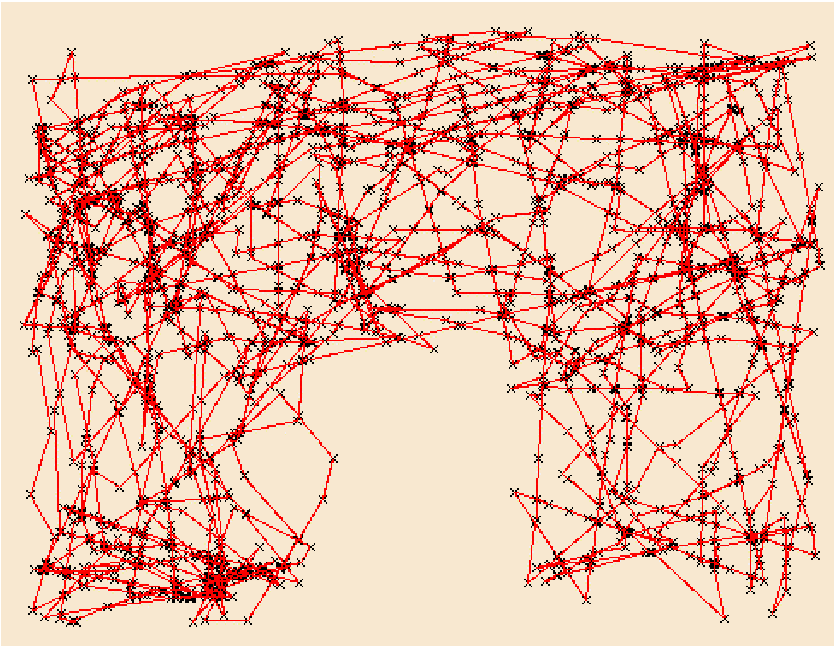


Abbildung 25 ***Topologische Karte der „Hufeisenanordnung"***

9.1.2.2 Versuchsergebnis

In dieser Versuchsumgebung befinden sich in den rechten unteren Ecken beider Hufeisenbügel Landmarken des Typs 2. Da bei diesen Standortschätzungen für die Position (27,40) damit eine Verwechslungsmöglichkeit mit dem rechte Bügel der Versuchsumgebung besteht führen wir diesen Versuch durch. Obwohl hier, wie auch in den vorangegangenen Versuchen, keine Standortinformation durch die Radsensoren verfügbar ist, tritt keine Verwechslung auf. Dies liegt daran, daß durch die unterschiedlichen Landmarkenkonfigurationen die Landmarke des Typs 2 in unterschiedlichen Kontexten auftritt. Diese unterscheiden sich durch die zwei weiteren Landmarken so stark, daß keine Verwechslung mit der anderen Seite der Testumgebung auftritt (μ=(24.141, 37.1079) , σ= (6.91092, 9.24037)).

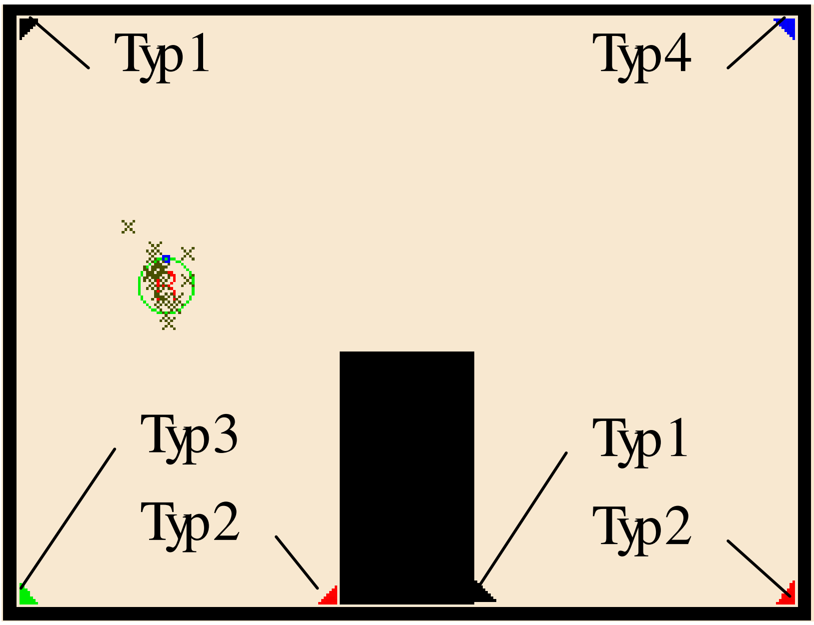


Abbildung 26 ***Lokalisationsergebnis bei der „Hufeisenanordnung“***

9.1.3 2 gleiche Räume, gleiche Landmarken

9.1.3.1 Versuchsbeschreibung

Diese Versuchsanordnung enthält zwei Räume gleicher Größe, die durch einen Gang miteinander verbunden sind. Die beiden Räume haben in jeder der 4 Ecken eine Landmarke unterschiedlichen Typs. Beide Räume gleichen sich bezüglich der Positionierung und der Typen der Landmarken. Im Gang sind zwei zusätzliche Landmarken anderer Typen angebracht, die jedoch aus den Räumen heraus nicht zu sehen sind. Der Roboter wird in zwei Versuchen in jeweils einen der beiden Räume gestellt. Dieser ermittelt daraufhin seinen Standort (von den Positionen (70,20) und (70,70)). Diese Experimente werden bei Verwendung von Radsensordaten wiederholt. Wenn der Roboter an der angegebenen Position aufgestellt wird, stehen ihm keine Radsensordaten über seine aktuelle Position zur Verfügung. Um eventuell früher aufgetretene Odometriefehler zu simulieren erzeugen wir verschieden stark von der tatsächlichen Position abweichende Pfadintegriererwerte.

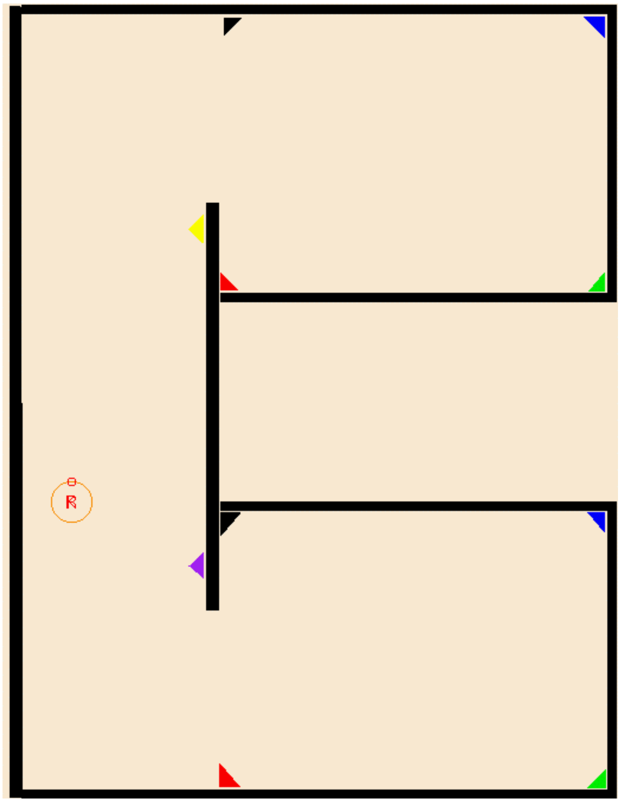


Abbildung 27 ***Simulationsumgebung mit 2 gleichen Räumen***

Dazu wird die tatsächliche Position durch eine zufällige Abweichung mit vorgegebenem Maximalwert von 4 Einheiten pro Richtung verfälscht. Diese, wenn auch ungenauen, Pfadintegriererwerte können das Navigationssystem vor Verwechslung von Landmarken bewahren.

Untersucht werden soll damit das Verhalten des Navigationssystems bezüglich der Verwechslung gleichartiger Umgebungen und stellt ein Problem, das ebenso in Umgebungen mit ununterscheidbaren Landmarken in untereinander ähnlichen Landmarkenkonfigurationen besteht. Wir betrachten dabei den Einfluß, den zur Berechnung genutzte Pfadintegriererwerte auf die Genauigkeit der Standortbestimmung haben. Die Erkundung der Umgebung durch den Roboter erfolgt dabei wiederum unter optimalen Bedingungen.

