„Navigation mobiler Systeme in Indoor-Umgebungen"

Diplomarbeit an der Fachhochschule Brandenburg
Fachbereich Informatik und Medien

vorgelegt von Jörg Dreßler

1. Gutachter: Prof. Dr. J. Heinsohn
2. Gutachter: Dipl.-Inform. I. Boersch

Brandenburg, den 3. August 2001

Aufgabenstellung

„Navigation mobiler Systeme in Indoor-Umgebungen"

Zielstellung des Themas ist die Untersuchung von Selbstlokalisierungsverfahren mobiler Systeme in Gebäuden (insbesondere Kalman-Filter) zur intelligenten Verknüpfung mehrerer Informationsquellen und die Implementierung, sowie Evaluation auf einem realen Roboter. Die Umsetzung soll besonderen Wert auf die Erweiterbarkeit des Systems über die herkömmlichen Sensorquellen (Sonar, Odometrie) hinaus auf zusätzliche Sensorquellen (z.B. Kompass, Gyroskop, DGPS) legen.

Danksagung

An dieser Stelle möche ich Herrn Dipl.-Inform. I. Boersch für die Unterstützung meiner Arbeit danken. Dafür, daß er in zahlreichen Diskussionen beratend zur Seite stand und wertvolle Hinweise und Anregungen beisteuerte.

Inhaltsverzeichnis

Abbildungsverzeichnis

Teil I

Einleitung

Im Folgenden wird eine Einführung zu dieser Diplomarbeit gegeben. Dies umfaßt sowohl die allgemeine Betrachtung von Servicerobotern und die Beschreibung der Voraussetzungen, wie zum Beispiel die Laborumgebung, aber auch den Aufbau der Arbeit.

1 Serviceroboter

1.1 Science Fiction contra Realität

Isaac Asimov beschreibt in seinem Roman, [Asi97], einen Roboter in einer fernen Zukunft, wie folgt.

Robot QT1 saß unbeweglich da. Die polierten Platten seines Körpers leuchteten, und das glühende Rot der photoelektrischen Zellen seiner Augen war ständig auf den Erdenmenschen gerichtet, der ihm am Tisch gegenübersaß.

Diese Zukunftsvision geht natürlich weit über das hinaus, was heute technisch machbar ist. Bevor uns Roboter als menschenähnliche und intelligente Wesen gegenübersitzen wird wohl noch eine lange Zeit vergehen. Trotzdem verdeutlicht dieser Auszug gewisse Grundlagen.
Schlägt man im Lexikon, [LIB94], den Begriff *Roboter* nach, so findet man folgende Erklärung: „*Maschinenmensch*, Bezeichnung für eine Apparatur mit äußerlicher Gestalt eines Menschen, die menschliche Bewegungen oder auch einfache Hantierungen ausführen kann". Roboter sind also Maschinen, die für uns Menschen gewisse Aufgaben erfüllen sollen. Sie sollen zum Beispiel monotone Tätigkeiten ausführen oder Aufgaben im gefährlichen Umfeld erfüllen.
Wenn diese Maschinen im direkten Kontakt mit Menschen und in unserem Lebensraum arbeiten, kommt ihnen eine besondere Stellung zu, sie werden als Serviceroboter bezeichnet.

Eine Definition was Serviceroboter sind, [SV96], ist folgende. Ein *Serviceroboter* ist eine freiprogrammierbare Bewegungseinrichtung, die teil- oder vollautomatisch Dienstleistungen verrichtet. Dienstleistungen sind dabei Tätigkeiten, die nicht der direkten industriellen Erzeugung von Sachgütern, sondern der Verrichtung von Leistungen an Menschen und Einrichtungen dienen.

Ein Serviceroboter besteht grundlegend aus zwei Komponenten, einer mobilen Plattform und den Anwendungsmodulen, [SV96]. Die folgende Abbildung veranschaulicht

dies noch einmal.

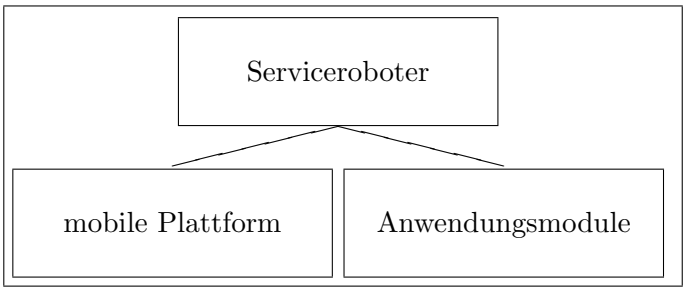

Abbildung 1: Die zwei grundlegendsten Komponenten eines Serviceroboters.

Die Entwicklung der Roboter kann, [SV96], in drei grundlegende Stufen eingeteilt werden.

- Industrie-Roboter
- Service-Roboter
- Personal-Roboter

Die Roboter dieser Stufen unterscheiden sich in ihrem Grad der Autonomie und in dem Grad der Unstrukturiertheit der Umgebung, in der sie agieren. Der autonomste und am wenigsten umgebungsabhänige ist der sogenannte Personal-Robot.

1.2 Aktueller Stand und Beispiele

Der ,,künstliche Mensch" ist und bleibt weit entfernt, [Hei00]. Wenn man Moravecs Buch, [Mor90], liest, kann man zu dem Schluß gelangen, daß es sich bei den heutigen Systemen um Maschinen handelt, die schwerfällig denken, die verschwommen sehen, ungeschickt greifen und ihre Welt zögerlich erkunden. Jedoch werden heute schon in den vielfältigsten Bereichen Serviceroboter eingesetzt. Diese sind in der Lage Aufgaben teil- bzw. vollautonom zu erfüllen. Typische Aufgaben für heutige Serviceroboter sind, zum Beispiel Reinigungsaufgaben, Aufgaben im Transport und Verkehr, im Baugewerbe, im Sicherheitsumfeld, in der Medizin aber auch der Einsatz im Bereich Hobby und Freizeit.

Im folgenden sollen zwei konkrete Beispiele für Serviceroboter dargelegt werden. Zum einen handelt es sich um den Roboter ,,Komm-Rein" und zum anderen um das System ,,SKYWASH".
Der Roboter ,,Komm-Rein" ist ein Serviceroboter aus dem Bereich Entertainment. Er hat im Museum für Kommunikation in Berlin die Aufgabe, so viele Besucher wie möglich zu begrüßen und über spezielle Events und Ausstellungen zu informieren. Sein Erscheinungsbild ist nachfolgend dargestellt.

Abbildung 2: Der Serviceroboter ,,Komm-Rein",[FHG].

Er ist in der Lage zwischen einzelnen Besuchern und Besuchergruppen zu unterscheiden. Die Besucher werden mit Hilfe eines Laserscanners anhand der Beine identifiziert. Außderdem ist er in der Lage sich bereits begrüßte Besucher zu merken. Eine ausführliche Darstellung ist unter [FHG] nachzulesen.

Das System ,,SKYWASH" dient zur Reinigung der Außenhaut von Flugzeugen. Das System besteht aus einem Basisfahrzeug und einem Manipulatorarm. Dieser besteht aus elf programmierbaren Achsen. Am Ende des Arms befindet sich eine Waschbürste zur Reinigung des Flugzeuges. Eine Darstellung des arbeitenden Systems ist in der folgenden Abbildung zu sehen.

Abbildung 3: Das Flugzeugreinigungssystem ,,SKYWASH", [FHG].

Das System kann praktisch an jedem Ort innerhalb und außerhalb des Hangars ohne Markierungen am Boden oder am Flugzeug arbeiten. Dies wird unter anderem durch den Einsatz eines 3-D Scanners erreicht. Je nach Verschmutzungsgrad kann das System langsam oder schnell reinigen. Eine detailierte Beschreibung ist in, [SV96] nachzulesen.

Weitere nennenswerte Serviceroboter sind zum Beispiel.

- Staubsauger für den Heimbereich von Panasonic oder Hitachi
- Das Kurierdienstsystem HelpMate
- Bestuhlungsroboter von Fujita

Alle diese Erläuterungen und Beispiele zeigen, daß das Gebiet der Serviceroboter ein Zukunftsmarkt ist. Die Entwicklungen auf diesem Gebiet schreiten rasch voran. Der Bedarf an Servicerobotern ist groß. Neue Forschungsrichtungen wie ,,artificial life" und ,,Genetisches Programmieren" führen zu einer immer höheren Innovationsgeschwindigkeit, [Hei00]. Vielleicht werden in naher Zukunft die Visionen von Asimov, Moravec, Kurzweil und vielen anderen Wissenschaftlern wahr.

2 Die Aufgabe

Diese Diplomarbeit befaßt sich grundlegend mit den folgenden Themengebieten:

- Navigation mobiler Roboter
- Sensor-Data-Fusion

Im Bereich der Navigation mobiler Roboter wird hier der Schwerpunkt auf die Selbstlokalisation gelegt, daß heißt, der Roboter soll in seiner Umgebung stets wissen, wo er sich befindet. Im Gebiet der Sensor-Daten-Fusion wird besonders auf den Kalman-Filter näher eingegangen, einem Werkzeug zur Fusion mehrerer sehr unterschiedlicher Quellen.
Im Ergebnis wird in dieser Arbeit eine ,,Fusions-Architektur" aufgezeigt, die es einem Roboter ermöglichen soll, seine Positionsbestimmung auch unter *schwierigen* Bedingungen durchzuführen. Mit Hilfe dieser *intelligenten Verknüpfung* soll es möglich sein, die unterschiedlichsten Fehler ausgleichen zu können. Beim Entwurf wurde besonderer Wert auf die Erweiterbarkeit des Systems gelegt.

3 Die Laborumgebung / Voraussetzungen

Die vorliegende Arbeit wurde im Fachbereich Informatik und Medien der FH Brandenburg erstellt. Die praktischen Arbeiten erfolgten im Labor für Künstliche Intelligenz, dem KI-Labor. Dieses verfügt über zwei Roboter des Types *Pioneer 2 CE*. Diese und die damit in Verbindung stehenden Komponten, wie die *Saphira*-Software und die Entwicklungsumgebung *MS Visual C++ 5.0* sind Gegenstand der nachfolgenden Betrachtungen.

3.1 Der Roboter Pioneer 2 CE

Der Roboter Pioneer 2 CE ist ein intelligenter Roboter, der von der amerikanischen Firma ActiveMedia, [AM], produziert und vertrieben wird. Die FH Brandenburg besitzt zwei solcher Roboter mit den Namen *Alfa* und *Romeo*. Jeder dieser beiden Roboter ist mit Odometrie-Encodern, Sonarsensoren, einem elektronischen Kompass und einer Kamera ausgestattet. Zudem besitzt jeder dieser Roboter als Aktor einen

Greifer. Die folgende Abbildung zeigt den Pioneer 2 CE Roboter *,,Romeo"* der FH Brandenburg.

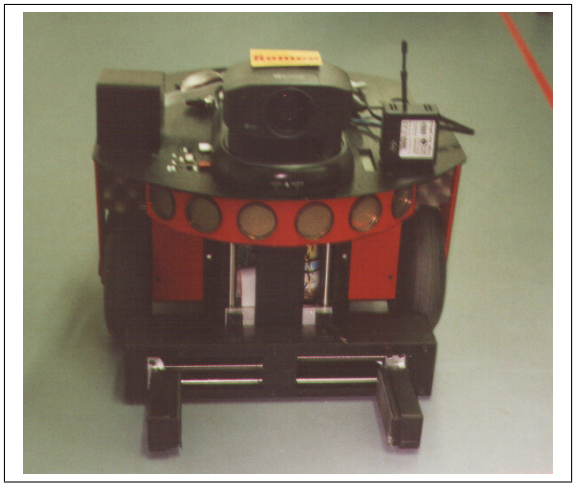

Abbildung 4: Der *,,Romeo"* Roboter Pioneer 2 CE.

Zu jedem Roboter gehört ein Hostrechner, ein Laptop. An der FH Brandenburg ist dies ein Compaq Armada 1750, PII mit 300 MHz und 64MB Arbeitsspeicher. An dem Laptop ist ein Funkmodem angeschlossen, welches die Kommunikation des Roboters mit der Saphira Software ermöglicht.
Die praktische Seite dieser Diplomarbiet, daß heißt die Entwicklung und Evaluation, wurde mit dem Pioneer 2 CE Roboter *Romeo* durchgeführt.

3.2 Die Saphira Software

Wie bereits erwähnt ist Saphira die Software, welche auf dem Laptop die Robotersteuerung übernimmt. Saphira ist eine Robotik-Anwendungs Entwicklungsumgebung. Sie wurde am SRI International's AI Center entwickelt, maßgeblich daran beteilig war Kurt Konolige.
Die Saphira Software ist eine Client Server Architektur. Der Client ist hier der Laptop mit der Saphira Umgebung, der Server ist der reale Pioneer Roboter bzw. der Simulator.

Die Control Architektur von Saphira besteht aus dem State Reflector, der den aktuellen Zustand des Roboters wiederspiegelt. Desweiteren werden zwei geometrische Räume unterschieden, der Local Perceptual Space, LPS, und der Global Perceptual Space, GPS. Der erste gibt die lokale Roboterumgebung, Roboter zentriert, wieder, der zweite die globale Roboterumgebung. Beide sind über die Roboterposition miteinander gekoppelt. Beide Umgebungen können sich wiedersprechen.
Saphira unterstützt fuzzy control und reaktives Planen, die gesamte Sensorik und Aktorik läßt sich auswerten und ansteuern. So zum Beispiel könnnen die Sonarinformationen ausgelesen und der Greifer angesprochen werden.
Eine Darstellung des graphischen Interfaces der Saphira Software ist in der folgenden Abbildung zu sehen.

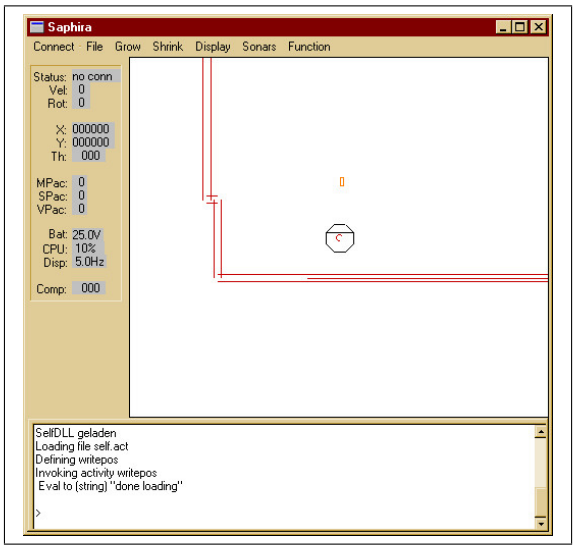

Abbildung 5: Das Anwendungsfenster der Saphira-Software.

3.3 MS Visual C++ 5.0

Microsoft Visual C++ ist eine integrierte Entwicklungsumgebung, eine integrated development enviroment, kurz IDE, daß heißt sie beinhaltet alle Komponenten zum Erstellen, Compilieren, Verknüpfen und Testen von Windows-Programmen. So besteht Visual C aus dem sogenannten Build-System, den Bibliotheken und sonstigen

Tools. Bei der Programmerstellung kommen zwei grundsätzliche Ansätze in Frage. Zum einen die sogenannte WIN32-Programmierung und zum anderen die objektorientierte Programmierung mit C++ und der MFC, der Microsoft Foundation Class Library.
In der vorliegenden Arbeit wurde die Programmentwicklung mit Microsoft Visual C++ Version 5.0 durchgeführt. Die Programme wurden ohne Hilfe der MFC geschrieben, also als reine WIN32-Programme in der Programmiersprache C. Dies hat den Vorteil, daß der Code sehr übersichtlich und auf das wesentliche beschränkt bleibt. Auch die fertig compilierten Anwendungen sind deutlich schlanker als unter Verwendung der MFC.
Eine Abbildung der Fenster, der Entwicklungsumgebung Visual C++ ist in der folgenden Abbildung dargestellt.

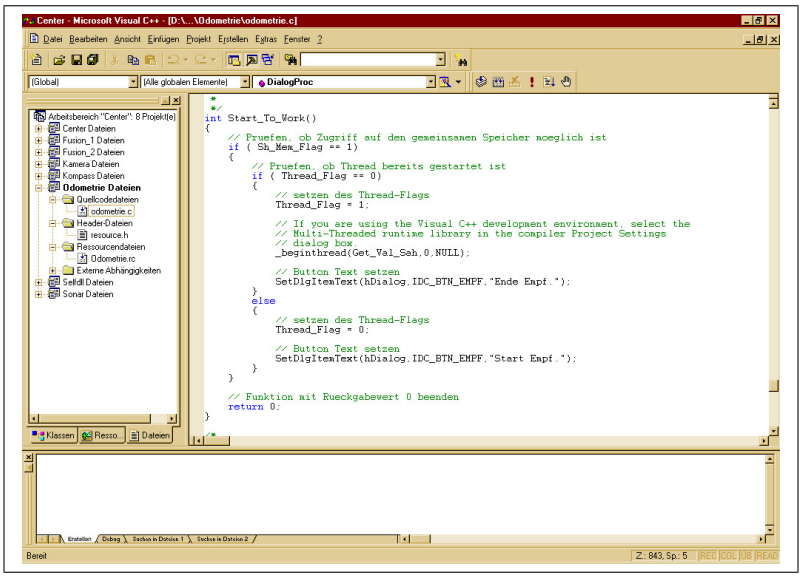

Abbildung 6: Die Entwicklungsumgebung MS Visual C++.

Um Erweiterungen für Saphira bzw. externe Anwendungen, die mit Saphira kommunizieren, zu erstellen, liegen der Saphira Software Librarys und Headerdateien bei. Die wichtigsten sind hier die Library `sf.lib` und die Headerdatei `sahira.h`. Diese müssen in jedes Projekt aufgenommen werden. Auf diese Weise sind auch die in dieser Arbeit besprochenen Applikationen entwickelt worden.

4 Aufbau der Arbeit

Die vorliegende Diplomarbeit gliedert sich in drei Hauptteile:

- $\frac{1}{3}$ Theorie
- $\frac{1}{3}$ Konzeption
- $\frac{1}{3}$ Implementierung

Im Theorieteil wird auf die Navigation mobiler Roboter, auf die Problematik der Sensor-Data-Fusion und ausführlich auf das Konzept des Kalman-Filters eingegangen.
Die Konzeption beschreibt unter Verwendung der zuvor erläuterten Konzepte, ein *,,Selbstlokalisierungsmodul"*, für einen autonomen mobilen Roboter.
Im letzen Teil, der Implementierung, wird niedergelegt, wie die praktische Umsetzung des in der Konzeption entwickelten *,,Selbstlokalisierungsmoduls* erfolgte.

Teil II

Theoretische Grundlagen

In diesem Kapitel der Diplomarbeit werden die theoretischen Grundlagen für die Umsetzung der genannten Aufgabenstellung gelegt.

Dieses Kapitel setzt sich aus den folgenden Teilen zusammen:

- Navigation mobiler Roboter
- Sensor-Data-Fusion
- Der Kalman-Filter

5 Navigation mobiler Roboter

Im Folgenden wird auf die *Navigation mobiler Roboter* eingegangen. Dies umfaßt sowohl die Betrachtung der Grundlagen und der theoretischen Aspekte, sowie die Untersuchung von *Selbstlokalisierungsverfahren*.

5.1 Grundlagen der Navigation mobiler Roboter

Von mobilen autonomen Robotern wird verlangt, daß sie sich selbstständig in unserer realen Welt zurechtfinden. Sie sollen zum Beispiel autonom Gebäude reinigen, Dienstbotengänge übernehmen oder als Blindenhunde ihren Dienst tun. Es spielen hierbei also die Autonomie und die Mobilität eine wichtige Rolle. Damit ein Roboter aber diese Aufgaben erfüllen kann, muß er in der Lage sein zu navigieren.
Der Begriff Navigation kommt aus der Schifffahrtskunde und bedeutet soviel, wie die allgemeine Bestimmung des Standortes und des Kurses [LIB94]. Ein Roboter soll also fähig sein, seine aktuelle Position zu bestimmen und ein bestimmtes Ziel anzusteuern. Unter der Position wird in dieser Arbeit der Ort und die Orientierung verstanden.
Nach [Hop92] wird generell unter Navigation die Problemstellung verstanden, ein bewegliches Objekt, ausgehend von einer momentanen Position, auf Basis unvollständiger Informationen, unter Berücksichtigung vorgegebener Randbedingungen zu einem vorgegebenen Ziel zu bringen.
Roboter sollen also effizient und kollisionsfrei in ihrer Umgebung navigieren und Strategien entwickeln. Sie sollen sich in verändernden Umgebungen autonom bewegen und Aufgaben lösen, dies wird auch intelligente autonome Navigation genannt. Die folgenden drei Grundfragen der Navigation, [LW91], muß ein autonomer mobiler Robter bei der Erfüllung seiner Aufgabe beantworten können. Sie sind in der folgenden Abbildung dargestellt.

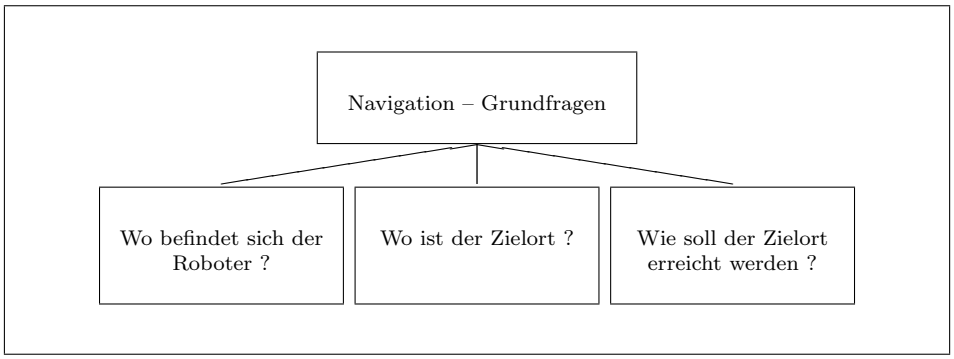

Abbildung 7: Die drei Grundfragen der Navigation [LW91].

Die Navigationskomponente eines autonomen mobilen Roboters nimmt eine entscheidende Rolle in der Kontrollstruktur ein. Sie soll gewährleisten, daß der Roboter kollisionsfrei, mit minimaler Fahrtzeit und Gesamtlänge, mit minimaler Gesamtrotation, mit minimaler Rechenzeit und maximaler Sicherheit sein Ziel erreicht [Mü98]. Oft kommt die Forderung der Echtzeitfähigkeit hinzu.
Navigationsaufgaben werden meist in zwei Phasen gelöst. In der ersten Phase wird ein Plan entwickelt, wie von einer Startposition zu einem Zielpunkt gelangt werden soll. Dies wird als Wegplanung bezeichnet und von der Planungskomponente ausgeführt. Hierbei wird eine abstrakte Aufgabe in eine Navigationsaufgabe umgesetzt. In der zweiten Phase wird der entwickelte Plan in die Tat umgesetzt. Dies wird als Fahrt bezeichnet und von der Pilotkomponente durchgeführt. Die folgende Abbildung verdeutlicht dies noch einmal.

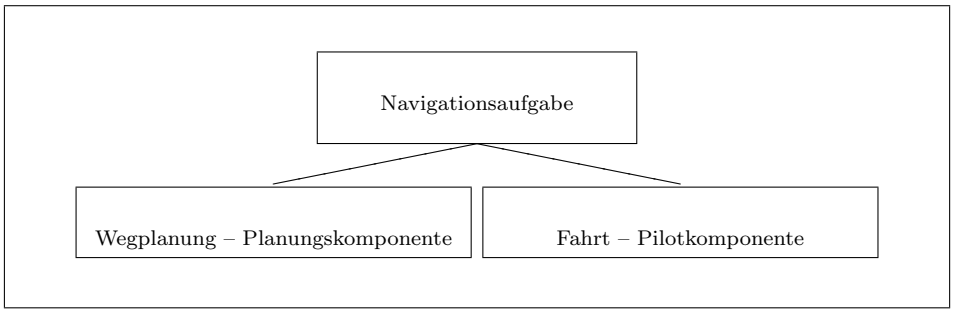

Abbildung 8: Die Hauptbestandteile einer Navigationsaufgabe.

5.2 Lokalisation und Planung

Lokalisation ist ein Vorgang, bei dem die Position eines Roboters in einer Umgebung aufgrund von Sensor-Daten ermittelt wird. Folgende Abbildung zeigt dies graphisch.

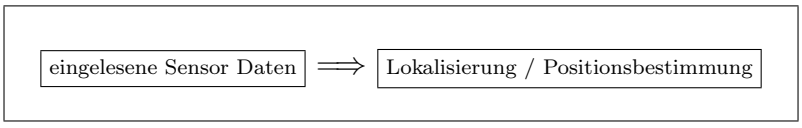

Abbildung 9: Der Vorgang der Lokalisierung.

Das Problem der Lokalisierung ist deshalb schwierig, weil Roboter in einer nicht vollständig bekannten Umgebung mit unsicheren Sensor-Daten agieren sollen, [GBFK].
Wie bereits im vorherigen Abschnitt erwähnt, spielt die Planung bei der Lösung einer Navigationsaufgabe eine wichtige Rolle. Die Planungskomponente ist die oberste in einer Top-down Architektur, gefolgt von der Navigationskomponente und der Pilotkomponente. Einige Planungsverfahren sollen nun kurz dargestellt werden.
Planung bedeutet hier allgemein, daß nach Möglichkeit ein optimaler Pfad von der aktuellen Position des Roboters zu einer Zielposition gefunden werden soll. Die Planung soll also die Grundfrage beantworten: ,,Wie soll der Zielort erreicht werden?". Man kann zwischen globaler und lokaler Pfadplanung unterscheiden. Bei der globalen Planung wird der zu fahrende Pfad komplett bestimmt, bevor der Roboter fährt.
Die bekanntesten Planungsmethoden sind die ,,Gummibandmethode", Potentialfelder und der A*-Algorithmus.
Die ,,Gummibandmethode" geht auf Hans Moravec zurück, [BHJ+00]. Hierbei wird mittels Triangulation die Position von Objekten bestimmt, zum Beispiel mit einer Kamera. Um diese Objekte wird eine Kugel konstruiert, deren Radius ein Maß für die Ungewißheit der bestimmten Position ist. Anschließend erfolgt eine Projektion auf eine zweidimensionale Karte. Um den Weg zwischen dem Start- und dem Endpunkt zu bestimmen, werden ,,Gummibänder" als Tangenten der Kreise gespannt. Das kürzeste Gummiband stellt den besten Weg dar.
Bei der Potentialmethode wird ebenfalls eine zweidimensionale Karte erzeugt. In dieser Karte erhalten Ziele ein anziehendes und Hindernisse ein abstoßendes Potential. Anschließend erfolgt eine Verrechnung zu einem Vektorfeld und die am stärksten resultierende Kraft gibt die Fahrtrichtung des Roboters an, [Rom96].
Der A*-Algorithmus ist ein Graphenalgorithmus, der als Eingabe einen Graphen, eine Kostenfunktion, einen Startknoten und eine Heuristik der tatsächlichen Kosten eines jeden Knoten erhält. Anhand der Kostenfunktion und der Heuristik wird nun der kürzeste Weg vom Start zum Ziel bestimmt, [Hei99].

5.3 Selbstlokalisation

5.3.1 Allgemeines

Selbstlokalisierung eines autonomen mobilen Roboters bezieht sich auf die Frage ,,Wo bin ich ?". Ohne die Beantwortung dieser Frage kann die zuvor angeführte Wegplanung und die sich anschließende Fahrt nicht durchgeführt werden. Die Komponente eines Roboters, die die Positionsbestimmung übernimmt, wird als *Selbstlokalisierungsmodul* bezeichnet. Es hat die Aufgabe, eine Positionsbestimmung bzw. eine Positionsschätzung (position estimation) durchzuführen.

5.3.2 Absolute und Relative Selbstlokalisation

Grundsätzlich ist zwischen absoluter und relativer Selbstlokalisation zu unterscheiden, siehe [Gut96]. Mit absoluter Selbstlokalisation ist gemeint, daß ein Roboter in der Lage ist, mit seinen Sensoren festzustellen, an welcher Position er sich befindet und zwar ohne Vorwissen. Wenn dagegen die ungefähre Position des Roboters bekannt ist und eine Positionskorrektur errechnet werden soll, so spricht man von relativer Selbstlokalisation.
Wie gut die Selbstlokalisation ist, die ermittelte eigene Position also, hängt auch von Anzahl und Art der Sensoren ab, mit denen der Roboter ausgestattet ist. Typische Sensoren, die bei den meisten mobilen Robotern verwendet werden, sind zum Beispiel Odometrie-Encoder, Ultraschall bzw. Sonar-Senoren und Videokameras.
Diese Sensoren können unterteilt werden in ihre Eignung zur relativen bzw. absoluten Positionsbestimmung. So zum Beispiel dienen Odometrie-Encoder und Gyroskope zur relativen Positionsermittlung und das GPS-System zur absoluten Positionsbestimmung, [Rom96].

