Räumlich und zeitlich ausgedehnte Regionen als Orientierungsmittel bei der Roboternavigation

Diplomarbeit am
Fachbereich Informatik
der Universität Hamburg

Ullrich A. Vogel
Lutterothstr. 55
20255 Hamburg

Erstbetreuer:
Prof. Dr. Bernd Neumann

Zweitbetreuer:
Prof. Dr. Christian Freksa

9. Oktober 1995

Kurzfassung

Wir betrachten einen autonomen mobilen Agenten, der einen Weg lernt, auf dem eine Ladung transportiert werden soll. Dazu muß er in der Lage sein, Routen zu lernen und zu reproduzieren. Er ist mit Abstandssensoren ausgerüstet, deren Messungen protokolliert werden. Eine Route besteht aus einer Folge von Landmarken, die hier sowohl zeitlich als auch räumlich ausgedehnte Objekte sind, die wir B-Regionen nennen. Prototypische B-Regionen werden dazu, abhängig von der Roboterwelt mithilfe von sog. B-Regionenmodellen definiert. Die Modelle enthalten räumliche und zeitliche Informationen über die Umgebung und Bewegung des Agenten, so wie sie sich im zeitlichen Verlauf der Abstandsmessungen darstellen. Durch den Abgleich der Messungen mit der Spezifikation der Modelle kann entschieden werden, ob die Messungen auf das Modell passen. Ist dies der Fall, so gilt die durch das B-Regionenmodell definierte Landmarke als erkannt. In der Akquisitionsphase werden die erkannten Landmarken zu einer Route zusammengefügt. In der Reproduktionsphase wird für das zur aktuellen Landmarke passende Modell geprüft, ob sich bei Ausführung der gelernten Bewegung die Umgebung wiedererkennen läßt. Ist dies nicht der Fall, so werden Recovery-Maßnahmen eingeleitet, die den Roboter wieder auf den richtigen Weg zurückbringen.

Inhaltsverzeichnis

1 Einleitung

Die Orientierung im Raum gehört zu den Basisfähigkeiten eines Roboters, der sich frei in seiner Umgebung bewegt. Dabei müssen Orte unterschieden werden und die eigene Lage bezüglich der Umgebung bestimmt werden. Grundlage dessen ist die Umgebungsrepräsentation, die sich je nach Aufgabe und Sensorik des Roboters unterscheiden kann. Hierzu existieren zum einen Ansätze, in denen die Welt in Form einer Rasterkarte, die zwischen Freiraum und Hindernissen unterscheidet, repräsentiert wird und die den Roboterstandort als Koordinate des Rasters auffassen.[1] Zum anderen gibt es qualitative Ansätze, die interessante Punkte in der Umgebung des Roboters identifizieren und die räumlichen Relationen zwischen den Landmarken in Form eines Graphen abspeichern. In beiden Fällen werden Punkte der Umgebung in die Repräsentation aufgenommen.

Ein Kriterium zur Beurteilung von Raumrepräsentationen ist deren Fähigkeit, mit den für die Roboterdomäne typischen Ungenauigkeiten umzugehen. Sowohl durch die Sensorik, bei der Toleranzen der Meßwerte unvermeidbar sind, als auch durch den Schlupf der Räder können teilweise erhebliche Abweichungen von der geplanten Position entstehen. Wir erhoffen uns von der vorgestellten Repräsentation eine erhebliche Erleichterung im Umgang mit solchen Abweichungen.

Wir wollen im Folgenden eine Repräsentationsform betrachten, die es ermöglicht, räumlich und zeitlich ausgedehnte Bereiche der Umgebung, sog. *Bewegungsregionen* (B-Regionen), zu repräsentieren um sie als Orientierungsmittel für die Navigation auszunutzen. B-Regionen sind Instanzen von B-Regionenmodellen, mit denen bedeutungstragende Ausschnitte der Domäne definiert werden. Erkannte B-Regionen können dann vom Roboter als Landmarken zur Umgebungsrepräsentation und zur Positionsbestimmung genutzt werden. Wir unterscheiden eine Akquisitionsphase, in der die erkannten Landmarken in einer Route aufgesammelt werden, und eine Reproduktionsphase, in der die gelernte Route nachvollzogen wird.

Eine Tür in einem Korridor, in die der Roboter einbiegt, kann beispielsweise in drei B-Regionen unterteilt werden, die unterschiedliche Bedeutungen haben. Abbildung 1 zeigt eine solche Situation. Wir unterscheiden eine Korridorregion, eine B-Region, in der der Roboter abbiegt und eine Türregion. Auf der gestrichelt dargestellten Trajektorie kann die Abbiegeregion nicht erkannt werden, da der Roboter hier zu wenig nach rechts dreht. Er weicht damit von der durch das B-Regionenmodell vorgegebenen Orientierung ab. Wir werden Korrekturmechanismen kennenlernen, die den Roboter wieder auf den „rechten Weg” zurückführen, wenn er die vorgesehene B-Region verläßt.

[1] [Davis 90], [McDermott, Davis 84]

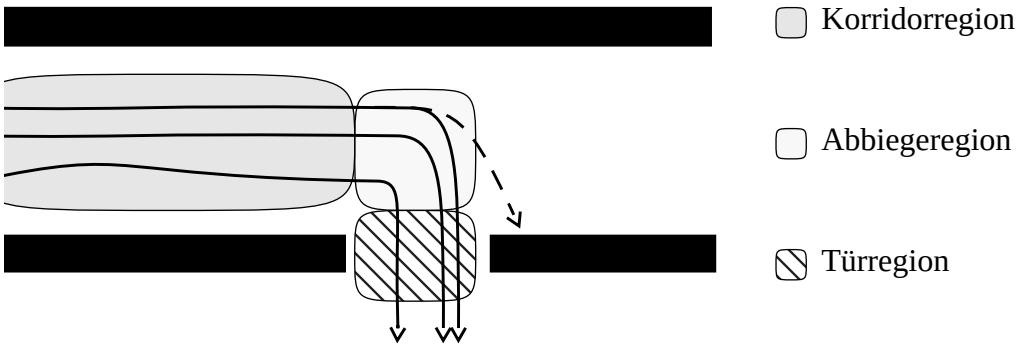

Abbildung 1: Aufteilung einer Tür in einem Korridor in drei B-Regionen. Die Pfeile deuten an, daß B-Regionen einen Ausschnitt aus dem fünfdimensionalen Kofigurationenraum des Roboters, der aus x-, y-Koordinate, Orientierung und Geschwindigkeit des Roboters besteht, definieren.

Als formale Grundlage für die Modellierung von B-Regionen stellen *Vorgangsmodelle* [Kockskämper, Neumann 94] den Ausgangspunkt für die Entwicklung von B-Regionenmodellen dar. Mit Vorgangsmodellen können interessante Vorgänge in den Meßwerten definiert und mithilfe der *Vorgangserkennung* [ebd.] erkannt werden. In der Roboterdomäne enthalten die Vorgangsmodelle qualitative räumliche und zeitliche Informationen über die Umgebung und Bewegung des Roboters, so wie sie sich im zeitlichen Verlauf der Meßwerte darstellen. Sie stellen damit eine Übersetzungsvorschrift der Sensorsignale in qualitative Termini dar.

Durch den hierarchischen Aufbau von Vorgangsmodellen kann das Modell der Welt strukturiert werden, so daß es möglich ist, viele Modelle domänenübergreifend zu definieren. Da die Umwelt des Roboters aber prinzipiell sehr viele verschiedene Formen annehmen kann, ist eine Anpassung der Modelle an die Erfordernisse der untersuchten Umwelt unumgänglich.

B-Regionenmodelle sind eine spezialisierte Form von Vorgangsmodellen, die in der Hierarchie auf der obersten Stufe stehen. Ihre erkannten Instanzen, die B-Regionen, werden in Routen als Landmarken verwendet

In der Akquisitionsphase wird der Roboter vom Trainer ferngesteuert. Dabei gleicht der Roboter die definierten B-Regionenmodelle mit den aufgenommenen Daten der Sensoren ab und setzt die erkannten B-Regionen zu einer Route zusammen.

Stellen wir uns einen Roboter vor, dessen Aufgabe es ist, eine Ladung zu transportieren. Dazu soll er den Transportweg lernen und möglichst fehlertolerant reproduzieren können. Eine solche Anwendung könnte man sich z. B. in Fabrikhallen, Krankenhäusern oder Bürogebäuden vorstellen. Wir benutzen die Simulation eines existierenden Roboters als Untersuchungsszenario. Die Sensorik besteht dabei aus 12 Abstandssensoren, die fortlaufend die Entfernung zum nächsten Hindernis in verschiedenen Richtungen messen. Ihre Funktion ist Ultraschallsensoren nachempfunden, wobei auf die Modellierung der Streu-, Beugungs- und Mehrfachreflexionseffekte verzichtet wurde. Die Bewegung des Roboters wird insoweit vereinfacht, daß wir nur eine konstante Geschwindigkeit zulassen. Weiterhin schließen wir Trägheitseffekte beim Anfahren und Bremsen in der untersuchten Simulation aus.

Die Möglichkeiten für Landmarken werden durch die verwendeten Sensoren auf solche Umgebungs- und Bewegungselemente eingeschränkt, die sich auf das Umgebungsprofil der Abstandssensoren auswirken.

B. Kuipers[2] hat mit dem NX-Roboter ein ähnliches Szenario untersucht. Dabei erzeugen signifikante Sensordaten, wie z. B. Gleichheit zweier Sensorwerte, die Landmarken. Sie werden abgespeichert und zur Identifizierung mit der numerischen Signatur aller Sensoren an dieser Stelle versehen. Für die Wiedererkennung von Landmarken wird ein Hill-Climbing-Algorithmus verwendet. Zusätzlich werden die Aktionen, die von einer Landmarke zur nächsten führen, wie z.B. „Wand verfolgen", abgelegt. In seinem Ansatz wird eine Karte in Form eines Graphen erzeugt, in dem die punktförmigen Landmarken als Knoten und die Bewegungsform zwischen den Landmarken als Kanten im Graph dargestellt sind. Dadurch wird der Raum in einer Weise partitioniert, die nicht sehr intuitiv ist: Der Roboter kann nicht die Region, in der er sich befindet erkennen, sondern nur spezielle Referenzpunkte im Raum. Der Rest der Umwelt wird nicht berücksichtigt. Nachteilig ist dabei, daß die Landmarken bei ihrer Erzeugung überspezifiziert werden, denn sie sind von der genauen Position und Ausrichtung des Roboters beim ersten Besuch abhängig. Dadurch muß für die Wiedererkennung einer Landmarke per Hill-Climbing solange rangiert werden, bis der Roboter auf genau der gleichen Stelle steht, wie bei der Aufzeichnung der Landmarke.

Wir wollen dagegen untersuchen, wie sich eine Route aus einer Folge von Regionen aufbauen läßt. Typischerweise kann man Regionen nicht mit einer einzelnen Messung erfassen. Nur wenn wir mehrere, aufeinanderfolgende Messungen (bzw. Meßwertverläufe) während der Bewegung des Agenten betrachten, ist es möglich, ausgedehnte Objekte zu identifizieren. Dabei verändern sich die Werte der Sensoren fortlaufend. Um in diesen Meßwertverläufen bedeutungstragende Ausschnitte zu erkennen, benutzen wir die Vorgangserkennung, die die eingehenden Messungen ständig mit den vordefinierten Modellen abgleicht. Die Vorgangsmodelle dienen dazu, die qualitativ bedeutsamen Vorgänge in der Roboterdomäne zu definieren. Sie beschreiben die B-Regionen, aus denen die Umgebung des Agenten besteht so, wie sie sich dem Roboter während seiner Bewegung darstellt. Wegen der Verwendung einer konstanten Geschwindigkeit, können wir die Zeit, die der Roboter für eine Strecke benötigt, ohne Umrechnung als Entfernungsmaß benutzen.

Die B-Regionen werden mit der Bewegungsinformation und den Zeitdauerangaben des erkannten Vorgangs zu einer Route zusammengesetzt. Dazu interpretieren wir die Propositionen der Modelle, die die (bei erkanntem Vorgang) möglichen Aussagen über Umgebung und Bewegung des Roboters enthalten. Die Umgebungsproposition stellt die wiedererkennbare Landmarke dar und die Bewegungsproposition liefert, zusammen mit den Zeitdauerangaben, die Roboterbewegung, die in der Reproduktionsphase auszuführen sind. Das folgende Beispiel macht das Vorgehen in der Akquisitionsphase deutlich:

BEISPIEL: Der Roboter lernt den in Abbildung 2 gezeigten Weg. Dazu wird er entlang der eingezeichneten Linie ferngesteuert. Er erkennt dabei hintereinander die folgenden Vorgänge: „Am Anfang der Sackgasse stehen", „vom Ende der Sackgasse aus wegfahren", „rechtwinklig zur hinteren

[2] [Kuipers 87]

Wand", „Korridor verfolgen", „Tür rechts im Korridor", „Korridor verfolgen", „rechts keine Wand mehr", „Rechtskurve bis rechtwinklig zur Wand hinten", „rechtwinklig zur hinteren Wand", ...

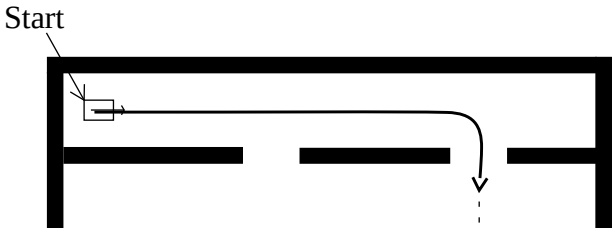

Abbildung 2: Räumliche Darstellung einer Route. Der Roboter wird während der Akquisitionsphase auf dem eingezeichneten Weg gefahren. Dabei wurde die in Tabelle 1 dargestellte Information aus den Vorgangsmodellen extrahiert und für die Route gespeichert.