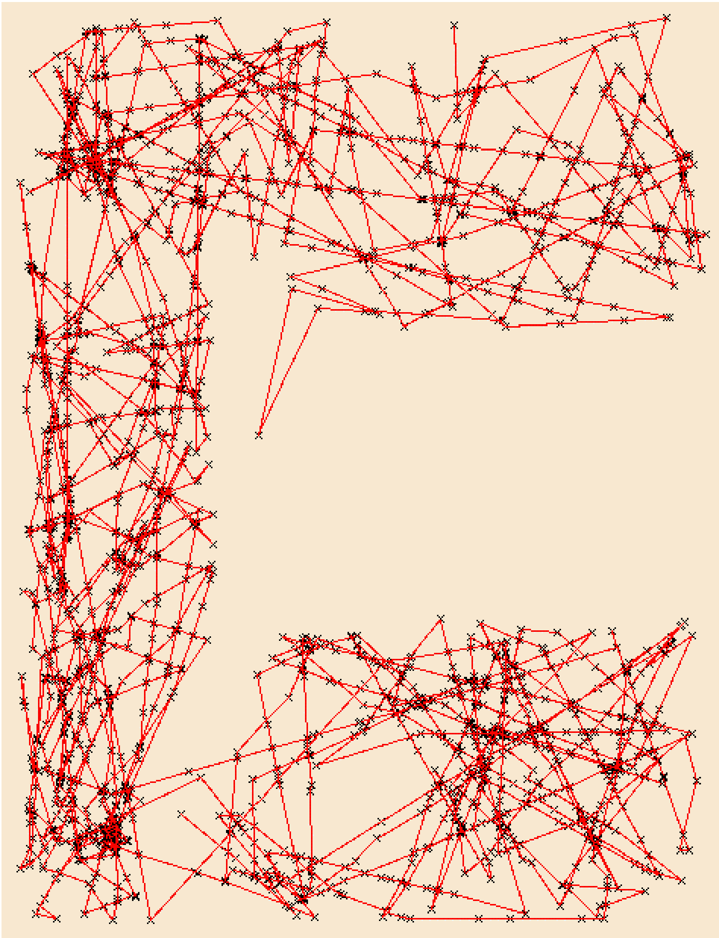


Abbildung 28 ***Topologische Karte der zwei gleichen Räume***

9.1.3.2 Versuchsergebnis

Wir untersuchen in diesem Experiment weiter Verwechslungsmöglichkeiten für das Navigationssystem. Die vorliegende Testumgebung wurde so konzipiert, daß sich die beiden Räume bezüglich der aus ihnen heraus sichtbaren Landmarken nicht unterscheiden. Die unterschiedlichen Öffnungen, durch die der Roboter Zutritt erhält, beeinflussen das Versuchsergebnis nicht, da die Standortbestimmung ausschließlich über die sichtbaren Landmarken bewerkstelligt wird.
Im ersten Teilexperiment plazieren wir den Roboter im unteren der beiden Räume (an der Position (70,20)). Die Abbildung 29 zeigt deutlich, wie das Navigationssy-

stem die beiden Räume verwechselt. Die etwas stärkere Clusterung der Standortschätzungen im unteren Raum begründet sich in der dort intensiver erfolgten Erkundung (μ=(67.6726, 29.5913) , σ=(8.99924, 14.204)). Dies ist anhand von Abbildung 28 auf Seite 78 nachvollziehbar. Die Schätzungen häufen sich jedoch nicht nur an den äquivalenten Positionen innerhalb der beiden Räume, sondern bilden eine Spur zwischen beiden Standortmöglichkeiten. Dies erklärt sich daraus, daß bei der Positionsbestimmung eine Mittelwertbildung durchgeführt wird. Sind nun sowohl Place-Units aktiv, die für einen Standort innerhalb des unteren Raums feuern, als auch solche, die für den oberen Raum stimmen, bewirkt eine Mittelung der Positionen, daß Standortschätzungen im gesamten Positionsbereich zwischen den Schätzungen auftreten können. Je nachdem wie viele Place-Units wie stark (Aktivierung) gegensätzlich stimmen verschiebt sich die resultierende Positionsschätzung zwischen den beiden Positionen. In diesen Versuchen werden jeweils alle Place-Units aktiv, die auf vorhandene Landmarkenmuster passen, da keine Radsensorinformationen benutzt wird. Erst durch Verwendung einer ungefähr bekannten Roboterposition kann das Place-Unit-Netz näher liegende Place-Units bevorzugen.

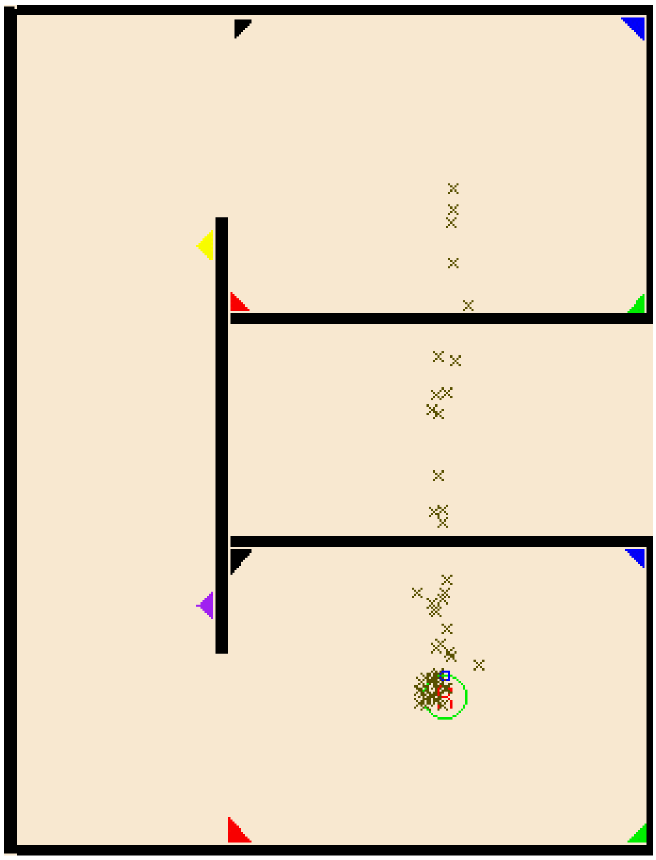


Abbildung 29 ***Ergebnis des Versuchs mit 2 gleichen Räumen mit Aufstellung im unteren Raum***

Das gleiche Experiment wurde für den oberen Raum wiederholt (die Aufstellposition ist (70,70)). In Abbildung 30 ist die vergleichbare Verteilung der Positionsschätzungen gut erkennbar, die sich ebenfalls im unteren Raum häufen (μ=(65.916, 29.7689), σ=(13.6489, 16.1486)). Diese Raumverwechslung kann bei ununterscheidbaren Landmarken nur durch stark unterschiedliche Anordnung der Landmarken oder durch Verwendung unterschiedlicher Landmarkentypen vermieden werden.

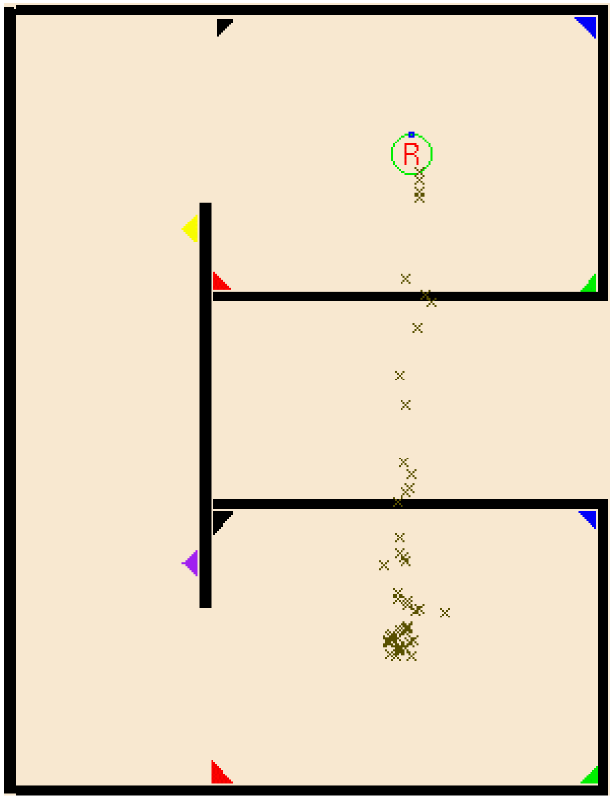


Abbildung 30 ***Ergebnis des Versuchs mit 2 gleichen Räumen mit Aufstellung im oberen Raum***

Die beiden folgenden Teilexperimente wiederholen die vorigen Versuche unter Verwendung von Radsensordaten, die künstlich mit fehlerhaften Werten generiert werden. Wiederum wird der Roboter an den Positionen (70,20) und (70,70) aufgestellt. Die simulierte Abweichung kann 4 Koordinaten-Einheiten in jeder Richtung betragen. Die Differenzen zwischen Radsensorpositionen und Place-Unit-Positionsangaben wurden mit Faktor 4 in ihrer Auswirkung auf die Positionsschätzung verstärkt. Die Versuche zeigen (Abbildung 31), daß die Räume einwandfrei unterschieden werden, obwohl sie von den Landmarkenanordnungen her noch immer ununterscheidbar sind (μ=(68.3809, 19.7623), σ=(2.10011, 1.86496)).