5.3.3 Sensorik

Nach [Bor91] können folgende Kriterien zum Vergleich von Selbstlokalisierungsansätzen herangezogen werden.

- Die Genauigkeit der ermittelten Position
- Das Messverfahren

- Die Kosten
- Auflösung bzw. sampling rate
- notwendige Rechenzeit

5.3.4 Odometrie

Eine der meist genutzten Techniken zur Positionsbestimmung eines mobilen Roboters ist die sogenannte Odometrie. Hierbei befinden sich Odometrie-Encoder an einem oder mehreren Rädern des Roboters, die die Drehbewegung der Räder messen können. Man unterscheidet allgemein inkrementelle und absolute Odometer. Diese Art der Positionsbestimmung wird auch als *Dead Reckoning* oder als Koppelnavigation bezeichnet. Hierbei wird aus der alten bekannten Position des Roboters, durch Verarbeitung der Encoderticks in der Odometrieberechnung, die neue Position ermittelt, [BHJ+00].
Der Vorteil beim Einsatz der Odometrie ist, daß die Odometrie-Sensoren vollkommen unabhängig sind und eine Positionsschätzung für einen Roboter zulassen. Ihr großer Nachteil ist jedoch, daß Fehler ohne eine obere Schranke anwachsen können, [Bor91]. Generell wird zwischen systematischen Fehlern, zum Beispiel der Auflösung, und unsystematischen Fehlern, wie Oberflächenbeschaffenheit, unterschieden, [BHJ+00]. Um diese Fehler zu reduzieren, werden Verfahren wie zum Beispiel Data-Sensor-Fusion angewandt, [Hei99].

5.3.5 Ultraschall

Ultraschallsensoren, auch als Sonar-Sensoren bezeichnet, beruhen auf der Messung von Signallaufzeiten. Sie liefern ein sogenanntes zweidimensionales Abstandsprofil oder auch einen Scan als Output. Sie dienen zur Entfernungsbestimmung, [Gut96]. Um aus den Daten der Ultraschallsensoren die Position des Roboters zu bestimmen, muß eine Scanüberdeckung durchgeführt werden, daß heißt, die Werte des Scans müssen mit dem Modell der Umgebung in Deckung gebracht werden. Dabei werden Objekt-Modelle mit Bild-Modellen verglichen.
Vorteile von Sonarsensoren sind ihre einfache Verwendung und die Kostengünstigkeit. Ein Nachteil der Ultraschallsensoren ist, daß die gemessene Entfernung ziemlich

unsicher ist. Die Glaubwürdigkeit der Sonarinformation sinkt exponentiell mit der gemessenen Entfernung, [MZ]. Des Weiteren ist die Richtung aufgrund des Öffnungswinkels nicht genau bestimmbar.

5.3.6 GPS

GPS steht für Global Positioning System, einem Satellitennavigationsverfahren. Hierbei erfolgt die Positionsbestimmung durch Trilateration, also durch Entfernungsmessung zu den Satelliten. Dies erfolgt durch Time of flight-Sensoren (ToF) und es sind mindestens drei Satelliten notwendig. Die Messung ist von der Sichtbarkeit der Satelliten abhängig.
Eine Verbesserung der ermittelten Position des GPS ist mit Hilfe des sogenannten Differential GPS möglich, [BHJ+00].

5.3.7 Transponder

Transponder können für die Selbstlokalisation eines mobilen Roboters als aktive Landmarken dienen. Bei Transpondern, Transmitter Respondern, also Sende- und Empfangseinrichtungen, handelt es sich um hermetisch gekapselte Datenträger. Sie bestehen aus einem Mikrocontroller und einer Antennenspule. Diese Marken sind absolut verschleißfrei und besitzen eine nahezu unendliche Lebensdauer. Sie werden auch als RFID, als Radio frequency identification System bezeichnet. Näheres zum Thema Transponder ist in [Fin00] nachzulesen.

5.3.8 Bekannte Ansätze und Einteilung

Nach [Gut96] kann man die Ansätze zur Selbstlokalisation unterteilen in scanbasierte, Verwendung von geometrische Merkmalskarten und Nutzung von künstlichen Landmarken. In der folgenden Abbildung wird dies noch einmal veranschaulicht.

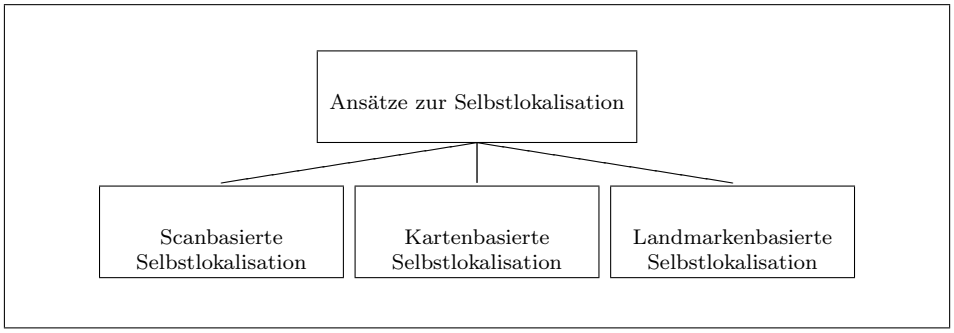

Abbildung 10: Die drei wichtigsten Ansätze zur Selbstlokalisation.

Bei der scanbasierten Selbstlokalisation, wie zum Beispiel von einem Laserscanner erzeugt, ist das Ziel, zwei aufgenommene Scans der Umgebung in Übereinstimmung zu bringen. Anhand dessen, wie die Scans in Übereinstimmung gebracht werden, (also durch Rotation und Translation) ist es möglich, eine Korrektur zu bestimmen und die Position des Roboter zu korrigieren.
Die Nutzung von Karten heißt, es wird ein Modell der Umwelt erzeugt und dem Roboter als Wissen zu Verfügung gestellt, das a-priori-Modell. Für eine Positionsbestimmung müssen die aufgenommenen Sensorwerte, wie zum Beispiel das Sonarbild, in Korrespondenz mit der internen Karte gebracht werden. Oft wird hier das Verfahren der occupancy grids, der Belegungsgitter angewandt.
Um dem Roboter eine bessere Positionsbestimmung zu ermöglichen, und wenn die Manipulation der realen Umgebung gestattet ist, bietet sich die Verwendung von künstlichen Landmarken an. Landmarken sind eindeutige wiedererkennbare Orientierungspunkte, wie zum Beispiel Führungsstreifen, Induktionsschleifen oder auch Transponder. Hierbei spielt die Wahrnehmbarkeit und der Informationsgehalt eine wichtige Rolle. Landmarken können in lokale und globale Orientierungsmarken unterschieden werden. Ein mobiler Roboter muß nun diese Marken erkennen und anschließend seine Position bestimmen. Dieses Verfahren eignet sich gut zur Verbesserung einer relativen Positionsbestimmung.
Zur Positionsbestimmung anhand von Landmarken werden allgemein zwei Verfahren unterschieden. Dies sind die Trilateration und die Triangulation. Bei der Trilateration wird die Position durch Entfernungsmessung zu bekannten Marken, Beacons,

bestimmt. Hierzu sind drei oder mehr Marken durch den Roboter zu identifizieren. Die Triangulation bedient sich der Winkelmessung zu bekannten Marken.
Des Weiteren existieren verhaltensbasierte Ansätze und dichte Sensordaten vergleichende Ansätze. Verhaltensbasierte Ansätze legen eine Interaktion der autonomen mobilen Roboter mit der Umwelt zugrunde. Beim dichte Sensordaten vergleichenden Ansatz wird eine Position unter Nutzung aller verfügbaren Sensordaten bestimmt, so zum Beispiel Überdeckung von Meßpunkten mit der Karte.

5.3.9 Markov Lokalisation

Die Markov Lokalisation ist ein ,,Werkzeug" zur Positionsbestimmung eines autonomen mobilen Roboters. Sie ist benannt nach Markov, einem russischen Mathematiker, der auf den Gebieten der Zahlentheorie und der Wahrscheinlichkeitsrechnung tätig war.
Die Markov Lokalisation besitzt folgende allgemeine Eigenschaften. Sie nutzt Wahrscheinlichkeitsverteilungen auf einem Gitter zur Darstellung der Roboterposition. Es erfolgt eine Berechnung über alle möglichen Positionen und die Wahrscheinlichkeitsneuberechnung erfolgt ohne Hintergrundwissen über den aktuellen Zustand bzw. wie dieser erreicht wurde. Bei der Markov Lokalisation wird ein updateing der Wahrscheinlichkeitsverteilungen über alle im Raum möglichen Roboterpositionen, basierend auf den Roboterbewegungen und einer gegebenen Karte durchgeführt. Sie beruht auf zwei Schritten, dem prediction step und dem update step. Eine grundlegende Darstellung ist in [TGB+] nachzulesen.
Als Vorteile der Markov Lokalisation lassen sich die Robustheit und die Tatsache anführen, daß verschiedene Positionen, an denen sich der Roboter befinden könnte, aufgezeigt werden. Als Nachteil sind die aufwendigen Berechnungen, der große Speicherbedarf und die sehr großen Datenstrukturen für die Karten zu nennen.
Eine Verbesserung der reinen Markov Lokalisation besteht in der Verwendung der sogenannten Correlation-Based Markov Lokalisation, siehe [CBM].

5.3.10 Ablauf zur Selbstlokalisierung

Die Selbstlokalisation, also der Prozess der Bestimmung der eigenen Position des Roboters, kann in mehrere Abschnitte unterteilt werden. Im ersten Schritt werden

die Daten der Sensoren ausgelesen. Diese werden nun im Vorverarbeitungsschritt aufbereitet und ausgewertet. Hier können zum Beispiel Filter angewendet werden. Im letzten Schritt werden diese Informationen mit dem Weltwissen des Roboters, zum Beispiel einer Karte, verglichen. Ergebnis ist die aktuelle Position des Roboters.

5.4 Navigation in Gebäuden

Gebäude können als ein bestimmter Umgebungstyp angesehen werden. Nach [Rom96] ist eine Unterteilung der Umgebungstypen wie folgt möglich. Es kann unterschieden werden zwischen innerhalb und außerhalb von Gebäuden, zwischen ebenen und unebenen Terrain, zwischen strukturierten oder unstrukturierten Umgebungen. Des Weiteren spielt es eine wichtige Rolle, ob es sich um eine bekannte oder unbekannte Umgebung handelt.
Die Navigation in Gebäuden, auch als Indoor-Navigation bezeichnet, unterscheidet sich stark von der Navigation im Freien, der Outdoor-Navigation. Anwendungen mobiler Roboter im Indoor-Bereich beziehen sich meist auf Bürolandschaften. Diese sind meist dadurch gekennzeichnet, daß sie einfach strukturiert sind und rechtwinklige Strukturen aufweisen. Typische Strukturen sind zum Beispiel Abzweigungen, Türdurchfahrten, Korridore und ein Aufzug. Auch ist der Untergrund und die zu überwindenden Hindernisse für mobile Roboter besser geeignet als zum Beispiel Wege im Freien. Die Auswahl der Sensoren ist ebenfalls unterschiedlich, so eignet sich zum Beispiel das GPS-System nicht im Indoor-Bereich.

5.5 Probleme der Navigation mobiler Roboter

Zum gegenwärtigen Zeitpunkt gibt es nicht *das* Navigationsverfahren. Die verschiedenen Vor- und Nachteile müssen für den jeweiligen Einsatzzweck eines autonomen mobilen Roboters ausgewogen werden. Wie zum Beispiel der Aufwand beim Einrichten der Umgebung oder die notwendige Rechenleistung für ein exakteres Verfahren. Während der Fahrt eines Roboters durch unsere reale Welt wird dieser von seinem intern ermittelten Kurs abgelenkt. Dies kann zum Beispiel durch Änderung der Oberfläche, auf der der Roboter fährt, durch ungenaue und ungleichmäßige Ansteuerung der Antriebsräder oder durch falsche Sensorwerte erfolgen. Diese Ablenkung kann dazu führen, daß ein Roboter schon nach relativ kurzer Zeit Schwierigkeiten hat,

seine Position einzuschätzen und sich sozusagen verläuft, wie in [Stö97] beschrieben. Es existiert also eine Differenz zwischen der gedachten Position des Roboters in seiner internen Darstellung und der tatsächlichen Position in der realen Umgebung. Nach [Kon98] können die Fehler bei der Fahrt eines mobilen Roboters in drei Hauptgruppen klassifiziert werden. Dies sind:

- *Range error* bzw. Entfernungsfehler
- *Turn error* bzw. Drehfehler
- *Drift error* bzw. Driftfehler

In der folgenden Abbildung ist graphisch die Auswirkung dieser Fehler veranschaulicht.

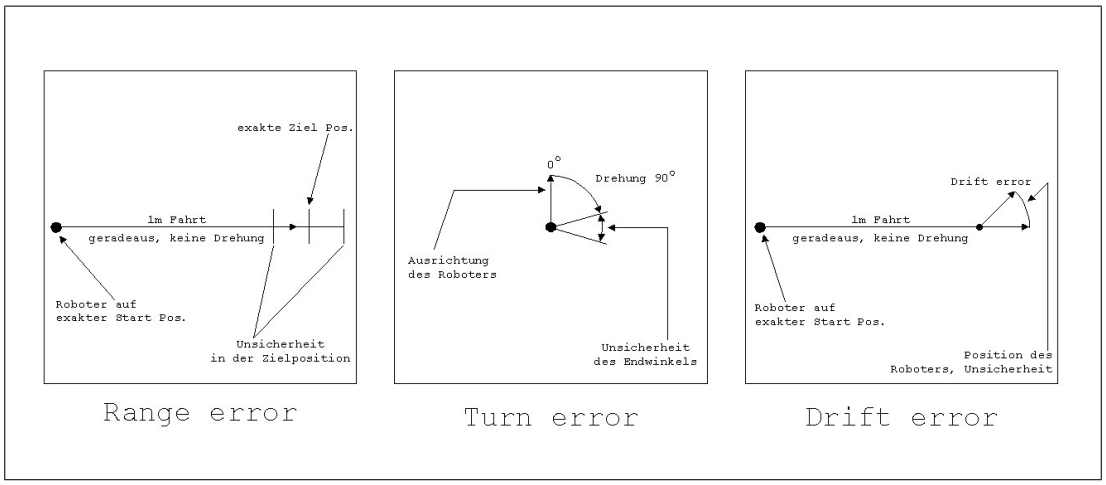

Abbildung 11: Die Fehlerarten und ihre Auswirkung.

Der Range error ist der Fehler, der beim Zurücklegen einer geraden Strecke auftritt. Der Drehfehler tritt bei der Drehung des Roboters auf und der Driftfehler ist der Orientierungsfehler beim Zurücklegen einer geraden Strecke.
Hierbei ergibt sich das Problem einer wachsenden Positionsunsicherheit durch zwei Faktoren. Zum Ersten durch die Addition neuer Positionsfehler und zweitens durch die Vergrößerung bestehender Fehler beim Bewegen.

Des Weiteren haben die jeweiligen Sensoren, wie zum Beispiel die Odometrie-Encoder und die Sonar-Sensoren, ihre eigenen Fehlercharakteristiken. Bei Odometrie-Sensoren kann hier zum Beispiel die Encoderauflösung und die Encoder Sampling Rate genannt werden. Sonar-Sensoren können, wenn sie als Array angeordnet sind, Reflektionen anderer Sonars auffangen und dadurch Fehlinformationen liefern.
Festzustellen ist, daß Sensordaten unzuverlässig sind, [Rom96]. Bei der Umsetzung der Selbstlokalisation muß die Unsicherheit der jeweiligen Sensoren berücksichtigt werden. Nach dem heutigen Stand der Technik ist eine Fehlereliminierung nicht möglich, wie am Beispiel der Sonar-Sensoren gezeigt. Jedoch existieren viele Verfahren, die eine Reduzierung der Fehler ermöglichen. Hier sind der später noch besprochene Kalman-Fiter und das Verfahren der Sensor-Data-Fusion, zu nennen.
Ein weiteres Problem, das im Zusammenhang mit der Navigation zu sehen ist, ist das sogenannte *Wake-Up-Problem*, das Aufwachproblem. Hierbei muß ein autonomer mobiler Roboter in der Lage sein, nachdem er in Betrieb genommen wurde und sich an einer beliebigen Position befindet, diese festzustellen. Er muß also eine Selbstlokalisation durchführen und die Frage ,,Wo bin ich ?" beantworten. Dies ist für die meisten heute verwendeten Systeme nicht gelöst, sie müssen von einem definierten Punkt starten.

6 Sensor-Data-Fusion

Im Folgenden wird auf die Problematik der *Sensor-Data-Fusion* näher eingegangen. Es werden die Grundlagen und Aufgaben, sowie konkrete Fusionsmethoden dargestellt.

6.1 Grundlagen und Aufgaben der Sensor-Data-Fusion

6.1.1 Einführung

Zur Einführung sollen hier Beispiele aus der Biologie dienen, [Var98]. Grillen besitzen die Fähigkeit zur akustischen Ortung und Orientierung. Dies wird auch Phonotaxis genannt. Das Richtungshören beruht hierbei auf der Differenz der Erregung beider Ohren. Die Sinneszellen, die in diesem Fall Omegazellen genannt werden, könnten direkt Ausgaben zur Orientierung bzw. Ortung liefern. Dies ist aber nicht der Fall, sondern das Gehirn wacht darüber, ob die Signale der Omegazellen sich mit Meldungen von anderen Sinnesorganen, wie zum Beispiel Augen oder Fühlern, womöglich widersprechen.
Auch Schmetterlinge bedienen sich zur Orientierung einer Kombination ihrer aus der Umwelt aufgenommenen Informationen. So nutzen sie die Kombination von Wind, Duft und visueller Orientierung, um zu navigieren.
Ebenfalls bedienen sich Eulen beim Beutefangen neben dem Gehör auch anderer Sinne. Näheres zu diesem Thema ist in [Var98] nachzulesen.
Die Sinnesorgane und das Gehirn der Tiere können hier mit den Sensoren bzw. mit dem Steuerungsrechner eines autonomen mobilen Roboters verglichen werden. Die Kombination und Verarbeitung der Informationen von verschiedenen Sensorquellen ist also Gegenstand der Sensor-Data-Fusion.
Ziel der Sensor-Data-Fusion ist es also, Informationen so zu verarbeiten, daß eine korrekte Wahrnehmung der Umwelt erfolgt. Das heißt, es sollen falsche bzw. fehlerhafte Informationen, sowie widersprüchliche Wahrnehmungen aussortiert werden. Richtige Wahrnehmungen von verschiedenen Quellen müssen dagegen verstärkt werden.

6.1.2 Grundlagen

Wie im vorhergehenden Abschnitt dargestellt, sind Sensoren Informationsquellen über den aktuellen Zustand der Umwelt.
Nach [Wü] kann ein Sensor in der allgemeinsten Weise, wie in der folgenden Abbildung dargestellt, betrachtet werden.

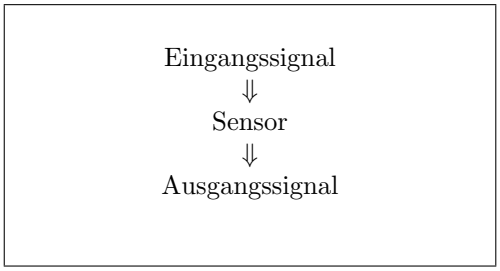

Abbildung 12: Die allgemeinste Betrachtung der Arbeitsweise eines Sensors.

Technisch können Sensoren zum Beispiel nach Eingangsgrößen, Wechselwirkung und Technologie klassifiziert werden.
Des Weiteren sind Sensoren zu unterscheiden in Eignung für bestimmte Aufgaben, Art und Größe der systembedingten Meßfehler und Art und Auswirkung äußerer Störeinflüsse.
Zum Eingangssignal und Ausgangssignal kommt noch ein Steuereingang für Korrekturgrößen und eine automatische Kalibrierung, und ein Störsignal, zum Beispiel gelegentlich auftretende Störungen und Drifterscheinungen, hinzu.
Die in einem mobilen Roboter eingesetzten Sensoren sind nicht perfekt, sie liefern unterschiedliche, unpräzise oder direkt unsichere Angaben.
Die Informationen, die ein einzelner Sensor liefert, sind meist unvollständig und unpräzise. Daher werden Methoden benötigt, um die Informationen verschiedener Sensoren zu kombinieren.
Methoden also, zur Handhabung von Unsicherheit und wissensbasierte Verfahren für Schlußfolgerungen und zur Entscheidungsfindung.
Nach [Dod98] läßt sich folgende Definition aufstellen. Der Prozeß der Kombination von Daten, in einer Art und Weise, daß das Ergebnis mehr Informationen bereitstellt als die Summe der einzelnen Teile, wird Data-Fusion genannt.
Definition der Daten-Fusion nach [Sie97]. Die Multisensor-Fusion oder Daten-Fusion bezeichnet die tatsächliche Kombination oder Verschmelzung der Daten mehrerer Sensoren.
Es ist zwischen den Begriffen der Fusion und der Integration zu unterscheiden. Fusion bedeutet allgemein soviel wie Vereinigung, Verschmelzung. Im Gegensatz ist Integration die Einbeziehung, Eingliederung in ein größeres Ganzes.

Nach [Sie97] gilt folgendes als Definition für Multisensor-Integration. Die Multisensor-Integration bezeichnet die synergetische Verarbeitung der Informationen mehrerer Sensoren zur Erfüllung der Aufgabe des Gesamtsystems. Dazu zählen insbesondere grundlegende Funktionen der Systemarchitektur, die die Zusammenarbeit der einzelnen Sensoren steuern.
Die Sensor-Data-Fusion nutzt Redundanz, das heißt mehrere verschiedene oder gleichartige Sensoren geben über einen bestimmten Zustand der Welt Auskunft. Ein Beispiel ist die Vereinigung der Bilder des rechten und des linken Auges zu einem einzigen Bild.

6.1.3 Die Taxonomie der Sensor-Data-Fusion

Die Taxonomie [1] der Sensor-Data-Fusion ist in der folgenden Abbildung dargestellt, [Dod98].

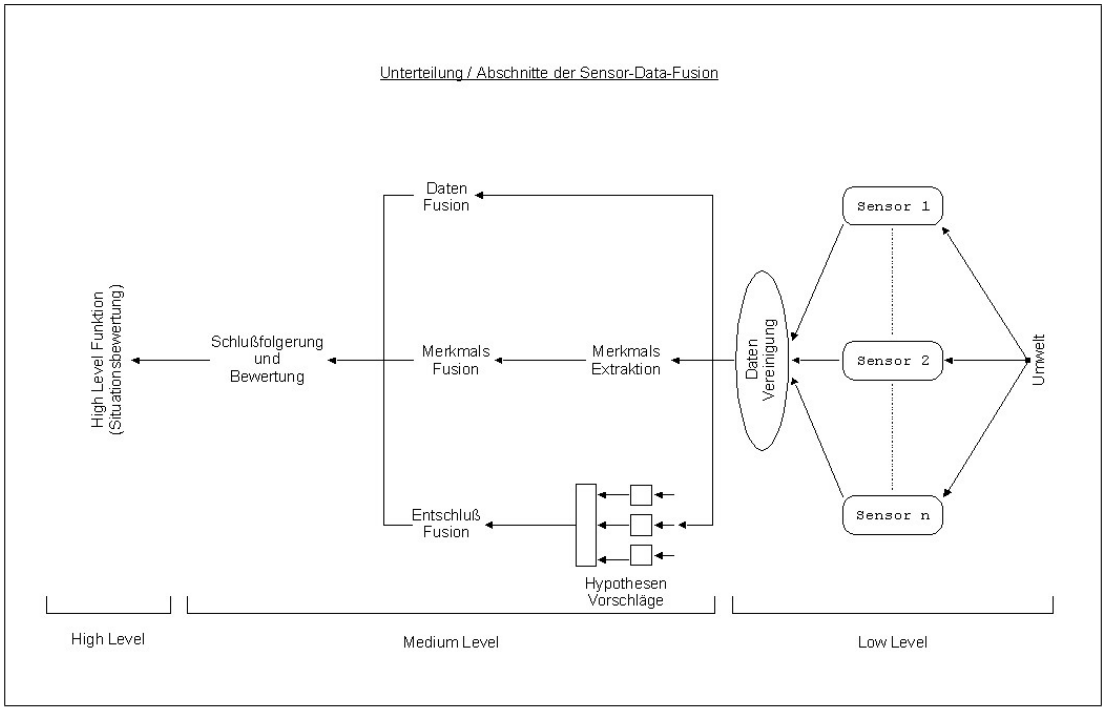

Abbildung 13: Die Taxonomie der Sensor-Data-Fusion.

Auf Low Level Ebene erfolgt eine Rohsensordatenverarbeitung und eine Anpassung bzw. Aufbereitung der Daten für nachfolgende Prozesse. Datenquellen sind hierbei zum Beispiel Sensoren wie Radar oder Sonar. Die Ebene Medium Level hat die Aufgabe einer Merkmalsextraktion und der Vorschlagformulierung. Die letzte Ebene, die High Level Ebene, übernimmt das Situationsmanagement. Hier können zum Beispiel Fragen beantwortet werden, wie ,,Ist dies eine Gefahr ?”.

[1] Einordnung, Segmentierung und Klassifikation; Teilgebiet der Linguistik, auf dem man durch Segmentierung und Klassifikation sprachlicher Einheiten den Aufbau eines Sprachsystems beschreiben will, [DUD94].

6.1.4 Fusions-Ebenen

Nach [Sie97] können folgende Ebenen der Sensor-Data-Fusion unterschieden werden. Diese Unterteilung erfolgt nach dem Grad der Abstraktion der Daten in den jeweiligen Ebenen:

- – Signal-Ebene
  – Pixel-Ebene
- Merkmals-Ebene
- Symbol-Ebene

Die niedrigste Abstraktionsebene, die Signal-Ebene, kombiniert die Signale eines Sensors. Auf Pixel-Ebene werden Messungen bildgebender Sensoren, wie zum Beispiel einer Kamera fusioniert. Hier kommen Bildverarbeitungsmethoden zum Einsatz. Die Merkmals-Ebene bietet Verfahren zur Fusion extrahierter Merkmale an. Die Symbol-Ebene, die höchste Abstraktionsebene, ist in der Lage, Sensoren, die sehr unterschiedliche Arten von Informationen liefern, zu kombinieren.

6.2 Anwendungsgebiete

Anwendungsgebiete für die Sensor-Data-Fusion sind zum Beispiel Medizinische Diagnose, Feuerlöschsysteme, Navigationssysteme für die Binnenschiffahrt [Sie97] und Robotik.
Roboter müssen in der Lage sein, Objekte, Hindernisse und Landmarken sicher zu erkennen. Und dies auf Grundlage unsicherer Informationen in einer Welt mit statischen und dynamischen Objekten.

6.3 Mathematische Fusionsmethoden

Grundsätzlich kann zwischen zwei Arten der Sensor-Data-Fusion unterschieden werden. Zum einen die Fusion mehrerer gleichartiger oder verschiedener Sensoren, zum anderen die Fusion eines einzelnen Sensors über einen bestimmten Zeitraum hinweg. Die folgende Abbildung stellt dies nochmals graphisch dar.

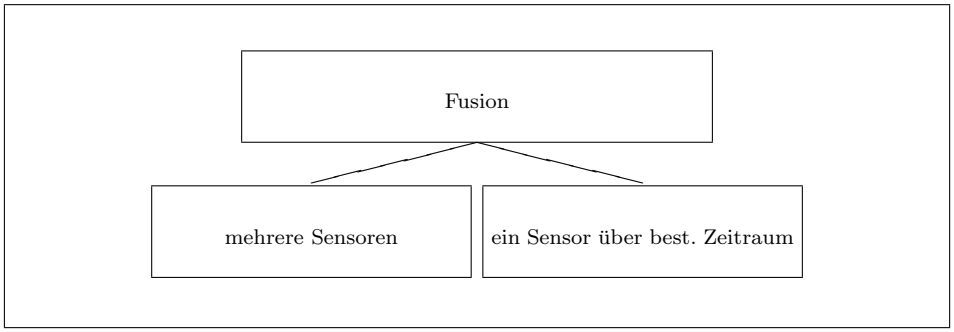

Abbildung 14: Die zwei grundsätzlichen Arten der Sensor-Data-Fusion.