Aus den Vorgangsmodellen kann dabei die relevante Information für die Route extrahiert werden:

Name des B-Regionenmodells	#	Umgebung	Translation	Rotation	Dauer
front-free-rest-blocked-m-00	1	Sackgassenbeginn			1
vert-rear-wall-f0	1	Wand	vorwärts		24
front-free-rest-blocked-p-f0	1	Sackgassenende	vorwärts		4
left-right-wall-p-f0	1	Korridor	vorwärts		11
right-open-left-p-f0	1	Tür rechts im Korridor	vorwärts		4
left-right-wall-p-f0	2	Korridor	vorwärts		10
du-r-fr	1	Rechts keine Wand	vorwärts		1
curve-90-fr	1	Wand	vorwärts	rechts	6
vert-rear-wall	2	Wand			1
...		...	...	...	...

Tabelle 1: Abfolge der erkannten Regionen bei der Roboterfahrt aus Abbildung 2.

Die Informationen aus Tabelle 1 werden durch gerichtete Kanten partiell geordnet, wodurch eine reproduzierbare Route entsteht, in der auch Alternativen ausrückbar sind, wenn das zur Korrektur von Fehlern notwendig ist.

Die Aufgabe der vorliegenden Diplomarbeit ist es, die prinzipielle Möglichkeit aufzuzeigen, Roboternavigation mit qualitativen sensorbasierten Modellen der

Roboterbewegung durchzuführen. Dazu werden wir formale Konzepte entwickeln, die eine Formulierung robust reproduzierbarer Modelle für die Roboterbewegung ermöglichen, und deren Grundlagen darstellen. Wir betrachten, wie der Roboter mithilfe dieser Konzepte eine Route lernen und reproduzieren kann. Darüberhinaus werden wir einen formalen Rahmen für die Definition von Reparaturmaßnahmen angeben, die den Roboter wieder auf die gelernte Route zurückbringen, wenn er so wesentlich von der gelernten Bewegung abweicht, daß die B-Region verlassen wird.

Der technische Rahmen wird durch einen simulierten Roboter gegeben, der mit seiner Sensorik Daten liefert. Typischen Verläufen dieser Daten werden mittels B-Regionenmodellen räumliche Bedeutungen zugeordnet, die zu einer Route zusammengesetzt werden. Damit wird es möglich, dem Roboter eine Route durch Teach-In-Programmierung beizubringen, die er reproduzieren kann. Da die einzelnen Elemente der Route B-Regionen sind, ist schon ein gewisser Freiraum in der Bewegung des Roboters impliziert. Weicht jedoch die Trajektorie über die Toleranzgrenzen hinaus von der Route ab, so kann der Roboter den Fehler erkennen und korrigieren. Dadurch ist der Roboter in der Lage z. B. eine Transportaufgabe zu erfüllen, indem die Route, die er wiederholen soll, einmal von einem Trainer gesteuert wird.

Kapitel 2 beschreibt den Roboter, seine Simulation und die Umgebung. Hier werden wir die Begriffe klären, die zur Beschreibung der Roboterbewegung und der Verarbeitung von Meßwerten notwendig sind. Dabei werden wir Einblicke in die Freiheitsgrade des Roboters gewinnen, die für die Entwicklung von geeigneten Modellen von Bedeutung sind. Im Kapitel 3 werden wir die geometrischen Gesetzmäßigkeiten der Abstandsmessung kennenlernen. Danach wenden wir uns einigen einfachen Experimenten zu, welche die für die Modellierung der B-Regionen verwendbaren Invarianten anschaulich machen. Die formalen Grundlagen für die qualitative Repräsentation von räumlich und zeitlich ausgedehnten B-Regionen werden in Kapitel 4 erläutert: dies sind Prädikate, Vorgangsmodelle und B-Regionenmodelle. Im Rahmen der Vorgangserkennung, die als Basis für die Erkennung von bedeutungstragenden Umgebungsmerkmalen dient, werden wir dort auch auf die Verarbeitung von Meßwerten eingehen, die sowohl bezüglich ihres Wertes als auch bezüglich des Zeitpunktes ihrer Aufnahme unsicher sind. Die Navigation mithilfe von B-Regionen werden wir in Kapitel 5 in drei Teile gliedern: In der Akquisitionsphase baut der Roboter eine Route aus B-Regionen auf. Diese Route wird in der Reproduktionsphase nachvollzogen, wobei das Recovery in gewisser Weise unterstützend wirkt, indem es qualitative Abweichungen des Roboters aufgrund einer Fallbasis von Recovery-Tools ausgleicht. Kapitel 6 schließlich wird diese Arbeit mit verwandten Arbeiten in Beziehung setzen. Danach schließen wir mit einer Bewertung und dem Ausblick.

2 Die Domäne

Für die Entwicklung eines Algorithmus zur Roboternavigation müssen wir die Fähigkeiten des Roboters, mit seiner Umwelt zu interagieren, untersuchen, denn der Algorithmus muß erstens die Daten, die der Roboter aus seiner Umgebung gewinnt, als Eingabe verarbeiten können, zweitens müssen die Steueranweisungen des Algorithmus in Aktionen umsetzbar sein und drittens müssen wir die Interpretation der Daten und der Wirkung von Aktionen, auf die Art der Umgebung des Roboters stützen. - Der Algorithmus ist von der sensomotorischen Ausstattung des Roboters und von der Welt des Roboters abhängig. Deshalb stellen wir in diesem Kapitel den Roboter, seine Simulation und die daraus resultierenden Definitionen der Problembeschreibung vor.

Um die Vorraussetzungen für die Navigation auf eine möglichst realistische aber handhabbare Basis zu stellen, verwenden wir die Simulation eines physischen Roboters als Versuchsplattform. Dazu stand uns ein, in Zusammenarbeit mit der Ecole National de Technique Advances (ENSTA) in Paris entwickelter Simulator zur Verfügung, der auf dem im folgenden Abschnitt vorgestellten „Robuter" basiert.

2.1 Der physische Roboter

Abbildung 3 zeigt den „Robuter". Wir können die Aufbauten, mit dem Arm und der Steuereinheit, sowie die Basis, mit den Rädern und den Sensoren, unterscheiden. Für die hier behandelte Navigationsaufgabe sind die Aufbauten von untergeordneter Bedeutung, da sich der Roboter nur in der Ebene bewegen kann. Daher ist beim Simulator auf die Modellierung der Aufbauten verzichtet worden.

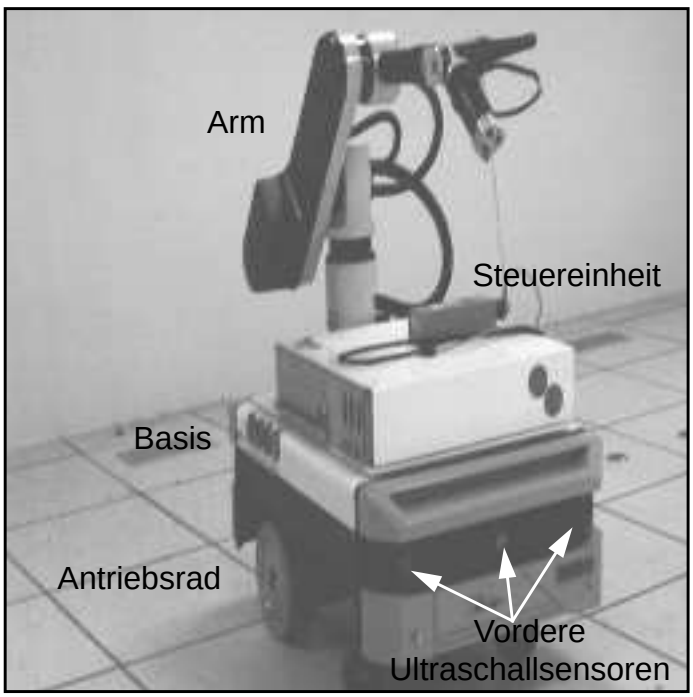

Abbildung 3: Roboter „Robuter" der ENSTA in Paris. Das Bild zeigt den Roboter von vorne links.

Aktorik

Der „Robuter" hat eine Grundfläche von 1030x700mm. Er wird an den Hinterrädern angetrieben. Die Vorderräder sind, wie bei einem Einkaufswagen, um eine vertikale Achse frei drehbar gelagert und laufen nur mit. Die Lenkung erfolgt mittels Differentialsteuerung: dabei kann Geschwindigkeit und Drehrichtung der beiden Antriebsräder unabhängig voneinander angesteuert werden. Dadurch kann jeder beliebige Kurvenradius, bis zur Drehung auf der Stelle gefahren werden. Kurven können durch verschiedene Geschwindigkeiten der beiden Räder erreicht werden. Drehungen auf der Stelle sind bei dieser Bauart möglich, indem die beiden Antriebsräder in entgegengesetzte Richtungen drehen.

DEFINITION 1: Der **Referenzpunkt** des Roboters ist der Punkt um den sich der Roboter dreht, wenn beide Antriebsräder mit gleicher Geschwindigkeit in die engegengesetzte Richtungen drehen.

Durch die Aktorik des Roboters sind zwei Richtungen, die Vorwärts- und Rückwärtsrichtung, bei der die beiden Antriebsräder die gleiche Geschwindigkeit haben, als Referenzrichtungen prädisponiert. Als Vorwärtsrichtung, die Richtung in die sich der Roboter meistens bewegt, wählen wir die Seite des Roboters an der die drei Sensoren zueinander um einen Winkel von 30^o versetzt angeordnet sind (s. u.), da sich daraus die beste Abdeckung der Ebene durch die Sensoren ergibt.

DEFINITION 2: Die **Referenzrichtung** des Roboters ist die Richtung in die der Sensor 0 mißt.

Sensorik

Der Roboter ist mit 12 Ultraschallsensoren ausgerüstet, von denen je drei an den vier Seiten der Basis angebracht sind. Die Sensoren rechts, links und hinten messen jeweils senkrecht zu der Seitenwand des Roboters auf denen sie angebracht sind. Dagegen sind die vorderen Sensoren jeweils um 30^o versetzt, so daß der mittlere Sensor genau in Fahrtrichtung mißt, während die beiden seitlichen Sensoren jeweils 30^o nach links bzw. rechts verdreht messen. Damit wird durch den Öffnungswinkel der Sensoren erreicht, daß vor dem Roboter ein Bereich von 45^o links und rechts der Referenzrichtung von den Sensoren abgedeckt wird. In Abbildung 3 kann man die Lage der Sensoren leider nur erahnen, da sich die Sensoren farblich fast nicht vom Hintergrund auf dem sie montiert sind abheben. Sie liegen alle in Höhe des dunklen Streifens auf der Vorderseite des Roboters. Abbildung 4 zeigt die Anordnung und den Winkel der Sensoren schematisch aus der Draufsicht. Die Winkel- und Versatzangaben beziehen sich auf die Referenzrichtung und den Referenzpunkt des Roboters.

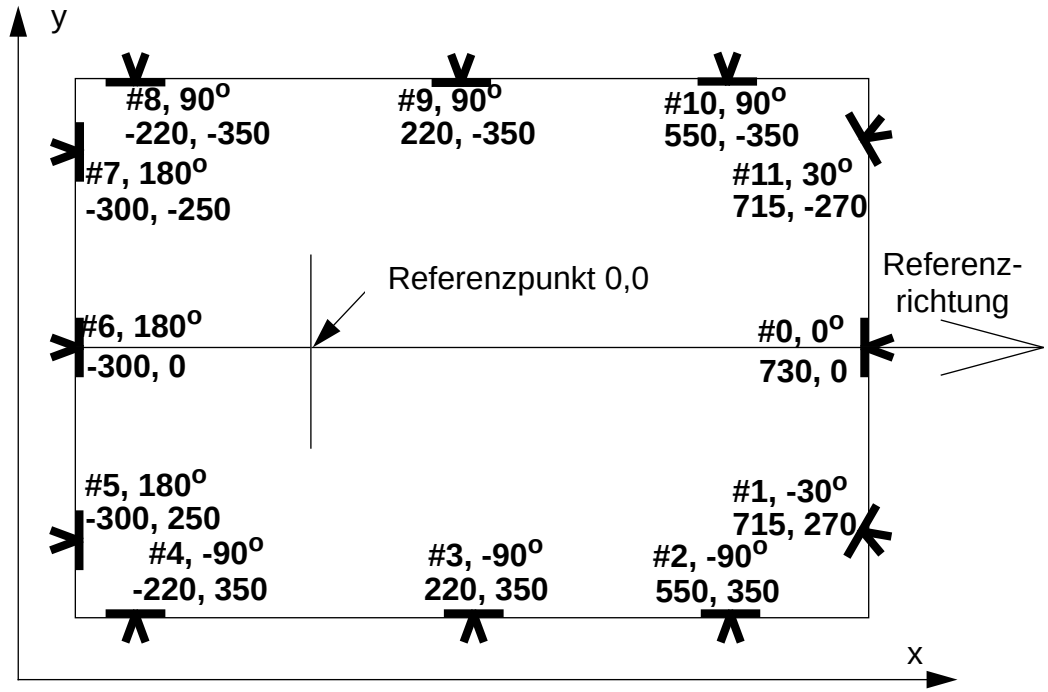

Abbildung 4: Schematische Darstellung der Sensoranordnung des „Robuters". Die Beschriftungen an den Sensoren beschreiben die Sensornummer, den Winkel, um den das Zentrum der Abstrahlkeule gegenüber der Ausrichtung des Roboters verdreht ist in Grad, sowie darunter den Versatz des Sensors in x- und y-Richtung in Bezug auf den Referenzpunkt in Millimetern.