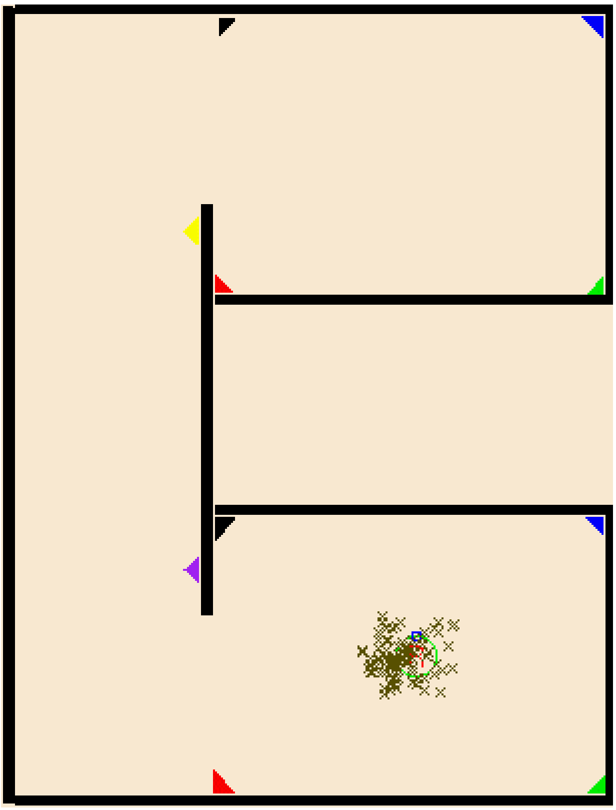


Abbildung 31 ***Ergebnis mit Aufstellung im unteren Raum und bekannter Radsensorposition***

Der obere Raum wird ebenfalls zweifelsfrei identifiziert (Abbildung 32), obwohl bei den vorangehenden Versuchen eine stärkere Häufung von Ergebnissen auf den unteren Raum verwies. Der größere Fehler der Standortbestimmung, der durch intensivere Erkundung des unteren Raums entsteht zeigt sich bei μ=(68.0643, 57.368) und σ=(2.00997, 9.78464) für die Meßdaten dieses Versuchs.

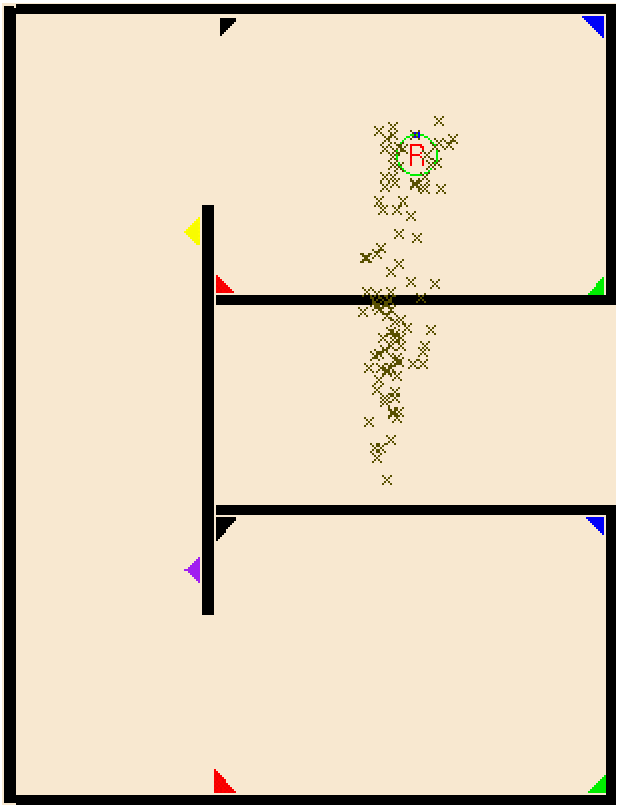


Abbildung 32 ***Ergebnis mit Aufstellung im oberen Raum und bekannter Radsensorposition***

9.1.4 2 gleiche Räume, verschiedene Anordnung

9.1.4.1 Versuchsbeschreibung

Die simulierte Umgebung dieses Versuchs gleicht der in 9.1.2.2 verwendeten. Hier jedoch haben wir die Landmarken des unteren Raums an die jeweils gegenüberliegende Ecke getauscht.

Wir untersuchen damit, welchen Einfluß die geänderte Anordnung der gleichen Landmarken auf das Ergebnis des Versuchs hat. Dazu wiederholen wir die beiden ersten Versuche aus 9.1.2.2. Die Rahmenbedingungen der Simulation wurden ebenfalls übernommen.

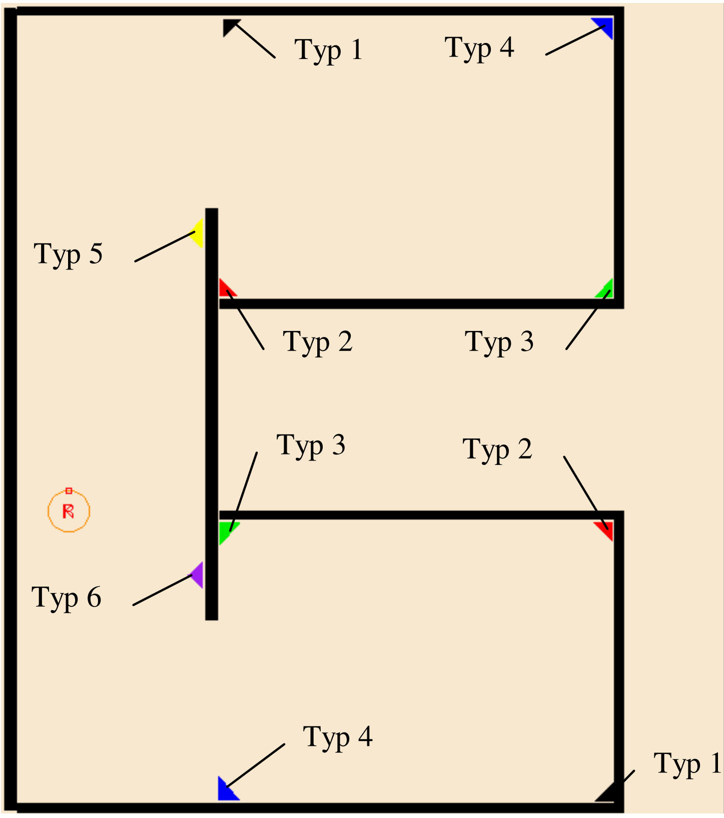


Abbildung 33 ***Zwei Räume mit unterschiedlichen Landmarkenanordnungen***

9.1.4.2 Versuchsergebnis

Die Umverteilung der Landmarken des unteren Raums gegenüber den vorangehenden Experimenten macht eine erneute Erkundung der Umgebung erforderlich. Das Ergebnis des ersten Versuchs zeigt, daß das Navigationssystem nach Vertauschen der Landmarkenanordnung die Räume zweifelsfrei unterscheidet. (Abbildung 34). Der Mittelwert der Meßdaten für den Versuch beträgt μ=(69.8578, 21.4889), die Standardabweichung σ=(2.86202, 1.8944). Abbildung 35 zeigt das Ergebnis für die Aufstellung im oberen Raum (μ=(69.6675, 70.6283), σ=(1.39952, 1.17522)).

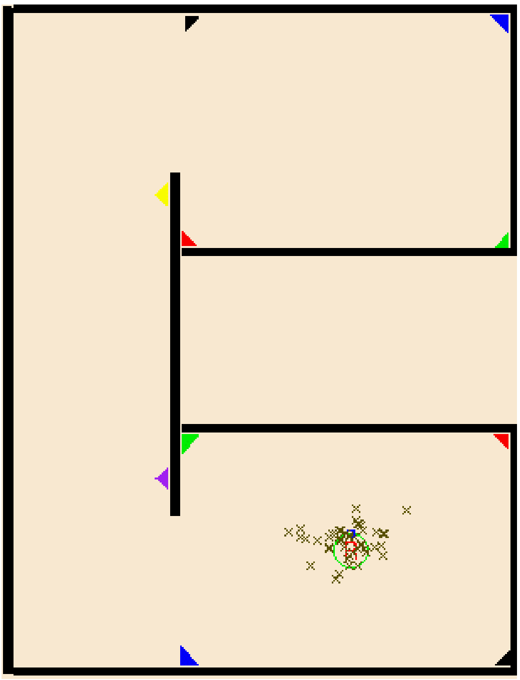


Abbildung 34 ***Lokalisationsergebnis bei Positionierung im unteren Raum***

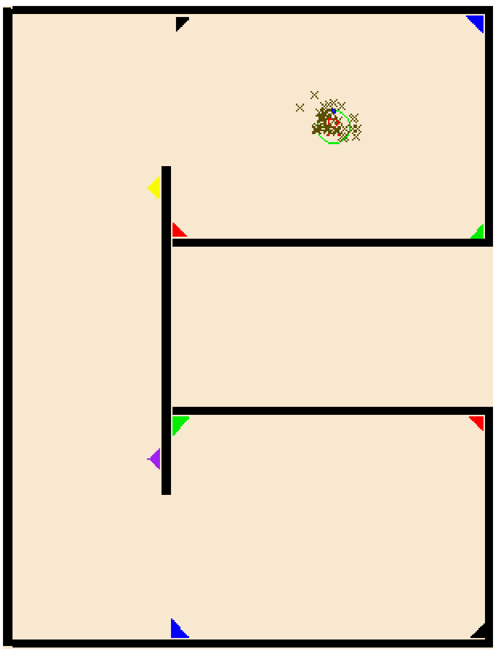


Abbildung 35 ***Lokalisationsergebnis bei Positionierung im oberen Raum***

9.1.5 2 gleiche Räume, unterschiedliche Landmarken, simulierte Fehler

9.1.5.1 Versuchsbeschreibung

Wir benutzen hier wieder die Simulationsumgebung, in der zwei Räume durch einen Gang miteinander verbunden sind. Die Landmarken sind diesmal eindeutig unterscheidbar (Abbildung 36). Die geometrische Anordnung der Landmarken wird nicht verändert.