Die Sensor-Data-Fusion auf der Low Level Ebene bzw. Signal-Ebene, ist eine mathematische Aufgabe. Hier kommen meist statistische Verfahren zum Einsatz.
Eine der meist genutzten Methoden zur Fusion stellt das Werkzeug des später noch ausführlich betrachteten Kalman-Filters dar. Hier sind der ,,einfache" Kalman-Filter und der erweiterte Kalman-Filter (EKF) zu unterscheiden, [BW00].

6.3.1 Gewichteter Mittelwert

Der gewichtete Mittelwert aus n Sensoren-Messungen x_i mit Gewichten $0 \leq g_i \leq 1$ ist:

$$\bar{x} = \sum_{i=1}^{n} x_i g_i \tag{1}$$

Durch die Gewichte ist es möglich, die unterschiedlichen Eigenheiten, zum Beispiel Präzision oder Fehleranfälligkeit, zu berücksichtigen. Bei der Verwendung von nur einem Sensor über die Zeit ist es möglich, aktuellere Messungen stärker zu gewichten, [Sie97].

6.3.2 Bedingte Wahrscheinlichkeiten

Die Sensor-Data-Fusion kann vom statistischen Standpunkt wie folgt betrachtet werden, [WJW97]. Gegeben sind zwei Zufallsvariablen A und B. Was sagt die Beobachtung B über A aus ? Hier kommen *Bedingte Wahrscheinlichkeiten* zum Einsatz.

$$P(B \mid A) = \frac{P(A \cap B)}{P(A)} \tag{2}$$

Die Wahrscheinlichkeit des Ereignisses B unter der Bedingung, daß A schon eingetreten ist, heißt Wahrscheinlichkeit von B unter der Bedingung A, genannt *Bedingte Wahrscheinlichkeit*, [Bar97].

6.3.3 Bayes Regel

Formel von Bayes. Die bedingten Wahrscheinlichkeiten der Ereignisse A_i unter der Voraussetzung, daß B bereits eingetreten ist, betragen, [Hei99]:

$$P(A_i \mid B) = \frac{P(A_i) \cdot P(B \mid A_i)}{\sum_{j=1}^{n} P(A_i) \cdot P(B \mid A_j)} \tag{3}$$

Hierbei kann $P(A_i)$ als Hypothese und $P(B \mid A_i)$ als Überzeugungsgrad betrachtet werden.

6.3.4 Dempster

Nach [Hei99], besteht das Wahrscheinlichkeitsmodell nach Dempster aus einer endlichen Menge von Sensoren. Diese Menge wird mit O bezeichnet. Die Zuverlässigkeit eines Sensors ist gegeben durch $\mu(o)$. Jeder Sensor liefert Ausgaben in einem Ereignisraum Ω. Nun wird eine sogenannte Beobachtungfunktion eingeführt $\Gamma : O \rightarrow 2^{\Omega}$. Bei Dempsters Modell werden die Basiswahrscheinlichkeiten aus den Sensordaten berechnet.
Beobachtungen gewichteter Sensoren können so zusammengefaßt werden, daß man Vertrauensintervalle für Ereignisse erhält.
Dempsters Kombinationsregel bietet die Möglichkeit der Kombination unterschiedlicher Informationsquellen in Form verschiedener Massenverteilungen auf der gleichen Potenzmenge. Sie enthält zwei Masseverteilungen m_1 und m_2 und berechnet aus diesen eine neue Massenverteilung $m = m_1 \oplus m_2$, [Tim98].

$$(m_1 \oplus m_2)(\emptyset) = 0 \tag{4}$$

$$(m_1 \oplus m_2)(C) = \frac{\sum\limits_{A \cap B = C} m_1(A) \cdot m_2(B)}{\sum\limits_{A \cap B \neq \oslash} m_1(A) \cdot m_2(B)} \tag{5}$$

Die Regel von Dempster stellt einen Formalismus zur Verknüpfung unabhängiger Informationsquellen dar. Es können Ausgaben von Sensorgruppen oder Bewertungen von Experten kombiniert werden.

6.4 Weitere Fusionsmethoden

Im vorhergehenden Abschnitt wurden nur Fusionsansätze für die Phase der ,,Datenvereinigung" auf Low Level Ebene aufgezeigt. Sensor-Data-Fusion kann aber viel mehr sein.
Genannt seien hier die Fusion mit Hilfe von Experten, die Nutzung bzw. Überprüfung von Hypothesen, Fusion mit Regelsystemen.

6.4.1 Fusion mit Experten

Unter einem Experten kann hier eine Instanz verstanden werden, die über Wissen eines bestimmten Sachverhalts verfügt. So könnte zum Beisiel ein Experte genau darüber Bescheid wissen, wie ein autonomer mobiler Roboter ausgerichtet ist. Andere Experten können über das gleiche oder ergänzendes Wissen verfügen.
Fusioniert man nun diese Expertenmeinungen, so soll zum Beispiel die Positionsschätzung für einen Roboter, durch die einzelnen speziellen Experten, verbessert werden.

6.4.2 Fusion mit Regeln

Um die Fusion mit Hilfe mathematischer Formeln zu erweitern, bietet sich die Verwendung von Regeln an. Regeln stellen Wissen über allgemeine Beziehungen zwischen Sachverhalten dar, [Hei99]. Sie können in der einfachsten Form als *Wenn*, *Dann* Regeln aufgefaßt werden.

Als einfaches Beispiel kann hier die Verarbeitung von Sonardaten dienen. Das Sonar hat die Eigenschaft, daß es nur auf kurze Entfernungen genau ist, bei weiten Entfernungen sind die Sonardaten sehr unglaubwürdig, siehe [MZ]. Eine einfache Regel zum Aussondern unglaubwürdiger Sonarwert könnte zum Beispiel wie folgt aussehen:

Wenn `Entfernung` größer `5m` *Dann* `Verwerfen`.

Aus vielen dieser einfachen Regeln läßt sich nun ein Regelsystem aufstellen, mit dessen Hilfe eine intelligente Fusion möglich ist.
Um nun Unsicherheit, wie ungenaue oder unzuverlässig arbeitende Sensoren, zu modellieren, bietet sich die Nutzung von Fuzzy-Logik an. Der Einsatz von Fuzzy Systemen ist dort geboten, wo kein Modell für einen Prozess existiert oder ein solches Modell nur mit sehr hohem Aufwand erstellt werden kann, [Hei99].

6.5 Probleme und Vorteile der Sensor-Data-Fusion

6.5.1 Vorteile

Sensoren besitzen Ungenauigkeiten und können gar defekt sein. Mit Hilfe von Fusionsmethoden ist es möglich, erwartete Fehler genau zu schätzen. Die kognitiven[2] Fähigkeiten eines Systems, wie zum Beispiel eines autonomen mobilen Roboters lassen sich erhöhen. Die Verknüpfung der Ausgaben mehrerer Sensoren führt zu präziseren Informationen über Aspekte und Merkmale der Umgebung. Die redundate Information von einer Gruppe von Sensoren, wie zum Beispiel dem Sonar-Ring am Pioneer 2 CE Roboter, ermöglicht die Reduzierung der Unsicherheiten in der Messung. Zudem wird die Zuverlässigkeit des Gesamtsystems erhöht, es kann zum Beispiel Sensorfehler ausgleichen.

6.5.2 Probleme

Probleme der Sensor-Data-Fusion sind zum Beispiel die Modellierung der Sensorinformation und der zugehörigen Unsicherheit in der Messung, [Sie97].

[2]kognitiv: die Erkenntnis betreffend; erkenntnismäßig; Entwicklung all der Funktionen beim Kind, die zum Wahrnehmen eines Gegenstandes oder zum Wissen über ihn beitragen, [DUD94].

Bei Sensoren, die redundante Informationen über die Umgebung liefern, liegt das Problem in dem Prozess, der die Daten in Übereinstimmung bringen muß.
Bei der Modellierung des Sensorfehlers, wird häufig als Ursache für den Fehler ein zufälliger Rauschprozeß angenommen. Dieser wird als Näherung, als Gaußsche Normalverteilung aufgefaßt. Hier bietet sich der Einsatz der schon besprochenen Fusionstechniken an.
Es müssen also Verfahren gefunden werden, die mit vertretbarem mathematischen Aufwand brauchbare Ergebnisse liefern.

7 Der Kalman-Filter

Auf den nächsten Seiten soll betrachtet und erklärt werden, was unter einem *Kalman-Filter* zu verstehen ist. Hierbei werden der historische Hintergrund, die theoretischen Grundlagen und Anwendungen erläutert. Des Weiteren wird ein Beispiel für einen *Kalman-Filter* dargelegt.

7.1 Einführung

Für den Kalman-Filter existiert ein weites Anwendungsfeld, zum Beispiel für Navigations- und Leitsysteme, Radarortung und Satellitenpositionsbestimmung. Des Weiteren seien hier folgende Anwendungen, wie Meteorologie, Hydrologie und Wasserressourcenmanagement, genannnt. Ohne Kalman-Filter gäbe es zum Beispiel keine Jumbojets und keine Fernsehsatelliten, [vR97].

7.1.1 Historischer Hintergrund

Rudolf Emil Kalman wurde am 19. Mai 1930 in Budapest geboren. Er studierte am Massachusetts Institute of Technology (MIT) und promovierte an der Columbia University 1957.
Zwischen 1960 und 1961 entwickelten Rudolph E. Kalman und Richard Bucy lineare Filtertechniken, um Rauschen aus einem Datenstrom zu filtern. Dies erfolgte durch den gezielten Übergang zu einem dynamischen Modell für ein Signal oder Grundsystem, das durch Systemeingangsrauschen und Meßrauschen und durch weiße Geräusche modelliert wurde.
Der daraus resultierende Kalman-Filter, der auf Zustandsbeschreibungstechniken und rekursiven Algorithmen beruht, revolutionierte das Gebiet der Systemtheorie, [Kal99]. Seitdem ist der Kalman-Filter ein weitreichendes Forschungsgebiet, unter anderem auf dem Feld der autonomen Navigation.

7.1.2 Was ist der Kalman-Filter ?

Der Kalman-Filter läßt sich allgemein und vereinfacht wie folgt beschreiben.
Er ist ein mathematisches Werkzeug, ein Verfahren zur optimalen Schätzung eines Zustandes, ein stochastischer Filter. Der Kalman-Filter liefert einen linearen und minimum-Varianz rekursiven Algorithmus, [Fra98].
Mit dem Kalman-Filter kann eine Störsignalunterdrückung erfolgen ohne einen Verlust an Nutzinformation, ohne das Nutzsignal signifikant zu beeinflußen, im Vergleich zu anderen Filterverfahren, wie zum Beispiel Frequenzfilter und adaptive Filterverfahren, [BGR98].
Der Kalman-Filter wird in dem allgemeineren Rahmen des Beobachterentwurfs gesehen. Für die Anwendung muß ein Zustandsmodell aufgestellt werden, [Sch92].

7.1.3 Anwendungen des Kalman-Filters in der Technik

Die NASA verwendet die Kalman-Filtertechnik für die Verfolgung von Satellitenbahnen. Spektakulär war der Einsatz des Kalman-Filters in der Mondlandefähre von Apollo 11, [Wen00]. Vier Dopplerradar-Stationen verfolgten die aktuelle Position der Fähre. An Bord bestand nun die Aufgabe, aus diesen Schätzwerten die aktuelle Position zu ermitteln.
Die Kombination von GPS mit INS (inertial navigation system) und dem Kalman-Filter ermöglicht eine wesentliche Verbesserung der Navigation. GPS hat einen minimalen Drift aber ein starkes Rauschen. INS dagegen liefert eine rauschlose Ausgabe, die langsam verdriftet. Der Kalman-Filter, der die statistischen Modelle beider Systeme berücksichtigt, kann die Vorteile beider nutzen, um den Einfluß ihrer nachteiligen Merkmale optimal zu minimieren, [Lev98].
In den Anwendungsbereich des Kalman-Filters gehören auch mit Kalman-Beobachtern arbeitende Lenksysteme. Hier erfolgt die Signalverarbeitung im Flugkörperlenksystem mit Kalman-Filtern zur Erfassung, Diskrimination und Verfolgung von Zielen unter komplexen und stark gestörten Umgebungsbedingungen, [Sch92].

7.2 Theoretische Grundlagen

Der Kalman-Filter ist wie schon gesagt ein mathematisches Werkzeug und demzufolge werden im folgenden mathematische bzw. theoretische Grundlagen des Kalman-Filters dargelegt.

7.2.1 Stochastische Prozesse

Ein stochastischer Prozess ist jeder zufällige Prozess, der sich über die Zeit entwickelt, [Dos]. Ein Prozess, der in jedem Schritt durch Wahrscheinlichkeiten gesteuert wird.
Eine Zufallsvariable $X(t)$ beinhaltet hierbei zwei Komponenten, zum einen einen Erwartungswert und zum anderen einen Zufallsanteil.
Ein stochastischer Prozess kann durch die folgende Gleichung definiert werden.

$$X(t) = E[X(t)] + fehler(t) \tag{6}$$

In der oberen Gleichung sind $E[X(t)]$ die Vorhersage der Zufallsvariable $X(t)$ und $fehler(t)$ der vorhergesagte Fehler, meist eine Wahrscheinlichkeitsverteilung, [Dos].

7.2.2 Methode der kleinsten Quadrate

Gaußsche Minimumbedingung:

$$Q := \sum_{i=1}^{n} (y_i - f(x_i))^2 \tag{7}$$

Die Methode der kleinsten Quadrate ergibt eine Näherungsfunktion, bei der die Summe der Quadrate der Abweichungen der Funktionswerte $f(x_i)$ von den statistischen Werten y_i ein Minimum ergibt, [Bar88].
Die nachfolgende Abbildung veranschaulicht dies nachmals grafisch.

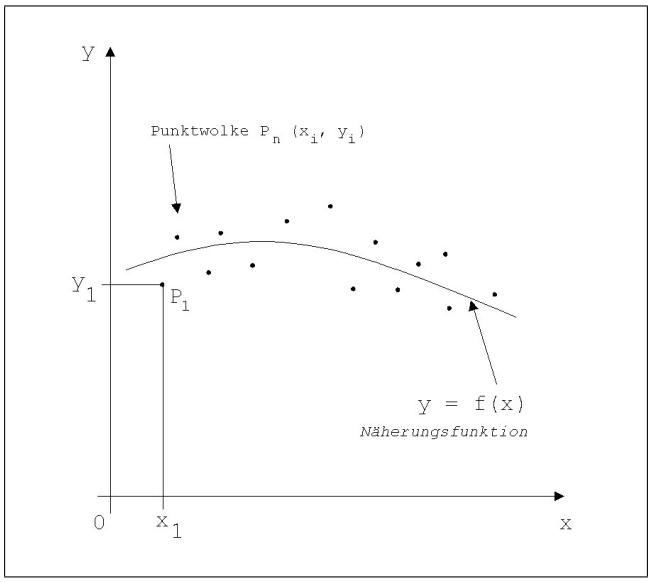

Abbildung 15: Die Methode der kleinsten Quadrate.

7.2.3 Verknüpfung zweier Zufallsvariablen

Ein Kernstück bzw. ein Grundprinzip des Kalman-Filters ist die Verknüpfung zweier Zufallsvariablen zu einem Schätzwert minimaler Varianz. Die Zufallsvariablen bzw.

die zu modellierenden Fehler, werden als Gaußsche Normalverteilung aufgefaßt.

$$f(x) = \frac{1}{\sqrt{2\pi\sigma}} \cdot e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2} \tag{8}$$

Hierbei sind μ der Erwartungswert oder auch Mittelwert und σ die Standardabweichung. Die folgende Abbildung skizziert diesen Sachverhalt nochmals anschaulich.

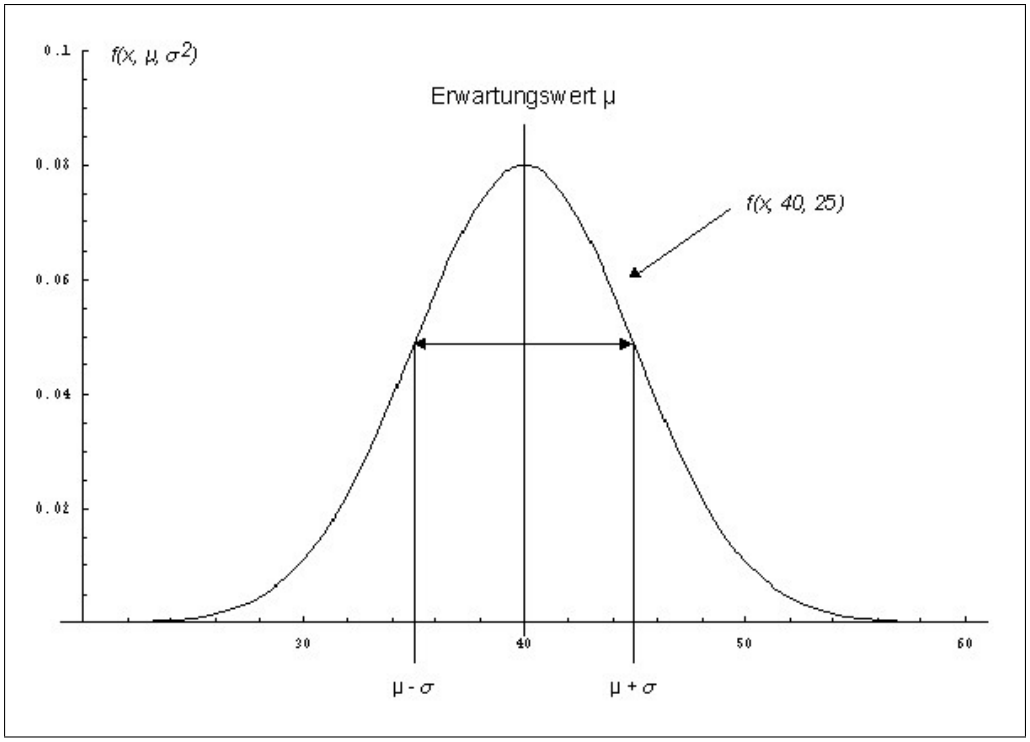

Abbildung 16: Die Gaußsche Normalverteilung (Gaußsche Glockenkurve).

Gegeben sind nun zwei Normalverteilungen mit jeweils einem Erwartungswert μ_1 und μ_2 und einer Standardabweichung σ_1 und σ_2.
Gesucht ist nun ein erwartungstreuer Schätzwert mit minimaler Varianz.

$$\mu_{opt} = (1 - K) \cdot \mu_1 + K \cdot \mu_2 \tag{9}$$

Wobei für K gilt $1 - K + K = 1$ bzw. $(1 - K) + (K) = 1$, d.h die Summe der Faktoren vor μ_1 und μ_2 muß eins ergeben.

Die optimale Gewichtung K_0 läßt sich nach [Sch92] wie folgt berechnen.

$$K_0 = \frac{\sigma_1^2}{\sigma_1^2 + \sigma_2^2} \tag{10}$$

Die angesetzte Schätzfunktion geht in die folgende Form über.

$$\begin{aligned}
\mu_{opt} &= (1 - K_0) \cdot \mu_1 + K_0 \cdot \mu_2 \\
&= (1 - \frac{\sigma_1^2}{\sigma_1^2 + \sigma_2^2}) \cdot \mu_1 + \frac{\sigma_1^2}{\sigma_1^2 + \sigma_2^2} \cdot \mu_2 \\
&= \frac{\mu_1}{1} - \frac{\sigma_1^2 \mu_1}{\sigma_1^2 + \sigma_2^2} + \frac{\sigma_1^2}{\sigma_1^2 + \sigma_2^2} \cdot \mu_2 \\
&= \frac{(\sigma_1^2 + \sigma_2^2) \cdot \mu_1 - \sigma_1^2 \mu_1}{\sigma_1^2 + \sigma_2^2} + \frac{\sigma_1^2}{\sigma_1^2 + \sigma_2^2} \cdot \mu_2 \\
&= \frac{\sigma_1^2 \mu_1 + \sigma_2^2 \mu_1 - \sigma_1^2 \mu_1}{\sigma_1^2 + \sigma_2^2} + \frac{\sigma_1^2}{\sigma_1^2 + \sigma_2^2} \cdot \mu_2
\end{aligned}$$

Die optimale Schätzfunktion bzw. der resultierende Erwartungswert berechnet sich wie folgt, [Sch92].

$$\mu_{opt} = \frac{\sigma_2^2}{\sigma_1^2 + \sigma_2^2} \mu_1 + \frac{\sigma_1^2}{\sigma_1^2 + \sigma_2^2} \mu_2 \tag{11}$$

Die aus der Verknüpfung resultierende minimale Varianz, kann mit der folgenden Formel berechnet werden, [Sch92].

$$\sigma_{min}^2 = \frac{\sigma_1^2 \sigma_2^2}{\sigma_1^2 + \sigma_2^2} \tag{12}$$

In der folgenden Abbildung ist die Verknüpfung zweier Zufallsvariablen anhand von Funktionsausdrücken nochmals veranschaulicht.

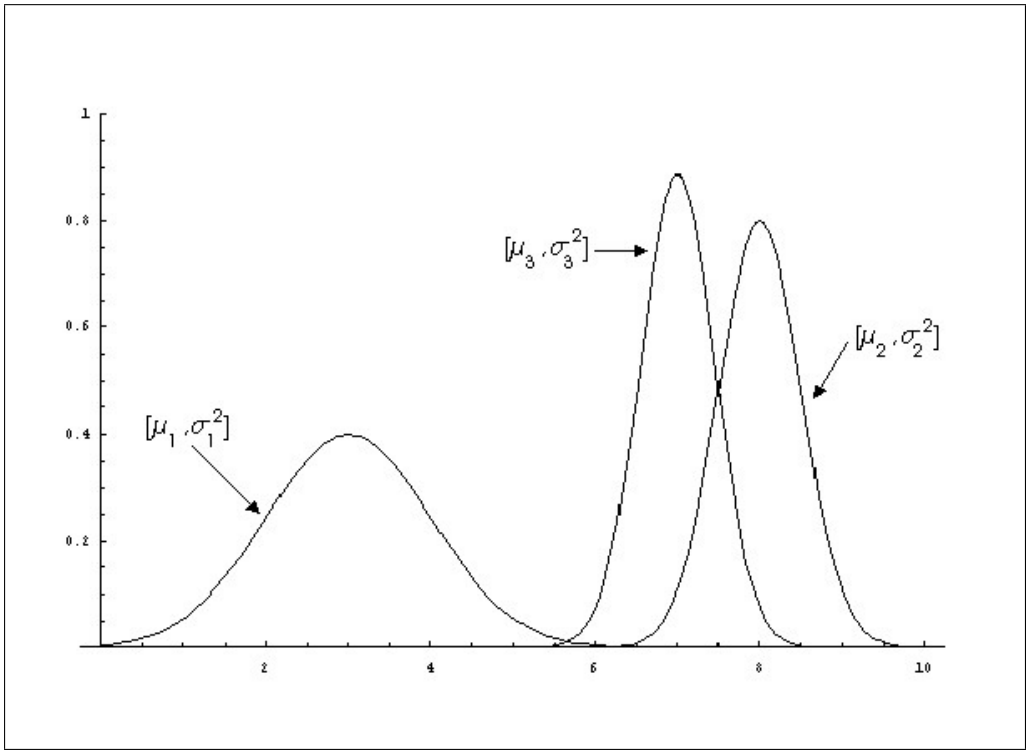

Abbildung 17: Die Verknüpfung von zwei Zufallsvariablen.

Hier wurden zwei Zufallsvariablen, die linke und die rechte Kurve, vereinigt, zu einem neuen Schätzwert mit minimaler Varianz, die mittlere Kurve. Die konkreten Werte der jeweiligen Normalverteilung betragen $\mu_1 = 3$, $\sigma_1^2 = 1$ und $\mu_2 = 8$, $\sigma_2^2 = 0,25$ sowie $\mu_3 = 7$ und $\sigma_3^2 = 0,20$.

7.2.4 Ein Beispiel

Als einfaches Beispiel soll hier die Positionsbestimmung auf See, [May79], dienen. Als Modifikation wird es auch als Schiffs-Beispiel in [GG99] vorgestellt.
Man befindet sich während der Nacht auf See und kennt seine eigene Position nicht. Zur Schätzung der Position wird der Sternenhimmel zu Hife genommen. Zum Zeitpunkt t_1 wird die Position z_1 ermittelt. Auf Grund von Meßfehlern, menschlichen Fehlern und ähnlichem hat das Ergebnis eine bestimmte Ungenauigkeit, mit der Standardabweichung $\sigma_{z_1}^2$. Nun stellt ein geübter Navigator, zur Zeit t_2, unabhängig

die Position z_2, mit einer Varianz σ_{z_2} fest.
Daraus folgt, daß die Wahrscheinlichkeiten von $x(t_1)$ und $x(t_2)$ der Positionen zu den Zeiten t_1 und t_2 dargestellt werden können.
Nutzt man nun die im vorhergehenden Abschnitt besprochenen Verfahren, so kann eine neue Positionsschätzung bestimmt werden. Hierbei ist die resultierende Varianz σ geringer als σ_{z_1} und σ_{z_2}. Die Ungenauigkeit in der Schätzung der Position wurde durch die Kombination der einzelnen Informationen vermindert.

7.3 Der diskrete Kalman-Filter

Kalman-Filterung erlaubt eine Vorhersage und Schätzung eines neuen Zustandes eines Systems, durch *einfache* Matrizenrechnungen. Dieses System wird beim diskreten Kalman-Filter mit linearen Gleichnungen beschrieben.
Gegeben sei ein System, das wie in der nächsten Abbildung dargestellt, gekennzeichnet ist.

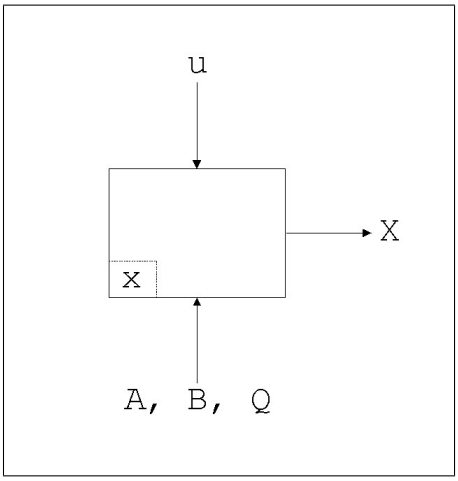

Abbildung 18: Darstellung des gegebenen Systems.

Hierbei stellen die Variable u die Eingaben bzw. Steuerung des Systems, A die Übergangsmatrix, B die Stellmatrix, Q das Prozeßrauschen und X den Systemzustand dar. Klein x veranschaulicht den nicht meßbaren internen Zustand.