Da sich Schall kugelförmig ausbreitet, nimmt die Intensität quadratisch mit der Entfernung ab. Dadurch ist die Reichweite von Ultraschallsensoren begrenzt. Sie ist in gewissen Grenzen durch die Intensität des Signals, also dem Schalldruck der ausgesendet wird, sowie der Eingangsempfindlichkeit beeinflußbar. Außerdem ist sie von der Beschaffenheit der Reflexionsstelle abhängig. Beispielsweise reflektiert eine flache Glasscheibe wesentlich besser, als ein Stoffvorhang. Aus praktischen Überlegungen wurde die Reichweite der Sensoren auf 10m festgelegt, indem Reflektierte Signale deren Laufzeit über 0,06s liegt nicht mehr verarbeitet werden.

Die Reichweite der hier verwendeten Ultraschallsensoren, die für die Erzeugung und den Empfang von Signalen die gleiche Membran verwenden, ist auch nach unten begrenzt, denn bevor ein Signal empfangen werden kann, muß die Eigenschwingung des Lautsprechers abgeklungen sein. Man muß demnach eine bestimmte Zeit abwarten, bevor man den Empfänger aktiviert. Dies bewirkt eine untere Grenze für die Abstandsmessung. Leider waren über diese Effekte keine detaillierten Angaben verfügbar. Die Untergrenze für die Abstandsmessung wurde durch einen *Sicherheitsabstand* modelliert. Dabei wird eine weitere Annäherung an ein Hindernis verhindert, wenn die Abstandsmessung eines Sensors dadurch kleiner als 50mm werden würde. Wenn das Hindernis in einem toten Winkel der Sensoren liegt kann allerdings ein kleinerer Abstand zur Wand erreicht werden.

Umgebung

Wir nehmen als Umgebung des Roboters eine relativ einfach gestaltete zweidimensionale Umgebung an, die an eine Büro-, Krankenhaus- oder Fabrikumgebung angelehnt ist. Sie besteht allein aus Freiraum und Hindernissen. Es wird nicht zwi-

schen verschiedenartigen Reflexionscharateristiken für Ultraschall unterschieden. Die Hindernisse ähneln Wänden, die in einen Grundriß eingezeichnet sind. Sie unterteilen den Freiraum in verschiedene Räume, die untereinander mit Durchlässen (Türen) verbunden sind. Die Roboterwelt ist statisch: Der Roboter ist das einzige bewegliche Element. Alle Hindernisse sind unbeweglich und persistent. Daraus folgt unter anderem, daß es z. B. keine Türen gibt, die einmal offen und einmal geschlossen sind. Wie in dem Grundriß eines Gebäudes, sind die meisten Wände rechtwinklig zu den Außenwänden angeordnet.

2.2 Der Simulator

Der Simulator „lsim" wurde in Zusammenarbeit mit Jean-Luc Marion von der ENSTA auf der Basis von „xsim" entwickelt. Dies wurde notwendig, da sich der Code des in Paris entwickelten Simulators xsim nicht auf der Hamburger Rechnerkonfiguration compilieren ließ. Ohne dies wären Änderungen, die für unsere Experimente notwendig waren, nicht durchführbar gewesen.

Abbildung 5 zeigt das Simulatorfenster mit den beiden Menüs. Alle Linien, die in dem Fenster dargestellt sind, sind Reflexionsstellen für die Ultraschallsensoren, egal in welchem Winkel sie zum einfallenden Signal stehen. Auch die Außenbegrenzung ist als Hindernis modelliert, so daß die dargestellte Szene abgeschlossen ist.

Zur Durchführung der Experimente wurde die Funktionalität des Simulators erweitert. Es wurden Funktionen zur Fern- und Tastatursteuerung der Roboterbewegung, sowie zur Datenübertragung zum Beobachtungsrechner implementiert, die es ermöglichen, die Simulation und die Navigation auf verschiedenen Rechnern zu vollziehen. Dies hat zum einen den Vorteil der klaren Trennung der Komponenten und zum anderen den der Steigerung der Rechengeschwindigkeit, weil der Rechner, auf dem die Navigationskomponente läuft, nicht auch noch mit der Simulation belastet ist.

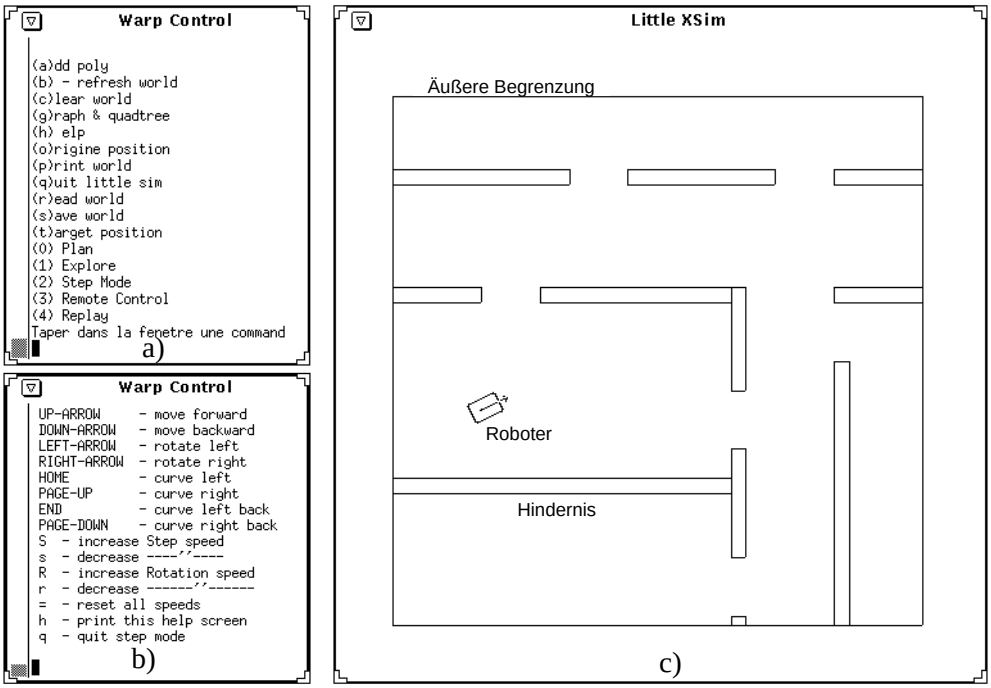

Abbildung 5: Bildschirmkopie des Robotersimulators „lsim". Links oben das Hauptmenü, darunter das Menü für die Tastatursteuerung und rechts die Darstellung des Roboters in seiner Umgebung.

Die Sensorik in der Simulation

In der Realität wird die Entfernungsmessung mit Ultraschall von vielen Faktoren beeinflußt. Angefangen von der Luftfeuchtigkeit, die einen starken Einfluß auf die Dichte der Luft und damit auf die Schallgeschwindigkeit hat, über die Anordnung der Hindernisse, die die Möglichkeiten für Mehrfachreflexionen bestimmt, bis hin zu deren Material, das die Intensität der Reflexion bestimmt. Um alle diese Einflüsse in der Simulation adäquat abzubilden, ist ein Aufwand notwendig, der den Rahmen dieser Arbeit sprengen würde. Wir haben deshalb angenommen, daß die Messungen der Ultraschallsensoren soweit vorverarbeitet sind, daß wir Abstandsmessungen in einer bestimmten Richtung erhalten: Ein Ultraschallsensor hat einen bestimmten Winkel, in dem er Schall aussendet und empfängt. Üblicherweise liegt dieser bei 30^{o}. Wenn ein Echo empfangen wird, so kann man durch die Laufzeit des Schallsignals den Abstand der Reflexionsstelle berechnen. Bei einer Laufzeit von ca. 0,06s liegt die Reflexionsstelle, bei einer Schallgeschwindigkeit von 330m/s auf einem 5,24m langen Kreisbogen mit einem Radius von 10m. Um von dieser langen Strecke einer möglichen Reflexion auf den tatsächlichen Punkt zu schließen, bedarf es einer recht aufwendigen Verarbeitung der Ultraschalldaten. Dies liegt nicht im Fokus dieser Arbeit. Daher wurde in der Simulation angenommen, daß die Reflexionsstelle im Zentrum des Abstrahlkegels liegt.

Diese Vereinfachungen scheinen unrealistische Anforderungen an die Vorverarbeitung der Sensordaten zu stellen. Allerdings existiert mit Laser-Range-Scannern eine Sensorik, deren physikalische Eigenschaften die Vereinfachungen der Simulation exakt widerspiegeln. Hierbei wird die Entfernung zur Reflexionsstelle über die Phasenverschiebung des reflektierten Laserstrahls zum ausgesendeten, sehr präzise

ermittelt. Der Einsatz dieser Sensoren in der Praxis hat allerdings den Nachteil, daß sie im Augenblick noch wesentlich teurer sind, als die üblichen Ultraschallsensoren.

Versuchsaufbau

Beim Aufruf des Simulators werden die Parameter für die Toleranzen übergeben. Im einzelnen sind die Größen der folgenden Ungenauigkeiten parametrierbar: Abweichung der Entfernungsmessung, Abweichung der Translation und Abweichung der Rotation. Dabei geben wir die Breite der Variationsintervalle prozentual zum tatsächlichen Wert an. Die Abweichungen sind innerhalb des Intervalls gleich verteilt.

Der Roboter wird in der simulierten Umgebung per Maus positioniert. Dabei wird durch die Festlegung eines Vektors, sowohl Position als auch Orientierung gewählt. Dann wird die Schrittweite für die Rotation und Translation des Roboters festgelegt. Sie entscheidet darüber, in welchen Abständen Meßwerte vom Roboter aufgenommen werden. Eine größere Schrittweite bedeutet eine größere Granularität der Messungen; denn das Fortschreiten der Zeit wurde in der Simulation als Zähler modelliert, der bei jedem Simulationsschritt hochgezählt wird.

Bei den Versuchen wird der Roboter per Tastatur gelenkt. Jeder Druck auf die Cursortasten erzeugt dabei zunächst eine Messung und dann die Bewegung des Roboters auf die nächste Position. Für jeden Simulationsschritt sind neun Aktionen möglich: Stillstand, Vorwärts, Rückwärts, Linksdrehung, Rechtsdrehung, sowie Kombinationen aus Translation und Rotation, z. B. Linkskurve. In Abbildung 5 b) sind die Tastenbelegungen für die Bewegungen aufgeführt.

Der Simulator prüft vor jedem Schritt, ob der angesteuerte Punkt im Freiraum liegt. Ist dies nicht der Fall, so wird der Schritt so weit ausgeführt, wie dies die Abmessungen des Roboters plus dem Sicherheitsabstand erlauben.

Für die Generierung und Reproduktion von Routen wurde angenommen, daß die Geschwindigkeit des Roboters für die Translation und Rotation konstant ist. Dies vereinfacht die Modellierung der Bewegung des Roboters, da die Bewegung des Roboters diskret mit neun Zuständen beschreibbar ist. Das wird dadurch erreicht, daß die Schrittlänge für Translation und Rotation im Verlauf der Versuche nicht verändert wird.

2.3 Definitionen

Aufbauend auf den in Abschnitt 2.1 und 2.2 vorgestellten Eigenschaften des Roboters und der Simulation können wir nun Definitionen einführen, die die bisher intuitiv verwendeten Begriffe für die folgenden Darstellungen präzisieren.

Wir verwenden für die Beschreibung der Lage des Roboters in seiner Umgebung ein globales kartesisches Koordinatensystem, dessen Ursprung in der linken unteren Ecke des Simulationsfensters liegt. Die x-Achse ist wie üblich die Waagerechte und die y-Achse die Senkrechte. Die Richtung 0^o ist die der x-Achse.

2.3.1 K-Raum

Mit dem Konfigurationenraum oder K-Raum werden die Freiheitsgrade des Roboters definiert.[1] Dabei sind die Anzahl und die Wertebereiche jedes einzelnen Frei-

[1] vgl. [Davis 90] S. 282ff.

heitsgrades festzulegen. Da wir die Bewegung eines Roboters in der Ebene beschreiben wollen, nehmen wir Lage und Bewegung in den K-Raum auf:

DEFINITION 3: Der **Konfigurationenraum** oder **K-Raum** des Roboters ist ein fünfdimensionaler Raum, bestehend aus:
$x \in X := \mathbb{R}^+$ der X-Koordinate,
$y \in Y := \mathbb{R}^+$ der Y-Koordinate,
$\phi \in \Phi := [0, ..., 2\pi[\subseteq \mathbb{R}^+$ der Orientierung,
$vt \in VT := \{-1, 0, 1\}$ der Translationsbewegung und
$vr \in VR := \{-1, 0, 1\}$ der Rotationsbewegung des Roboters.
Eine **Konfiguration K** des Roboters läßt sich damit als Quintupel $(x, y, \phi, vr, vt) \in$ K-Raum vollständig beschreiben.

Der Roboter ist im Simulator mit einem Rechteck und einem Pfeil dargestellt. Die Basis des Pfeiles beginnt am Referenzpunkt des Roboters und zeigt in die Referenzrichtung. Die Lage des Roboters ist bezüglich dieses Punktes beschrieben. Die Bewegung 1 ist für die Translation die Bewegung in Pfeilrichtung und für die Rotation eine Drehung in die mathematisch positive Richtung (entgegen dem Uhrzeigersinn).

Oft benutzen wir auch Teilräume des K-Raums, um einen bestimmten Aspekt der Roboterkonfiguration herauszustellen. Dazu verwenden wir die Begriffe Position, Lage und Bewegung:[2]

DEFINITION 4: Die **Position** des Roboters ist eine Projektion der Konfiguration auf die X- und Y-Achse. Sie wird mit dem Paar $(x, y) := (x, y, 0, 0, 0) \in$ K-Raum bezeichnet.