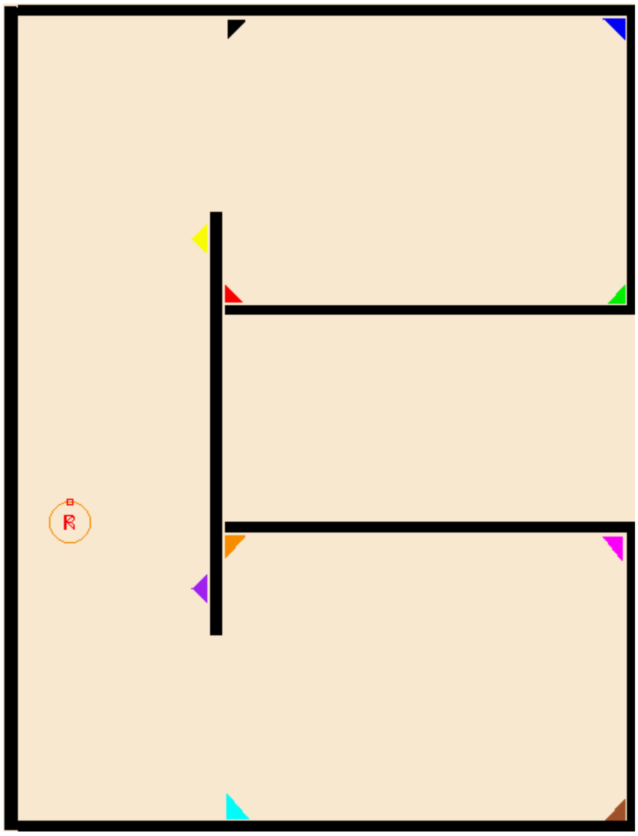


Abbildung 36 ***Zwei Räume mit eindeutig unterscheidbaren Landmarken***

Die Erkundung der Versuchsumgebung findet hierbei nicht unter den gleichen optimalen Bedingungen der vorigen Experimente statt. Simuliert werden Odometriefehler, so daß mit zunehmender Erkundungsdauer eine steigende Abweichung der Positionsangabe des Pfadintegrierers zu erwarten ist. Wir ließen das Navigationssystem zudem die Positionsschätzungen vom Place-Unit-Netz übernehmen,

obwohl die Umgebung noch nicht erkundet ist, so daß der Fehlereinfluß auf die dem Navigationssystem bekannte Information bei der Erkundung groß ist.
Unter diesen Bedingungen wiederholen wir die ersten beiden Versuche aus 9.1.2.2, positionieren also den Roboter jeweils einmal im unteren und oberen Raum.
Wir untersuchen damit die Veränderung der Positionsschätzungen des Place-Unit-Netzes unter Fehlereinfluß in unbekanntem Terrain.
Die simulierten Fehler in der Fortbewegung und die Positionsfehlschätzungen des unfertigen Netzes zeigen sich deutlich durch die Abweichung der Kartenform von der Form der Simulationsumgebung (Abbildung 37).

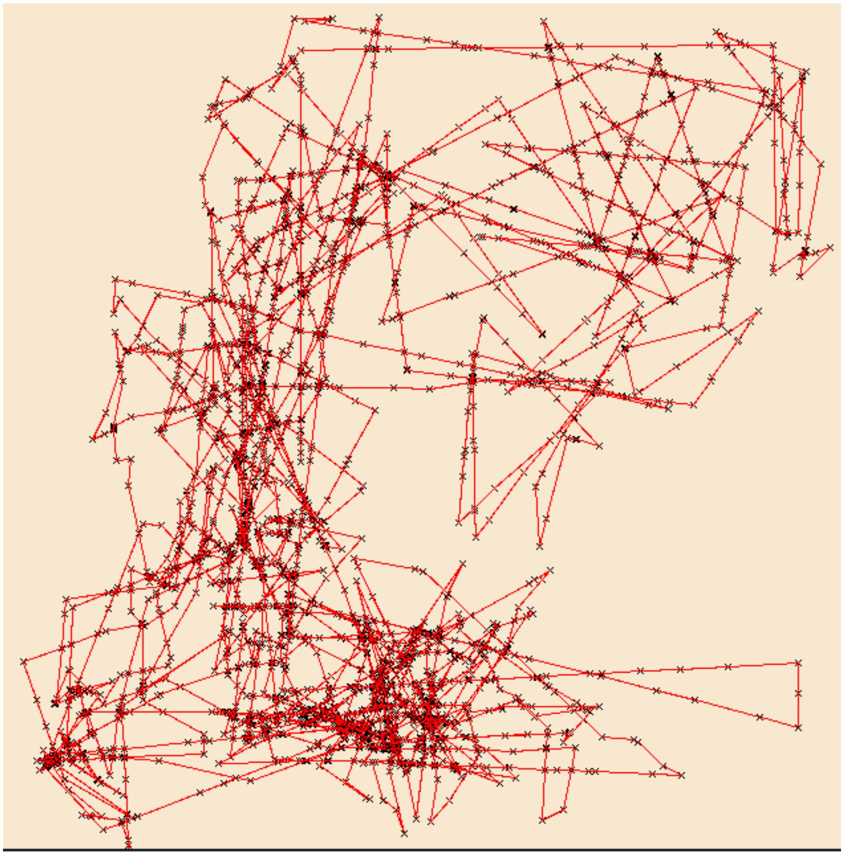


Abbildung 37 ***Topologische Karte der zwei unterscheidbaren Räume unter Fehlereinfluß***

9.1.5.2 Versuchsergebnis

Wir überprüfen nun das Verhalten des Navigationssystems bei Einwirkung von Odometriefehlern und Übernahme von Positionsschätzungen des untrainierten Place-Unit-Netzes während der Erkundungsphase. Die entstehende topologische Karte (siehe Abbildung 37 auf Seite 87) vermittelt einen Eindruck der vom Place-Unit-Netz gelernten, fehlerhaften Positionsangaben. Mögliche Verwechslungen der Räume haben wir durch Verwendung eindeutig unterscheidbarer Landmarken ausgeschlossen, so daß die entstehenden Abweichungen nur Ergebnis der veränderten Erkundungsbedingungen sind.

Die Abweichungen lassen keine präzise Bestimmung des Roboterstandorts zu. Räume, in denen sich der Roboter befindet bleiben jedoch unterscheidbar (Abbildung 38 und Abbildung 39).

Neben den simulierten Odometriefehlern sind fehlerhafte Daten der intern verwalteten Roboterposition für die Abweichungen verantwortlich, da ein nicht vollständig trainiertes Netz bei bisher unbesuchten Stellen der Umgebung mit Positionsschätzungen aus dem bereits erkundeten Bereich antwortet. Dadurch entsteht eine Diskrepanz zwischen der tatsächlichen Position und der Standortschätzung, die bei weiterer Erkundung des neuen Gebietes erhalten bleibt und für fehlerhafte Werte in den neu angelegten Place-Units verantwortlich ist. Aus diesen unpräzisen Werten kann keine korrekte Position mehr bestimmt werden. Die Fehler in den Werten der Place-Units werden auch im späteren Betrieb des Roboters aus dem Place-Unit-Netz übernommen. Für das erste Experiment ergeben sich μ=(51.007, 14.057) und σ=(7.72949, 3.20578), für das zweite μ=(60.574, 56.5213) und σ=(7.99736, 10.7227).

So wird erkenntlich, daß für die Funktionsfähigkeit des Navigationssystems eine gewisse Genauigkeit der Radsensorinformation vonnöten ist, um das Place-Unit-Netz korrekt aufzubauen. Redish und Touretzky erwähnen dieses Problem in [RedTou97] und streben eine Erforschung der von Ratten zur Lösung eingesetzten Strategie an.

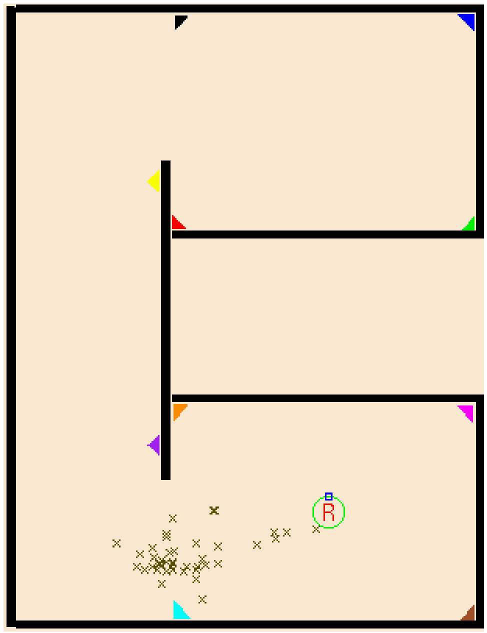


Abbildung 38 ***Ergebnis bei Aufstellung im unteren Raum bei Erkundung mit Fehlereinwirkung***

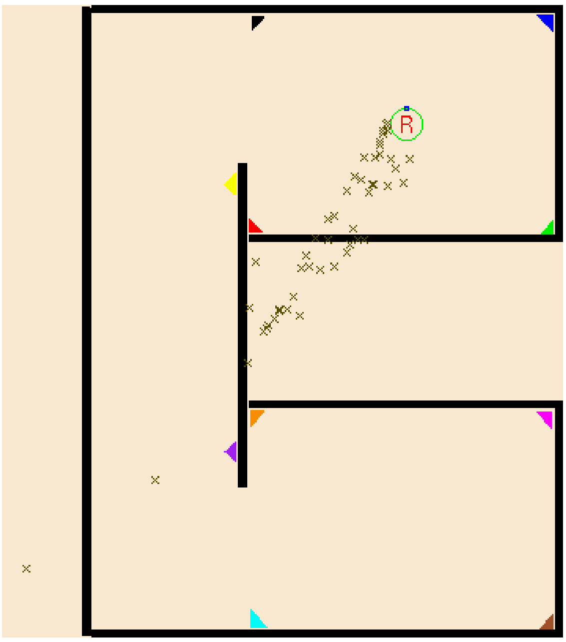


Abbildung 39 ***Ergebnis bei Aufstellung im oberen Raum bei Erkundung mit Fehlereinwirkung***

9.1.6 3 ununterscheidbare Landmarken

9.1.6.1 Versuchsbeschreibung

Wir verwenden hier die Versuchsanordnung aus dem Versuch in 9.1.1. Die Landmarken sind diesmal jedoch ununterscheidbar.