Das System und dessen Zustände lassen sich mit der folgenden Gleichung beschreiben.

$$x_{k+1} = A_k x_k + B u_k \tag{13}$$

A hat die Form einer $n \times n$ Matrix und B die Form $n \times l$. Die Matrizen A und B müssen für jedes System, entsprechend der Eigenschaften dieses Systems, entwickelt werden. Sie spiegeln das System idealisiert wieder. Im wirklichen System treten jedoch bestimmte Fehler auf, die modelliert werden müssen.
Die allgemeine Form einer Gleichung für die Zustandsbeschreibung eines Systems lautet.

$$x_j = a_{1j} \cdot x_1 + a_{2j} \cdot x_2 + \ldots a_{nj} \cdot x_n + b_1 \cdot u_1 + b_2 \cdot u_2 \ldots \tag{14}$$

Als Beispiel sei hier ein System gegeben, daß durch die folgenden Matrizen A und B beschrieben wird.

$$A = \begin{pmatrix} 20 & 60 & 0 \\ 0 & 10 & 30 \\ 0 & 50 & 40 \end{pmatrix} B = \begin{pmatrix} 1 & 4 \\ 8 & 1 \\ 5 & 1 \end{pmatrix}$$

Der aktuelle Systemzustand x zum Zeitpunkt k sei wie folgt gegeben,

$$x_k = \begin{pmatrix} 100 \\ 1500 \\ 90 \end{pmatrix}$$

Die Steuerung bzw. Eingabe u in das System zum Zeitpunkt k sei die folgende,

$$u = \begin{pmatrix} 20 \\ 5 \end{pmatrix}$$

Nun läßt sich der neue Systemzustand zum Zeitpunkt $k+1$ wie folgt berechnen.

$$x_{k+1} = \begin{pmatrix} 20 & 60 & 0 \\ 0 & 10 & 30 \\ 0 & 50 & 40 \end{pmatrix} \cdot \begin{pmatrix} 100 \\ 1500 \\ 90 \end{pmatrix} + \begin{pmatrix} 1 & 4 \\ 8 & 1 \\ 5 & 1 \end{pmatrix} \cdot \begin{pmatrix} 20 \\ 5 \end{pmatrix}$$

$$= \begin{pmatrix} 92000 \\ 17700 \\ 78600 \end{pmatrix} + \begin{pmatrix} 40 \\ 165 \\ 105 \end{pmatrix}$$

Also ist der neue Systemzustand zum Zeitpunkt $k+1$ der folgende.

$$x_{k+1} = \begin{pmatrix} 92040 \\ 17865 \\ 78705 \end{pmatrix}$$

Die Modellierung der Fehlervorhersage, stellt die folgende Gleichung dar.

$$P_{k+1} = A_k P_k A_k^T + Q_k \tag{15}$$

Hier sind die Matrizen A_k, P_k und A_k^T der modellierte Systemfehler und die Matrix Q_k der Prozeßfehler.
Die Matrix P wird hier als *Kovarianz-Matrix* bezeichnet. Die Kovarianz ist eine Kenngröße für die statistische Verwandtschaft von Zufallsvariablen, [Sch92]. Liegen keine Abhänigkeiten der Komponenten vor, so nimmt die Kovarianzmatrix eine Diagonalform an. Sie hat dann, im Falle von drei Komponenten, die folgende Form:

$$covar(w_1, w_2, w_3) = \begin{pmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & \sigma_3 \end{pmatrix}$$

Anderenfalls sieht die allgemeine Form der Kovarianzmatrix, für drei Komponenten, wie folgt aus:

$$covar(w_1, w_2, w_3) = \begin{pmatrix} \sigma_1 & cov(w_1, w_2) & cov(w_1, w_3) \\ cov(w_2, w_1) & \sigma_2 & cov(w_2, w_3) \\ cov(w_3, w_1) & cov(w_3, w_2) & \sigma_3 \end{pmatrix}$$

Die Matrix Q kann als Kovarianzmatrix des Prozeßrauschens verstanden werden. Eine mögliche allgemeine Form ist:

$$Q = \begin{pmatrix} \sigma_{w_1} & 0 & 0 \\ 0 & \sigma_{w_2} & 0 \\ 0 & 0 & \sigma_{w_3} \end{pmatrix}$$

Zur Verdeutlichung soll das im folgenden dargelegte Beispiel dienen.
Es sei angenommen, daß die Matrix P zum Zeitpunkt k wie folgt gegeben ist.

$$P_k = \begin{pmatrix} 0.5 & 0 & 0 \\ 0 & 0.8 & 0 \\ 0 & 0 & 0.1 \end{pmatrix}$$

Die Matrix Q zum Zeitpunkt k sei die folgende.

$$Q_k = \begin{pmatrix} 0.4 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 0.8 \end{pmatrix}$$

Nun läßt sich die neue Fehlermatrix P zum Zeitpunkt $k+1$, unter Verwendung der bereits gegebenen Matrix A, wie folgt berechen.

$$P_{k+1} = \begin{pmatrix} 20 & 60 & 0 \\ 0 & 10 & 30 \\ 0 & 50 & 40 \end{pmatrix} \cdot \begin{pmatrix} 0.5 & 0 & 0 \\ 0 & 0.8 & 0 \\ 0 & 0 & 0.1 \end{pmatrix} \cdot \begin{pmatrix} 20 & 0 & 0 \\ 60 & 10 & 50 \\ 0 & 30 & 40 \end{pmatrix} + \begin{pmatrix} 0.4 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 0.8 \end{pmatrix}$$

$$= \begin{pmatrix} 3080 & 480 & 2400 \\ 480 & 170 & 520 \\ 2400 & 520 & 2160 \end{pmatrix} + \begin{pmatrix} 0.4 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 0.8 \end{pmatrix}$$

Also ergibt sich P zum Zeitpunkt $k+1$ wie folgt

$$P_{k+1} = \begin{pmatrix} 3080.40 & 480 & 2400 \\ 480 & 170.10 & 520 \\ 2400 & 520 & 2160.80 \end{pmatrix}.$$

Die Gleichungen für den Kalman-Filter können in zwei Gruppen eingeteilt werden, [BW00]. Zum einen ist dies die Schätzung des Systemzustandes und zum anderen die Korrektur. Dies ist in der foldenden Abbildung graphisch veranschaulicht.

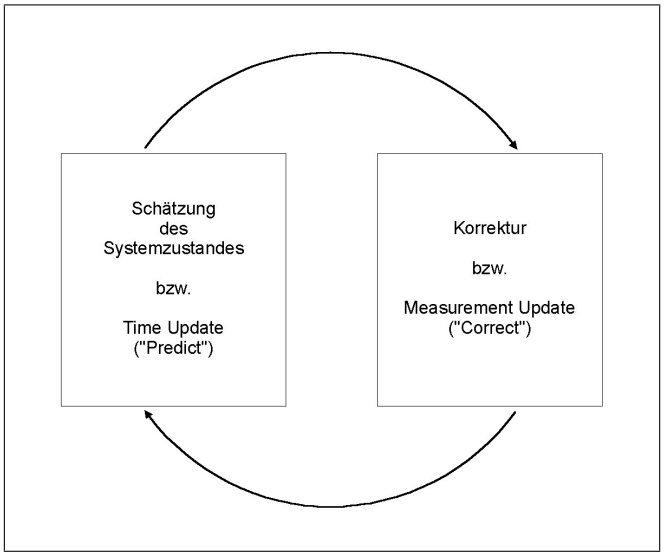

Abbildung 19: Die Unterteilung des Kalman-Filters in zwei Abschnitte, [BW00].

In dieser Abhandlung wurde nur auf die erste Gruppe, die Schätzung des Systemzustandes eingegangen.
Festzuhalten bleibt, daß die Anwendbarkeit des diskreten Kalman-Filters auf der

Bedingung beruht, daß es sich beim dem zu modellierenden System um ein *lineares* System handelt.

7.4 Der erweiterte Kalman-Filter

Der diskrete Kalman-Filter ist nur in der Lage mit linearen Systemen umzugehen. Diese Einschränkung wird beim *erweiterten Kalman-Filter* (EKF) aufgehoben. Auch hier werden die Gleichungen in zwei Gruppen eingeteilt, die Schätzung und die Vorhersage.
Beim erweiterten Kalman-Filter wird der Systemzustand nicht mehr mit Matrizen modelliert, sondern als Funktion ausgedrückt. Das Systemmodell läßt sich wie folgt beschreiben.

$$x_{k+1} = f(x_k, u_k, w_k) \tag{16}$$

Diese Funktion kann nun eine nichtlineare Funktion sein und stellt so eine Verallgemeinerung des Kalman-Filters dar. Die Anwendung des Kalman-Filter-Prinzips für nicht lineare Systeme wird durch eine Näherungsfunktion, wie zum Beispiel durch Anwendung von Taylor Reihen, möglich.

Als Beispiel sei folgendes System gegeben.

$$f(\bullet) = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} \sin x_1 \\ \log x_2 \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \end{pmatrix}$$

Bei einem gegebenen Systemzustand von $x = \begin{pmatrix} 1 \\ 10 \end{pmatrix}$ zum Zeitpnkt k, läßt sich der neue Systemzustand zum Zeitpunkt $k+1$ entsprechend der oberen Funktionen berechnen.

$$x_{k+1} = \begin{pmatrix} \sin 1 \\ \log 10 \end{pmatrix} = \begin{pmatrix} 0.84 \\ 2.30 \end{pmatrix}$$

Die Fehlervorhersage wird mit der folgenden Gleichung beschrieben, [BW00].

$$P_{k+1} = A_k P_k A_k^T + W_k Q_k W_k^T \tag{17}$$

Hierbei ist die Matrix P die Fehlerkovarianzmatrix. Die Matrizen WQW^T stellen den Prozessfehler dar. Wobei die Matrix W die Jacobian - Matrix des Prozessrauschens und Q die Kovarianzmatrix des Prozessrauschens zum Zeitpunkt k ist.

Die Matrix A ergibt sich aus den partiellen Ableitungen:

$$A = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \cdots \\ \vdots & \ddots \end{pmatrix} \tag{18}$$

Sie wird als *Jacobian-Matrix* bezeichnet.
Für das oben genannte Beispiel ergibt sich somit die folgende Matrix A.

$$\begin{pmatrix} \cos x_1 & 0 \\ 0 & \frac{1}{x_2} \end{pmatrix}$$

Als reines Rechenbeispiel, das heißt ohne Herleitung der entsprechenden Matrizen, soll die im folgenden dargelegte Rechnung dienen.

Gegeben ist also die Matrix A mit

$$A = \begin{pmatrix} \cos x_1 & 0 \\ 0 & \frac{1}{x_2} \end{pmatrix}.$$

Die Fehlerkovarianzmatrix P zum Zeitpunkt k sei wie folgt gegeben,

$$P_k = \begin{pmatrix} 0.75 & 0 \\ 0 & 0.375 \end{pmatrix}.$$

Die Modellierung des Prozessfehlers erfolgt durch die im Folgenden gegebenen Ma-

trizen W und Q.

$$W = \begin{pmatrix} 1.3 \cdot x_1 & 0 \\ 0 & 1 \end{pmatrix} \quad Q = \begin{pmatrix} 3 & 0 \\ 0 & \tan x_2 \end{pmatrix}$$

Der aktuelle Systemzustand ist mit $x = \begin{pmatrix} 1 \\ 10 \end{pmatrix}$ gegeben.

Die Berechnung der neuen Fehlerkovarianzmatrix erfolgt nun durch Einsetzen der entsprechenden Matrizen A, P und W, Q in die Gleichung

$$P_{k+1} = A_k P_k A_k^T + W_k Q_k W_k^T.$$

Somit kann man für P_{k+1} schreiben:

$$P_{k+1} = \begin{pmatrix} \cos x_1 & 0 \\ 0 & \frac{1}{x_2} \end{pmatrix} \cdot \begin{pmatrix} 0.75 & 0 \\ 0 & 0.375 \end{pmatrix} \cdot \begin{pmatrix} \cos x_1 & 0 \\ 0 & \frac{1}{x_2} \end{pmatrix} + \begin{pmatrix} 1.3 \cdot x_1 & 0 \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 3 & 0 \\ 0 & \tan x_2 \end{pmatrix} \cdot \begin{pmatrix} 1.3 \cdot x_1 & 0 \\ 0 & 1 \end{pmatrix}$$

Durch Einsetzen von x erhält man nun:

$$P_{k+1} = \begin{pmatrix} 0.54 & 0 \\ 0 & 0.1 \end{pmatrix} \cdot \begin{pmatrix} 0.75 & 0 \\ 0 & 0.375 \end{pmatrix} \cdot \begin{pmatrix} 0.54 & 0 \\ 0 & 0.1 \end{pmatrix} + \begin{pmatrix} 1.3 & 0 \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 3 & 0 \\ 0 & 0.648 \end{pmatrix} \cdot \begin{pmatrix} 1.3 & 0 \\ 0 & 1 \end{pmatrix}$$

Durch Matrizenmultiplikation erhält man die folgende Form.

$$P_{k+1} = \begin{pmatrix} 0.219 & 0 \\ 0 & 0.00375 \end{pmatrix} + \begin{pmatrix} 5.07 & 0 \\ 0 & 0.648 \end{pmatrix}$$

Durch Addition der Matrizen erhält man nun die neue Fehlerkovarianzmatrix zum

Zeitpunkt $k+1$.

$$P_{k+1} = \begin{pmatrix} 5.289 & 0 \\ 0 & 0.652 \end{pmatrix}$$

Auch in dieser Abhandlung wurde nur auf den ersten Schritt, die Schätzung des Systemzustandes, eingegangen. Genaueres ist in [BW00] nachzulesen.

7.5 Fazit

Durch Kalman-Filterung werden Meßwerte verschiedener Sensoren bezüglich der gleichen Beobachtung fusioniert. Hierbei sollen die Genauigkeit einer Messung erhöht und die Fehler minimiert werden.
Der Kalman-Filter verbindet alle ihm zur Verfügung stehenden Informationen und verarbeitet dabei alle Messungen, trotz ihrer verschiedenen Genauigkeiten, [GG99].

Als Vorteile sind zu nennen:

- Die *resultierende Positionsungenauigkeit ist geringer ist als die Ausgangsschätzung*
- Der Algorithmus ist sehr speichereffizient

Der Kalman-Filter eignet sich gut für Echtzeitanwendungen. Der Algorithmus nutzt nur die Daten des vorherigen Durchlaufs, es müssen nicht alle früher ermittelten Daten gespeichert werden. Der Kalman-Filter nimmt also nur die aktuellen Werte zur Berechnung, im Gegensatz zum Beispiel zur Markow-Lokalisation, hierbei werden alle wahrscheinlichen Positionen gespeichert und daraus die Beste ermittelt. Fährt ein Roboter mit Kalman-Filter durch die Welt, dann gehen auf diese Weise sämtliche früheren Schätzungen und Korrekturen in die neue Positionsschätzung ein.

Nachteile des Kalman-Filters sind:

- Voraussetzung für die Anwendung ist, daß die Fehler als *normalverteilte Fehler* vorliegen
- Das System muß sich mit linearen Zustandsgleichungen beschreiben lassen

Bei Sensoren ist nach dem Stand der Technik keine Fehlereleminierung möglich. Der Kalman-Filter stellt aber eine gute Möglichkeit zur Fehlerreduktion mittels Softwarekorrektur dar, [Rom96].

Teil III

Konzeption

Dieses Kapitel beinhaltet die Konzeption für ein *Selbstlokalisierungsmodul*, das in der Lage ist, die Position und die damit verbundene Positionsunsicherheit eines autonomen mobilen Roboters zu überwachen und zu beeinflußen.
Grundlegendes Ziel ist die Entwicklung eines Schätzverfahrens zur Positionsbestimmung des autonomen mobilen Roboters Pioneer 2 CE.

8 Einleitung

In den vergangenen Abschnitten, wurden theoretische Grundlagen zur Navigation autonomer mobiler Roboter, zur Data-Sensor-Fusion und speziell zur Kalman-Filtertechnik erläutert. Im folgenden Teil der Arbeit, wird nun die Konzeption für ein Selbstlokalisierungsmodul dargelegt, das auf diese Grundlagen aufbaut.
Dieses sogenannte *Selbstlokalisierungsmodul*, besteht selbst wieder aus einzelnen Modulen zur Vorverarbeitung und Verarbeitung der Sensorinformationen spezieller Sensoren, wie zum Beispiel der Odometrie-Sensoren und der Sonar-Sensoren.
Im ersten Teil wird also auf die Konzeption der einzelnen, sogenannten Module näher eingegangen. Im anschließenden Teil werden der Architekturentwurf, die Architektur und die Datenbeschreibung erläutert.

9 Konzeption der Module

Die Sensorik, die für die Konzeption der Module ausgewählt wurde, richtet sich nach den am Pioneer 2 CE Roboter tatsächlich vorhandenen Sensoren.
Es werden grundlegend zwei Arten von Modulen unterschieden. Erstens die *Sensor-Module*. Ihre Aufgabe ist die Verarbeitung der Sensor-Daten und Ermittlung eines Systemzustandes mit entsprechender Beurteilung. Konkret heißt dies, daß eine Beurteilung der aktuellen Position bzw. der möglichen Positionen erfolgt. Zweitens die sogenannten *Fusions-Module* mit der Aufgabe der globalen Sensorintegration.

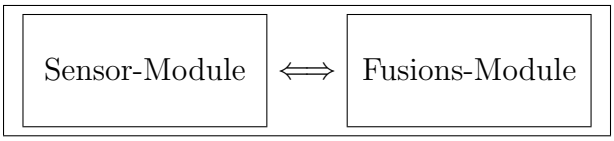

Abbildung 20: Die zwei grundlegenden Arten von Modulen.

Die Sensoren, die entsprechenden Sensor-Module und ihre Aufgaben, die nachfolgend genau beschrieben werden, sind:

- Odometrie-Modul

- Kompaß-Modul
- Sonar-Modul
- Kamera-Modul

9.1 Das Odometrie-Modul

Odometrie Sensoren sind wohl die am meisten verwendeten Sensoren für die relative Positionsbestimmung eines autonomen mobilen Roboters, Sensoren für *Dead Reckoning*. Sie haben jedoch den gravierenden Nachteil, daß die Fehler bei der Messung des zurückgelegten Weges sehr groß sind und die Positionbestimmung des Roboters sehr ungenau wird, besonders bei langen Strecken.
Eine Positionsermittlung mit Hilfe der Odometrie kann allgemein, wie in der folgenden Abbildung gezeigt, betrachtet werden.

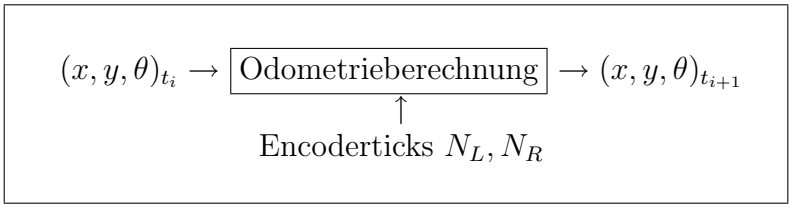

Abbildung 21: Allgemeine Odometriepositionsermittlung.

Nach [Kon98] kann die Odometrieberechnung, die Berechnung der aktuellen Position des Roboters, mit folgenden Gleichungen erfolgen. Die Position wird hier als Tripel (x, y, θ) verstanden, die neue Position, zum Zeitpunkt t_{i+1}, wird als (x', y', θ') geschrieben.

$$x' = x + \Delta s \cos\theta \tag{19}$$

$$y' = y + \Delta s \sin\theta \tag{20}$$

$$\theta' = \theta + \Delta\theta \tag{21}$$

Hierbei sind (x, y, θ) die aktuelle Position des Roboters, Δs der zurückgelegte Weg und $\Delta\theta$ die Richtungsänderung.
Diese Gleichungen funktionieren aber nur unter der Annahme, das Δs und $\Delta\theta$ sehr klein sind, also für geringe Bewegungen des Roboters.
Die auftretenden Fehler können allgemein in drei Hauptgruppen klassifiziert werden. Eine Abbildung dazu ist im Kapitel *Navigation Mobiler Roboter* zu sehen.

- Range error k_R
- Turn error k_θ
- Drift error k_D

Fehler können mathematisch als Zufallsvariablen behandelt werden. Sie besitzen eine Varianz bzw. eine Standardabweichung.
Wenn ein Roboter eine Wegstrecke s zurücklegt und um einen Winkel θ dreht, so ergeben sich folgende Varianzen.

$$\sigma_s = k_R s$$

$$\sigma_\theta = k_\theta + k_D s$$

Der Systemzustand, also die Position des Roboters $p = \begin{pmatrix} x \\ y \\ \theta \end{pmatrix}$, läßt sich wie folgt darstellen.

$$f(\bullet) : \begin{pmatrix} x \\ y \\ \theta \end{pmatrix} = \begin{pmatrix} x + s\cos\theta \\ y + s\sin\theta \\ \theta + \alpha \end{pmatrix} \tag{22}$$

p ist hierbei ein Vektor von Zufallsvariablen mit drei Komponenten. Es bleibt festzuhalten, daß die Komponenten x, y und θ nicht unabhängig sind, im Gegensatz zu s und $\Delta\theta$. So wirkt zum Beispiel die Varianz des Winkel θ auf die Varianz von x und y.

Zur Beschreibung der Fehler bzw. der Ungenauigkeitsmodellierung muß zwischen zwei Fällen unterschieden werden.

1. der Roboter befindet sich exakt auf seiner Position und Fehler entstehen während der Bewegung des Roboters

2. es liegt ein Fehler in der Ausgangsposition des Roboters vor, aber der Roboter führt seine Bewegungsbefehle exakt aus

Die folgende Abbildung verdeutlicht den Fall der exakten Position und den Einfluß einer fehlerhaften Bewegung. Eine fehlerhafte Bewegung bzw. eine fehlerhafte Wahrnehmung der Bewegung kann zum Beispiel durch die Samplerate der Odometrie-Encoder oder durch die Encoder-Auflösung verursacht werden.

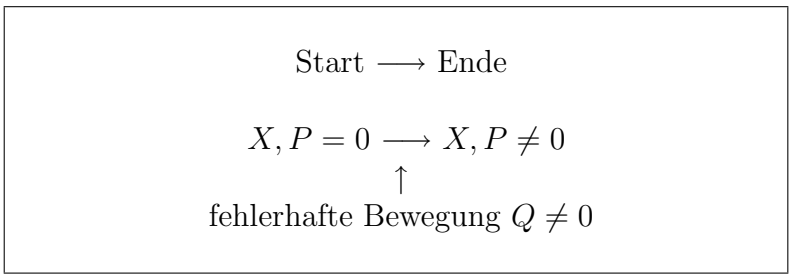

Abbildung 22: Einfluß einer fehlerhaften Bewegung bei exakter Position.

Die Kovarianzmatrix nach der Bewegung im Koordinatensystem des Roboters, dem Local Perceptual Space, LPS ist die folgende.

$$covar(p) = \begin{bmatrix} k_R s & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & k_\theta \theta + k_D s \end{bmatrix} \tag{23}$$

Das globale Koordinatensystem, der Global Perceptual Space, GPS ist um den Winkel θ gedreht. Dies läßt sich mit der folgenden Matrix beschreiben.

$$R(\theta) = \begin{pmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{24}$$

Die Transponierte Matrix zu $R(\theta)$ ist $R(\theta)^T = R(-\theta)$.

$$R(\theta)^T = R(-\theta) = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{25}$$

Somit läßt sich die Kovarianzmatrix im globalen Koordinatensystem, $covar(p')$, also die Fehlerdarstellung nach der Bewegung, wie folgt schreiben.

$$covar(p') = \begin{pmatrix} R(-\theta) \begin{bmatrix} k_R s & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & k_\theta\theta + k_D s \end{bmatrix} R(\theta) \end{pmatrix} \tag{26}$$

Die folgende Abbildung verdeutlicht den Sachverhalt bei einem Fehler in der Ausgangsposition.

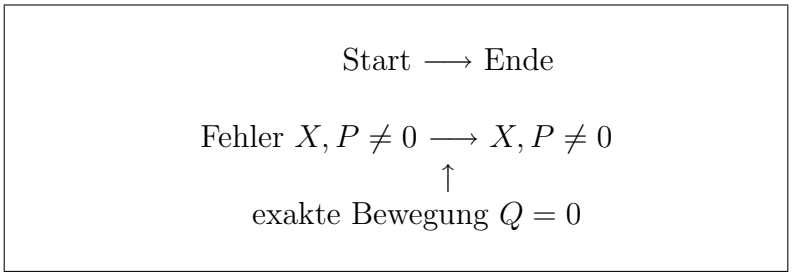

Abbildung 23: Fehler in der Ausgangsposition.

Hier liegt eine exakte Bewegung vor, d.h. der Roboter führt die ihm gegebenen Bewegungsbefehle genau aus und es gibt keine Fehler wie Range Error, Turn Error und Drift Error.
Der Sachverhalt, daß ein Fehler in der Ausgangsposition vorliegt und eine exakte Bewegung ausgeführt wird, kann wie folgt beschrieben werden.

$$covar(p') = F(\hat{p})covar(p)F(\hat{p})^T \tag{27}$$

Die *Jacobi - Matrix* ist allgemein wie folgt gegeben.

$$Jacobi - Matrix = \begin{pmatrix} \frac{\partial f_x}{\partial x} & \frac{\partial f_x}{\partial y} & \frac{\partial f_x}{\partial \theta} \\ \frac{\partial f_y}{\partial x} & \frac{\partial f_y}{\partial y} & \frac{\partial f_y}{\partial \theta} \\ \frac{\partial f_\theta}{\partial x} & \frac{\partial f_\theta}{\partial y} & \frac{\partial f_\theta}{\partial \theta} \end{pmatrix}$$

Konkret, d.h. nach Bildung der entsprechenden Ableitungen, sieht sie wie folgt aus.

$$F(\hat{p}) = A = \begin{bmatrix} 1 & 0 & -s\sin\theta \\ 0 & 1 & s\cos\theta \\ 0 & 0 & 1 \end{bmatrix} \tag{28}$$

Als Beispiel zur Bildung der Ableitungen sei hier die erste Zeile der Matrix A erläutert.

$$f_x = x + s\cos\theta$$

$$f'_{x/x} = 1$$

$$f'_{x/y} = 0$$

$$f'_{x/\theta} = -s\sin\theta$$

Die folgende endgültige Gleichung, zur Unsicherheitsmodellierung der Position eines autonomen mobilen Roboters, vereinigt nun die Aspekte der Integration der Fehler und der Roboterbewegung, und zwar durch Addition.

- $F(\hat{p}) \cdot covar(p) \cdot F(\hat{p})^T$ – Fehler in der Ausgangsposition

- $R(-\theta) \begin{bmatrix} k_R s & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & k_\theta\theta + k_D s \end{bmatrix} R(\theta)$ – Fehler bei der Bewegung

Jeder Teil dient zur Berechnung einer entsprechenden Kovarianzmatrix $covar(p')_{FehlerStartpos.}$ und $covar(p')_{FehlerBeweg.}$. Für den jeweiligen Fall werden so die

jeweiligen Elemente der Kovarianzmatrix manipuliert. Durch Addition lassen sich nun die fehlenden Elemente in der endgültigen Kovarianzmatrix $covar(p')$ einsetzen bzw. verstärken.

$$covar(p') = F(\hat{p}) \cdot covar(p) \cdot F(\hat{p})^T + R(-\theta) \begin{bmatrix} k_R s & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & k_\theta \theta + k_D s \end{bmatrix} R(\theta) \quad (29)$$

Zum Vergleich sei hier nochmals die Gleichung des erweiterten Kalman-Filters dargestellt.

$$P_{k+1} = A_k \cdot P_k \cdot A_k^T + W_k Q_k W_K^T \quad (30)$$

Als Fazit bleibt festzuhalten, daß unter Annahme eines Systemmodells aus dem aktuellen Δs und dem aktuellen $\Delta\theta$ aus P_k bzw. $covar(p')_{alt}$ ein neues P_{k+1} bzw. $covar(p')_{neu}$ berechnet werden kann. Das Systemmodell besteht hier, wie schon besprochen, unter anderem aus den Komponenten $k_R s$ sowie der Matrix A.

Die bis hier niedergelegte Darstellung, besonders die mathematische Entwicklung, zur Ermittlung der Positionsunsicherheit eines autonomen mobilen Roboters geht auf Kurt Konolige zurück, siehe [Kon98].

Das in der folgenden Abbildung dargestellte Blockdiagramm des Odometrie Sensor Moduls zeigt die Arbeitsweise konkret auf die Umgebung des Pioneer 2 CE Roboters bezogen.
Hierbei geht es schwerpunktmäßig um die Schätzung der Ungenauigkeit der von Saphira gelieferten Position. Die Daten von Saphira werden mit dem im oberen Teil beschriebenen Verfahren verarbeitet und bilden so die Meinung des Odometrieexperten.
Es ist zu sehen, daß dieses Modul initialisiert werden muß, und zwar mit der Ungenauigkeit der Startposition. Hierbei geht es aber nicht um die Lösung des sogenann-

ten Aufwachproblems.
Des Weiteren ist die rekursive Arbeitsweise hervorgehoben. Die Schätzung der Positionsunsicherheit zum Zeitpunkt k fließt in die Berechnung der neuen Positionsunsicherheit zum Zeitpunkt $k+1$ ein.