DEFINITION 5: Die **Lage** des Roboters ist eine Projektion der Konfiguration auf die X-, Y- und Φ-Achse. Sie wird durch das Tripel $(x, y, \phi) := (x, y, \phi, 0, 0) \in$ K-Raum bezeichnet.

DEFINITION 6: Die **Bewegung** des Roboters ist eine Projektion der Konfiguration auf die VR- und VT-Achse. Sie wird mit dem Paar $(vr, vt) := (0, 0, 0, vr, vt) \in$ K-Raum bezeichnet.

Eine Konfiguration (x, y, ϕ, vt, vr) beschreibt die Lage und die Bewegung des Roboters an einem Zeitpunkt. Aus diesen Angaben kann theoretisch die Lage des Roboters zum darauffolgenden Zeitpunkt berechnet werden, da die Schrittweite bekannt ist. Allerdings kann durch den Schlupf der Räder auf dem Boden oder durch Fehler der Motoransteuerung eine Abweichung entstehen. Hierdurch können auch kleine Fehler im Verlauf der Bewegung große Wirkungen haben. Z. B. hat eine Abweichung bei der Rotation, wenn danach eine lange Strecke geradeaus gefahren wird, eine große Abweichungen in der Position am Ende der Translation zur Folge. Daher kann die Vorhersage der nächsten Lage nicht ausgenutzt werden,

[2]Die Begriffe Lage und Position sind gemäß den in [Atiya 95] verwendeten definiert.

um die Anzahl der Dimensionen für die Beschreibung der Bewegung zu verringern. Wir stellen die Bewegung des Roboters durch Folgen von Konfigurationen sog. K-Folgen dar.

DEFINITION 7: Eine **Konfigurationenfolge** oder **K-Folge** ist mit Zeitpunkten versehene Folge von Konfigurationen (vgl. Definition 3). Sie beschreibt eine ausgezeichnete Trajektorie des Roboters in einem Zeitintervall. Die Konfigurationen sind dabei im Abstand von Gran abgetastet und gemäß den Zeitpunkten ihrer Beobachtung $T(k_i)$ indiziert: $<k_1, ..., k_n>$ mit $n \in \mathbb{R}$.

In der physikalischen Welt kann sich die Konfiguration des Roboters kontinuierlich in der Zeit verändern. Die Bewegung des Roboters wird durch Funktionen der Zeit für jede Dimension des K-Raums ausgedrückt. Wir können dies nur bis zu einem gewissen Grad im Computer abbilden. Dazu benutzen wir ein bestimmtes Zeitintervall, in dem die Werte der Zeitfunktionen ermittelt werden. Dieses Intervall ist durch Gran, die Granularität, definiert. Gran muß so klein gewählt sein, daß zwischen zwei Konfigurationen nichts wesentliches passiert. Gemäß Shannons Theorem kann damit aus der Konfigurationenfolge die reale Bewegung des Roboters ohne Informationsverlust rekonstruiert werden.

Wir benutzen in der Folge für alle Angaben von Positionen auf Millimetergenauigkeit gerundete Werte. Diese Auflösung hat sich im Verlauf der Arbeit als praktikabel erwiesen.

BEISPIEL: Der Roboter fährt von der Startposition (640, 544) in Richtung 45^o. Er fährt neun Zeitschritte geradeaus. Dies entspricht der in Tabelle 2 gezeigten K-Folge. Der Zeitpunkt der Beobachtung $T(k_i)$ ist in der ersten Spalte eingetragen.

Zeit	x / mm	y / mm	ϕ / rad	vt	vr
1.3	640	544	0.776	1	0
2.3	751	653	0.776	1	0
3.3	863	763	0.776	1	0
4.3	974	872	0.776	1	0
5.3	1086	982	0.776	1	0
6.3	1197	1091	0.776	1	0
7.3	1309	1201	0.776	1	0
8.3	1420	1310	0.776	1	0
9.3	1532	1420	0.776	1	0
10.3	1643	1529	0.776	1	0

Tabelle 2: Ausgabe einer K-Folge des simulierten Roboters bei einer Fahrt mit einer Schrittweite von 156,25mm in Richtung 45^o.

2.3.2 Trajektorien

Mit einer Konfigurationenfolge wird eine bestimmte Bewegung des Roboters durch den Raum in einem festen Zeitintervall beschrieben. Eine vom konkreten Zeitpunkt unabhängige Beschreibung einer Bewegung nennen wir Trajektorie.

DEFINITION 8: Eine **Trajektorie** T ist die Menge aller Konfigurationenfolgen (vgl. Definition 7) die, abgesehen von einem zeitlichen Offset c, gleich sind.

Man beachte bei der Definition 8, daß eine Trajektorie neben der Bewegung des Roboters auch die Schrittweite durch den Abstand zweier aufeinanderfolgenden Konfigurationen festlegt. Die Definition der Trajektorie wird durch die Verwendung einer konstanten Schrittweite erheblich vereinfacht, denn die Gleichheit zweier Trajektorien könnte sonst nicht durch die Gleichheit ihrer Elemente definiert werden.

SATZ 1: Seien K, K' zwei K-Folgen der Trajektorie T
und $c \in \mathbb{R}$. $\Rightarrow \forall$ i = 1, ..., n mit $k \in K$, $k' \in K'$ und $t_i, t_i' \in T$
gilt: $t_i = t_i' + c$.

Eine Trajektorie definiert also den räumlich konstanten Teil verschiedener K-Folgen, deren einzelne Konfigurationen gleich sind und sich nur im Zeitpunkt der Beobachtung unterscheiden. Die Trajektorie ist die zeitinvariante Beschreibung der besuchten Konfigurationen.

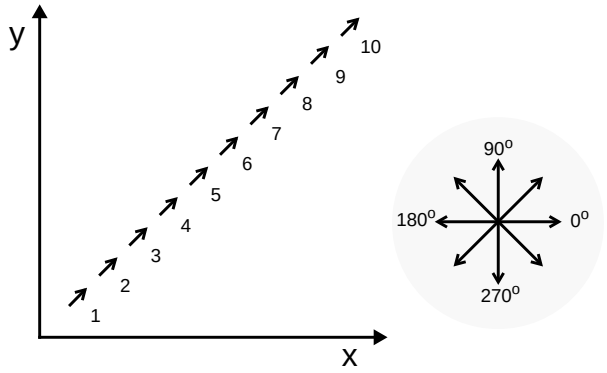

Abbildung 6: Die Abbildung der Trajektorie der K-Folge aus dem Beispiel auf Seite 16. Die Pfeile stellen die Position und die Orientierung des Roboters dar. Die Bewegung des Roboters an jeder Lage ergibt sich aus der Differenz zum folgenden Pfeil. Die Indizes an den Pfeilen sind nicht als Zeitpunkte, sondern als Ordnungsmittel für die Bestimmung des Nachfolgers gedacht.

Der Unterschied von K-Folgen und Trajektorien liegt allein darin, daß die K-Folgen mit konkreten Zeitpunkten versehen sind, während in Trajektorien nur die Reihenfolge der verschiedenen Konfigurationen festgelegt ist.

Wir haben in den vorangegangenen Abschnitten die Vorraussetzungen des Roboters und der Simulation betrachtet. Wir haben Begriffe eingeführt, mit denen wir den internen Zustand des Roboters beschreiben können, ohne dabei die Sensorinformation mit einzubeziehen. Wir werden nun die für die Betrachtung der Sensoren benötigten Begriffe definieren.

2.3.3 Meßraum

Die Sensorik des Roboters verbindet die Umwelt des Roboters mit seiner internen Repräsentation von ihr. Nur durch die Sensorik kann der interne Zustand des Roboters auf seine Gültigkeit hin überprüft werden. Daher definieren wir im Folgenden die für die Sensorik wichtigen Begriffe.

In der betrachteten Anwendung ist der Roboter mit Abstandssensoren ausgerüstet. Deren numerischen Messungen werden in konstanten Zeitabständen, der Granularität, aufgenommen und zusammen mit ihrem Zeitstempel in der quantitativen Historie abgespeichert. Ich setze für die weitere Verarbeitung dieser Daten voraus, daß das Zeitintervall zwischen den Messungen so klein ist, daß keine relevante Information verloren geht.[3]

DEFINITION 9: Sei $U_q = \{q_1,...,q_n\}$ die Menge der beobachtbaren Größen und $dom(q_i)$ der Wertebereichvon $q_i \in U_q$.
Eine **numerische Messung** NM (nach Kockskämper) ist definiert als Tripel (q, vr, tr), mit $q \in U_q$ ist ein Meßwert, $vr := [v_{min}, v_{max}] \subseteq dom(q) \cup \{nr\}$, gibt den Wertebereich von q während tr an.

---

[3]Gemäß dem Abtasttheorem von Shannon

$tr := [t_{min}, t_{max}] \subseteq \mathbb{R}$, gibt ein Zeitintervall an. Die Abkürzung nr, steht für „no reflection", also für eine Messung, bei der keine Entfernung bestimmt werden konnte; tr und vr drücken die Unsicherheit bzgl. der Zeit bzw. des Werts der Messung aus.

Die Reichweite der verwendeten Sensoren ist auf 10m begrenzt. Wir verwenden den Wert nr um auszudrücken, daß der Wert außerhalb der Reichweite des Sensors liegt. Durch die Annahme, daß wir von jedem Hindernis, das im Bereich des Sensors ist, eine Reflexion erhalten, egal in welchem Winkel die Oberfläche des Hindernisses zum Sensor ausgerichtet ist, können wir nr so interpretieren, daß kein Hindernis innerhalb der Reichweite des Sensors vorhanden ist.

DEFINITION 10: Ein **Meßraum MR** ist ein n-dimensionaler Raum. Dabei umfaßt jede Dimension von MR den Wertebereich einer Meßgröße vereinigt mit {nr}. Eine **Messung M** ist ein n-dimensionaler Vektor aus dem Meßraum.
$M := (v_1, ..., v_n) \in MR$ mit $v_i \subseteq dom(q) \cup \{nr\}$,
wobei $n \in \mathbb{N}$ und $1 \leq i \leq n$.

Der Meßraum des Roboters ist ein 12-dimensionaler Raum, dessen Dimensionen jeweils einen Wertebereich von 50.0 bis 10000.0 umfassen, da jeder der zwölf Sensoren einen Abstand zwischen 50mm und 10000mm messen kann. Erweitert wird jede Dimension noch durch den Wert nr der ausdrückt, daß die Messung außerhalb der Reichweite liegt.

Um die Entwicklung von Meßwerten im Verlauf der Zeit zu beobachten ist es notwendig, ein Protokoll der Werte zu führen. Wir nennen dieses Protokoll die *quantitative Historie*. Sie enthält alle Messungen, die seit Beginn der Beobachtung aufgenommen wurden. Hierbei sind, wie in der Definition der numerischen Messung vorgesehen, sowohl die Werte- als auch die Zeitunsicherheiten in Form von Intervallen abzuspeichern. Ein Element der quantitativen Historie hat die Form:

$$([t_{min}\ t_{max}] \text{ beobachtbare Größe } [v_{min}\ v_{max}])$$

Es definiert damit für jede beobachtbare Größe ein Intervall sowohl in der Dimension der Zeit als auch in der der Werte.

BEISPIEL: Wir sehen einen Ausschnitt aus der quantitative Historie einer Geradeausfahrt durch einen Korridor. Die Messungen erfolgen im Abstand von 1.0 Zeiteinheiten und geschehen in einer Reihenfolge, in der sich die Sensoren möglichst wenig gegenseitig beeinflussen können. Die Ultraschallmeßwerte sind mit us-0-val bis us-11-val entsprechend der Sensornummer benannt.

```
([1.0 1.1] us-0-val  'nr)
([1.0 1.1] us-6-val  [1110 1166])
([1.1 1.2] us-3-val  [1451 1524])
([1.1 1.2] us-9-val  [1360 1428])
([1.2 1.3] us-1-val  [3396 3566])
([1.2 1.3] us-7-val  [1120 1176])
([1.3 1.4] us-4-val  [1463 1536])
([1.3 1.4] us-10-val [1372 1441])
([1.4 1.5] us-5-val  [1097 1152])
```

```
([1.4 1.5] us-11-val [3087 3241])
([1.5 1.6] us-2-val  [1437 1509])
([1.5 1.6] us-8-val  [1340 1407])
([2.0 2.1] us-0-val  'nr)
([2.0 2.1] us-6-val  [2307 2422])
([2.1 2.2] us-3-val  [1404 1474])
([2.1 2.2] us-9-val  [1402 1472])
([2.2 2.3] us-1-val  [2779 2918])
([2.2 2.3] us-7-val  [2310 2426])
([2.3 2.4] us-4-val  [1415 1486])
([2.3 2.4] us-10-val [1404 1474])
([2.4 2.5] us-5-val  [2310 2426])
([2.4 2.5] us-11-val [3945 4142])
([2.5 2.6] us-2-val  [7832 8223])
([2.5 2.6] us-8-val [1425 1496])
```

Jede Messung, die wir in der physikalischen Welt durchführen, benötigt Zeit. Dies gilt insbesondere für Messungen mit Ultraschall, denn dabei wird die Laufzeit des reflektierten Schalls gemessen, um die Entfernung der Reflexionsstelle zu errechnen. Wir werden dies dadurch berücksichtigen, daß wir für jede Messung ein Beobachtungsintervall statt eines Zeitpunktes angeben. Bei Ultraschall dürfen sich die Messungen von Sensoren, die in die gleiche Richtung messen zeitlich nicht überlappen, weil sie sich sonst gegenseitig stören würden. Dadurch ergibt sich für die quantitative Historie, daß sie diskontinuierliche Intervalle des Meßraumes definiert. Abbildung 7 veranschaulicht den Intervallcharakter der Meßwerte.