Die simulierte Umgebung wird ohne simulierte Fehlereinflüsse erkundet. Ferner werden beim Anlegen der Place-Units während des Erkundungsprozesses wie in 9.1.1 die somit korrekt vorliegende Radsensorinformation genutzt.

Ebenso werden erneut in dieser Umgebung die 3 Landmarken-Aufgabe, die 3-1 Landmarken-Aufgaben und die 3+1 Landmarken-Aufgabe gestellt.

Es soll untersucht werden, wie das Ergebnis der Positionsbestimmung des Navigationssystems von der Unterscheidbarkeit der Landmarkentypen abhängt.

Die Erkundung fand unter den gleichen Rahmenbedingungen wie der erste Versuch statt.

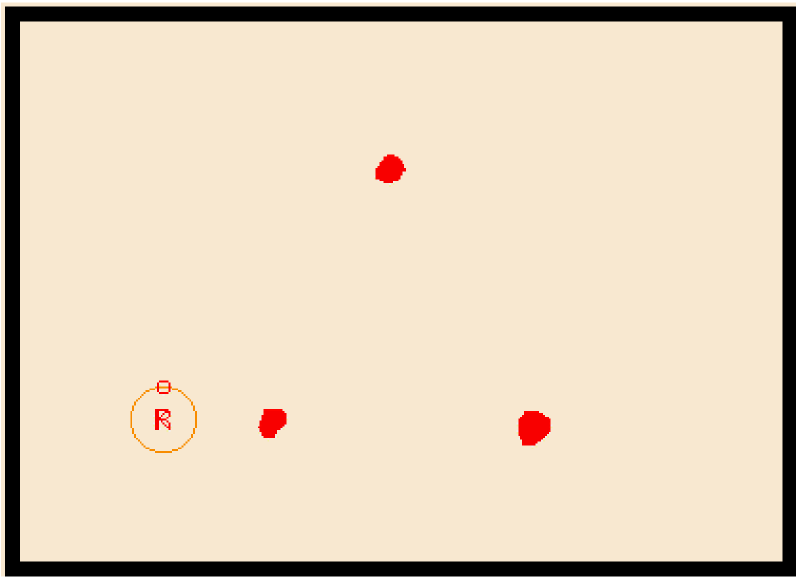


Abbildung 40 ***Umgebung wie im ersten Versuch, jedoch nur mit Landmarken eines Typs***

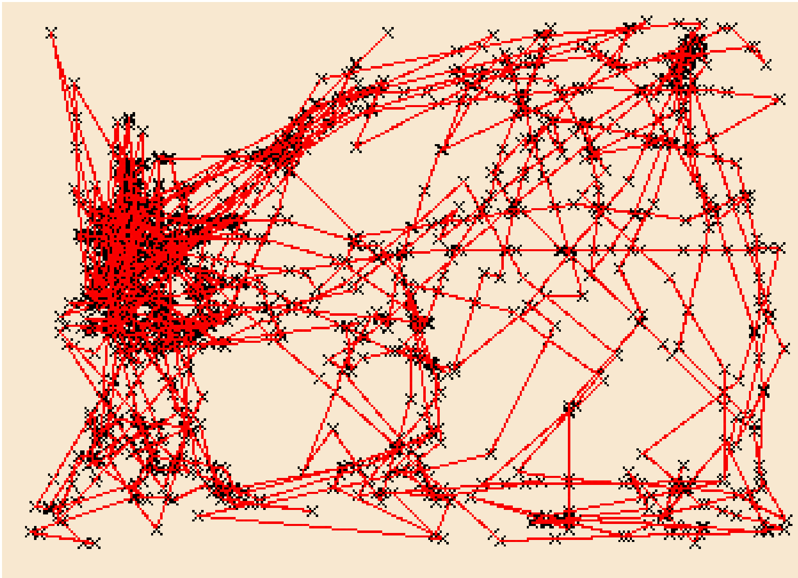


Abbildung 41 ***Topologische Karte der Aufgabe mit 3 ununterscheidbaren Landmarken***

9.1.6.2 Versuchsergebnis

Wir wiederholen hier die Experimente aus 9.1.1 mit ununterscheidbaren Landmarken. Anhand von Abbildung 42 ist erkennbar, daß die Bestimmung des Roboterstandorts eindeutig die Mitte zwischen den drei Landmarken identifiziert (Abbildung 42). Das zeigt sich auch in den Werten μ=(49.6381, 40.8706) und σ=(1.14275, 0.783669).

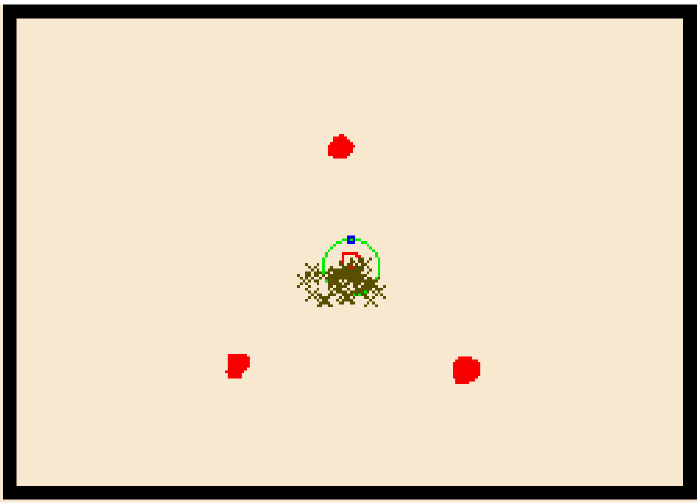


Abbildung 42 ***Ergebnis der 3 Landmarken-Aufgabe mit ununterscheidbaren Landmarken***

Bei der Lösung der 3-1 Landmarken-Aufgaben (siehe Abbildung 43) verschiebt sich das Zentrum der Schätzungshäufung etwas. Die richtige Stelle wird indes erkannt (μ=(48.1834, 41.5179), σ=(1.19273, 0.198661)). Dies ändert sich auch nicht bei Entfernen der rechten unteren Landmarke anstelle der oberen (μ=(49.2969, 40.243), σ=(0.196543, 0.364912)).

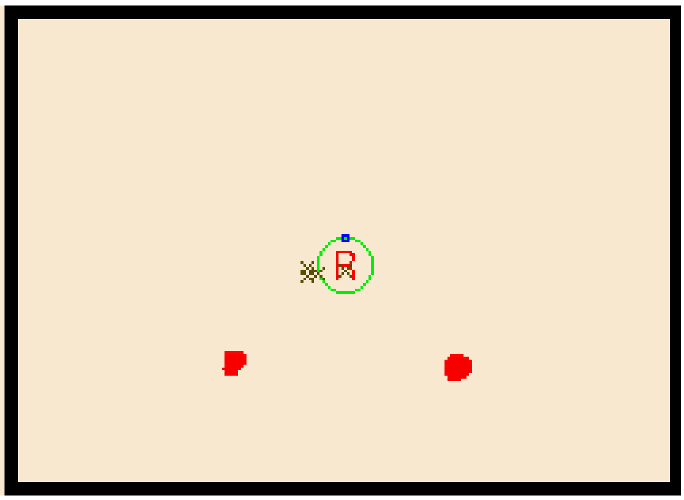


Abbildung 43 ***Ergebnis des 3-1 Landmarken-Problems mit ununterscheidbaren Landmarken***

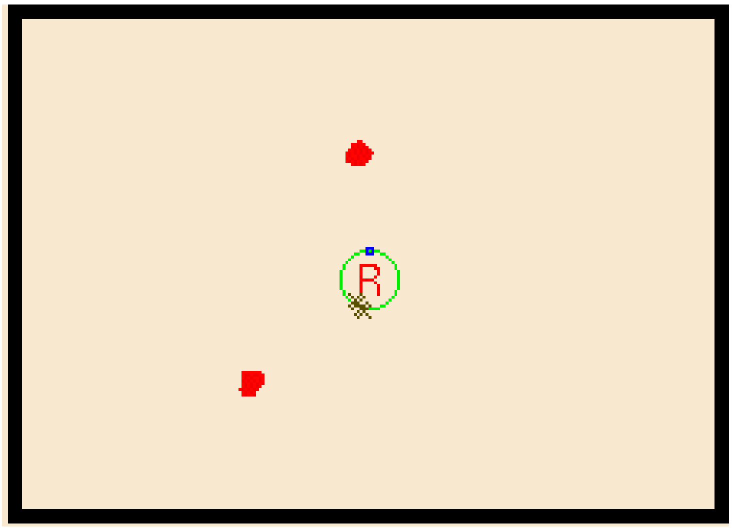


Abbildung 44 ***Ergebnis 3-1 Landmarken-Problems II mit ununterscheidbaren Landmarken***