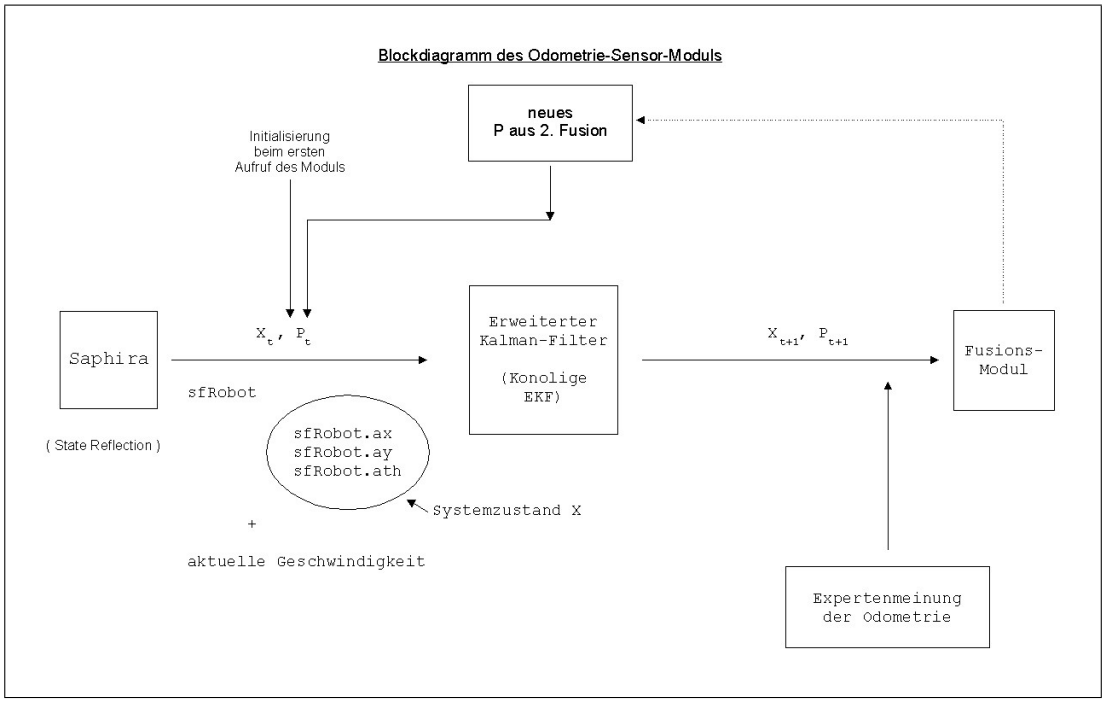

Abbildung 24: Das Blockdiagramm des Odometrie Sensor Moduls.

9.2 Das Kompass-Modul

Dieses Modul ist, wie der Name schon sagt, für die Verarbeitung der Kompasswerte zuständig. Der Kompass liefert die Ausrichtung θ, in die der Roboter ausgerichtet ist. Er liefert diese Information absolut. Der Kompass unterliegt aber gewissen Störeinflüssen.
Der elektronische Kompass am Pioneer 2 CE Roboter misst die Ausrichtung des Roboters relativ zum Norden des Erdmagnetfeldes.
In der folgenden Abbildung ist das Magnetfeld bzw. die Magnetfeldkarte des KI-Labors dargestellt, entnommen aus [GL00].

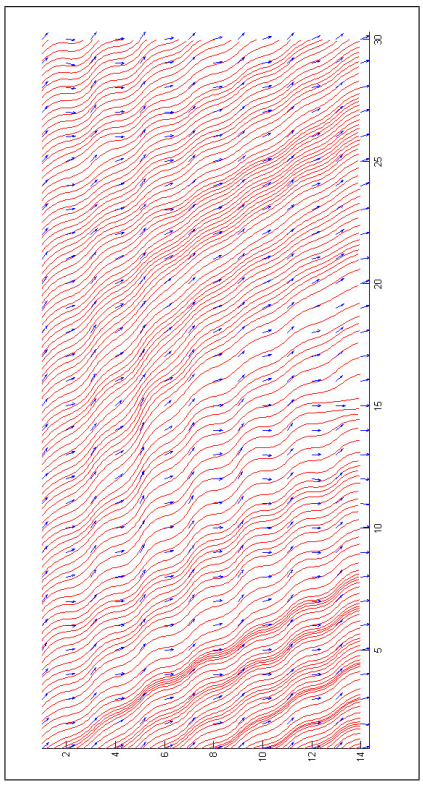

Abbildung 25: Magnetfeld mit Feldlinien.

Es ist zu sehen, daß eigentlich ein homogenes Magnetfeld vorliegt. Das dargestellte Magnetfeld hat sich durch alternierendes Abfahren von Streifen vor der Freifläche vor den Schränken im KI-Labor ergeben. Es ist wellenförmige und kann als systematischer Fehler betrachtet werden. Der Kompass kann in dieser Umgebung also zur Positionsbestimmung des Roboters herangezogen werden.
Der Kompass unterliegt aber einer bestimmten spezifischen Abweichung. Diese wurden in der Studienarbeit [GL00] untersucht und als Abweichungskurve aufgenommen. In der folgenden Abbildung ist die Abweichung der Kompasswerte von der tatsächlichen Ausrichtung des Roboters als Diagramm veranschaulicht.
Unter Verwendung dieser Abweichungskurve, kann der Kompass des Pioneer 2 CE Roboters in der Umgebung des KI-Labor zur Richtungsbestimmung dienen. Diese Kurve kann als LUT, als Look Up Table, angesehen bzw. benutzt werden. Aus dem gemessenen Kompasswert kann so die korrekte Richtung ermittelt werden.

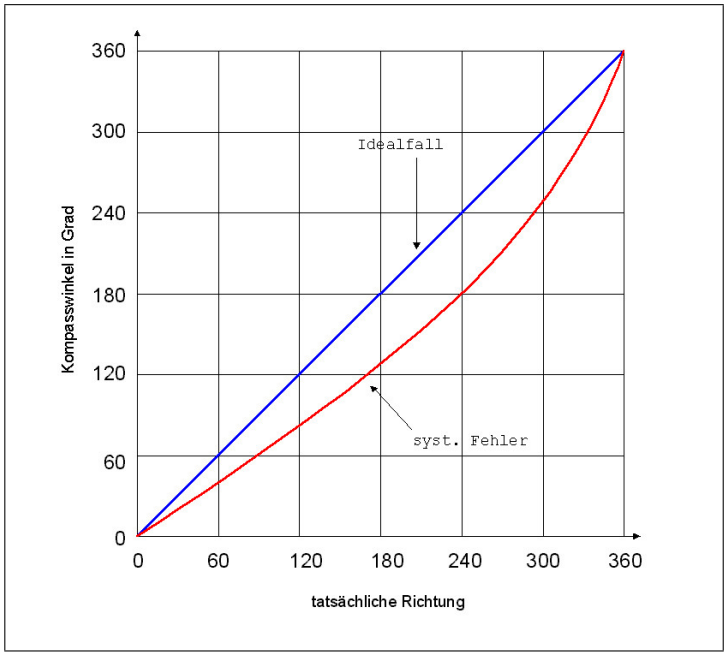

Abbildung 26: Kompasswinkel / Abweichungskurve.

9.3 Das Sonar-Modul

Das Sonar Sensor Modul hat die Aufgabe die Informationen der Sonarsenoren aufzubereiten und zu verarbeiten. Ein Kern dieses Moduls stellt der sogenannte *Matchingprozess* dar. Hierbei sind ein Objekt-Modell mit einem Bild-Modell in Übereinstimmung zu bringen.

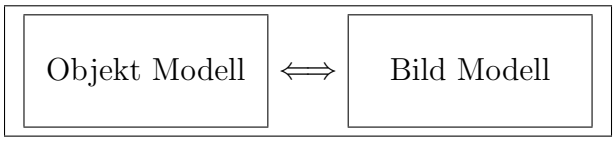

Abbildung 27: Matching.

Methoden, die für das Sonar Matching in Frage kommen, sind zum Beispiel die Bayessche Methode (BOK – Bayes optimaler Klassifikator) oder die Anwendung Heuristischer Funktionen.

Um Sonarwerte in Übereinstimmung bringen zu können, werden diese in eine Karte eingetragen. Dies erfolgt über einen bestimmten Zeitraum hinweg. Der so entstehende Scan muß nun, zur Positionsermittlung, mit einer Karte gematcht werden. Dies heißt Scanüberdeckung, d.h. die Punkte des Scans werden mit dem Modell der Umgebung in Deckung gebracht.

Hier bietet sich das Verfahren der Kreuzkorrelation an.

$$c_t := \sum_{k=0}^{n-1} y_k \cdot z_{(k+l)} \tag{31}$$

Der Matchingprozess kann allgemein in drei Phasen unterteilt werden, dies sind:

1. Rotation $\longrightarrow$ Drehen, bis die beste Übereinstimmung erreicht ist
2. Translation $\longrightarrow$ Verschieben
3. Bestimmung der Position des Roboters

Die folgende Abbildung zeigt nochmals die allgemeine Vorgehensweise beim Vergleich von Beobachtung und Karte.

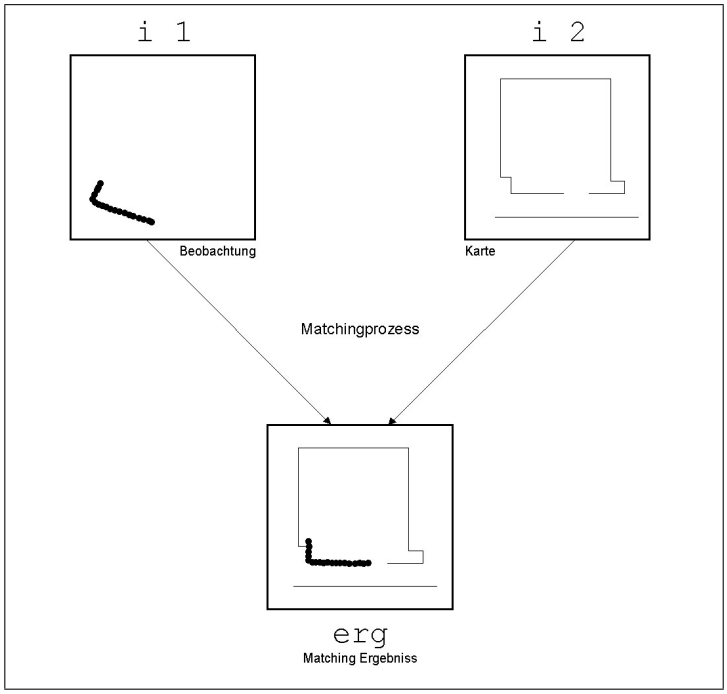

Abbildung 28: Darstellung des *,,Matching-Prozesses"* zum Vergleich von Beobachtung und Karte.

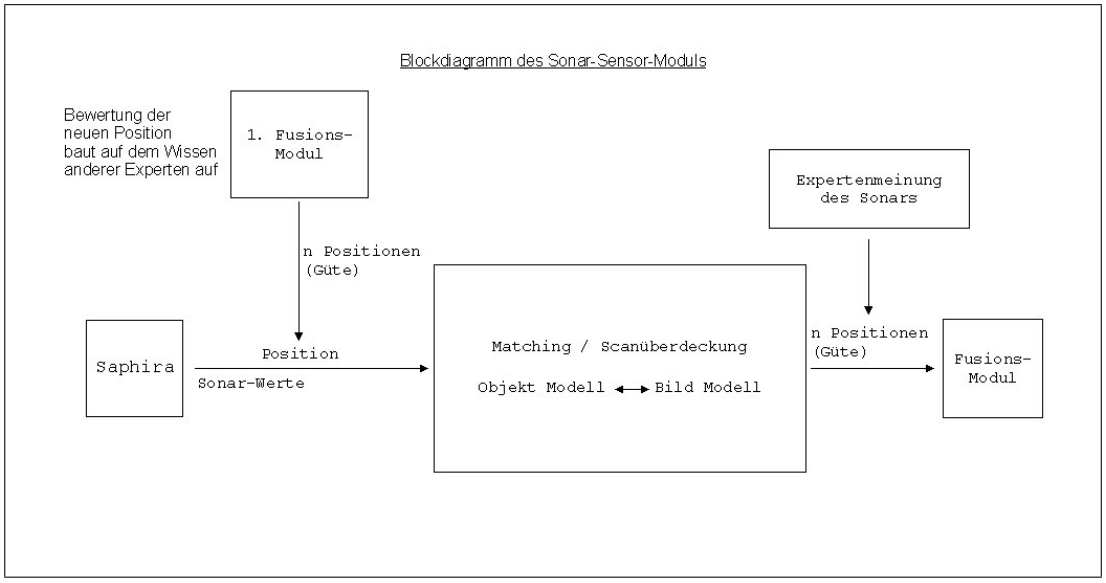

Abbildung 29: Das Blockdiagramm des Sonar (Ultraschall) Sensor Moduls.

Das in der obrigen Abbildung dargestellte Blockdiagramm beschreibt die Arbeitsweise des Sonar Sensor Moduls. Es ist zu sehen, daß die ,,reinen" Sonarwerte der Sonarsensoren am Pioneer 2 CE Roboter und die geschätzte Position des Roboters als Eingaben dienen. Dieses Sensormodul ist auf die Expertenmeinung anderer Module angewiesen, da die Position aus dem ersten Fusionsmodul bezogen wird. Intern wird nun der Matchingprozess ausgeführt und als Ausgabe die Expertenmeinung des Sonars geliefert. Dies können mehrere mögliche Positionen sein.

9.4 Das Kamera-Modul

Die Kamera ist ein bildgebender Sensor, der seine Umgebung als Farbbilder aufnimmt. Durch diesen Sensor wird keine konkrete Position geliefert. Es ergibt sich also die Aufgabe einer Merkmalsextraktion und die Nutzung dieser Merkmale zur Positionsbestimmung. Die Arbeitsweise ist in der folgenden Abbildung verdeutlicht.

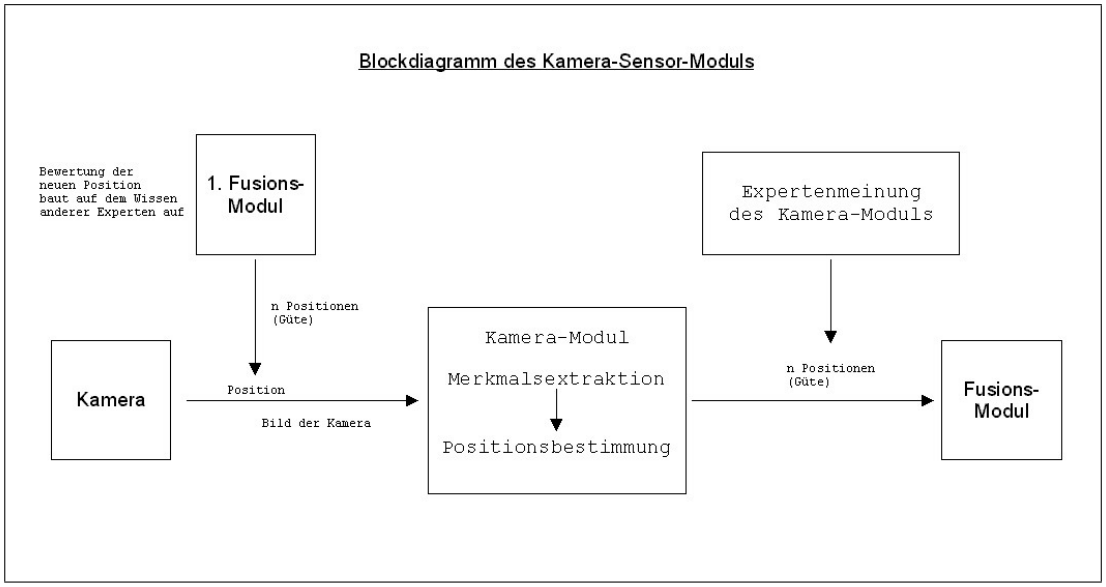

Abbildung 30: Die schematische Arbeitsweise des Kamera-Moduls.

Dieses Modul kann von der Art der Ausgabe mit dem Sonar bzw. mit dem Sonar Sensor Modul verglichen werden. Auch hier wird eine Menge von n möglichen Positionen, die wahrscheinlich sind, als Ausgabe geliefert.
Ein Beispiel für die Wahrnehmung der Umgebung durch die Kamera, ist in der folgenden Abbildung zu sehen.

Abbildung 31: Bild der Kamera des Pioneer 2 CE Roboters.

Aus diesem Bild müssen nun bestimmte Merkmale extrahiert werden, zum Beispiel die Scheuerleiste als Linie zwischen dem Boden und der weißen Wand. Anhand dieses Merkmals kann nun versucht werden, Vorschläge für den Ort des Roboters zu liefern. Hier Korridor vor dem KI-Labor.

9.5 Weitere mögliche Sensor-Module

Weitere denkbare Sensoren und somit Sensor-Module, wären zum Beispiel das Gyroskop, der Einsatz eines Laserscanner oder auch Transponder.
Das Gyroskop, oder auch der Kreiselkompaß, liefert sehr genau die Orientierung des Roboters, er unterliegt aber mit der Zeit Driftfehlern, so zum Beispiel bei großen Beschleunigungen.
Mit einem Laserscanner können zwei- und dreidimensionale Abstandsprofile erstellt werden. Diese sind aufgrund des sehr kleinen Öffnungswinkel des Laserscanners sehr genau, [Gut96].

9.6 Die Fusions-Module

Die Fusions-Module haben, wie schon gesagt, die Aufgabe der globalen Sensorintegration. Sie sollen die Daten der einzelnen Sensor-Module zusammenführen und eine Zustandsschätzung liefern.

Mit welcher Methode bzw. welchem Verfahren, kann eine Fusion erfolgen ? Hier können folgende Verfahren in Betracht gezogen werden.

- Statistische Verfahren
  - Bedingte Wahrscheinlichkeiten
  - Bayes
  - Normal Verteilungen
- Kalman-Filter Konzept
- Wissensbasiert; Wenn-Dann-Regeln

Um den Charakter der intelligenten Verknüpfung bzw. Fusion mehrer Sensoren bzw. Sensormodule hervorzuheben, wird im Folgenden von der sogenannten *Experten Fusion* gesprochen.
Experten Fusion heißt hier, zusätzlich zur mathematischen Fusion soll ein Experte eingesetzt werden, der entscheidet, ob zum Beispiel nach dem Kalman-Filterprinzip zusammengefaßt wird oder nicht. Ziel ist die korrekte Fusion, auch wenn nichtsystematische Fehler vorliegen.
Welche Ausgaben können die Sensoren liefern ? Hier können folgende drei grundlegende Fälle unterschieden werden. Es wird davon ausgegangen, daß Experten Ausgaben über die Position eines autonomen mobilen Roboters machen. Die folgende Abbildung veranschaulicht den ersten Fall.

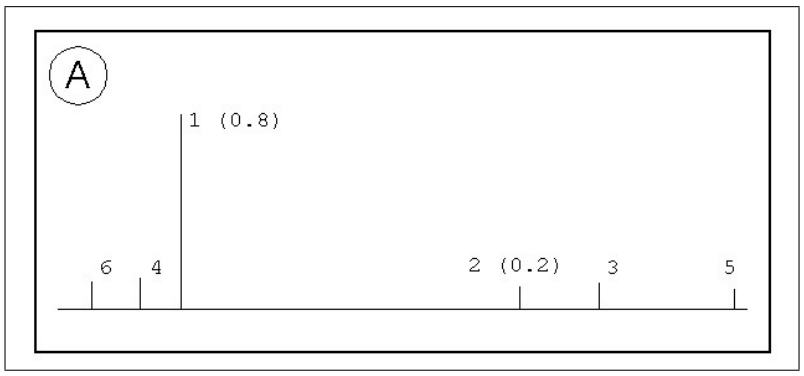

Abbildung 32: Fall A zur Expertenfusion.

Der Experte eins ist sich mit einer Wahrscheinlichkeit von 0.8 sehr sicher, daß sich der Roboter an dem von ihm beobachteten Ort befindet. Die übrigen Experten machen sehr unsichere Angaben, mit Wahrscheinlichkeiten von zum Beispiel 0.2. Zudem liegen die Expertenmeinungen sehr weit auseinander, so dass in diesem Fall nur auf die Expertenmeinung des ersten Experten zurückgegriffen werden kann. Dieser erzeugt in diesem Fall die Ausgabe. Aufgrund der Tatsache, daß nur ein Experte herangezogen werden kann, ergibt sich eine entsprechend geringe Sicherheit der Ausgabe, so zum Beispiel von 0.7.

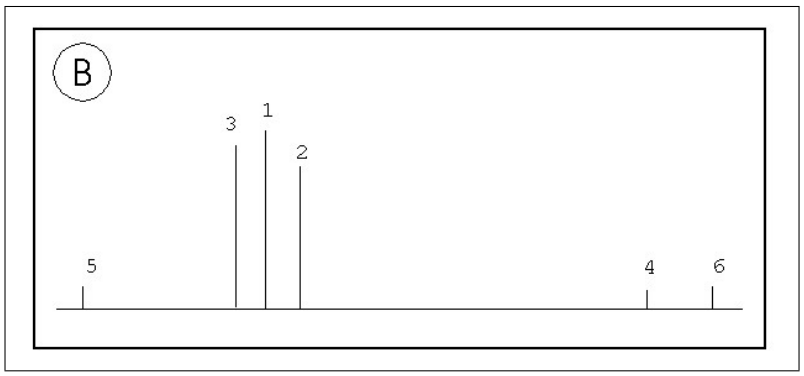

Abbildung 33: Fall B zur Expertenfusion.

Im Fall B sind die Experten eins, zwei und drei in etwa der selben Meinung, daß der Roboter sich an der von ihnen gedachten Position befindet. Die übrigen Experten

vier, fünf und sechs liegen zuweit auseinander und sind mit sehr geringen Wahrscheinlichkeiten zu unsicher. Hier wird das Ergebnis der Fusion aus den Meinungen der Experten eins, zwei und drei ermittelt. Dies kann zum Beispiel mit Hilfe des Kalman-Filterprinzips erfolgen, wobei die Meinungen der Experten eins, zwei und drei ,,optimal" zusammengefaßt werden. Die Sicherheit der Ausgabe, ist aufgrund der übereinstimmenden Meinung mehrerer Experten sehr hoch, zum Beispiel 0.99.

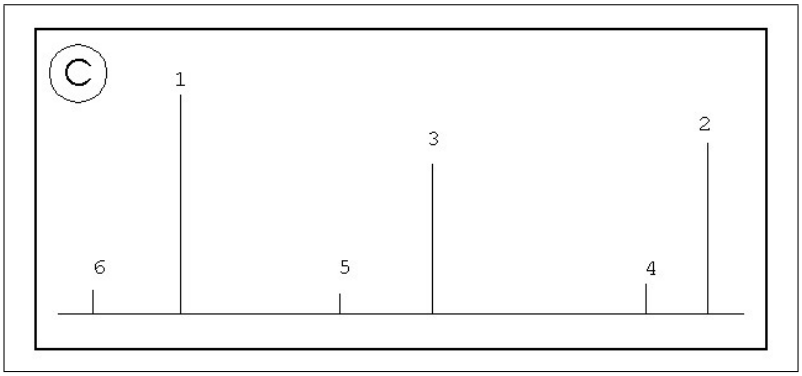

Abbildung 34: Fall C zur Expertenfusion.

Im letzten Fall sind drei Experten, mit hohen Wahrscheinlichkeiten, sehr unterschiedlicher Meinung. Die Experten eins, zwei und drei sind sich jeweils sehr sicher, daß sich der Roboter an der von ihnen gesagten Position befindet. Diese Expertenmeinungen liegen jedoch sehr weit auseinander, so das man den Experten beachtet, der sich am sichersten ist. Der Experte eins bestimmt in diesem Fall die Ausgabe. Die anderen Experten vier, fünf und sechs liegen ebenfalls sehr weit auseinander, haben aber sehr geringe Wahrscheinlichkeiten und tragen deshalb nicht zum Fusionsergebnis bei. Die Sicherheit der Ausgabe ist hier gering, da nur auf einen Experten, der sich aber sehr sicher ist, zurückgegriffen wird. Sie könnte zum Beispiel 0.5 betragen. Die drei dargelegten Fälle der Expertenfusion haben gezeigt, daß Expertenmeinungen, die ein Maxima darstellen, sehr großes Gewicht für die Fusion bekommen. Man kann deshalb diese Art der Fusion auch als *Maxima Experten Fusion* bezeichnen. Die Fälle A, B und C stellen hierbei sogenannte *Maxima Klassen* dar.
Die dargelegten drei Fälle zur Expertenfusion decken grundlegend alle möglichen auftretenden Fälle ab. Von diesen Fällen können nun wieder andere Fälle abgeleitet

werden. So zum Beispiel, wenn sich Expertengruppen bilden, die sich, wie der Fall C darstellen. Hier könnten mehrere Experten ähnliche Schätzungen wie der Experte eins abgeben und andere wieder eine Expertengruppen um den Experten zwei bilden. Die Expertengruppe, die am sichersten ist würden die Ausgabe bilden.

10 Architektur und Architekturentwurf

Im engeren Sinne wird unter einer Architektur die Aufteilung eines Softwaresystems in seine Komponenenten, meist Module, deren Schnittstellen, die Prozesse und Abhängigkeiten zwischen ihnen, sowie die benötigten Ressourcen verstanden, [RP97]. Dies soll in Bezug auf das zu entwickelnde *Selbstlokalisierungsmodul* Gegenstand des folgenden Abschnittes sein.

10.1 Übertragung der Problemstellung auf die Architektur

Als Grundkonzept für die Architektur dient die stufenweise Fusionierung der Sensoren, nach Art der Sensoren. Also die Fusionierung nach Art der Informationen, die ein Sensor liefert.

10.2 Arten von Sensoren

Es können generell zwei Arten von Sensoren bzw. Sensormodulen unterschieden werden.

1. Sensoren, die eine eindeutige Position oder eine Positionskomponente liefern, die sogenannten Primärsensoren
2. Sensoren, die n Positionsmöglichkeiten ausgeben, die sogenannten bildgebenden Sensoren

Im ersten Fall wird eine Sensorinformation verarbeitet und eine Position mit einer gewissen Unsicherheit ausgegeben.

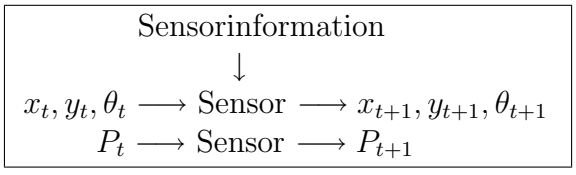

Abbildung 35: Beispielhafter Aufbau eines Primärsensors.

Zur ersten Kategorie zählen hier Sensoren, wie der Kompaß und das Gyroskop aber auch die Odometrie-Sensoren.
Sensoren bzw. Sensormodule, die mehrere mögliche Positionen für einen autonomen mobilen Roboter ausgeben können, werden dem zweiten Fall zugeordnet. Auch hier wird die mögliche Position mit einer Sicherheit bzw. Unsicherheit beschrieben. Die folgende Abbildung zeigt den Aufbau eines solchen Sensors.

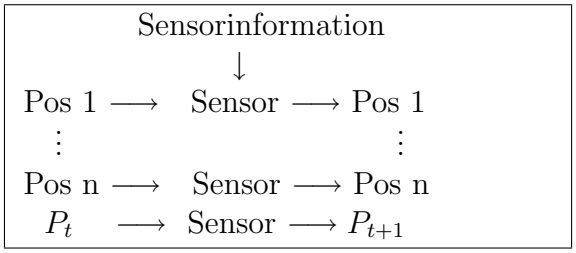

Abbildung 36: Beispielhafter Aufbau eines bildgebenden Sensors.

Solche Sensoren sind zum Beispiel die Kamera oder die Sonar- bzw. Ultraschallsensoren. Für die Verarbeitung der gelieferten Positionsschätzung spielt hierbei die Tatsache, wie der Winkel einbezogen wird eine wichtige Rolle. Es besteht zum einen die Möglichkeit, zu jeder Positionsangabe n mögliche Ausrichtungen des Roboters in Betracht zu ziehen. Zum anderen kann eine Beschränkung auf einen Winkel pro Position erfolgen. Die folgende Abbildung veranschaulicht dies nochmals.

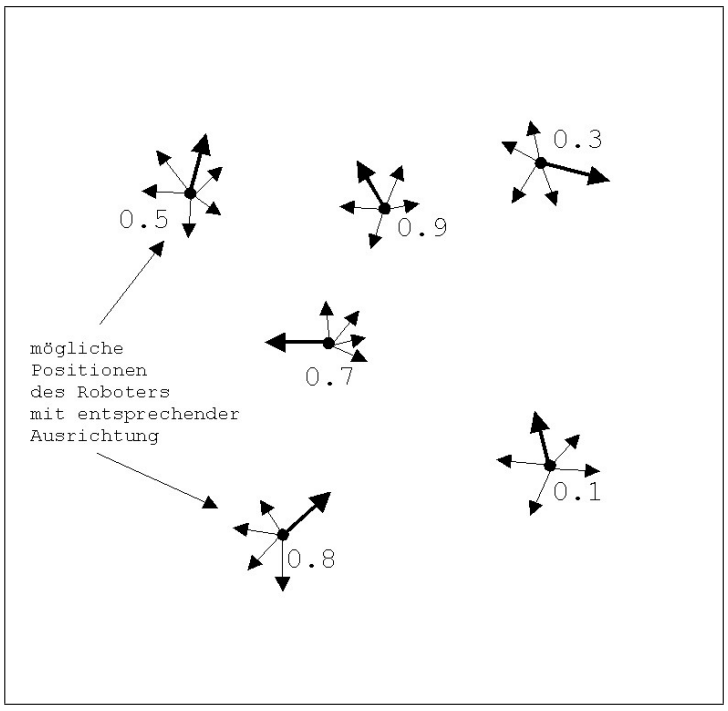

Abbildung 37: Zur Problematik Winkeleinbeziehung pro Position.