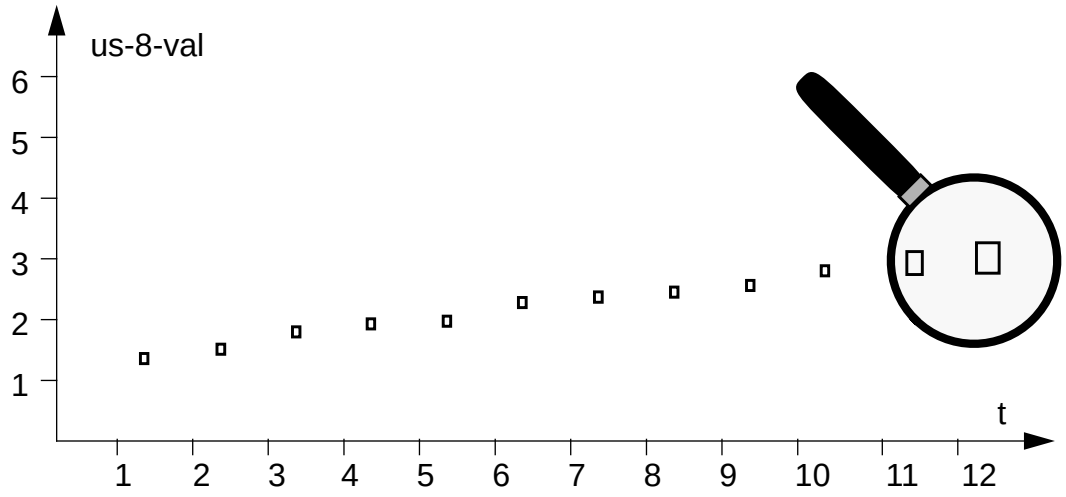

Abbildung 7: Graphische Darstellung der Meßwerte des Sensors links hinten (Sensor 8). Die einzelnen Meßwerte sind Intervalle, die sowohl in der Dimension der Zeit als auch in der des Wertes ausgedehnt sind. Die Meßwerte sind dem selben Versuch entnommen, der auch dem Beispiel der quantitativen Historie auf Seite 19 zugrunde liegt.

Wir gehen bei den Messungen davon aus, daß die Granularität so klein ist, daß nichts wesentliches zwischen zwei Messungen passiert. Dadurch kann, wie bei der K-Folge, der reale Verlauf der Meßwerte aus den aufgenommenen Werten rekonstruiert werden.

2.3.4 Meßwertverläufe

Bei der Erkennung von Vorgängen ist es oft notwendig, Meßwerte zu Zeitpunkten zu ermitteln, die nicht mit dem durch Gran vorgegebenen Zeitraster zusammenfallen. Daher liegt für genau diesen Zeitpunkt gar keine Messung vor. Gemäß dem Abtasttheorem können diese Werte aber aus den vorhandenen Messungen rekonstruiert werden, da sich innerhalb der Granularität keine signifikanten Veränderungen in den Meßwerten ereignen dürfen. Die sog. *Rekonstruktoren* stellen hier die Meßwerte für beliebige Zeitpunkte durch Interpolation zur Verfügung. Damit können trotz der gerasterten Aufnahme, Meßwerte für beliebige Zeitpunkte zur Verfügung gestellt werden.

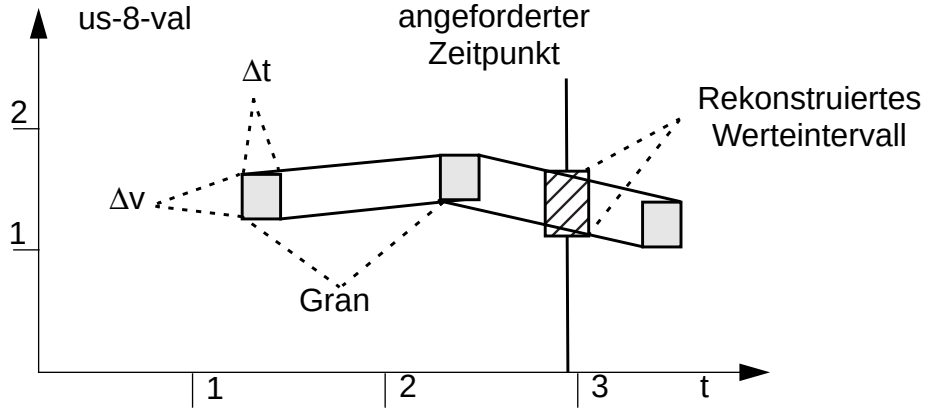

Abbildung 8: Lineare Interpolation der Meßwerte für Zeitpunkte, die zwischen den durch Gran vorgegebenen Punkten liegen.

Bei der Rekonstruktion der Meßwertverläufe aus den gerastert aufgenommenen Daten werden Werte für beliebige Zeitpunkte auf Anfrage zur Verfügung gestellt. Dabei werden Wertebereichsintervalle für Zeitpunkte konstruiert, die auch die zeitlichen Unsicherheiten berücksichtigen. Wie in Abbildung 8 zu sehen, wird ein Werteintervall erzeugt, das größer als Δv ist, denn das Minimum und das Maximum des Werteintervalls müssen im zeitlichen Intervall $[t_i+\Delta t\ t_i+\Delta t]$ gesucht werden.[4] Δt ist konstant und definiert die kleinste im System vorhandene Zeitspanne. Bei der Betrachtung von B-Regionenmodellen und Vorgangsmodellen werden wir darauf zurückkommen.

Die Messungen der Abstandsensoren werden in regelmäßigen Abständen abgetastet und mit ihren Zeitstempeln in der quantitativen Historie abgespeichert. Die Rekonstruktoren stellen dann Meßwerte für beliebige Zeitpunkte zur Verfügung und generieren damit Meßwertverläufe. Diese können dann zusammen mit ihren zeitlichen Einschränkungen durch primitive Vorgangsmodelle beschrieben werden. Nach der Generierung von Meßwertverläufen beginnt damit die qualitative Verarbeitung.

[4]Genaue Informationen über die Rekonstruktion von Meßwerten sind in [Lahres, Trowe 93] und [Kockskämper et al. 93] zu finden.

2.3.5 Bewegungszustand

In den Abschnitten 2.3.1 und 2.3.3 haben wir die Konfiguration und die Messung des Roboters kennengelernt. Während sich die Konfiguration auf den Roboter selbst bezieht, drückt die Messung die Beziehung des Roboters zu seiner Umgebung aus. Wir können die Konfiguration als Beschreibung des internen Zustandes und die Messung als Beschreibung des externen Zustandes des Roboters verstehen. Um den Gesamtzustand des Roboters zu beschreiben, fassen wir Konfiguration und Messung zur Zustandsbeschreibung zusammen.

DEFINITION 11: Der **Bewegungszustand** Z des Roboters ist ein Tripel
$\mathbf{Z} := (Tr, K, M)$ mit
$Tr = [t_{min}\ t_{max}] \subseteq \mathbb{R}$
$K = (x, y, \phi, vr, vt) \in$ K-Raum und
$M = (v_1, \ldots, v_{12}) \in$ Meßraum.

Ein *Bewegungszustand* (B-Zustand) beschreibt den internen und externen Zustand des Roboters für ein Zeitintervall. Hierbei bezieht sich t_{min} auf den Zeitpunkt zu dem die Konfiguration abgelesen wurde und t_{max} auf den größten in den Zeitintervallen der Meßwerte vorhandenen Zeitpunkt.

Wir verwenden den Bewegungszustand dazu, für jeden Zeitschritt den aktuellen Status des simulierten Roboters festzuhalten. Hierbei genügt es nur t_{min} anzugeben, da die Verzögerung und die zeitliche Ausdehnung aller Messungen im Versuchsverlauf konstant sind.

2.4 Zusammenfassung

In diesem Kapitel haben wir uns mit der Roboterdomäne, ihrer Simulation und mit den daraus resultierenden Begrifflichkeiten beschäftigt, die für die Interpretation von Meßverläufen von Bedeutung sind. Dabei haben wir die betrachteten Freiheitsgrade des Roboters sowie dessen Möglichkeiten, Meßwerte über seine Umgebung zu erfassen, kennengelernt. Danach erfuhren wir, wie mit Meßwerten umzugehen ist, die mit Unsicherheiten bzgl. ihres Wertes sowie bzgl. des Zeitpunktes ihrer Aufnahme behaftet sind. Die weitere Verarbeitung bis zur Modellierung und Erkennung von B-Regionen werden wir in den Kapiteln 4 und 5 kennenlernen. Doch zunächst werden wir im folgenden Kapitel einige Experimente vorstellen, die den Verlauf der Sensorwerte während der Bewegung des Roboters anschaulich machen sollen.

3 Versuche

Bevor wir uns der Modellierung von B-Regionen und der Vorgangserkennung zuwenden, werden wir in diesem Kapitel anhand von relativ einfachen Experimenten versuchen, einen Einblick in den Verlauf der Abstandsmessungen während der Fahrt des Roboters zu gewinnen. Wir werden dabei typische Merkmale der Meßverläufe (sog. Invarianten) identifizieren, die wir zum Teil als Prädikate[1] in den B-Regionenmodellen wiederfinden werden. Um die Identifikation zu erleichtern, wählen wir eine möglichst simple Roboterumgebung.

3.1 Geometrische Grundlagen

Zum Verständnis der experimentell gewonnenen Ergebnisse ist es von Vorteil, zunächst einige geometrische Gesetzmäßigkeiten zu klären, die bei der Abstandsmessung in Räumen mit glatten Wänden auftreten. Wir skizzieren daher die Ableitung einer, von der Rotation und Translation des Roboters abhängigen, Formel für die Veränderung des gemessenen Abstandes. Wir gehen dabei von der Definition des Tangens für spitze Winkel aus, die in Abbildung 9 dargestellt ist.

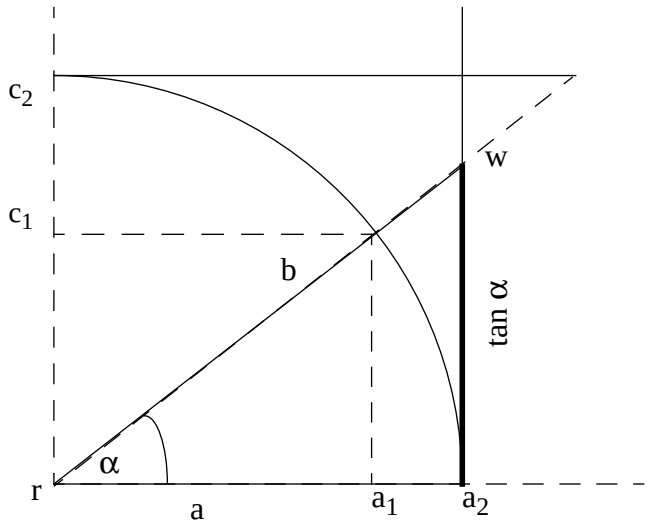

Abbildung 9: Einheitskreis mit den Definitionen des Tangens für spitze Winkel.

Wir übertragen diese Definitionen in die Welt des Roboters wie folgt:

- $a := \overline{r\,a_2}$, ist der kürzeste Abstand zwischen Roboter und Hindernis, also das Lot auf die Wand.
- $b := \overline{r\,w}$, ist die Entfernung zwischen Sensor und Wand, gemessen in der „Blickrichtung" des Roboters.
- α ist der Winkel um den die Blickrichtung des Sensors bzgl. der Richtung von a verdreht ist.

Aus der Definition des Tangens und dem Satz des Pythagoras folgt damit, daß sich b gemäß der folgenden Gleichung berechnen läßt:

[1] Siehe Abschnitt 4.1 (Prädikate)

$$b = a \cdot \sqrt{1 + \tan^2 \alpha} \qquad \text{(Gleichung 1)}$$

Unter Beibehaltung des Winkels, also bei einer reinen Translation, verhält sich der Abstand zur Wand linear, da sich nur a verändert. Bei einer reinen Rotation verändert sich α und a bleibt konstant; dann verhält sich die Funktion nicht linear. Die folgende Abbildung 10 macht den Verlauf der Funktion für zwei Veränderliche anschaulich:

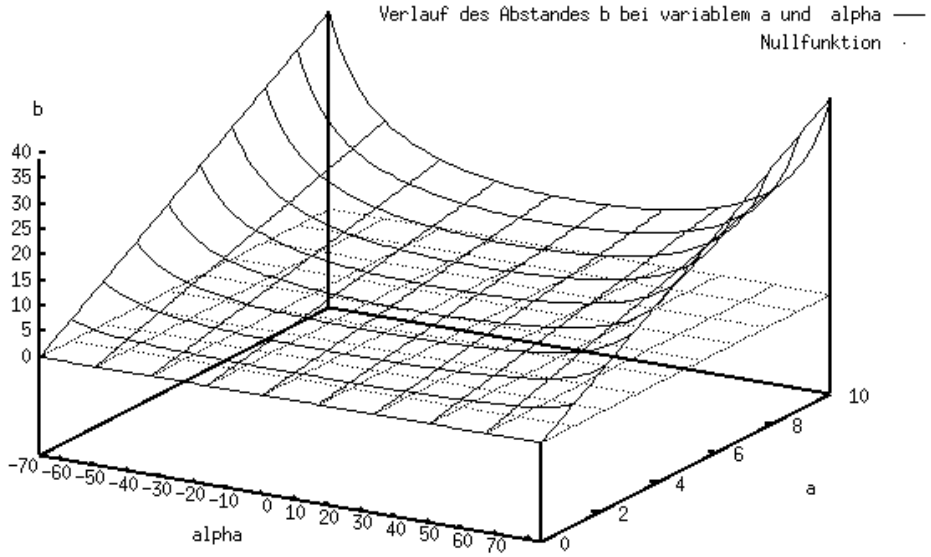

Abbildung 10: Dreidimensionaler Graph der Abstandsfunktion aus Gleichung 1 bei Veränderung beider Variablen. Der Rotationswinkel ist auf der mit alpha bezeichneten Achse in Grad angegeben. Der senkrechte Abstand zur Wand ist auf der mit a bezeichneten Achse abgetragen. Auf der Höhenachse b ist der Abstand abgetragen, der gemessen wird, wenn die Meßrichtung um alpha von der Senkrechten zur Wand abweicht. Das gepunktet gezeichnete Gitter bezeichnet die Nullfunktion. Sie ist eingetragen, um das ablesen der Höhe zu erleichtern.