Wird jedoch eine weitere Landmarke gleichen Typs hinzugefügt, entsteht ein interessantes Ergebnis (Abbildung 45). Es sei dabei an das Ergebnis aus 9.1.1.2 erinnert. Dort entstand ein zweites Cluster von Standortschätzungen. Dadurch, daß in diesem Versuch die Landmarken nicht unterscheidbar sind, entstehen weitere Möglichkeiten für ihre Verwechslung. Die entsprechenden „Phantompositionen“ sind in Abbildung 45 durch weitere Kreise gekennzeichnet. Es entsteht außer dem Cluster an der tatsächlich Aufstellposition ein weiteres an der linken Dreiecksseite, da der Roboter tatsächlich links unterhalb einer Landmarke steht und diese Position daher auch möglich ist (Abbildung 46). Zudem entstehen unterhalb der beiden unten im Bild aufgestellten Landmarken noch Cluster, die denen in der Bildmitte entsprechen. Diese sind, genau wie die dazugehörigen Landmarken, gegenüber den oberen seitlich verschoben. Zudem sind die Häufungen nicht so stark ausgebildet, da die entsprechenden Positionsschätzungen nur auf einer möglichen Landmarkensichtung rechts und links oberhalb des Roboters hin entstehen. Die zu den Positionen gehörenden Schätzungen unterhalb der unteren Landmarkenreihe haben, wie in Abbildung 46 zu sehen, die geringste Ähnlichkeit mit der tatsächlichen Roboteraufstellung, da die Positionsschätzung nur auf der relativen Position zu einer Landmarke beruht. Daher häufen sich die Standortschätzungen für diese Stellen nicht so sehr wie bei den beiden oberen Möglichkeiten. Die starke Verteilung von Standortschätzungen zeigt sich für Mittelwert und Standardabweichung der Meßdaten mit μ=(43.8563, 35.6528) und σ=(15.9289, 12.6247).

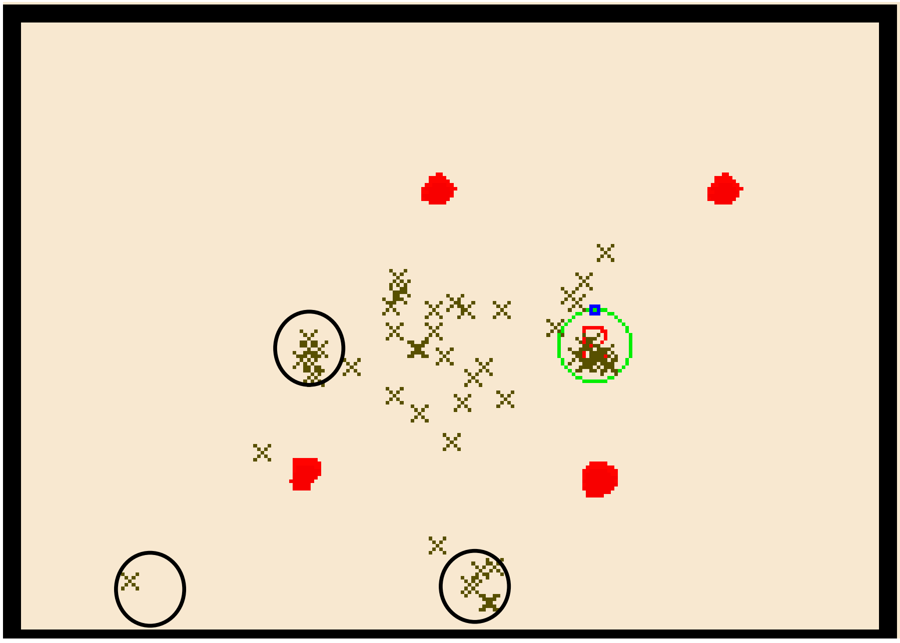


Abbildung 45 ***Ergebnis der 3+1 Landmarken-Aufgabe***

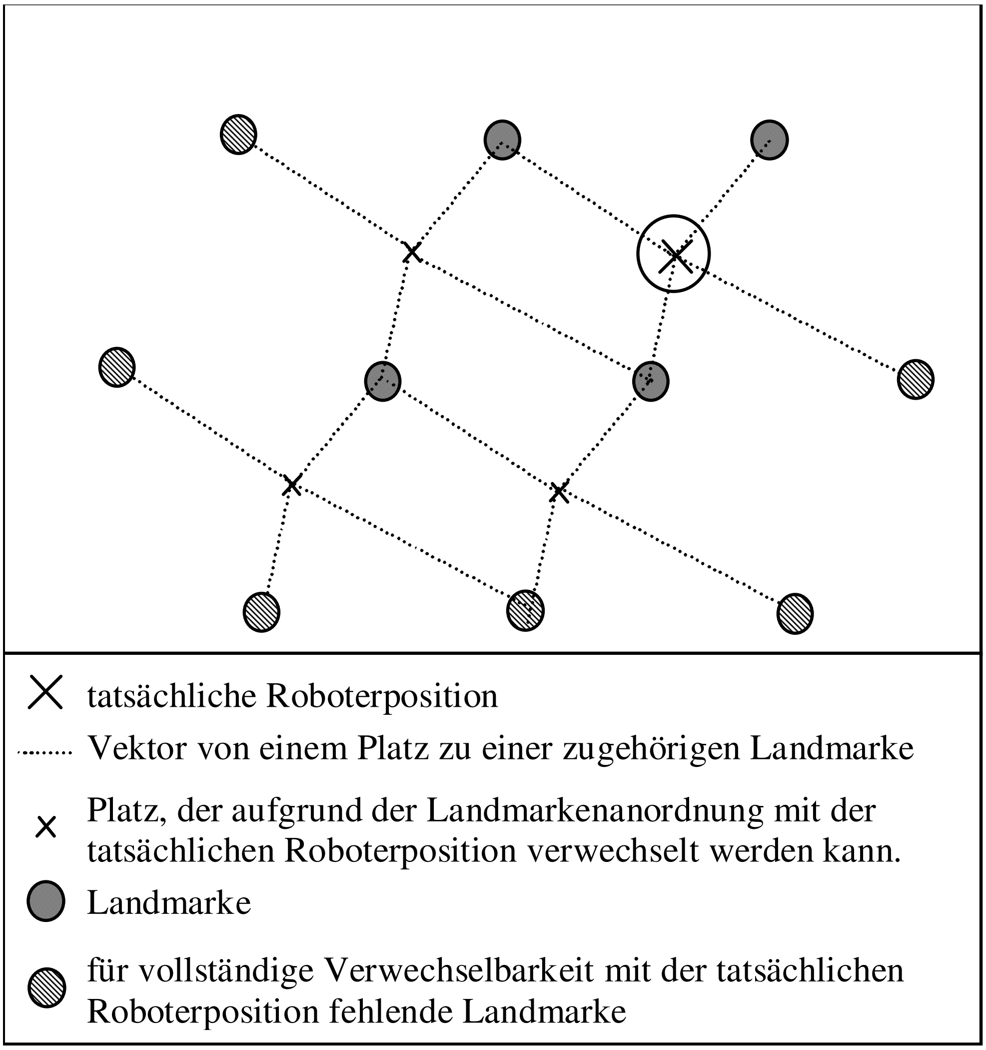


Abbildung 46 ***Positionsverwechslungen bei der 3+1 Landmarken-Aufgabe***

9.1.7 3 Landmarken, simulierte Fehler

9.1.7.1 Versuchsbeschreibung

In diesem Versuch benutzen wir wieder den simulierten Raum aus 9.1.1. Diesmal jedoch übernimmt der Roboter bereits während der Erkundung die Positionsschätzungen des Place-Unit-Netzes obwohl es gerade erst aufgebaut wird. Ferner werden Fehler in der Odometrie simuliert. Die Rahmenbedingungen der Simulation entsprechen somit denen aus 9.1.4.2. In diesem Experiment soll die Auswirkung von fehlerhaften Positionsbestimmungen während der Erkundung auf die resultie-

rende Erkennungsleistung des Place-Unit-Netzes an einem zweiten, kleineren, Beispiel, untersucht werden.
Die topologische Karte, die aus der Erkundung hervorging, ist in Abbildung 47 dargestellt. Simulierte Fortbewegungsfehler und Fehlpositionierungen während der Erkundung zeigen sich in der topologischen Karte durch einzelne Graphenknoten, die außerhalb der für den Roboter erreichbaren Simulationsumgebung liegen.