10.3 Das Blockdiagramm des Selbstlokalisierungsmoduls

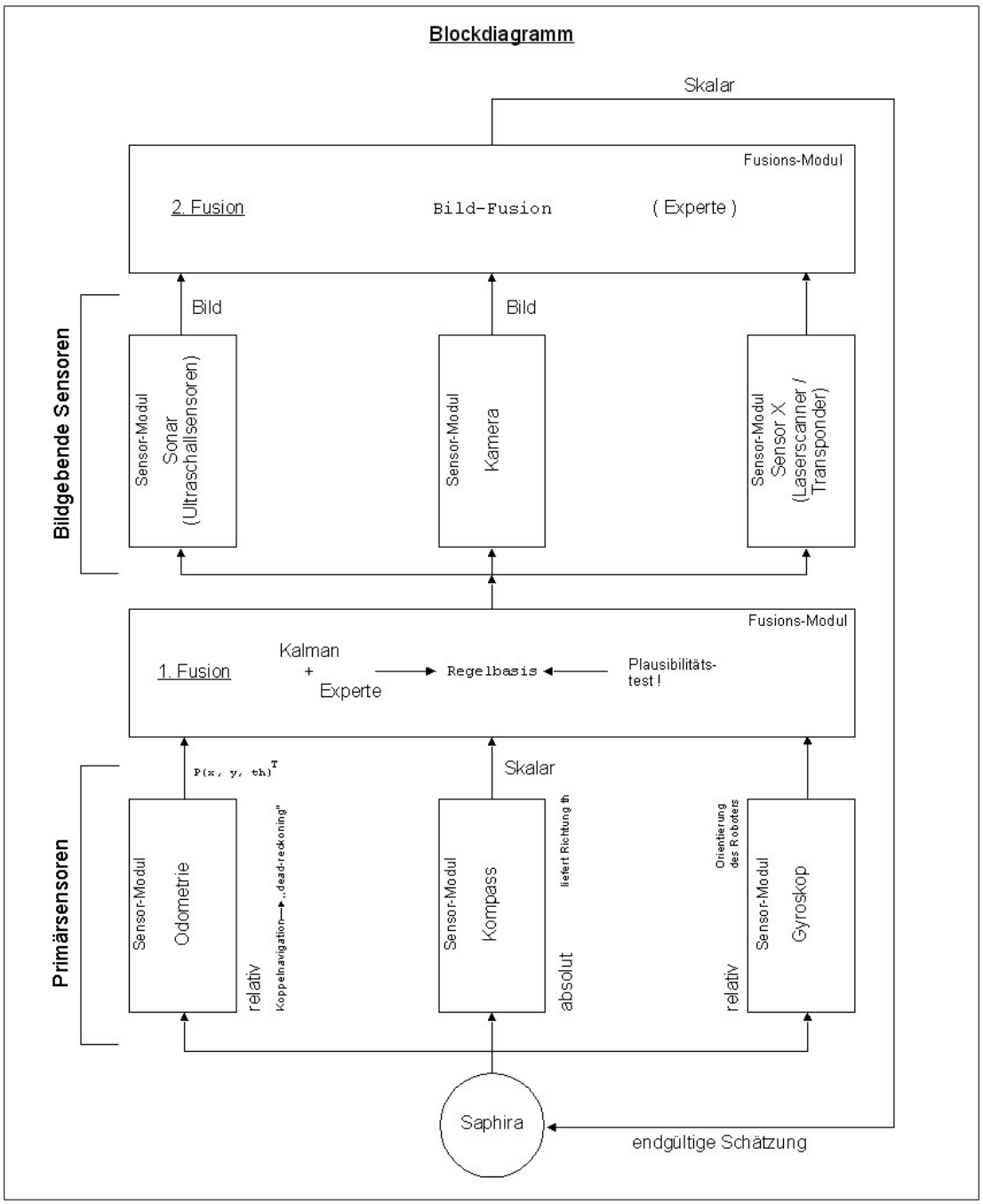

Abbildung 38: Das Blockdiagramm des gesamten Selbstlokalisierungsmoduls.

Das gezeigte Blockdiagramm des Selbstlokalisierungsmoduls läßt sich allgemein in vier Ebenen unterteilen. Die erste Ebene bilden Sensoren bzw. daraus resultierende Sensormodule, die eine Positionsschätzung liefern bzw. eine Positions- oder Geschwindigkeitsbestimmung ermöglichen. Sie werden hier als *Primärsensoren* bezeichnet.
Diese Sensoren werden durch das erste Fusionsmodul, die zweite Ebene, vereinigt. Hier kommt die bereits erwähnte Expertenfusion zur Anwendung. Hier werden n Positionen, die wahrscheinlich sind, als Eingabe für die dritte Ebene ausgegeben.
Die dritte Ebene stellen die sogenannten *bildgebenden Sensoren* dar. Diese können mehrere Positionen bewerten und haben ein Bild als Ausgabe.
Die vierte und letzte Ebene ist das zweite Fusionsmodul, das die Aufgabe hat, die bildgebenden Sensoren zu fusionieren und eine endgültige Positionsschätzung zu liefern.
Die Fusion der jeweiligen Ausgabe der Sensor-Module übernehmen das erste und das zweite Fusionsmodul. Bei der Ersten Fusion erfolgt hierbei eine Vereinigung der gelieferten Werte, zum Beispiel vom Odometrie-Modul und Kompass-Modul, nach dem Kalman-Filter-Prinzip. Zusätzlich kommt ein Experte hinzu, der über Weltwissen verfügt und die Fusion steuert. Dieses Weltwissen kann zum Beispiel das Wissen sein, das der Kompaß temporäre Aussetzer hat und ungenau bei Drehungen ist oder die maximale Geschwindigkeit des Roboters.
Die Zweite Fusion ist aufgrund der Tatsache, daß die bildgebenden Sensoren, wie zum Beispiel das Sonar oder die Kamera, als Ausgabe Bilder liefern, eine Bildfusion. Hierbei werden die erhaltenen Bilder, in denen die Positionsschätzungen eingetragen sind, fusioniert. Ein Experte kann auch diese Fusion beeinflußen bzw. steuern.
Es ist zu sehen, daß die Fusionsmodule, je nachdem, ob sie Primärsensoren oder bildgebende Sensoren fusionieren, unterschiedliche Fusionsansätze beinhalten.

10.4 Eigenschaften der Architektur

Wenn Sensoren bzw. Sensormodule wie in der aufgezeigten Architektur angeordnet werden, so ergeben sich folgende Vorteile.
Diese Architektur ist in der Lage, Aussetzer des Meßprinzips bzw. nichtsystematische Fehler einzelner Sensoren zu kompensieren. Allgemein wird bei einem Sensor eine

physikalische Größe in eine *interessierende* Größe überführt. Die allgemeine Arbeitsweise wurde auch schon im Kapitel *Sensor-Data-Fusion* beschrieben. Die folgende Abbildung verdeutlicht nochmals das Meßprinzip eines Sensors.

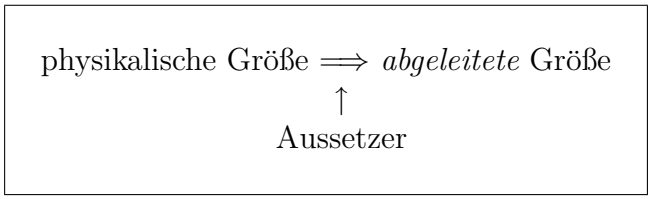

Abbildung 39: Zum Meßprinzip eines Sensors.

Ein Sensor überführt also eine physikalische Größe, wie zum Beispiel die Helligkeit oder die Laufzeit des Sonars in eine abgeleitete Größe. Für einen autonomen mobilen Roboter ist dies meist eine Position bzw. eine Positionskomponente. Zum Beispiel leitet der Kompass aus dem Vektor des Magnetfeldes die Richtung des Roboters ab. Sensoren können für einen bestimmten Zeitraum ausfallen und so falsche Ausgaben bzw. Informationen liefern.
Zum Beispiel wird beim Kompaß, der sich in einem Magnetfeld befindet, die Richtung des Roboters ausgegeben. Nichtsystematische Fehler treten hier zum Beispiel beim Überfahren eines Starkstromkabels auf.
Die Sonarsensoren bestimmen aus der Signallaufzeit die Entfernung zu einem Hindernis. Trifft das Sonar aber zum Beispiel auf eine schallisolierte Wand, so sind die gewonnenen Informationen falsch.
Die Kamera in einem zu dunklen Durchgang oder das Gyroskop, das einem Stoß ausgesetzt ist, liefern schlicht weg unwahre bzw. falsche Informationen.
Die Verknüpfung der einzelnen Sensoren, wie in der aufgezeigten Architektur, ermöglicht es, derartige Störungen intelligent zu verarbeiten und trotzdem eine zuverlässige Aussage über die Position eines autonomen mobilen Roboters zu liefern.
Des Weiteren ist zu sehen, daß diese Architektur relativ leicht erweiterbar ist. An jedes der beiden Fusionsmodule können theoretisch beliebig viele Sensormodule angeschlossen werden. Dazu muß ein neues Sensormodul klassifiziert werden und entsprechend als Eingabe für das erste bzw. zweite Fusionsmodul dienen.

Es ist ersichtlich, daß die Anzahl der Sensormodule die Genauigkeit der Positionsschätzung beeinflußt. Je mehr Sensormodule vorhanden sind, desso genauer kann das Selbstlokalisierungsmodul die aktuelle Position des Roboters ermitteln.

Teil IV

Implementierung

Dieser Abschnitt der Diplomarbeit befaßt sich mit der Implementierung des in der Konzeption besprochenen *Selbstlokalisierungsmoduls*.

11 Einleitung

Eine wichtige Zielstellung dieser Arbeit ist die Untersuchung von Selbstlokalisierungsverfahren autonomer mobiler Roboter. Kernpunkt dabei ist die *intelligente Verknüpfung* mehrerer Sensorquellen. Im Kapitel Konzeption dieser Arbeit wurde eine *Fusions-Architektur* aufgezeigt. Die praktische Umsetzung sowie die Implementierung auf dem Roboter Pioneer 2 CE, *Romeo*, sind nun Gegenstand dieses Kapitels.

12 Prinzipieller Aufbau

Im Rahmen der Lösung der praktischen Aufgabe hat sich das sogenannte *,,Selbstlokalisierungsmodul"* herausgebildet. Dieses Selbstlokalisierungsmodul besteht nun selbst wieder aus spezialisierten Modulen und Komponenten. Die wichtigsten sind das Odometrie-Modul, das Kompaß-Modul, das Sonar-Modul, die Fusions-Module 1 und 2 sowie die Saphira Umgebung.
Jedes dieser Module stellt eine separate Windowsanwendung dar. Diese ist als WIN32-Anwendung implementiert und zeigt sich dem Anwender als Windows-Dialog.
Die Kommunikation der Module untereinander sowie die Kommunikation mit Saphira erfolgt mit Hilfe eines gemeinsamen Speicherbereichs.
Die folgende Abbildung verdeutlicht nocheinmal den prinzipiellen Aufbau und die Implementierung des Selbstlokalisierungsmoduls.

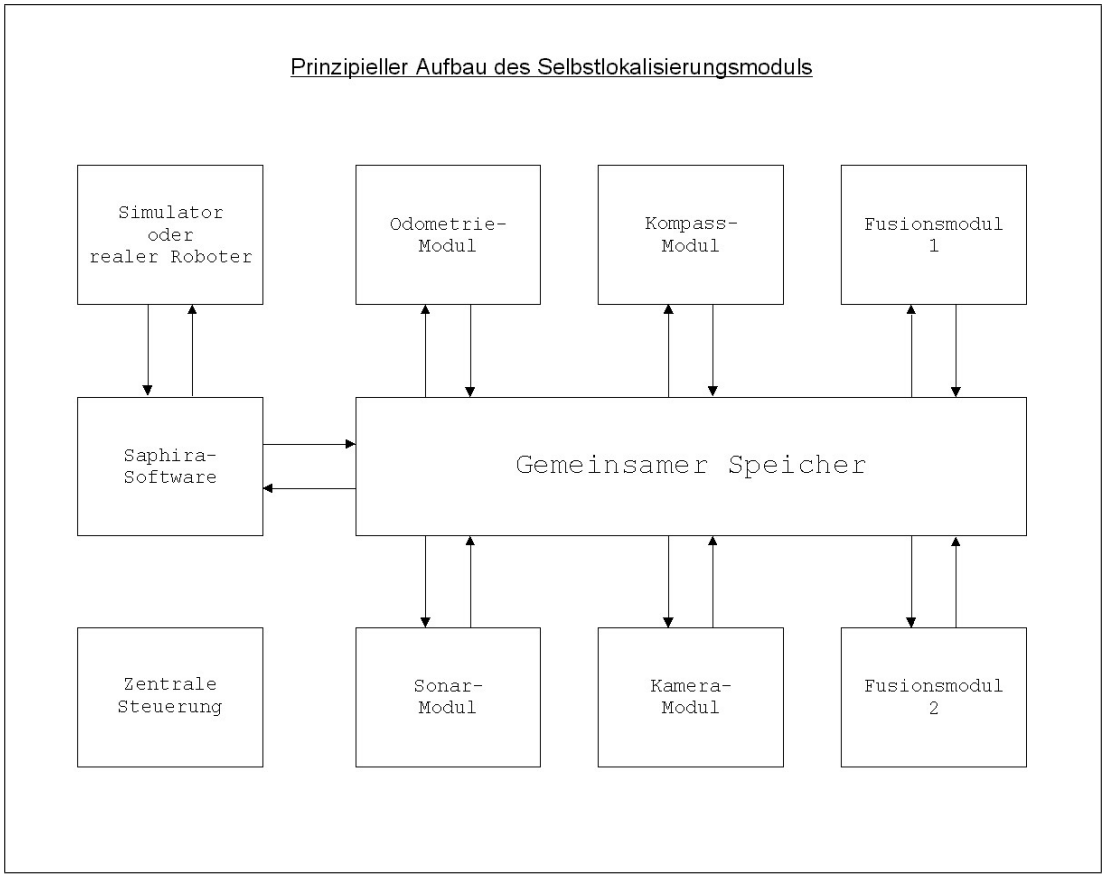

Abbildung 40: Der prinzipielle Aufbau des Selbstlokalisierungsmoduls.

Die Saphira Umgebung ist hierbei die Schnittstelle zum realen Roboter bzw. zum Simulator. Im Rahmen des Selbstlokalisierungsmoduls werden zwei Verhalten, die in der Datei `Self.act` definiert sind, geladen. Das erste, ist das Verhalten `writepos()`. Dieses sorgt für den Datenaustausch, so zum Beispiel der aktuell gelieferten Position, durch interagieren mit dem gemeinsamen Speicher. Es werden konkret folgende Daten aus Saphira in den gemeinsamen Speicher geschrieben:

- Die aktuelle Roboterposition $p_{akt} = \begin{pmatrix} x \\ y \\ \theta \end{pmatrix}$ aus der `sfRobot`-Struktur
- Die aktuelle Geschwindigkeit des linken und rechten Rades aus der `sfRobot`-

Struktur

- Die gemessene Entfernung der Sonarsensoren 0 bis 7
- Der aktuelle Kompaßwert

Diese Informationen werden von den jeweiligen Modulen ausgewertet und weiterverarbeitet. Saphira stellt also in dieser Umgebung einen Server da.
Das zweite Verhalten ist das Verhalten `setpos()`. Dieses hat die Aufgabe, den Roboter in der Karte mit der Funktion `sfMoveRobotPos()` zu versetzen. Die Position wird vom zweiten Fusionsmodul geliefert.

13 Die Komponenten des Selbstlokalisierungsmoduls

Im Folgenden wird die Implementierung einzelner Module des Selbstlokalisierungsmoduls besprochen. Dies sind konkret:

- das Odometrie-Modul
- das Kompaß-Modul
- das Erste Fusions-Modul
- das Kamera-Modul
- das Sonar-Modul
- das Zweite Fusions-Modul

Des Weiteren wird die Komponente ,,gemeinsamer Speicher" erläutert. Öffnet man unter Microsoft Visual C++ den Arbeitsbereich `center.dsw`, so werden alle Projekte des ,,Selbstlokalisierungsmoduls" bereitgestellt. Allgemein besteht hierbei ein Projekt aus drei Dateien. Dies sind `<modulname>.c`, die Quellcodedatei, die Headerdatei `resource.h` und die eigentliche Resourcendatei `<modulname>.rc`.

13.1 Der gemeinsame Speicher

Wie schon erwähnt, dient der gemeinsame Speicher zur Kommunikation der Module mit Saphira und der Kommunikation der Module untereinander. Compiliert man das Projekt `SelfDll`, so erhält man eine Dynamic Link Library, die Datei `SelfDll.dll`. Wird diese DLL in Saphira geladen, so wird ein gemeinsamer Speicherbereich unter Windows eingerichtet.
Dieser Speicherbereich, der shared memory, ist als Mappedfile implementiert. Hierzu wird beim laden der DLL, in der Funktion `sfLoadInit()` die Funktion

- ```
  HANDLE CreateFileMapping(HANDLE hFile,
  LPSECURITY_ATTRIBUTES lpFileMappingAttributes,
  DWORD flProtect,
  DWORD dwMaximumSizeHigh,
  DWORD dwMaximumSizeLow,
  LPCTSTR lpName)
  ```

aufgerufen. Als Parameter werden unter anderem die Größe und ein Name übergeben. Über diesen Namen können nun alle anderen Module den eingerichteten gemeinsamen Speicher benutzten. Die Größe wird anhand der Struktur des gemeinsamen Speichers bestimmt. Diese Struktur ist in der Headerdatei `self.h` definiert. Nachfolgend ist diese Struktur dargelegt.

- ```
  typedef struct sharedmem_struct {
  float saph_x; // Odometrie Pos.-> X  von Saphira
  float saph_y; // Odometrie Pos.-> Y  von Saphira
  float saph_th; // Odometrie Pos.-> Th von Saphira
  float saph_vleft; // aktuelle Geschwindigkeit von Saphira
  float saph_vright; // aktuelle Geschwindigkeit von Saphira
  float saph_sonar0; // ausgelesene Sonarwerte von Saphira Sonar 0
  float saph_sonar1; // ausgelesene Sonarwerte von Saphira Sonar 1
  float saph_sonar2; // ausgelesene Sonarwerte von Saphira Sonar 2
  float saph_sonar3; // ausgelesene Sonarwerte von Saphira Sonar 3
  float saph_sonar4; // ausgelesene Sonarwerte von Saphira Sonar 4
  float saph_sonar5; // ausgelesene Sonarwerte von Saphira Sonar 5
  float saph_sonar6; // ausgelesene Sonarwerte von Saphira Sonar 6
  float saph_sonar7; // ausgelesene Sonarwerte von Saphira Sonar 7
  float saph_comp; // ausgelesener Kompasswert von Saphira
  float odomet_x; // Ausgabewerte des Odometriemoduls Pos.-> X
  float odomet_y; // Ausgabewerte des Odometriemoduls Pos.-> Y
  float odomet_th; // Ausgabewerte des Odometriemoduls Pos.-> TH
  ```

```
float odo_sigma_x; // Varianz X  vom Odometriemodul
float odo_sigma_y; // Varianz Y  vom Odometriemodul
float odo_sigma_th; // Varianz Th vom Odometriemodul
float covar_old[3][3]; // berechneten Kovar.Matrix vom Odometrie-Modul
float covar_new[3][3]; // neue Kovar.Matrix berechnet vom 2. Fus.Modul
float compass; // Ausgabewert des Kompassmoduls
float comp_sigma; // Varianz des Kompasswertes vom Kompassmodul
float fusion1_x; // Ausgabe des ersten Fusionsmoduls Pos.-> X
float fusion1_y; // Ausgabe des ersten Fusionsmoduls Pos.-> Y
float fusion1_th; // Ausgabe des ersten Fusionsmoduls Pos.-> TH
float fus1_sigma_x; // Varianz der Fusion X-Wert
float fus1_sigma_y; // Varianz der Fusion Y-Wert
float fus1_sigma_th; // Varianz der Fusion Th-wert
int   pos_bild_sonar[POS_BILD][POS_BILD]; // ,,Positionsbild'' des Sonarmoduls
int   pos_bild_kam[POS_BILD][POS_BILD]; // ,,Positionsbild'' des Kameramoduls
float sonar_angle_array[POS_BILD][POS_BILD]; // ,,Winkel-Array'' des Sonar-Moduls
float kam_angle_array[POS_BILD][POS_BILD]; // ,,Winkel-Array'' des Kamera-Moduls
int   karten_bild[POS_BILD][POS_BILD]; // Array fuer Karte (Saphira -> Sonar-Modul)
int   sonar_start_x;   // Startposition des Sonar-Moduls -> X Wert
int   sonar_start_y; // Startposition des Sonar-Moduls -> Y Wert
float fusion2_x; // Ausgabe des zweiten Fusionsmoduls Pos.-> X
float fusion2_y; // Ausgabe des zweiten Fusionsmoduls Pos.-> Y
float fusion2_th; // Ausgabe des zweiten Fusionsmoduls Pos.-> TH
int   set_pos_flag; // Flag fuer versetzen des Roboters in der Karte
} sharedmem_struct;
```

Des Weiteren werden mit der Funktion `sfAddEvalVar()` Variablen festgelegt, die dem gemeinsamen Speicher zugeordnet sind und von Saphira aus genutzt werden können. Diese werden genutzt, um mit Saphira zu kommunizieren.

13.2 Die Zentrale Steuerung

Die *Zentrale Steuerung* ist die Komponente des Selbstlokalisierungsmoduls, welche zum aktivieren und zum steuern der einzelnen Module gedacht ist. Von hieraus lassen sich zum Beispiel alle Module starten. Eine Darstellung der Windowsanwendung ist in der folgenden Abbildung zu sehen.

Abbildung 41: Die zentrale Steuerung des Selbstlokalisierungsmoduls.

Gestartet wird die zentrale Steuerung durch Aufruf der Datei `center.exe`. Der Button `[Saphira]` startet die Saphira-Software, der Button `[Pioneer-Simulator]` den Simulator. Mit Hilfe der Buttons `[Odometrie-Modul]`, `[Kompass-Modul]`, `[1. Fusions-Modul]`, `[Sonar-Modul]`, `[Kamera-Modul]` und `[2. Fusions-Modul]` lassen sich die jeweiligen Komponenten des Selbstlokalisierungsmoduls starten. Der Button `[Start All]` starten alle Module des Selbstlokalisierungsmoduls. Durch den Button `[Reset All]` wird eine Nachricht an alle Module gesendet, die diese veranlaßt sich in den Startzustand zurückzusetzen. Mit dem Button `[Close All]` werden allen Fenster des Selbstlokalisierungsmoduls bis auf die zentrale Steuerung geschlossen.

13.3 Das Odometrie-Modul

Das *Odometrie-Modul* hat die Aufgabe, wie bereits in der Konzeption erwähnt, die Expertenmeinung der Odometrie zu bilden. Auch dieses Modul ist als eine WIN32-Anwendung implementiert und wird durch Aufruf der Datei `odometrie.exe` gestartet. Das Anwendungsfenster des Odometrie-Moduls ist nachfolgend zu sehen.

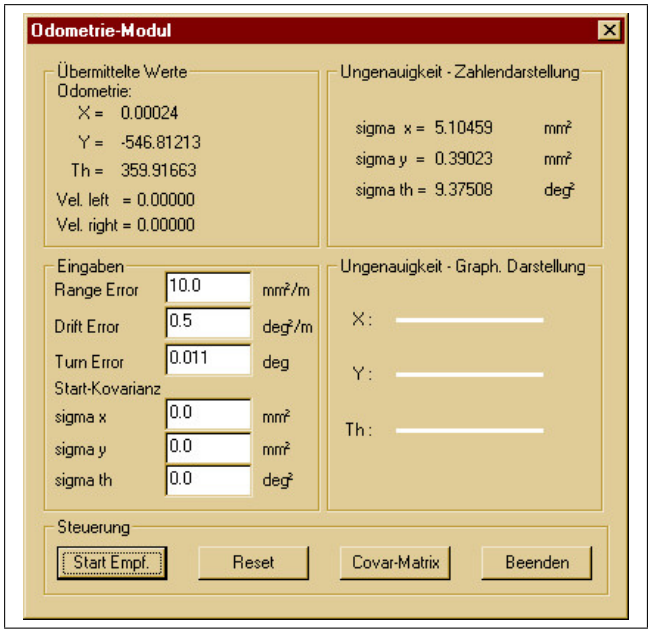

Abbildung 42: Das Anwendungsfenster des Odometrie-Moduls.

Als Kern dieses Moduls arbeitet der in der Konzeption ausführlich dargestellte erweiterte Kalman Filter nach Konolige, kurz Konolige EKF. Dieser ist in der Funktion

- `int Odometrie_EKF(double theta, double s, double alpha)`

implementiert. Aus Saphira bzw. aus dem gemeinsamen Speicher werden dazu die aktuelle Position und die aktuelle Geschwindigkeit gelesen. Wenn der gemeinsame Speicher eingerichtet und durch das Odometrie-Modul initialisiert wurde (mit Hilfe der Funktion `Init_Shared_Mem()`), dann wird ein Thread ausgeführt, der die Berechnungsfunktion permanent aufruft.
Das Odometrie-Modul gibt im Anwendungsfenster die Hauptdiagonale der aktuell berechneten Kovarianzmatrix aus, also die Werte σ_x, σ_y und σ_θ. Zusätzlich erfolgt eine graphische Ausgabe in Form von Balkendiagrammen.
Der Button `[Covar-Matrix]` öffnet ein Fenster, in dem die vollständige, aktuell berechnete, Kovarianzmatrix angezeigt wird. Nachfolgend ist dieses Fenster dargestellt.

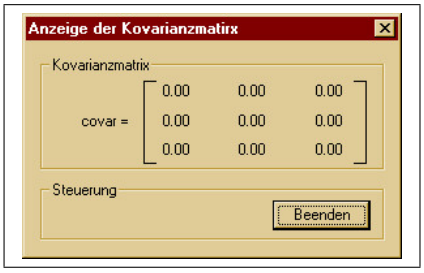

Abbildung 43: Die Anzeige der kompletten Kovarianzmatrix des Odometrie-Moduls.

Der Button `[Reset]` dient zum Rücksetzen des Odometrie-Moduls, daß heißt die Kovarianzmatrix wird auf Null zurückgesetzt. Hierbei werden aber die Werte der Eingabefelder für die Start-Kovarianz berücksichtigt.
Anschließend hat die Kovarianzmatrix die Form $P = \begin{pmatrix} \sigma_{xStart} & 0 & 0 \\ 0 & \sigma_{yStart} & 0 \\ 0 & 0 & \sigma_{\theta Start} \end{pmatrix}$.

Im laufenden ,,Selbstlokalisierungsmodul" erfolgt ein Update der Kovarianzmatrix mit Hilfe der Funktion `Read_New_Covar()`. Diese liest die vom zweiten Fusionsmodul neuberechnete Kovarianzmatrix ein. Die Ausgabe der aktuellen Kovarianzmatrix übernimmt die Funktion `Write_Old_Covar()`.
Mit Hilfe des Buttons `[Start Empf.]` bzw. `[Ende Empf.]` wird der Thread, der Daten aus dem gemeinsamen Speicher liest und die Odometrie-EKF Berechnung anstößt, gestartet bzw. gestoppt. Damit ist auch die Möglichkeit gegeben, im laufenden ,,Selbstlokalisierungsmodul" die Arbeit zu stoppen und Werte in Ruhe auszuwerten.

13.4 Das Kompass-Modul

Für die Verarbeitung der Kompasswerte ist das sogenannte *Kompass-Modul* zuständig. Dieses liest aus dem gemeinsamen Speicher den aktullen Kompasswert von Saphira. Wie alle anderen Module, ist auch dieses Modul als WIN32-Anwendung implementiert und stellt einen Dialog dar. Gestartet wird dieses Modul durch Aufruf der Datei `kompass.exe`. Die folgende Abbildung zeigt das Anwendungsfenster des Kompass-Moduls.