Eine Translation entspricht hier einer Parallelen zur Achse a auf der Fläche des Graphen. Eine reine Rotation entspricht einer Parallelen zur Achse alpha auf der Fläche des Graphen. Die Höhe b des Graphen gibt dann den vom Sensor gemessenen Abstand zur Wand an.

Bei einer Kurvenfahrt verändern sich Winkel und Abstand gleichzeitig. Wenn der Roboter auf die Wand zufährt, so verhält sich der Abstand zur Wand gemäß der Höhe einer Kurve, die auf dem Graphen von hinten nach vorne verläuft. Entfernt sich der Roboter während der Kurve von der gemessenen Wand, so verläuft die Kurve in die entgegengesetzte Richtung.

Wenn der Abstand während der Kurve weitgehend konstant bleibt, also etwa für einen seitlichen Sensor, so ist die Krümmung bei einem großen Abstand zur Wand größer als bei einem kleineren Abstand.

3.2 Umgebung und Parameter

In den folgenden Experimenten entspricht die simulierte Umgebung einem Raum von 8x8m ohne Hindernisse. Der Winkel des Roboters zum Raum wird immer bezüglich der horizontalen angegeben. Der in Abbildung 11, 15 und 19 isoliert stehende Roboter dient als Referenz und steht genau in Richtung 0 Grad. Die Abwei-

chungen der Sensorik und der Aktorik der Robotersimulation wurden für die in diesem Kapitel gezeigten Experimente minimiert, damit der typische Verlauf der Meßwerte nicht verwischt wird.

3.3 Translation

Bei diesem Versuch wird der Roboter in einem leeren, quadratischen Raum bei einem Startwinkel von 23 Grad mit einer Schrittlänge von 312,5mm zwischen den Messungen gestartet. So ergeben sich für die zurückgelegte Strecke von ca. 7200mm 24 Messungen. Nach dem 24. Schritt sind die Messungen konstant, da der Roboter wegen der Wand keinen ganzen Schritt der vorgegebenen Länge mehr machen kann.

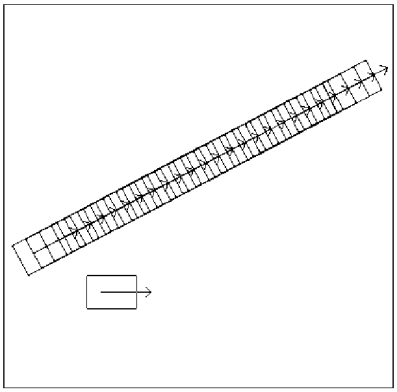

Abbildung 11: Geradeausfahrt des Roboters durch einen leeren, quadratischen Raum von 8x8m. Startwinkel 23 Grad von der Horizontalen.

3.3.1 Meßwerte der vorderen Sensoren

Wie zu erwarten verlaufen die Meßwerte monoton und linear, solange keine Ecke gemessen wird. Abbildung 12 zeigt den Meßverlauf der vorderen drei Sensoren. Die Meßwerte sinken, bis der Roboter die Wand erreicht hat. Die Steigung der Geraden kann als Geschwindigkeit für die Annäherung zur Wand interpretiert werden. Zusätzlich konvergieren der rechte vordere Sensor und der Sensor in Fahrtrichtung, da sie sich auf die selbe Wand beziehen. Die Kurve des Sensors vorne links schneidet die beiden anderen Sensorverläufe, weil dieser die obere Wand mißt.

Aus der Betrachtung der Meßwerte lassen sich bereits folgende Invarianten für die gleichmäßige Vorwärtsfahrt ableiten: Die Kurve des Sensors 0 ist monoton fallend und linear. Die Steigung der Kurve gibt die Geschwindigkeit der Annäherung an die gemessene Wand an. Falls sich mehrere vordere Sensoren auf die selbe Wand beziehen, so konvergieren ihre Meßkurven. Ecken, bewirken eine Änderung der Steigung der Kurve, also eine Unstetigkeit in der Ableitung. Da nicht sicher festzustellen ist, ob sich die Messungen zweier Sensoren auf die gleiche Wand beziehen, verwenden wir das Konvergenzkriterium nicht.

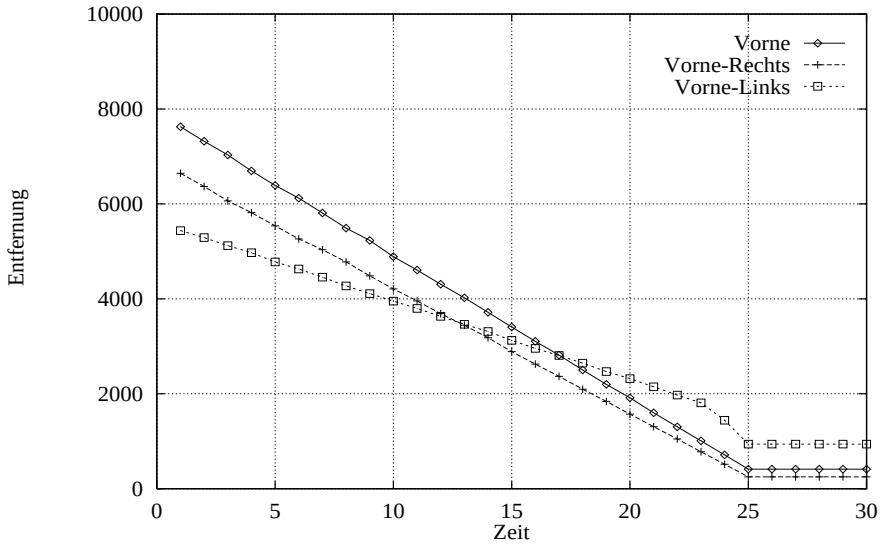

Abbildung 12: .Meßverlauf der vorderen drei Sensoren bei einer Translation durch einen leeren, quadratischen Raum (siehe Abbildung 11). Vorne bezeichnet den Sensor 0, Vorne-Rechts bezeichnet den Sensor 1 und Vorne-Links bezeichnet den Sensor 11.

3.3.2 Meßwerte der seitlichen Sensoren

Für die Erkennung von Ecken die seitlich vom Roboter liegen ist die Betrachtung mehrerer Sensoren, die in die gleiche Richtung messen interessant. So wie bei der folgenden Abbildung 13, in der die Messungen der linken Sensoren zusammengefaßt sind.

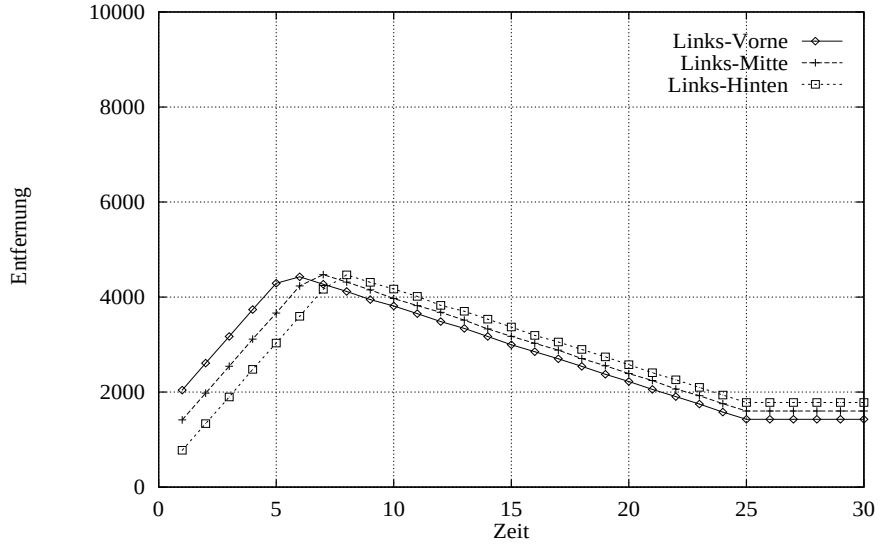

Abbildung 13: Meßverlauf der linken Sensoren beim Versuch aus Abbildung 11. Links-Vorne bezeichnet den Sensor 10, Links-Mitte bezeichnet den Sensor 9 und Links-Hinten bezeichnet den Sensor 8.

Hier erkennt man die obere linke Ecke des Raumes, die der Roboter zwischen den Schritten 5 und 8 passiert, durch den aufeinanderfolgenden Wechsel der Steigungen aller drei Meßverläufe. Dieser Wechsel geschieht zudem in Fahrtrichtung; also zuerst am vorderen, dann am mittleren und zuletzt am hinteren Sensor.

3.3.3 Integration von verschiedenen Richtungen

Bei der Translation kann man, um die Sicherheit der Erkennung zu steigern, die vordere und die hintere Abstandsmessung miteinander in Beziehung setzen (siehe Abbildung 14).

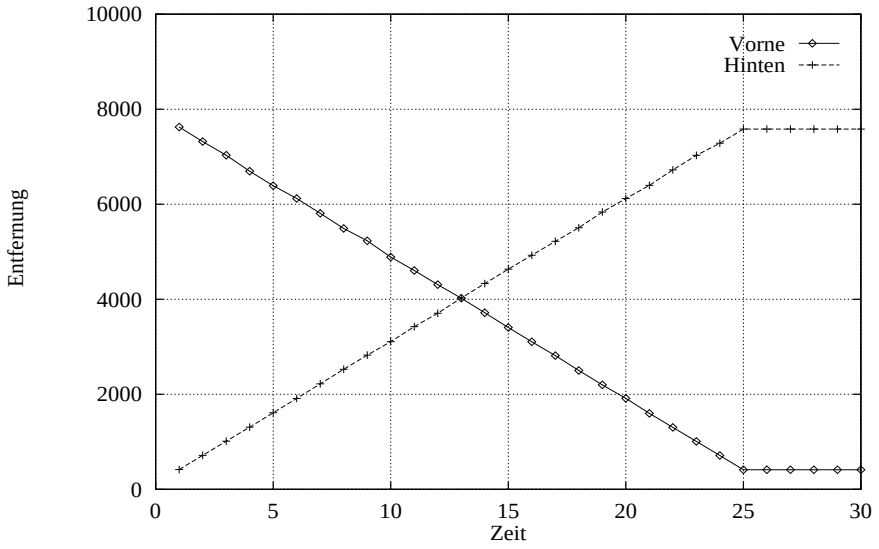

Abbildung 14: Vergleich der Messungen eines vorderen mit einem hinteren Sensors beim Versuch aus Abbildung 11. Vorne bezeichnet Sensor 0 und Hinten bezeichnet Sensor 6.

Die Steigung des vorderen Sensors ist, bis auf das Vorzeichen, gleich der Steigung des hinteren Sensors. Dies muß bei jeder Translation der Fall sein, bei der zwei unbewegliche Hindernisse vorne und hinten gemessen werden. Denn die Steigung der Kurve ist ein Maß für die Geschwindigkeit der Annäherung an das Hindernis. Und die Geschwindigkeit auf das vordere Hindernis zu und die Geschwindigkeit vom hinteren Hindernis weg müssen gleich sein.

3.4 Rotation

Bei dem in Abbildung 15 gezeigten Experiment wird der Roboter in dem bekannten 8x8m Raum gegen den Uhrzeigersinn vom Startwinkel von 324 Grad bis zu einem Winkel von 290 Grad rotiert. Danach wird er gegen die rechte Seitenwand gefahren. Die Rotationsgeschwindigkeit ist hier so gewählt, daß zu jedem Zeitschritt eine Messung erfolgen kann. Sie beträgt ca. 3 Grad.

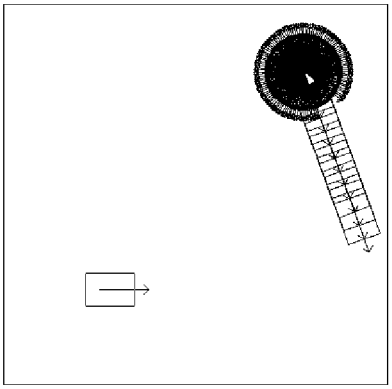

Abbildung 15: Rotation in einem leeren, quadratischen Raum. Der Startwinkel beträgt 324 Grad.

3.4.1 Rechte Sensoren

In der folgenden Abbildung 16 wird der Meßverlauf der rechten drei Sensoren bei der Rotation gezeigt. Solange ein Sensor die gleiche Wand mißt, verläuft die Messung gemäß Gleichung 1. Die Schwierigkeit hierbei liegt für die Erkennung einer so komplizierten Vorschrift darin, daß sehr viele Meßwerte erforderlich sind um die Gesetzmäßigkeit zu erkennen.