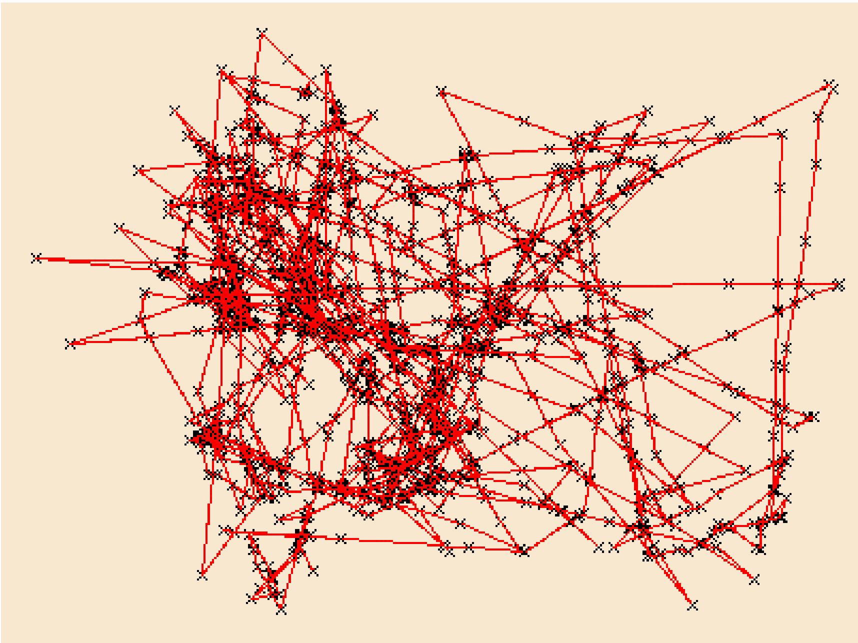


Abbildung 47 ***Topologische Karte der 3 Landmarken-Aufgabe unter Fehlereinfluß***

9.1.7.2 Versuchsergebnis

In diesem Versuch zeigen wir, wie sich simulierte Fehler in der Odometrie und die Übernahme von Positionsschätzungen des untrainierten Place-Unit-Netzes bei der Erkundung einer kleinen Versuchsumgebung auswirken. Als Versuchsumgebung wurde bewußt die Versuchsumgebung aus 9.1.1 gewählt, die sich als besonders unproblematisch erwiesen hat. Wie Abbildung 48 zu entnehmen ist, treten trotzdem ähnliche Fehler wie in 9.1.4.2 auf, die sich in Mittelwert und Standardabweichung der Meßdaten bemerkbar machen (μ=(35.6687, 31.134), σ=(12.5591, 10.1252)). Auch für kleine Umgebungen ist also eine möglichst präzise Umgebungserkundung nötig.

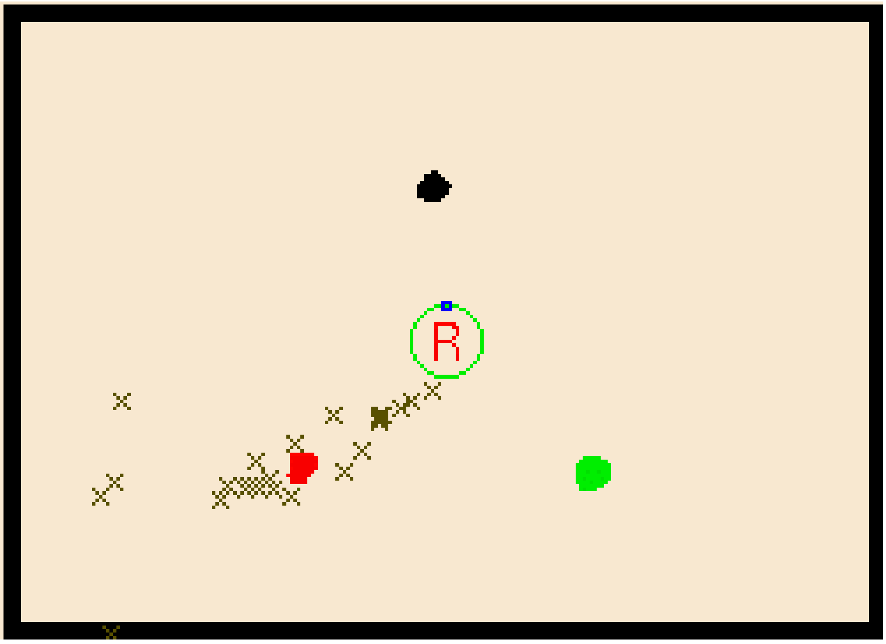


Abbildung 48 Ergebnis der 3 Landmarken-Aufgabe unter Fehlereinfluß bei der Erkundung

Kapitel 10 Diskussion

Wir haben ein System vorgestellt, das diverse Aspekte der Rattennavigation nachbildet. Es besteht aus gekoppelten Komponenten für Platzerkennung, Pfadintegration. Grundlage hierfür ist das in [TouWan94] vorgestellte Verfahren.
Die Arbeit von Touretzky wurde erweitert um Erkennung verschiedener Landmarkentypen, was eine höhere Positionierungspräzision erlaubt. Sind Landmarkenkonfigurationen eindeutig unterscheidbar, muß für eine Positionsidentifizierung nicht einmal eine ungefähre Position bekannt sein.
Die Place-Units führen eine Fuzzy-Auswertung durch, bei der nicht verfügbare Terme von Eingabekomponenten einfach wegfallen. Plätze können aufgrund beliebiger Kombinationen von allozentrischen Winkeln zu Landmarken (bei bekannter Roboterausrichtung), Entfernungen zu Landmarken, Landmarkentypen (bei erfolgreicher Erkennung), egozentrischen Winkeldifferenzen zwischen Landmarkenpaaren und Koordinaten des Pfadintegrierers erkannt werden. Je mehr dieser Hinweise dabei verfügbar sind, desto höher ist die Erkennungsleistung.
Diese beliebige Kombinierbarkeit wurde durch Benutzung einer Verfügbarkeitsmatrix erreicht, mit der sich beliebige Merkmalskomponenten ausblenden lassen.
Daß Umgebungsbereiche sich nicht vollkommen gleichen dürfen, zumindest bezüglich der betrachteten Umgebungselemente, um eine Unterscheidung zu ermöglichen, trifft gleichsam auf Lebewesen zu. Im Falle vollständiger Ununterscheidbarkeit ziehen Lebewesen ebenfalls eine Kenntnis über die ungefähre Position hinzu.
Einen der größten Nachteile des Verfahrens von Touretzky et al., nämlich die hohe Anzahl benötigter Place-Units, konnten wir deutlich mindern, indem nur Place-Units neu angelegt werden dürfen, die bisher nicht im Netz vorhanden sind. Die Trainingsumgebungen der hier präsentierten Experimente hatten zwischen ca. 7000 und 12000 Place-Units. Werden einfach beliebige neue Place-Units angelegt, steigt die Zahl der für größere Umgebungen benötigten Units leicht auf Vielfache von 10^5 an. Dadurch bedingt arbeitet die Standortbestimmung schneller, da weniger Aktivierungen berechnet werden müssen.

Die Verwaltung der topologischen Karte ist derzeit recht aufwendig, da beim Anlegen neuer Verbindungsstrecken Überkreuzungen mit bestehenden Verbindung überprüft werden. Da hierzu Positionsdaten der Place-Units benutzt werden, ist die Erkundung auch bezüglich der Kartenkorrektheit von der Genauigkeit der Erkundung abhängig. Fehlerhafte Kreuzungen in der topologischen Karte haben jedoch keine negativen Auswirkungen, da Strecken, die nicht passierbar sind, bei einem Benutzungsversuch als blockiert markiert werden können.
Während des Durchfahrens von unbekannten Stellen der Umgebung können sich Abweichungen in der Roboterausrichtung und Position akkumulieren. Eine Lösung dieses Problems ist nötig und Gegenstand der aktuellen Forschung [RedTou97].

Kapitel 11 Ausblick

Das vorgestellte System wurde nicht auf einem echten Robotersystem getestet. Dazu sind weitere Arbeiten nötig. So muß ein Bildverarbeitungssystem die Erkennung von Hindernissen und Landmarken unterschiedlicher Typen bewältigen. Die nötige Vorverarbeitung erfordert ein leistungsfähiges Bildverarbeitungssystem, das die Arbeit in einer praxisgerechten Geschwindigkeit leisten kann. Die Erkennung unterschiedlicher Landmarkentypen könnte durch neuronale Klassifizierer erfolgen.

Der Problematik, genaue Positionsangaben des Pfadintegrierers während der Erkundungsphase zur Verfügung zu stellen, könnte bei einem echten Robotersystem mit verschiedenen Mitteln begegnet werden. Zum einen könnten Radsensoren sowohl an angetriebenen als auch an nicht-angetriebenen Rädern angebracht werden. Die Bestimmung der Roboterausrichtung könnte mit Hilfe eines elektronischen Kompaß unterstützt werden. Während der Erkundungsphase wäre sogar der Einsatz eines differential GPS-Systems (DGPS) denkbar (Beim DGPS wird ein Referenzempfänger mit bekannter Position benutzt, der zur Kompensation der im GPS eingebrachten Ungenauigkeiten verwendet wird).

Ein aufwendigeres Erkundungsverfahren könnte den Erkundungsprozeß beschleunigen, indem gezielt unerkundete Bereiche der Karte angesteuert werden, um einen Erkundungsnutzen zu maximieren. Da dabei kürzere Wegstrecken für die Erkundung entständen, würden sich Odometriefehler weniger stark auswirken.

Für eine schnellere Erkundung simulierter Umgebungen wäre ein Trainingsverfahren denkbar, das gezielt Place-Units für eine Testumgebung generiert (wie von Touretzky et al. benutzt). Versuche mit dem Simulator benötigen dann weniger Vorbereitungszeit.

Die Wegplanung für den Roboter ist in dem Navigationssystem vorgesehen, jedoch noch nicht implementiert. Dazu gehören eine Graphensuche und die Generierung gezielter Steuerbefehle für das Fahrsystem anhand der topologischen Karte.