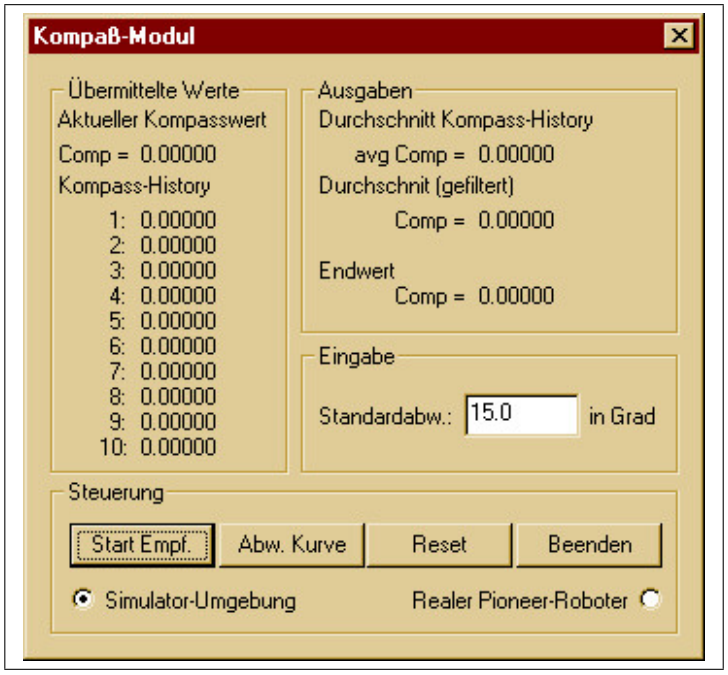

Abbildung 44: Das Anwendungsfenster des Kompass-Moduls.

Die Kernfunktion, die Funktion, die die Expertenmeinung des Kompass-Moduls bildet ist die Funktion:

- `int Compass_Calculation(void *dummy)`.

Hier erfolgt das Erstellen der Kompasshistory, die Aussonderung von ungültigen Werten und die endgültige Ausgabe. Das Kompass-Modul kann sowohl mit dem Pioneer-Simulator als auch mit dem realen Roboter betrieben werden. Diese Einstellung erfolgt mit den Radiobuttons `( ) Simulator-Umgebung` bzw.
`( )Realer Pioneer-Roboter`. In der Simulator-Umgebung wird der Kompasswert mit normalverteilten Fehler angenommen. Wird der reale Roboter eingesetzt, so wird die in der Studienarbeit,[GL00], aufgenommene Abweichungskurve zur Bestimmung des endgültigen Wertes benutzt.
Der Button `[Abw. Kurve]` öffent ein Fenster zur Darstellung der Abweichungskurve. Dieses ist in der folgenden Abbildung zusehen.

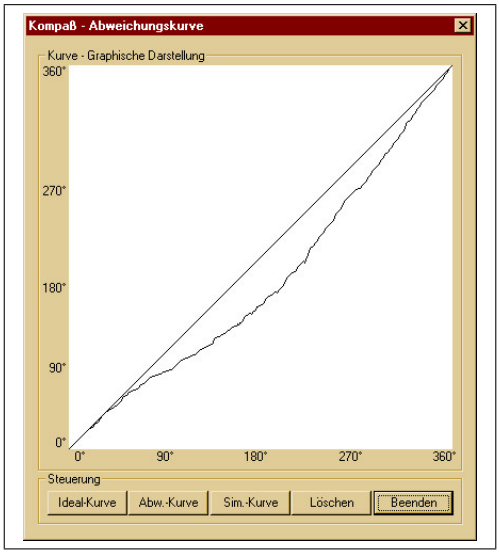

Abbildung 45: Das Abweichungskurvenfenster des Kompass-Moduls.

Mit dem Button `[Abw.-Kurve]` wird die Abweichungskurve graphisch dargestellt. Diese wird aus der Datei `comp.dat` geladen. Des Weiteren kann mit den Buttons `[Ideal-Kurve]` und `[Sim.-Kurve]` die jeweilige graphische Ausgabe veranlaßt werden. Der Button `[Loeschen]` dient lediglich zum löschen des graphischen Bereichs für die Kurvenausgabe.
Der Button `[Reset]` dient zum Zurücksetzen des Kompass-Moduls in den Startzustand. Hierzu wird unter anderem die Kompasshistory gelöscht.
Mit Hilfe des Buttons `[Start Empf.]` bzw. `[Ende Empf.]` wird der Thread, der die Berechnungen und den Datenaustausch durchführt, gestartet bzw. gestoppt.

13.5 Das Erste Fusions-Modul

Die Fusion der Primärsensoren erfolgt im sogenannten *Ersten Fusions-Modul*. Konkret werden hier die Werte des Odometrie-Moduls und des Kompass-Moduls fusioniert. Auch dieses Modul ist als WIN32-Anwendung implementiert. Gestartet wird es durch Aufruf der Datei `fusion_1.exe`. Das Anwendungsfenster des ersten Fusions-Moduls ist in der folgenden Abbildung dargestellt.

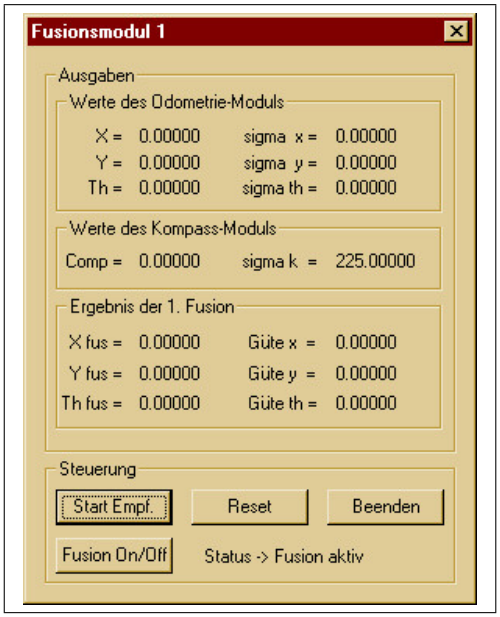

Abbildung 46: Das Anwendungsfenster des Ersten Fusions-Moduls.

Die Kernfunktion dieses Moduls ist die Funktion:

- `int Fusions_1()`.

Hier erfolgt die Fusion, wobei hier drei Fusionen zum Einsatz kommen. Dies sind die X-Fusion, die Y-Fusion und die θ-Fusion.
Der Button `[Fusion On/Off]` dient zum Ein- bzw. Ausschalten der Fusion. Ist die Fusion ausgeschaltet, so werden die vom Odometrie-Modul gelieferten Werte als Ausgabe durchgeschleift. Im eingeschaltetem Zustand wird eine Fusion nach dem Prinzip der Verknüpfung von zwei Zufallsvariablen durchgeführt. So wird der Ausgabewinkel $\theta_{Fusion1}$, des ersten Fusions-Moduls wie folgt berechnent:

$$\theta_{Fusion1} = \frac{\sigma^2_{Komp}}{\sigma^2_{Odomet} + \sigma^2_{Komp}} \cdot \theta_{Odomet} + \frac{\sigma^2_{Odomet}}{\sigma^2_{Odomet} + \sigma^2_{Komp}} \cdot \theta_{Komp}$$

Die Güte des Ausgabewinkels wird wie folgt ermittelt:

$$Güte_{\theta_{Fusion1}} = \frac{\sigma^2_{Odomet} \cdot \sigma^2_{Komp}}{\sigma^2_{Odomet} + \sigma^2_{Komp}}$$

Der Button `[Reset]` dient zum Zurücksetzen des Ersten Fusions-Moduls.
Der Button `[Start Empf.]` bzw. `[Ende Empf.]` startet oder stoppt den Thread, der für die Berechnung der Fusionswerte verantwortlich ist.

13.6 Das Kamera-Modul

Das sogennannte *Kamera-Modul* ist in dieser Arbeit nur ein ,,Dummy-Modul". Das heißt, es verarbeitet nicht wirklich die Informationen der Kamera, sondern liefert zufällige Werte für die zweite Fusion. Dieses Modul ist ebenfalls eine WIN32-Anwendung und wird durch Aufruf der Datei `kamera.exe` gestartet. Das Anwendungsfenster des Kamera-Moduls ist in der folgenden Abbildung zu sehen.

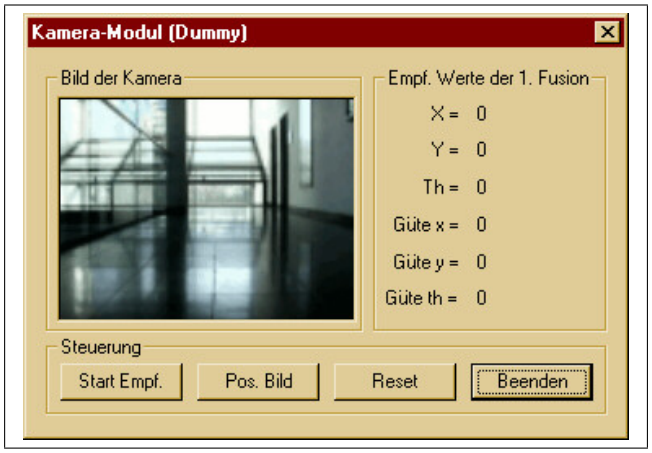

Abbildung 47: Das Anwendungsfenster des Kamera-Moduls.

Als Kernfunktion dieses Moduls soll die Funktion:

- `int Cal_Kamera()`

arbeiten. Sie hat die Aufgabe das Bild der Kamera auszuwerten und Positionsschätzungen zu liefern. Diese Positionsschätzungen werden anschließend in das sogenannte Positionsbild eingetragen. Zur Anzeige dieses Positionsbildes dient der Button `[Pos. Bild]`. In der folgenden Abbildung ist das Positionsbildfenster des Kamera-Moduls dargestellt.

Abbildung 48: Das Positionsbildfenster des Kamera-Moduls.

Die Positionsschätzungen werden als Grauwertpunkte in das Bild eingetragen. Wobei dieser Grauwert ein Maß für die Unsicherheit ist. Ist eine Positionsschätzung sehr sicher, dann wird ein weißer Punkt eingetragen. Dieses Positionsbild ist die Ausgabe des Kamera-Moduls. Zusätzlich wird ein sogenanntes Winkelarray ausgegeben, dieses ermöglicht es, zu jeder Positionsschätzung einen Winkel anzugeben.
Der Button `[Reset]` dient zum Zurücksetzen des Kamera-Moduls in den Startzustand.
Mit dem Button `[Start Empf.]` bzw. `[Ende Empf.]` kann der Thread, der das Positionsbild generiert und das Winkelarray füllt gestartet bzw. gestoppt werden.

13.7 Das Sonar-Modul

Das *Sonar-Modul* dient zur Verarbeitung der von Saphira gelieferten Sonarinformationen. Es bildet die Expertenmeinung des Sonars. Dieses Modul ist eine WIN32-Anwendung und wird durch Aufruf der Datei `sonar.exe` gestartet. Das Anwendungsfenster des Sonar-Moduls ist in der folgenden Abbildung zu sehen.

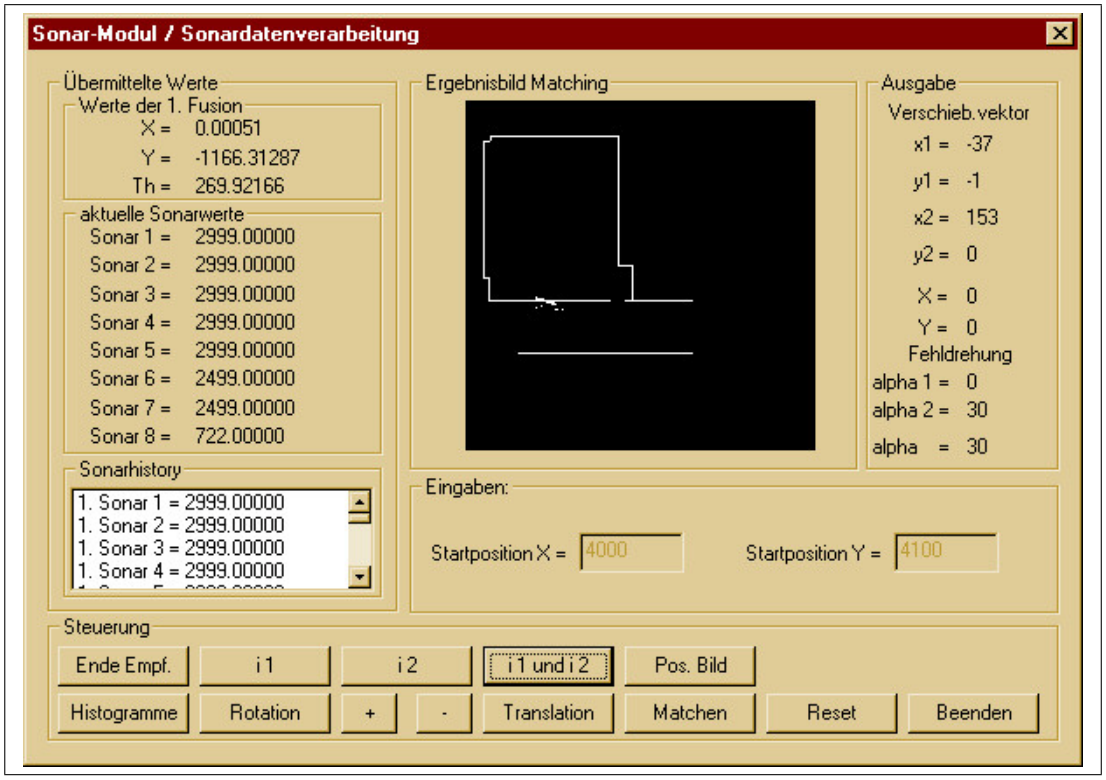

Abbildung 49: Das Anwendungsfenster des Sonar-Moduls.

Als Kernfunktion dieses Moduls kann die Funktion:

- `Cal_Match()`

angesehen werden. Diese führt den sogenannten Matchingprozess durch. Der Matchingprozess erfolgt in mehreren Phasen. In der ersten wird das Bild rotiert, bis die beste Übereinstimmung erreicht ist. Hierzu dient die Funktion `Cal_Rotation()`. In der zweiten Phase wird das Bild verschoben, bis wieder eine beste Übereinstimmung erreicht ist. Dazu wird die Funktion `Cal_Translation()` aufgerufen. Anschließend erfogt eine endgültige Bestimmung der Roboterposition. Diese Position, daß heißt die X Werte und Y Werte werden in das Positionsbild des Sonar-Modul eingetragen, welches die Ausgabe des Sonar-Moduls ist. Zusätzlich wird im Winkelarray des Sonar-Moduls zu jeder ermittelen X, Y Komponente ein Winkel ausgegeben. Das Winkelarray und das Positionsbild dienen als Eingabe für das Zweite Fusions-Modul.

Das Positionsbild kann mit Hilfe des Buttons `[Pos. Bild]` angezeigt werden. Ein Beispiel für ein solches Positionsbild vom Sonar-Modul ist in der folgenden Abbildung zu sehen.

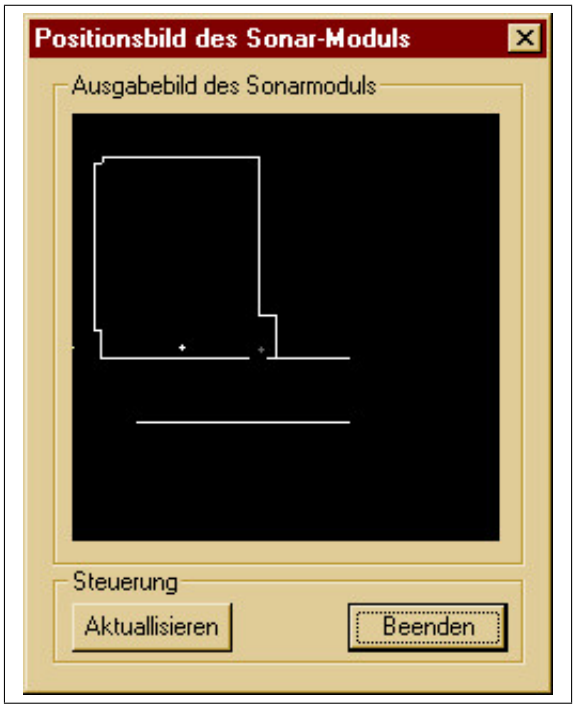

Abbildung 50: Ein Positionsbild vom Sonar-Modul.

Die eingetragenen Grauwert Punkte stellen die Positionsunsicherheit dar. Der weiße Punkt ist die wahrscheinlichste Positionsschätzung des Sonar-Moduls.
Um sich die aktuelle Scanhistory, die mit der Karte gematcht werden soll, anzeigen zu lassen, dient der Button `[i 1]`. Die Karte kann, im graphischen Ausgabebereich, mit dem Button `[i 2]` angezeigt werden. Eine Überblendung von Scan und Karte wird mit Hilfe des Buttons `[i1 und i2]` angezeigt. Diese Bezeichnungen sind analog zur Konzeption gewählt worden.
Der Button `[Reset]` dient zum Zurücksetzen des Sonar-Moduls in den Startzustand. Dies führt unter anderem zum Löschen der sogenannten Scanhistory. Der Button `[Start Empf.]` bzw. `[Ende Empf.]` dient zum Starten oder Stoppen der Arbeit des Sonar-Moduls. Durch einen Stopp im laufenden Selbstlokalisierungsmodul ist die

Gelegenheit gegeben, Werte in Ruhe auszuwerten oder zur Veranschaulichung den Matchingprozess manuell durchzuführen. Dazu dienen die folgenden Buttons.
Der Button `[Rotation]` ruft die Funktion `Cal_Rotation()` auf. Als Ausgabe kann die errechnete Fehldrehung beobachtet werden. Die Buttons `[+]` und `[-]` ermöglichen ein manuelles Drehen der aktuellen Scanhistory, um jeweils 5 Grad. Die Scanhistory wird dabei um die aktuelle Position des Roboters gedreht. Der Button `[Translation]` dient zur Ermittlung der Verschiebung, hierbei wird die Funktion `Cal_Translation()` aufgerufen. Anschließend kann der errechnete Verschiebungsvektor betrachtet werden. Mit Hilfe des Buttons `[Matchen]` wird der Matchingprozess manuell ausgelöst, daß heißt es werden die Funktionen `Cal_Rotation()` und `Cal_Translation()` aufgerufen.
Zur Verdeutlichung der Arbeitsweise des Sonar-Moduls, bzw. des Matchingprozesses, dient der Button `[Histogramme]`. Dieser öffnet das Histogrammfenster des Sonar-Modul. In der folgenden Abbildung ist dieses Fenster dargestellt.

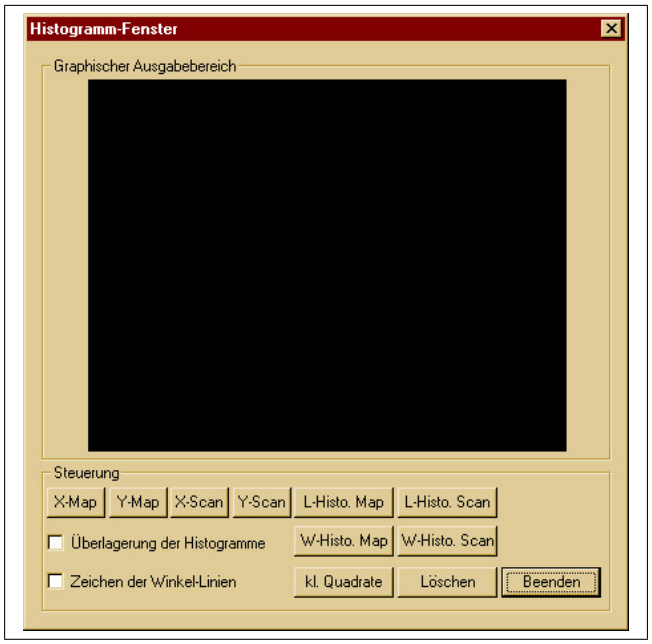

Abbildung 51: Das Histogrammfenster des Sonar-Moduls.

Die Buttons `[X-Map]`, `[Y-Map]`, `[X-Scan]` und `[Y-Scan]` dienen zur Anzeige der

jeweiligen X- bzw. Y-Histogramme der Karte und des Scans. Das heißt das Bild wird spalten- bzw. zeilenweise durchlaufen und vorhandene Einträge gezählt. Dieses Verfahren ist aber nur unter der Bedingung rechtwinkliger Strukturen gültig.
Die Buttons `[L-Histo. Map]` und `[L-Histo. Scan]` dienen zur Erzeugung und Anzeige der Längenhistogramme von Karte und Scan. Hierbei werden von der aktuellen Position des Roboters Strahlen ausgesendet und die Entfernung bis zum Scaneintrag oder Karteneintrag gemessen. Diese Histogramme werden aber nicht für den Matchingprozess genutzt.
Die Buttons `[W-Histo. Map]` und `[W-Histo. Scan]` erzeugen die Winkelhistogramme der Karte und des Scans und zeigen diese im Histogrammfenster an. Bei einem Winkelhistogramm werden Strahlen von der aktuellen Roboterposition ausgesendet und der Winkel zwischen zwei Eintragungen der Karte bzw. im Scan bestimmt. Auch dieses Verfahren hat sich für den Matchingprozess nicht durchgesetzt.
Der Button `[kl. Quadrate]` ist ebenfalls experimentell. Dieses Verfahren hat keinen Einfluß auf den Matchingprozess. Hier wird die Methode der kleinsten Quadrate nach Gauss, auf die Längenhistogramme der Karte und des Scan angewendet.
Die angeklickte Checkbox `[ ] Ueberlagerung` führt zur Überblendung der Histogramme, so kann zum Beispiel das X-Histogramm der Karte und des Scan gleichzeitig betrachtet werden.
Die gesetzte Checkbox `[ ] Zeichnen der Winkel-Linien` dient zum besseren Lesen der Winkelhistogramme. Hierbei werden senkrechte Linien eingezeichnet und zwar an den Stellen 90 Grad, 180 Grad und 270 Grad.
Der Button `[Loeschen]` hat lediglich die Funktion, den graphischen Ausgabebereich des Histogrammfensters zu löschen.

Das vorliegende Sonar-Modul nutzt für den Matchingprozess nur die sogenannten X- / Y-Histogramme. Die Rotation, die Berechnung des Fehlwinkels, beruht auf dem Maximum im X-Histogramm oder im Y-Histogramm. Liegt zum Beispiel, wie in der folgenden Abbildung dargestellt, eine verdrehte Scanhistory vor, dann ergibt sich kein eindeutiges Maximum im X- oder Y-Histogramm.

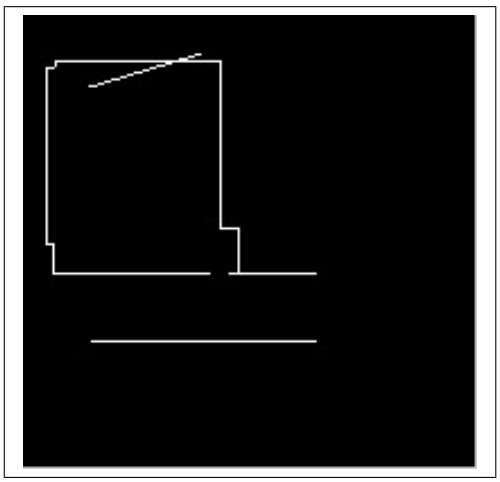

Abbildung 52: Zu matchende Scanhistory.

Die folgende Abbildung zeigt die Überblendung des X-Histogramms der Karte und des Scans. Außer den vier Picks der Karte kann kein Maximum festgestellt werden.

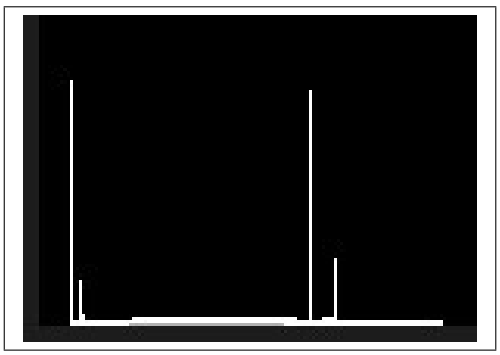

Abbildung 53: Überblendung X-Map und X-Scan.

Das gleich trifft für die Y-Histogramme der Karte und des Scans zu. Außer den Picks der Karte kann auch hier kein eindeutiges Maximum festgestellt werden. Die folgende Abbildung zeigt die Überblendung beider Histogramme.

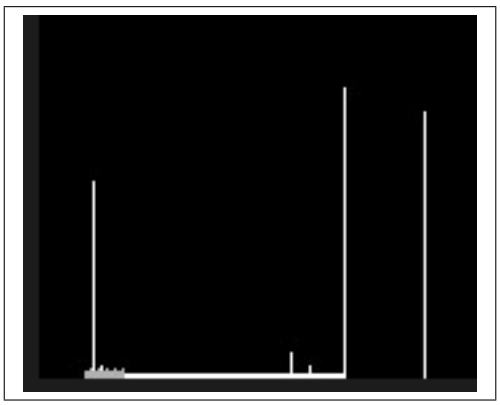

Abbildung 54: Überblendung Y-Map und Y-Scan.

Die Berechnung des Fehlwinkels erfolgt indem die aktuelle Scanhistory, jeweils in 5 Grad Schritten um die aktuelle Roboterposition gedreht wird. Dabei wird von -45 Grad bis +45 Grad gedreht und jeweils ein X-Histogramm und ein Y-Histogramm des Scans erstellt. Nach jeder Drehung werden die Maxima der Histogramme ermittelt. Der Fehlwinkel ist letztlich der, bei dem der größte maximale Wert festgestellt wurde. Dies ist in der Funktion `Cal_Rotation()` implementiert. Im dargelegten Beispiel ergibt sich im X-Histogramm kein Maximum. Dies ist in der folgenden Abbildung zu sehen.

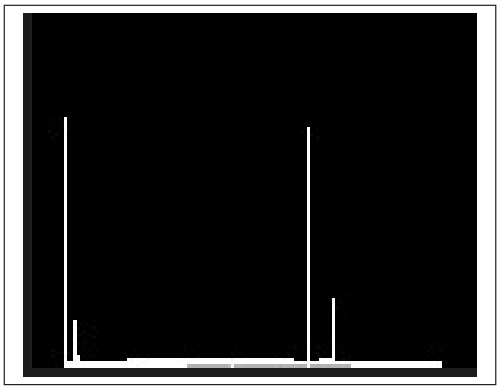

Abbildung 55: Überblendung X-Map und X-Scan nach der Rotationsberechnung.

Im Y-Histogramm des Scans ist dagegen ein deutliches Maximum zu erkennen. Der

weiße Pfeil zeigt auf dieses Maximum. Die folgende Abbildung zeigt die Überblendung der Y-Histogramm von Karte und Scan.

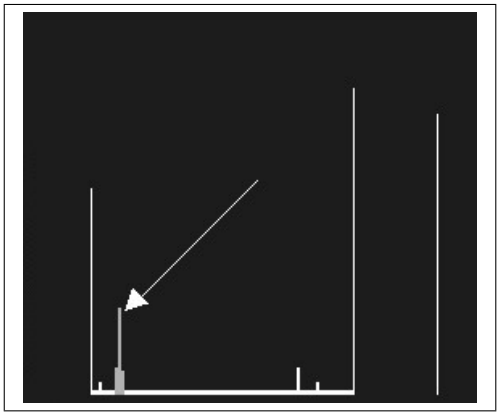

Abbildung 56: Überblendung Y-Map und Y-Scan nach der Rotationsberechnung.

Der somit ermittelte Winkel wird als Fehlwinkel im Sonar-Modul ausgegeben. Dieses Verfahren ist jedoch nur in der Lage Verdrehungen in 5 Grad Schritte zu bestimmen.

Nun kann die Berechnung der Translation erfolgen. Diese beruht, wie in der Konzeption erwähnt auf dem Verfahren der Kreuzkorrelation. Hierbei werden die X-Histogramme der Karte und des Scans, sowie die jeweiligen Y-Histogramme korreliert. Ein Bedingung dabei ist, daß das Histogramm des Scan ein Maximum auf weist. Das X-Histogramm des Scans, weist jedoch hier kein Maximum auf und somit ist auch keine Verschiebung in X Richtung notwendig.
Das Y-Histogramm des Scans besitzt wie schon gesagt ein Maximum, dieses wird durch die Kreuzkorrelation nach links auf das dort vorhandene Maximum verschoben. Die folgende Abbildung zeigt die Überblendung der Y-Histogramme der Karte und des Scans nach der Translationsberechnung.

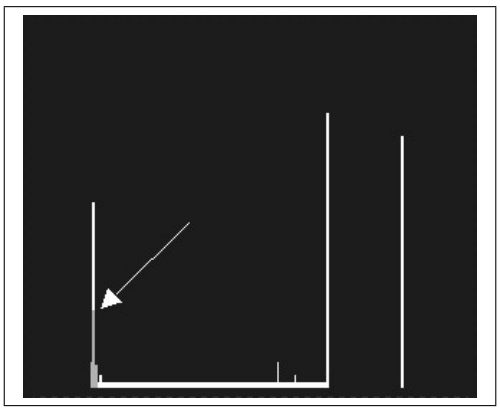

Abbildung 57: Überblendung Y-Map und Y-Scan nach der Translationsberechnung.