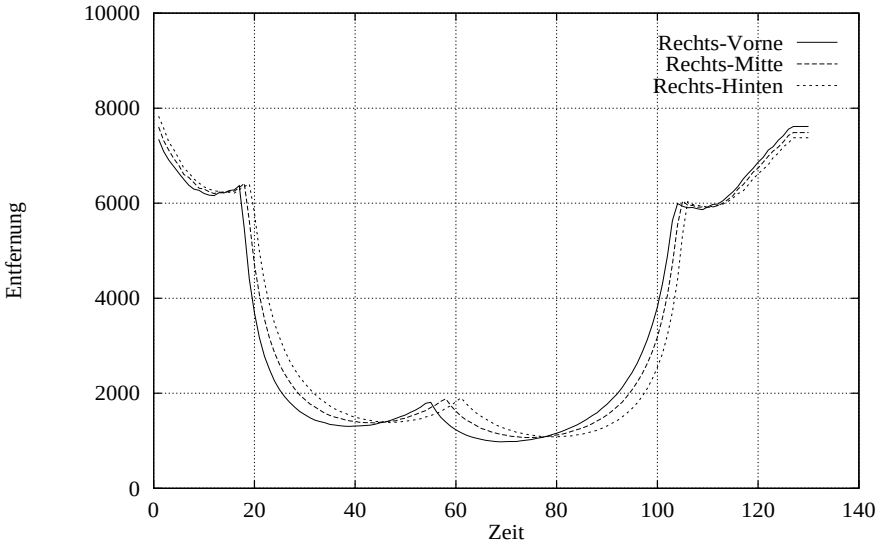

Abbildung 16: Meßverlauf der rechten Sensoren bei der Rotation aus Abbildung 15.

Für die Praxis ist der Vergleich der Sensorwerte von parallel messenden Sensoren relevant: Messen alle drei Sensoren die gleiche Entfernung zum Hindernis (z. B. Zeitpunkt 79 in Abbildung 16), so ist die Seite des Roboters auf der sich die Sensoren befinden parallel zur Oberfläche des Hindernisses ausgerichtet.

In bestimmten Fällen könnte es für einen Roboter relevant sein, die Ecken eines Raumes zu finden. Dies ist durch die Beobachtung der Meßwerte bei einer Rotation leicht möglich. Sie äußern sich, genau wie bei der Translation, durch Unstetigkeitsstellen der ersten Ableitung.

3.4.2 Vordere Sensoren

In Abbildung 17 wird der Verlauf der vorderen drei Sensoren gezeigt, die jeweils um 30 Grad zueinander verdreht sind.

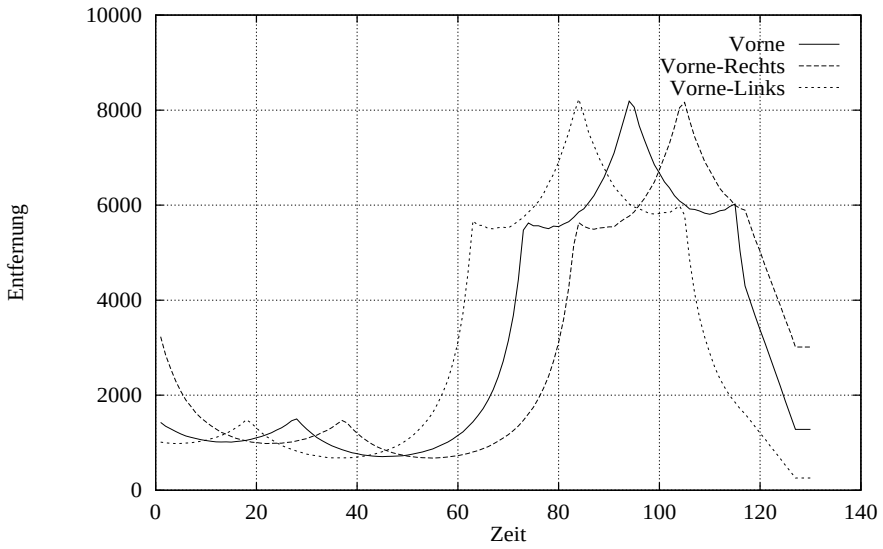

Abbildung 17: Meßverlauf der vorderen drei Sensoren. Vorne bezeichnet Sensor 0, Vorne-Rechts bezeichnet Sensor 1 und Vorne-Links bezeichnet Sensor 11.

Hierbei fällt auf, daß die drei Kurven eine sehr ähnliche Form haben. Sie sind nur um den Verdrehungswinkel der Sensoren zueinander verschoben. Dies ist ein gutes Kriterium für die Erkennung von Rotationen, die auf der Stelle erfolgen, und kann zwischen sämtlichen Sensoren benutzt werden. Dadurch daß der Winkel zwischen den Sensoren bekannt ist, läßt sich dadurch der Rotationswinkel exakt ermitteln. Allerdings sollte man noch zusätzlich andere Kriterien verwenden, damit zufällige Übereinstimmungen der Messungen, die durch Symmetrien in der Umgebung ausgelöst werden, nicht zu Fehlinterpretationen führen. Dies geschieht bei der Modellierung der B-Regionen durch die Überprüfung der Bewegungsproposition. Wir gehen dabei davon aus, daß der Roboter nur Rotationen erkennen können muß, die durch die eigene Aktorik ausgelöst werden.

Aus einem Schnitt zweier Meßverläufe läßt sich schließen, daß der kürzeste Abstand zur Wand in der Winkelhalbierenden der beiden Sensoren befindet (z. B. Zeitpunkt 100 in Abbildung 17). Dieses Kriterium kann aber nur dann genutzt werden, wenn sicher ist, daß die beiden Sensoren die gleiche Wand messen, also am besten bei Sensoren, die in die gleiche Richtung messen.

3.4.3 Integration verschiedener Richtungen

Daß die Integration von verschiedenen Meßrichtungen ein gutes Kriterium zur Erkennung von Rotationen ist, zeigt die Abbildung 18 deutlich, in der vier Richtungen zusammen dargestellt sind.

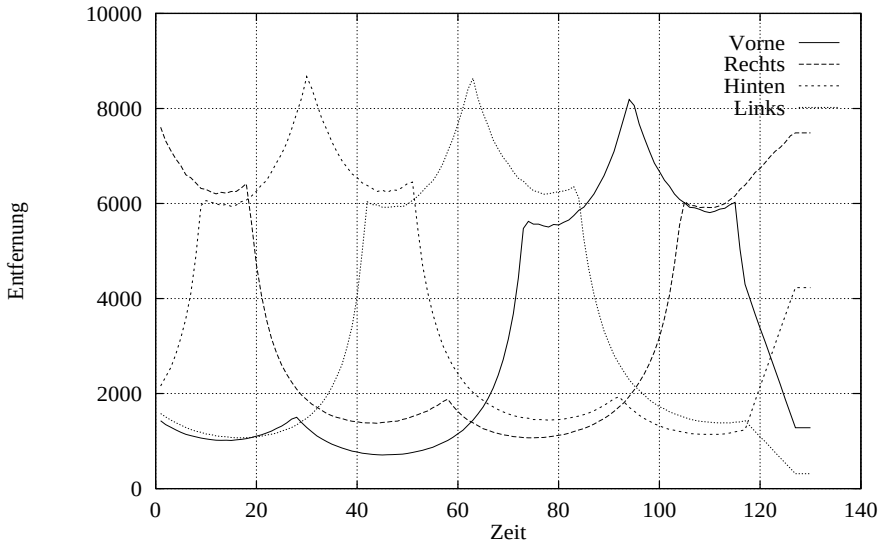

Abbildung 18: Integration von vier verschiedenen Richtungen. Vorne bezeichnet Sensor 0, Rechts bezeichnet Sensor 3, Hinten bezeichnet Sensor 6 und Links bezeichnet Sensor 9.

Die Form der Kurven wiederholt sich für alle Sensoren entlang der Rotationsrichtung. Da der Roboter sich entgegen dem Uhrzeigersinn dreht, ist die Folge der Sensoren für eine bestimmte Form: Links, Vorne, Rechts, Hinten.

3.5 Kurvenfahrt

Das Experiment „Kurvenfahrt” zeigt, wie sich die Messungen verhalten, wenn Rotation und Translation gleichzeitig erfolgen. Also wenn der Roboter eine weiche Kurve fährt. Die folgende Abbildung beschreibt den entsprechenden Versuch in dem schon bekannten Raum:

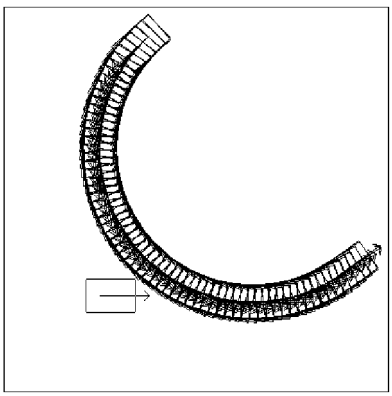

Abbildung 19: Kurvenfahrt durch einen leeren, quadratischen Raum. Der Startwinkel beträgt 222 Grad, der Endwinkel 33 Grad.

Im Versuch wurde der Roboter bei einem Winkel von 222 Grad mit einer Schrittlänge von 156,25mm und einer Rotationsgeschwindigkeit von 2,81 Grad gesteuert.

3.5.1 Interpretation der vorderen Sensoren

Für die Kurvenfahrt ist typisch, daß die Meßkurven nicht symmetrisch bezüglich ihrer Minima sind. Die Steigung der Kurve nimmt rechts vom lokalen Minimum schneller zu, als links davon. Dies erklärt sich dadurch, daß sich bei der Kurvenfahrt die Länge des Lots vom Roboter zur Wand verkürzt. Zur Veränderung des Abstandes siehe Abbildung 10 auf Seite 24.

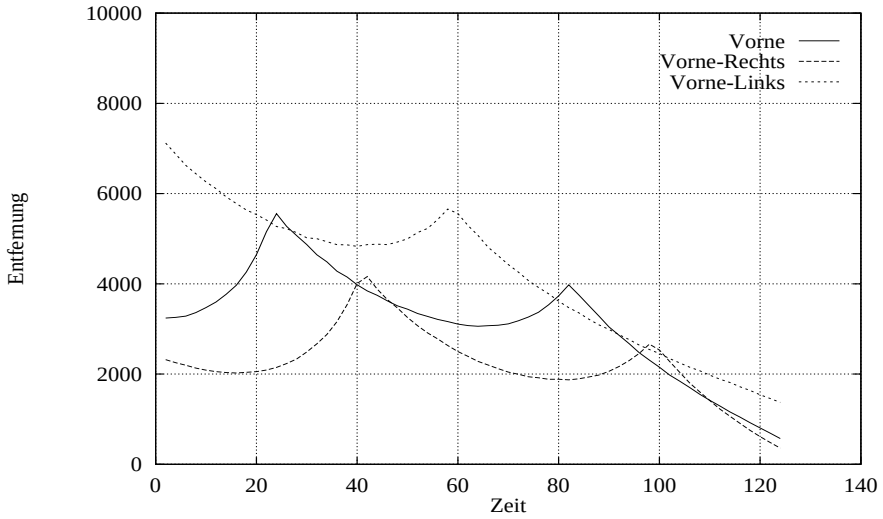

Abbildung 20: Vordere Sensoren bei der Kurvenfahrt aus Abbildung 19. Vorne bezeichnet Sensor 0, Vorne-Rechts bezeichnet Sensor 1 und Vorne-Links bezeichnet Sensor 11.

Bei den vorderen Sensoren kann man verschiedene Invarianten erkennen:

- Ecken: Unstetigkeit in der Ableitung der Kurve.
- Roboter steht im Winkel von 90^o zur Wand: Sensor 1 und Sensor 11 messen den gleichen Abstand. Dies gilt in diesem Experiment zum Zeitpunkt 102, wo der Roboter auf die rechte Wand zufährt.
- Die Vorderseite des Roboters steht im Winkel von 15^o zur Wand: Ein seitlicher Sensor hat den gleichen Wert gemessen wie der mittlere Sensor (Sensor 0).

Die beiden letzten Kriterien sind nur unter der Vorraussetzung gültig, daß sich die Messungen der Sensoren auf die gleiche Wand beziehen. Sie sind daher nur für sehr geringe Abstandsmessungen brauchbar.

Speziell können wir aus der Messung der vorderen drei Sensoren ablesen, daß sich der Roboter etwa zwischen t=58 und t=82 um 30 Grad gedreht hat und sich um ca. 1800mm vorwärts bewegt hat. Denn die rechte untere Ecke des Raumes, die von den Sensoren Vorne-Links und Vorne gemessen wurde, kann als Orientierungspunkt benutzt werden. Natürlich kann aus den Sensordaten alleine nicht bestimmt werden, ob dies tatsächlich die selbe Ecke ist, aber wenn sich für die Hypothese dieser Fortbewegungsart noch weitere Evidenzen finden lassen, dann kann man sie wohl identifizieren. Bei der Modellierung von B-Regionen wurde allerdings auf

eine solche Verwendung von Ecken verzichtet, da wir Hindernisse nicht als eigenständige Objekte Modelliert haben; dies wäre aber Vorraussetzung für die Identifikation.

Wir gehen bei der Untersuchung der Roboterdomäne davon aus, daß die Geschwindigkeit des Roboters konstant ist. Allerdings ist es interessant zu betrachten, wie wir bei einer variablen Geschwindigkeit die von der Aktorik angesteuerte Geschwindigkeit anhand der Sensordaten überprüfen können. Die Differenz zwischen der angesteuerten und der gemessenen Geschwindigkeit würde Rückschlüsse auf den Schlupf der Räder zulassen. Bei einer reinen Translation oder Rotation des Roboters haben wir schon Möglichkeiten zur Bestimmung der Geschwindigkeit kennengelernt. Für die Kurvenfahrt ist es etwas aufwendiger, den Rotations- und Translationsanteil zu bestimmen. Die Translationsgeschwindigkeit läßt sich ermitteln, indem wir charakteristische Werte der Funktionen, wie etwa lokale Maxima, Minima und Wendepunkte, für die verschiedenen Sensoren miteinander durch Geraden verbinden. Die Steigung dieser Geraden kann dann als linearer Anteil, wie bei der Translation interpretiert werden. Die Rotationsgeschwindigkeit kann durch den Vergleich entsprechender Punkte ermittelt werden: Zwei aufeinanderfolgende lokale Maxima von benachbarten Sensoren, die sich in ihrer Höhe durch den Translationsanteil der Bewegung unterscheiden, geben den Rotationswinkel an, um den sich der Roboter in der Zeit gedreht hat. Beispielsweise hat sich der Roboter in der Zeit von 58 bis 82 um 30^o gedreht (siehe Abbildung 20).