Erweiterungen der Steuerungsmöglichkeiten des Simulators durch Interaktion mit einem Benutzer ist wünschenswert. So könnte der Benutzer gezielt die Verfügbar-

keit von Radsensordaten, Landmarkentypinformationen oder auch aller optisch wahrgenommenen Komponenten während einer laufenden Simulation beeinflussen.
Das Navigationssystem kann um Möglichkeiten erweitert werden, nicht zusammenhängende Kartenfragmente, die zu verschiedenen, nicht zwangsläufig zusammenhängenden Umgebungsteilen gehören, zu verwalten. Bei Auffinden von Überlappungen könnten diese Karten zusammengeführt werden. Damit wäre es beispielsweise auch möglich, Karten verschiedener Bürostockwerke zu pflegen.

Kapitel 12 Abbildungsverzeichnis

Abbildung 1 Funktionelle Organisation von Orientierungs- und Wahrnehmungsmechanismen der Ratte (aus: [WanTou94]) 19
Abbildung 2 Pixelverschiebung durch Bewegung des Roboters 25
Abbildung 3 Die drei Komponenten eines Roboternavigationssystems. 28
Abbildung 4 Module der drei Roboterkomponenten 29
Abbildung 5 Detailliertes Schema des Modells 30
Abbildung 6 Positionsschätzung mit Bildung eines gewichteten Positionsmittels 51
Abbildung 7 Der Roboter erkundet eine Sackgasse. 53
Abbildung 8 Kreuzung eines bereits befahrenen Streckenabschnitts 54
Abbildung 9 Anlegen eines neuen Kreuzungspunktes 54
Abbildung 10 Bedeutung der in Klassendiagrammen verwendeten Symbole 57
Abbildung 11 Klassendiagramm des Environment-Moduls 58
Abbildung 12 Klassendiagramm des Object-Moduls 60
Abbildung 13 Klassendiagramm des Order-Moduls 61
Abbildung 14 Klassendiagramm des Placeunits-Moduls 62
Abbildung 15 Klassendiagramm des Placemap-Moduls 63
Abbildung 16 Klassendiagramm des Roboter-Moduls 64
Abbildung 17 Die simulierte Umgebung des ersten Versuchs 67
Abbildung 18 Die topologische Karte der 3 Landmarken-Aufgabe 68
Abbildung 19 Lokale Ansicht der Position, an der wir die Versuchsergebnisse ermitteln 68
Abbildung 20 Versuchsergebnis der Aufgabe mit 3 eindeutig unterscheidbaren Landmarken 70
Abbildung 21 Lokalisationsergebnis der ersten 3-1 Landmarken-Aufgabe 71
Abbildung 22 Lokalisationsergebnis der zweiten 3-1 Landmarken-Aufgabe 72
Abbildung 23 Lokalisationsergebnis der 3+1 Landmarken-Aufgabe 73
Abbildung 24 Die Simulationsumgebung „Hufeisenanordnung“ 74
Abbildung 25 Topologische Karte der „Hufeisenanordnung“ 75
Abbildung 26 Lokalisationsergebnis bei der „Hufeisenanordnung“ 76
Abbildung 27 Simulationsumgebung mit 2 gleichen Räumen 77
Abbildung 28 Topologische Karte der zwei gleichen Räume 78
Abbildung 29 Ergebnis des Versuchs mit 2 gleichen Räumen mit Aufstellung im unteren Raum 80
Abbildung 30 Ergebnis des Versuchs mit 2 gleichen Räumen mit Aufstellung im oberen Raum 81
Abbildung 31 Ergebnis mit Aufstellung im unteren Raum und bekannter Radsensorposition 82
Abbildung 32 Ergebnis mit Aufstellung im oberen Raum und bekannter Radsensorposition 83
Abbildung 33 Zwei Räume mit unterschiedlichen Landmarkenanordnungen 84
Abbildung 34 Lokalisationsergebnis bei Positionierung im unteren Raum 85
Abbildung 35 Lokalisationsergebnis bei Positionierung im oberen Raum 85
Abbildung 36 Zwei Räume mit eindeutig unterscheidbaren Landmarken 86
Abbildung 37 Topologische Karte der zwei unterscheidbaren Räume unter Fehlereinfluß 87
Abbildung 38 Ergebnis bei Aufstellung im unteren Raum bei Erkundung mit Fehlereinwirkung 89
Abbildung 39 Ergebnis bei Aufstellung im oberen Raum bei Erkundung mit Fehlereinwirkung 89
Abbildung 40 Umgebung wie im ersten Versuch, jedoch nur mit Landmarken eines Typs 90
Abbildung 41 Topologische Karte der Aufgabe mit 3 ununterscheidbaren Landmarken 91
Abbildung 42 Ergebnis der 3 Landmarken-Aufgabe mit ununterscheidbaren Landmarken 92
Abbildung 43 Ergebnis des 3-1 Landmarken-Problems mit ununterscheidbaren Landmarken 92
Abbildung 44 Ergebnis 3-1 Landmarken-Problems II mit ununterscheidbaren Landmarken 93

Abbildung 45 Ergebnis der 3+1 Landmarken-Aufgabe ________________________________ *94*
Abbildung 46 Positionsverwechslungen bei der 3+1 Landmarken-Aufgabe ________________ *95*
Abbildung 47 Topologische Karte der 3 Landmarken-Aufgabe unter Fehlereinfluß __________ *96*
Abbildung 48 Ergebnis der 3 Landmarken-Aufgabe unter Fehlereinfluß bei der Erkundung ___ *97*

Kapitel 13 Literaturverzeichnis

[BacWax94] Ivan A. Bachelder und Allen M. Waxman, Mobile Robot Visual Mapping an Localization: A view-based neurocomputational architecture that emulates hippocampal place learning, Neural Networks, Vol. 7, Nos. 6/7, pp. 1083-1099, 1994

[BuhBur95] J.Buhmann, W.Burgard , A.B.Cremers, D.Fox, T.Hofmann, F.Schneider, J.Strikos, S.Thrun: The mobile robot Rhino, AI Magazine, 16:1, Spring 1995

[ConMah93] Jonathan H. Connell, Sridhar Mahadevan: Rapid task learning for real robots, Robot Learning, 1993, pp. 105-140

[Kurz95] Andreas Kurz: ALEF: An autonomous vehicle which learns basic skills and constructs maps for navigation, Robotics ans Autonomous Systems 14, 1995, pp. 171-183

[MalSch95] Mallot, Schölkopf: Learning of cognitive maps from sequences of views, Invited Paper at ESANN’95: 3rd European Symposium on Artificial Neural Networks, Brussles, April 19-21, 1995

[Matari92] Maja J. Mataric, Integration of representation into goal-driven behavior-based robots, IEEE Transactions on robotics and automation, Vol. 8, No. 3, June 1992, pp. 304-312

[MenKak93] Min Meng und A.C.Kak, NEURO-NAV: A neural network based architecture for vision-guided mobile robot navigation using non-metrical models of the environment, Proceedings of the 1993 IEEE International Conference of Robotics and Automation

[Nehmzo95] Ulrich Nehmzow: Animal and robot navigation, Robotics and Atonomous Systems 15, 1995, pp. 71-81

[RedTou96] A.D. Redish and D.S. Touretzky: Modeling interactions of the rat's place and head direction systems. Advances in Neural Information Processing Systems 8, D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, eds., MIT Press.

[RedTou97] A.D. Redish and D.S. Touretzky: Navigating with landmarks: computing goal locations from place codes. In K. Ikeuchi and M. Veloso, (eds.), Symbolic Visual Learning, Oxford University Press.

[Röfer93] Thomas Röfer, Controlling a robot with image-based homing, Proceedings of the 1993 IEEE International Conference On Neural Networks, Vol. I, Zentrum für Kognitionswissenschaften Universität Bremen, Tagungsband des Workshops Kognitive Robotik

[SchöMal94] Schölkopf und Mallot: View-based cognitive mapping and path planning: view graph, Technical Report No. 7, Nov. 1994, Max-Planck-Institut für biologische Kybernetik, Tübingen.

[Thrun93] Sebastian B. Thrun: Exploration and model building in mobile robot domains, Proceedings of the 1993 IEEE International Conference of Robotics and Automation, pp. 175ff

[TouWan94] D.S. Touretzky, H.S. Wan, A.D. Redish: Neural Representation of Space in Rats and Robots, in: Computational Intelligence: Imitating Life. Proceedings of the symposium held at 1994 IEEE World Congress on Computational Intelligence, 1994.

[VesTsc93] Sjur J. Vestli, Nadine Tschichold-Gürman, Henrik Andersson: Learning control and localisation of mobile robots, in: P. Levi Th. Bräunl (Hrsg.): Autonome Mobile Systeme, 1994, Springer Verlag, pp. 202-213

[WalHal93] Ashley Walker, John Hallam, David Willshaw: Bee-havior in a Mobile Robot: The construction of a self-organized cognitive map and its use in robot navigation within a complex, natural environment, Proceedings of the 1993 IEEE International Conference of Robotics and Automation, pp. 1451ff

[WanTou94] H. S. Wan, D. S. Touretzky, A. D. Redish: Towards a computational theory of rat naigation, in: M. Mozer, P. Smolensky, D. Touretzky, J. Elman and A. Weigend, eds., Proceedings of the 1993 Connectionist Models Summer School, pages 11-19, Lawrence Earlbaum Associates

Hiermit erkläre ich, daß ich die vorliegende Arbeit eigenständig verfaßt und keine anderen als die angegebenen Quellen und Hilfsmittel genutzt habe.

Kiel, den 10.12.1997

(Unterschrift Oliver Harwart)