Die Verschiebung des Maximums des Y-Histogramms ist nochmals durch den weißen Pfeil veranschaulicht. Die ermittelten Werte der Translationsberechnung werden im Sonar-Modul als Verschiebungsvektor ausgegeben. Die Translationsberechnung ist in der Funktion `Cal_Translation()` implementiert.
Nach dem Matching, daß heißt nach Rotation und Translation, liegt nun folgendes Bild vor. Die Überblendung von gematchtem Scan und Karten nach dem Matching ist in der folgenden Abbildung zu sehen.

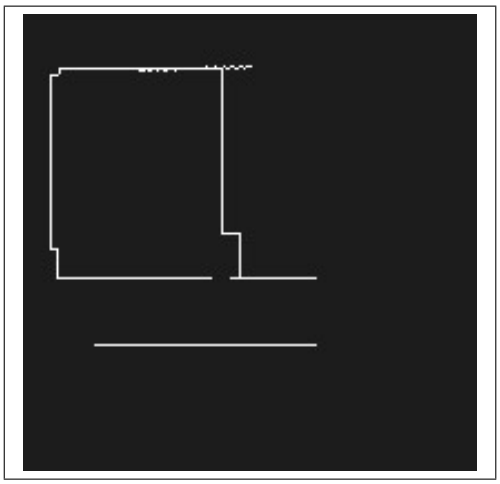

Abbildung 58: Scanhistory und Karte nach dem Matching.

Zusammenfassend ist festzustellen, daß durch die beschriebene Rotation und Translation ein Matching von Scan und Karte möglicht ist. Dieses wird im laufenden Sonar-Modul durch einen Thread in Intervallen durchgeführt. Es wird die Funktion `Cal_Match()` aufgerufen.

13.8 Das Zweite Fusions-Modul

Das *Zweite Fusions-Modul* ist verantwortlich für die Fusion der bildgebenden Sensoren. Für die vorliegende Implementierung sind dies konkret die Informationen des Sonar-Moduls und des Kamera-Moduls. Die WIN32-Anwendung wird durch Aufruf der Datei `fusion_2.exe` gestartet. Das Anwendungsfenster des Zweiten Fusions-Moduls ist in der folgenden Abbildung zu sehen.

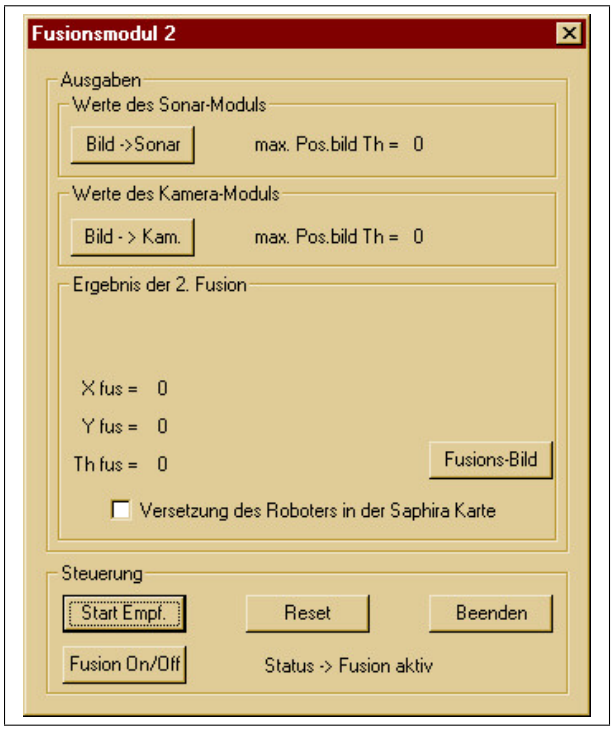

Abbildung 59: Das Anwendungsfenster des Zweiten Fusions-Moduls.

Die Kernfunktion dieses Moduls ist die Funktion:

- `int Fusion_2()`

Diese Funktion hat die Aufgabe, die endgültige Positionsschätzung des Selbstlokalisierungsmoduls auszugeben. Dazu erfolgt eine Fusion der sogenannten Positionsbilder vom Sonar-Modul und vom Kamera-Modul. Mit den Buttons `[Bild->Sonar]` und `[Bild->Kam.]` kann eine Anzeige der jeweiligen Positionsbilder erfolgen. Neben

diesen Buttons wird jeweils ein Winkel ausgegeben, der dem Winkel an der wahrscheinlichsten Stelle im Positionsbild, dem Maximum, entspricht. Ein Beispiel für ein Positionsbild vom Sonar-Modul ist in der folgenden Abbildung zu sehen.

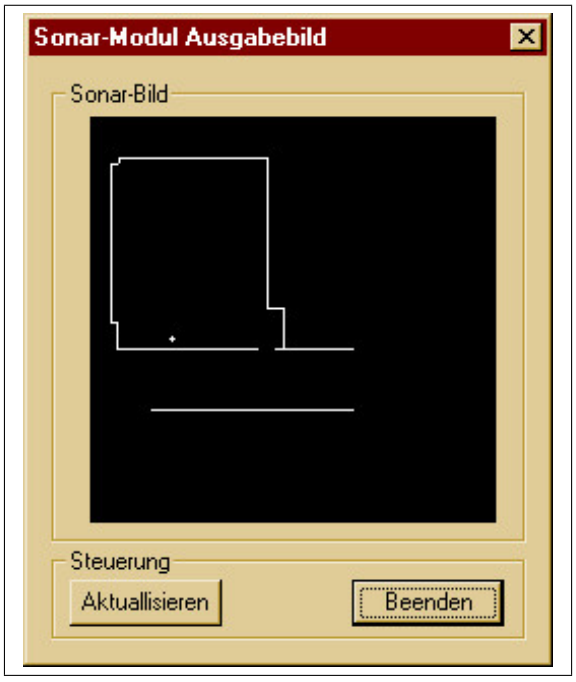

Abbildung 60: Ein Positionsbild vom Sonar-Modul.

Die folgende Abbildung zeigt ein vom Kamera-Modul generiertes Positionsbild. Die eingetragenen Grauwert Punkte sind die jeweiligen Positionsvorschläge.

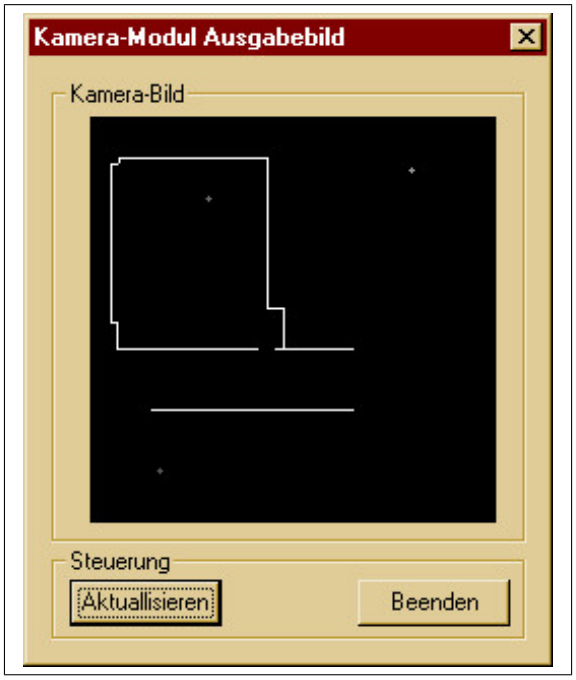

Abbildung 61: Ein Positionsbild vom Kamera-Modul.

Diese beiden Positionsbilder werden nun zu einem sogenannten Fusionsbild zusammengefasst. Dazu ruft die Funktion `Fusion_2()` die Funktion `Create_Fusions_Pic()` auf. Die Überdeckung beider Bilder erfolgt durch eine Addition. Ein Beispiel für ein Fusionsbild, das mit Hilfe des Buttons `[Fusions-Bild]` angezeigt werden kann, ist in der folgenden Abbildung zu sehen.

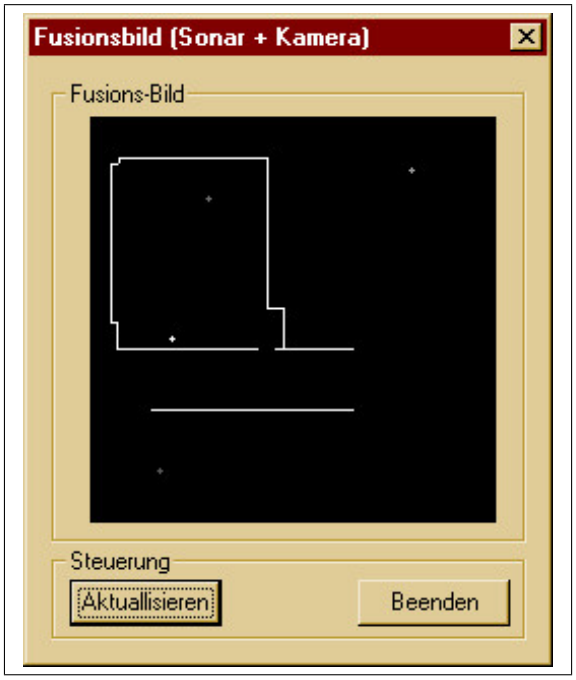

Abbildung 62: Ein Fusionsbild des Zweiten Fusions-Moduls.

Aus diesem Fusionsbild wird nun mit Hilfe der Funktion `Get_PosFusions_Pic()` die Ausgabeposition, daß heißt der X Wert und der Y Wert ermittelt. Hierzu wird in dem Fusionsbild eine Maximum Suche durchgeführt.
Die letzte Komponente, der Ausgabewinkel θ, wird durch die Fusion der gelieferten Winkelarrays ermittelt. Dazu wird die Funktion `Angle_Fusion()` aufgerufen. Hierbei werden die zuvor ermittelten Werte und die Tatsache, welches Positionsbild den stärkeren Einfluß hat, berücksichtigt.
Soll eine Versetzung des Roboters in der Saphira Karte erfolgen, so muß die Checkbox `[ ] Versetzen des Roboters` angeklickt sein. Dies führt zum Setzen des Flags `set_pos_flag`, welches vom Verhalten `set_pos()` ausgewert wird und so eine Versetzung veranlaßt.
Mit dem Button `[Fusion On/Off]` kann die Arbeitsweise des Zweiten Fusions-Moduls eingestellt werden. Im Zustand Fusion aktiv, werden die ermittelten Werte aus den Positionsbildern und den Winkelarrays als endgültige Ausgabe geliefert. Im inaktiven Zustand werden die Odometriedaten als Ausgabe durchgeschleift.

Der Button `[Reset]` dient, wie bei allen anderen Modulen, zum Zurücksetzen des Zweiten Fusions-Moduls in den Startzustand. Der Button `[Start Empf.]` bzw. `[Ende Empf.]` startet oder stoppt den Thread für die Berechnung der Fusionwerte.

14 Erweiterbarkeit

Die Aufgabenstellung dieser Diplomarbeit legt besonderen Wert auf die Erweiterbarkeit des in dieser Arbeit vorgestellten Selbstlokalisierungsmoduls. Ziel ist es möglichst problemlos neue Sensorquellen bzw. Sensor-Module in das System einzufügen. Dazu sind allgemein folgende Schritte notwendig.

- Klassifizierung des neuen Sensors
- Erstellung des Entsprechenden Sensor-Moduls
- Erweiterung des gemeinsamen Speichers
- Anschluß des neuen Sensor-Moduls an ein Fusionsmodul
- Neu compilieren alle Projekte des Selbstlokalisierungsmoduls

Klassifizierung heißt hier der neue Sensor muß einem Typ zugeordnet werden, entweder zu den Primärsensoren oder zu den bildgebenden Sensoren. Ein denkbarer neuer Primärsensor wäre zum Beispiel ein Gyroskop, ein bildgebender Sensor könnte ein Laserscaner sein.

Als nächstes muß nun ein Sensor-Modul geschrieben werden, das die gelieferten Werte des Sensors verarbeitet und ein Positionsschätzung ausgibt. Es sind entsprechende Schnittstellen zu implementieren. Das Modul kann sich, wie die hier gezeigten Module als Windowsanwendung mit Fenster darstellen oder auch als Dynamic Link Libaray.

Das neue Sensor-Modul muß in der Lage sein mit anderen Modulen zu kommunizieren, dazu muß der gemeinsame Speicher entsprechend erweitert werden. In der Headerdatei `self.h` ist die Struktur `sharedmemstruct` entsprechend zu bearbeiten.

Je nach Klassifizierung ist das neue Sensor-Modul an das Erste bzw. Zweite Fusions-Modul anzuschließen. Für das Gyroskop müßte zum Beispiel die θ-Fusion des Ersten Fusions-Modul angepasst werden. Das Ausgabebild des Laserscanners muß in die Bildfusion des Zweiten Fusions-Moduls einbezogen werden.

Nach dem Erstellen des neuen Sensor-Moduls und der Modifikation der entsprechenden Projekte des Selbstlokalisierungsmoduls, müssen alle Projekte, auch die nicht modifizierten, neu erstellt bzw. compiliert werden. Dies ist notwendig, da die Struktur des gemeinsamen Speichers verändert wurde und sich somit Adressen verschoben haben. Als letzter Schritt bleibt dann nur noch der Test des gesamten Selbstlokalisierungsmoduls, jetzt mit einem neuen Sensor-Modul.

15 Test und Evaluation

Zum Testen des hier vorgestellten Sebstloklisierungsmoduls wurde ein Verhalten, das Verhalten `drive.act`, geschieben. Dieses veranlaßt, daß der Roboter eine Strecke auf und abfährt. Praktisch wurde dies mit dem Pioneer 2 CE Roboter ,,Romeo" vor den Schränken im KI-Labor durchgeführt. Die folgende Abbildung veranschaulicht diesen Sachverhalt.

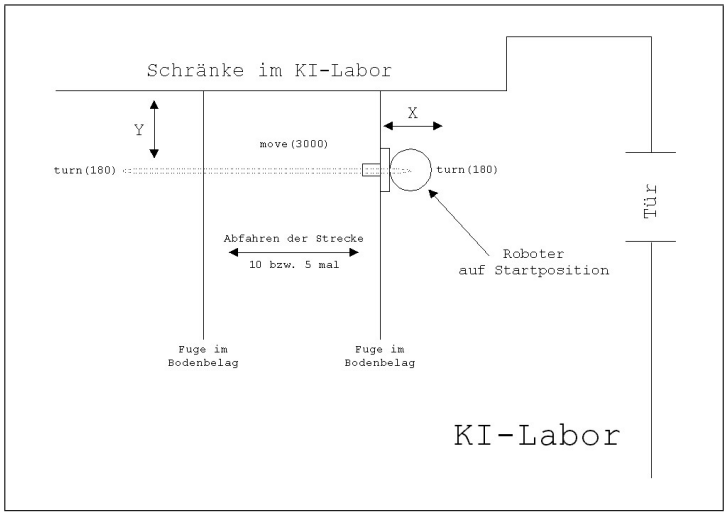

Abbildung 63: Test des Sebstlokalisierungsmoduls mit dem Roboter ,,Romeo".

Bei dem Abfahren der Streck verdriftet der Roboter langsam, so zum Beispiel durch die 180 Gard Wende am Ende der Strecke. Aufgabe des Selbstlokalisierungsmoduls ist nun, die Kontrolle über die Position des Roboters zu behalten.

Hierbei wurde das Selbstlokalisierungsmodul parallel zu Saphira aktiviert. Das heißt es wurden alle Module gestartet, es erfolgte jedoch keine Versetzung des Roboters in der Saphira Karte. Es mußte bei den Tests festgestellt werden, daß das vorliegende Selbstlokalisierungsmodul noch nicht in der Lage ist, den oben genannten Anspruch zu erfüllen. Dies liegt vor allem an dem Sonar-Modul. Dieses arbeitet aufgrund der Auflösung des Positionsbildes nur mit einer Genauigkeit von ± 100mm und das Matching schlägt in einigen Fällen fehl. Das Sonar-Modul gibt unter Umständen also falsche Werte aus. Das Zweite Fusions-Modul kann, aufgrund des fehlenden Kamera-Moduls oder anderer Quellen, dies nicht korrigieren. Demzufolge wird die Roboterposition durch das Selbstlokalisierungsmodul falsch beurteilt und durch die Auflösungsbegrenzung unter Umständen verschlechtert.
Bei den Test zeigte sich aber, daß die Fusion der Primärsensoren sehr wirkungsvoll ist. So konnte zum Beispiel, der durch die Abweichungskurve bestimmte Kompasswert, gut zur Positionsverbesserung beitragen. Die Werte der Ersten Fusion wurden jedoch durch das Sonar-Modul wieder verfälscht. Es hat sich gezeigt, daß das Sonar-Modul relativ gut die Winkelabweichung ermittelt. Die Translation , besonders in X Richtung, schlägt hier fehl.

16 Zusammenfassung und Ausblick

Zusammenfassend läßt sich sagen, daß die vorliegende Arbeit sich mit der Navigation mobiler Roboter, hier besonders mit der Selbstlokalisierung und mit Fusionstechniken befaßt hat. Im Ergebniss wurde eine *Fusionsarchitektur* entwickelt und praktisch umgesetzt. Hierbei wurde auf die Punkte intelligente Verknüpfung und Erweiterbarkeit des Systems wertgelegt.
Das vorliegende Selbstlokalisierungsmodul ist noch nicht vollständig ausgereift, wie die Tests gezeigt haben. Es bleibt jedoch festzuhalten, daß die Fusion von Sensoren bzw. Sensor-Modulen, wie in dieser Arbeit beschrieben, ein großes Potential besitzt. Ansatzweise konnte dieses Potential, wie bei der θ Fusion des Ersten Fusions Mo-

duls, aufgezeigt werden. Im Ganzen bleibt dieses Potential des Selbstlokalisierungsmoduls in der vorliegenden Version verborgen. Dies liegt vor allem an der einfachen Implementierung der einzelen Module. In dieser Arbeit wurde nicht Wert auf ein bestimmtes Modul gelegt, sondern auf die Funktion des Ganzen. Somit ist die vorhandene Implementierung eher eine Funktionsskizze als ein praktisch einsetzbares System.
Als Ausblick sei gesagt, das es noch jede Menge zu tun und zu implementieren gibt. Jedes der aufgezeigten Modul kann in geeigneter Weise überarbeitet und erweitert werden.
Wenn dieses Selbstlokalisierungsmodul einmal ausgereift und getestet ist, dann könnte es als Dynamic Link Libaray verpackt, unauffällig seinen Dienst tun. Windowsfenster, wie in dieser Arbeit, könnten dann verborgen bleiben. Diese sind jetzt jedoch zur Veranschaulich der Arbeitsweise, zur Steuerung und zur Kontrolle von Werten notwendig.
Das Modul, welches den größten Umfang aufweist und trotzdem noch nicht zu friedenstellend arbeitet, ist das Sonar-Modul. Dieses Modul sollte als nächster Schritt, im Zusammenhang mit dem Selbstlokalisierungsmodul, bearbeitet werden. Wenn dieses Modul zuverlässige Positionsschätzungen liefert, ist das Selbstlokalisierungsmodul bereits in der Lage, genauer zu arbeiten. Das Kamera-Modul ist für die zweite Fusion dringend notwendig. Jedoch stellt dieses Modul eine besondere Herausforderung dar, wenn es in der Lage sein soll, an einer beliebigen Position des Roboters eine Positionsschätzung vorzunehmen und nicht auf vordefinierte Landmarken angewiesen ist.
Je mehr Sensorquellen an das Selbstlokalisierungsmodul angeschlossen werden, desto genauer und zuverlässiger kann es arbeiten. Ziel muß es daher für die Zukunft sein, so viele Sensor-Module wie möglich in das Selbstlokalisierungsmodul zu integrieren. Bereits erwähnte neue Sensor-Module könnten ein Gyroskop-Sensor-Modul oder ein Transponder-Sensor-Modul sein.

Literatur

[AM] *ActiveMedia*. http://www.activrobots.com.

[Asi97] Isaac Asimov. *Meine Freunde die Roboter*. Band I, Wilhelm Heyne Verlag München, 1997.

[Bar88] Hans-Jochen Bartsch. *Mathematische Formeln*. 15. Auflage, Buch- und Zeit-Verlagsgesellschaft Köln, 1988.

[Bar97] Hans-Jochen Bartsch. *Taschenbuch mathematischer Formeln*. 17., neubearbeitete Auflage, Fachbuchverlag Leipzig im Carl Hanser Verlag, 1997.

[BGR98] BGR. *Referat B3.21: Seismische Meßverfahren, Methodenentwicklung: Kalman-Filter*. Bundesanstalt für Geowissenschaften und Rohstoffe, http://bzax04.bgr.de/b321/html/b321e1.html, 1998.

[BHJ+00] I. Boersch, J. Heinsohn, K.-H. Jänicke, H. Loose, F. Mündemann und H. Kanthack. *Vorlesung „Applikationen Intelligenter Systeme"*. Lehrveranstaltungsbegleitendes Material, Fachhochschule Brandenburg, 2000.

[Bor91] Johann Borenstein. *Navigating mobile Robots, Systems and Techniques*. A. K. Peters, Ltd., Wellesley, MA, 1991.

[BW00] Gary Bishop und Greg Welch. *An Introduction to the Kalman Filter*. Department of Computer Science University of North Carolina, pdf, 2000.

[CBM] *Markov Localization using Correlation*. Content Areas: robotics, perception, Tracking Number: A595, pdf.

[Dod98] Tony Dodd. *An Introduction to Multi-Sensor Data Fusion*. Department of Electronics & Computer Science University of Southampton, 1998.

[Dos] Prashant Doshi. *A Brief Introduction to Stochastic Processes*. Drexel University.

[DUD94] DUDEN. *Das Große Fremdwörterbuch, Herkunft und Bedeutung der Fremdwörter*. DUDENVERLAG Mannheim Leipzig Wien Zürich, 1994.

[FHG] *Robotersysteme.* http://www.ipa.fhg.de/300/320/Robotersysteme.php3.

[Fin00] Klaus Finkenzeller. *RFID-Handbuch, Grundlagen und praktische Anwendung induktiver Funkanlagen, Transponder und kontakloser Chipkarten.* 2. Auflage Carl Hanser Verlag München Wien, 2000.

[Fra98] Elena Franzini. *Was ist der Kalman Filter ?, Eine kurze Einleitung.* Zentralanstalt für Meteorologie und Geodynamik Wien, http://www.zamg.ac.at/wetter/modell/Kalmanfilter.html, 1998.

[GBFK] Jens-Steffen Gutmann, Wolfram Burgrad, Dieter Fox und Kurt Konolige. *An Experimental Comparison of Localization Methods.* pdf.

[GG99] Jörg Grawe und Oliver Groht. *Entwicklung und Realisierung eines Garbage-Collecting Serviceroboters für partiell bekannte Räume.* Diplomarbeit an der FH Hamburg, pdf, 1999.

[GL00] Daniel Gülow und Sebastian Lindner. *Studienarbeit – Applikation Intelligenter Systeme, Abschlußaufgabe 8.* Fachhochschule Brandenburg, 2000.

[Gut96] Jens-Steffen Gutmann. *Vergleich von Algorithmen zur Selbstlokalisierung eines mobilen Roboters.* Diplomarbeit an der Universität Ulm, 1996.

[Hei99] Jochen Heinsohn. *Wissensverarbeitung, Eine Einführung.* Spektrum Akademischer Verlag Heidelberg Berlin, 1999.

[Hei00] Jochen Heinsohn. *R2-D2 im Kommen.* Märkische Allgemeine Zeitung, IQ Brandenbug, 9. November, 2000.

[Hop92] Peter Hoppen. *Autonome Mobile Roboter, Echtzeitnavigation bei autonomen mobilen Robotern in bekannter und unbekannter Umgebung.* BI-Wiss.-Verl., 1992.

[Kal99] *Rudolf Emil Kalman.* http://www.cs.unc.deu/~welch/kalmanBiblio.html bzw. http://www.cs.unc.edu/~welch/kalman/siam_sontag.html, 1999.

[Kon98] Kurt Konolige. *Geometry, Robot Motion, and Proprioception.* Project 1, Copyright by Kurt Konolige, 1998.

[Lev98] Larry J. Levy. *The Kalman Filter: Navigation's Integration Workhorse.* The Johns Hopkins University, Applid Physics Laboratory, http://www.gpsworld.com/columns/0997Innov/0997kalman.htm, 1998.

[LIB94] Lexikon-Institut-Bertelsmann. *Bertelsmann Lexikon in 15 Bänden.* Bertelsmann Lexikothek Verlag, 1994.

[LW91] Leonard und Durrant Whyte. *Mobile robots localization by tracking geometric beacons.* IEEE Trans. Robotics and Automation, 1991.

[May79] Peter S. Maybeck. *Stochastic models, estimation, and control.* Copyright ©1979, by Academic Press, Inc., pdf, 1979.

[Mor90] Hans Moravec. *Mind Children.* Der Wettlauf zwischen menschlicher und künstlicher Intelligenz, Hoffmann und Campe, 1990.

[Mü98] Michael Müller. *Navigation von autonomen mobilen Robotern.* Literatur zum Hauptseminar Mobile Roboter – Schwerpunkt Navigation, TU München, 1998.

[MZ] Alexander Mojaev und Andreas Zell. *Sonardaten-Integration für autonome mobile Roboter.* Universität Tübingen, Wilhelm-Schickard-Institut für Informatik, Lehrstuhl Rechnerarchitektur.

[Rom96] Erich Rome. *Einführungsvortrag Über Navigation autonomer, mobiler Roboter (AMR).* KI-Klausur, ©GMD, pdf, 1996.

[RP97] Peter Rechenberg und Gustav Pomberger. *Informatik-Handbuch.* Carl Hansa Verlag München Wien, 1997.

[Sch92] Herbert Schlitt. *Systemtheorie für stochastische Prozesse, Statistische Grundlagen, Systemdynamik, Kalman-Filter.* Springer Verlag Berlin Heidelberg, 1992.

[Sie97] Martin Sielaff. *Untersuchung grundlegender Methoden zur Integration verschiedenartiger Sensoren am Beispiel einer Kartengenerierung.* Diplomarbeit an der Universität Stuttgart, 1997.

[Stö97] Sascha A. Stöter. *Robothleten.* Roboterwettkampf zeigt alternative Staubsauger und Marsroboter, C't magazin für computertechnik, Heft 10, 1997.

[Sto98] Markus Stoye. *Navigation mit Hilfe von eindimensionalen Bildern.* Technische Universität München, 1998.

[SV96] Rolf Dieter Schraft und Hansjörg Volz. *Serviceroboter.* Innovative Technik in Dienstleistung und Versorgung, Springer-Verlag Berlin Heidelberg, 1996.

[TGB+] Sebastian Thrun, Jens-Steffen Gutmann, Wolfgang Burgard, Benjamin J. Kuipers und Dieter Fox. *Integrating Topological and Metric Maps for Mobile Robot Navigation: A Statistical Approach.*

[Tim98] Ingo Johannes Timm. *Dempster-Shafer Evidenztheorie, Dempsters Kombinationsregel.* http://www.informatik.uni-bremen.de/~kcr/paper/pius97/node17.html, 1998.

[Var98] Dezsö Varju. *Mit den Ohren sehen und den Beinen hören, die spektakulären Sinne der Tiere.* Verlag C. H. Beck München, 1998.

[vR97] Gero von Randow. *Roboter Unsere nächsten Verwandten.* Rowohlt Taschenbuch Verlag, 1997.

[Wei98] Thilo Weigel. *Roboter Fußball: Selbstlokalisierung, Weltmodellierung, Pfadplanung und verhaltensbasierte Kontrolle.* Diplomarbeit an der Universität Freiburg, 1998.

[Wen00] Lothar Wenzel. *Kalman-Filter, Ein mathematisches Modell zur Auswertung von Messdaten für die Regelungstechnik.* Elektronik, Heft 6, 2000.

[WJW97] Olle Wijk, Patric Jensfelt und Bo Wahlberg. *Sensor Fusion for Mobile Robot Navigation – A First Subjective Discussion.* Royal Institute of Technology Stockholm, 1997.

[Wü] Wünschmann. *Sensorik-Vorlesungsskript*. http://elvis.inf.tu-dresden.de/asc2html/sensorik/h-000003.htm.

Erklärung

Ich versichere, die vorliegende Arbeit allein und nur unter Zuhilfenahme der im Literaturverzeichnis genannten Quellen angefertigt zu haben.

Jörg Dreßler
Brandenburg, den 3. August 2001