3.5.2 Hintere Sensoren

Die Werte der hinteren Sensoren verlaufen tendenziell steigend, woraus wir ableiten können, daß sich der Roboter vorwärts bewegt. Ein Fallen der Messungen erfolgt immer in Verbindung mit einer großen Differenz zwischen den Sensoren, die ja in die gleiche Richtung zeigen. Das heißt, daß sich der Roboter in einem spitzen Winkel zur Wand befindet. In diesem Fall ist der nichtlineare Anteil aus Gleichung 1 besonders groß.

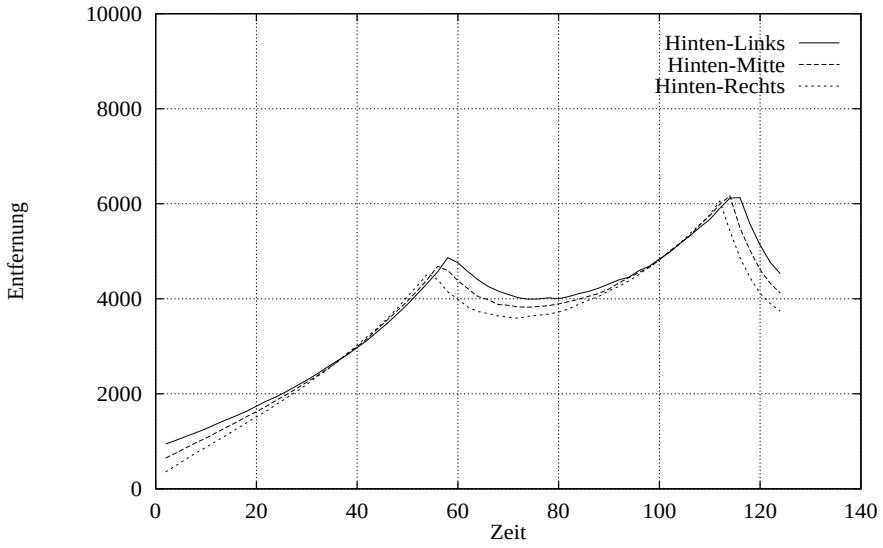

Abbildung 21: Hintere Sensoren bei der Kurvenfahrt aus Abbildung 19. Hinten-Links bezeichnet den Sensor 7, Hinten-Mitte bezeichnet den Sensor 6 und Hinten-Rechts bezeichnet den Sensor 5.

Wir können die Schnittpunkte der Meßverläufe wie in Abschnitt 3.4 wieder dazu verwenden, um zu ermitteln, wann die Roboterseite auf der die Sensoren montiert sind zum Hindernis parallel ausgerichtet sind.

3.5.3 Integration verschiedener Richtungen

Bei der Betrachtung der vier Hauptrichtungen kann man gut erkennen, wie sich die gleichzeitige Translation und Rotation auf den Verlauf der Messungen auswirkt.

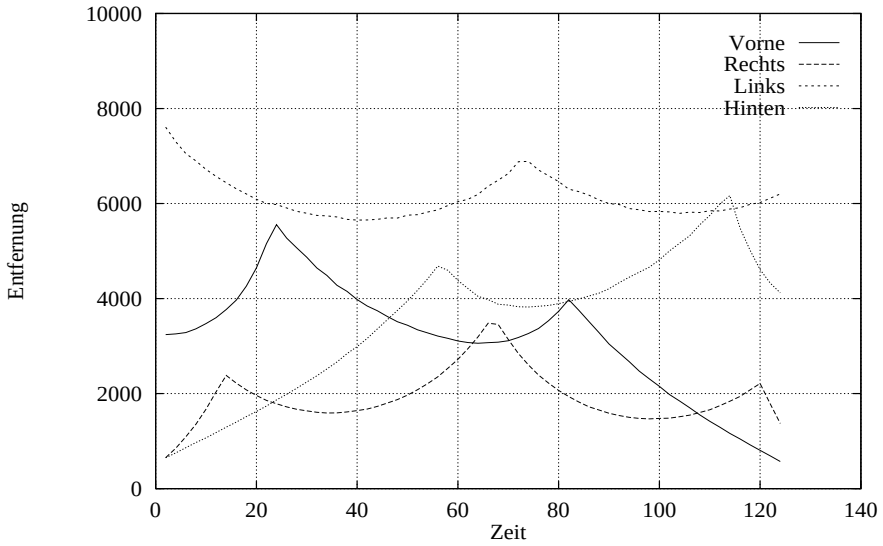

Abbildung 22: Rundumblick in die vier Hauptrichtungen bei der Kurvenfahrt aus Abbildung 19. Vorne bezeichnet Sensor 0, Rechts bezeichnet Sensor 3, Links bezeichnet Sensor 9 und Hinten bezeichnet Sensor 6.

Die Krümmung der Graphen (2. Ableitung) für die seitlichen Sensoren sind symmetrisch bezüglich ihrer lokalen Minima. Dagegen sind bei den Sensoren für vorne und hinten gegensätzliche Verzerrungen um die Minima zu beobachten, die aus der Translation folgen. Bei dem Graphen für den hinteren Sensor nimmt die Krümmung nach dem Minimum ab, während sie bei dem Graphen für den vorderen Sensor zunimmt.

3.6 Zusammenfassung

Wir haben in diesem Kapitel einen Einblick in das Verhalten von Meßwerten während verschiedener Bewegungsformen des Roboters gewonnen. Dabei haben wir verschiedene Invarianten der Meßwertverläufe erkennen können, die für bestimmte Bewegungs- bzw. Umgebungsformen typisch sind. Solche Merkmale sind entscheidend, um Prädikate und Vorgangsmodelle zu definieren, die die Bedeutung der Merkmale festlegen.

- Ein Stillstand des Roboters kann durch die Konstanz der Werte aller zwölf Sensoren festgestellt werden.
- Eine Vorwärtsfahrt des Roboters kann durch die Beobachtung der Werte von Sensor 0 und 6 erkannt werden. Die Werte von Sensor 0 sind linear fallend und die Werte von Sensor 6 sind im gleichen Maße linear steigend.

- Eine Annäherung an z. B. die linke Wand während einer Geradeausfahrt des Roboters, kann durch die Beobachtung der Werte der linken Sensoren detektiert werden. Hierbei sind die Werte aller drei Sensoren mit der gleichen Steigung linear fallend. Fährt der Roboter dabei vorwärts, so sind die Werte der Sensoren zu jedem Meßzeitpunkt wie folgt geordnet: Wert von Sensor 10 < Wert von Sensor 9 < Wert von Sensor 8.
- Die Rotation des Roboters allein aufgrund der Abstandssensoren zu bestimmen hat sich als schwierig herausgestellt, da die Erkennung eines Meßverlaufs, der Gleichung 1 folgt, sehr viele Messungen erfordert, die sich auf die gleiche, gerade Wand beziehen. Die Erkennung ist demnach stark von der Umgebung abhängig. Wir werden daher zur Erkennung der Rotation die Motoransteuerung in die Definition der Modelle einfließen lassen. Wir beschränken uns damit darauf, nur die durch den Roboter selbst ausgelösten Rotationen zu erkennen.
- Wir können bestimmte Winkel von Rotationen bestimmen, indem wir den bekannten Montagewinkel der Sensoren ausnutzen und die Werte von Sensoren zu verschiedenen Zeitpunkten miteinander vergleichen. So muß beispielsweise der Sensor 5 nach einer Linksdrehung um 90^o den gleichen Wert messen, wie der Sensor 3 vor der Drehung. Dabei muß noch die Differenz des Versatzes zum Referenzpunkt des Roboters von 30mm berücksichtigt werden. Für diese Vergleiche bieten sich die Sensoren 3, 5 und 8 sowie die Sensoren 4, 7 und 9 an, da ihr Versatz nach Drehungen auf der Stelle minimal ist.
- Bestimmte Winkel zur Oberfläche eines Hindernisses können ermittelt werden, indem die Werte von Sensoren, die in die gleiche Richtung messen verglichen werden. Wir nutzen das aus, um den Roboter an seiner Umgebung auszurichten:
  - Der Roboter steht vertikal zur hinteren Wand, wenn alle hinteren Sensoren die gleiche Entfernung messen.
  - Der Roboter ist parallel zu einer Seitenwand ausgerichtet, wenn alle rechten bzw. linken Sensoren den gleichen Wert messen.
- Ecken können bei der Translation, Rotation und Kurvenfahrt durch lokale Maxima des Meßverlaufs erkannt werden.

Neben den oben aufgeführten Punkten können auch Schwellwerte definiert werden. Sie können sich auf den Meßwert selbst oder auf dessen Ableitungen beziehen.

Die Untersuchung der Invarianten kann nicht erschöpfend sein, da in verschiedenen Umgebungsformen immer auch verschiedene Invarianten wichtig werden. Aber wir haben durch die Experimente einen Einblick bekommen, welche Möglichkeiten für den Roboter bestehen, anhand der Meßwerte einfache Merkmale seiner Umgebung oder seiner Bewegung zu erkennen. Wir werden im folgenden Kapitel betrachten, wie wir aus solchen einfachen Merkmalen komplexe Vorgänge und B-Regionen modellieren können.

4 Bewegungsregionen

In den vorangegangenen Kapiteln haben wir die Domäne und einige der in ihr erkennbaren Invarianten von Meßwertverläufen für bestimmte Bewegungsvorgänge kennengelernt. Nun werden wir die Modellierungshierarchie von den Prädikaten bis zu den Bewegungsregionen (B-Regionen) vorstellen und zeigen, wie damit Bausteine für die Umgebung des Roboters modelliert werden können.

Prädikate sind Abbildungen, die Meßwertverläufe auf Wahrheitswerte abbilden. Durch ihren Namen wird den betreffenden Messungen eine qualitative Bedeutung zugeordnet. Darauf aufbauend sind die primitiven Vorgangsmodelle definiert, die die zeitlichen Gültigkeitsintervalle der Prädikate mit ihrem Beginn und Ende repräsentieren. Zusammengesetzte Vorgangsmodelle können dann aus primitiven oder aus ebenfalls zusammengesetzten Vorgangsmodellen aufgebaut werden, um komplexere Einheiten zu beschreiben. B-Regionenmodelle sind spezielle Vorgangsmodelle, die einen bedeutungstragenden Teil der Roboterdomäne qualitativ beschreiben. Sie definieren die räumlichen Bausteine, aus denen die Umgebung des Agenten besteht, so wie sie sich dem Roboter während seiner Bewegung darstellt. Sie enthalten Informationen über die Bewegung und die Umgebung des Roboters; dadurch wird es möglich aus den Sensorinformationen die Bewegungsproposition und die Umgebungsproposition zu isolieren. Diese Bausteine können dann zusammen mit den Zeitdauerangaben zur Route zusammengesetzt werden.

4.1 Prädikate

Zur Erkennung von bedeutungstragenden Vorgängen benötigen wir eine Abstraktion der Meßwertverläufe, die es uns erlaubt deren Verlauf über die Zeit qualitativ zu beschreiben. Wir können damit die für unsere Betrachtungsweise wesentlichen Merkmale der Messung herausstellen und die unwesentlichen Merkmale offen lassen. Diese Abstraktion leisten die Prädikate, die die Meßwertverläufe auf Wahrheitswerte abbilden.

DEFINITION 12: Sei $P = \{p_1, ..., p_m\}$ die Menge aller Prädikate und $\text{dom-int}(q_i) = \{[a, b] \mid a, b \in \text{dom}(q_i)\}$ die Menge aller Wertebereichsintervalle über $\text{dom}(q_i)$.
Ein **Prädikat** (nach Kockskämper) ist eine Abbildung aus dem Meßraum in Wahrheitswerte
$\mathbf{p} : \text{dom-int}(q_{i1}) \times ... \times \text{dom-int}(q_{ik}) \rightarrow \{\text{true, false, unknown}\}$, wobei $1 \leq i_1 \leq ... \leq i_k \leq m$.

Prädikate bilden damit einen m-dimensionalen Quader aus dem Meßraum auf die Wahrheitswerte ab. Bei der Definition der Prädikate können Dimensionen des Meßraumes unspezifiziert bleiben, wenn sie keinen Einfluß auf den Wert des Prädikats haben.

Im Falle des hier betrachteten Roboters ist es zur Interpretation der Meßwerte oft wichtig, auch die aktuelle Motoransteuerung zu betrachten. Beispielsweise haben konstante Messungen aller Sensoren eine andere Bedeutung, wenn der Roboter vorwärts fahren sollte, als wenn der Roboter sowieso stillstehen soll. Denn im ersten Fall kann man davon ausgehen, daß die Roboterbewegung durch irgendetwas blockiert ist, während im zweiten Fall die Konstanz der Sensorwerte völlig normal ist. Daher verwenden wir die Werte der vt und vr aus der Konfiguration des

Roboters ebenfalls zur Formulierung der Prädikate. Damit ist für die hier betrachtete Anwendung $1 \leq m \leq 14$, da der Meßraum 12 Dimensionen umfaßt und dazu noch die Dimensionen für vt und vr kommen.

BEISPIEL: Lokales Maximum einer Entfernungsmessung: Bei der Definition des Prädikates mithilfe von *defpredicate* wird der Name, der Bezug der Messung und die Testfunktion festgelegt.[1]

```
(defpredicate local-max
    :roles ((?prop :is-a distance))
    :test-func (and (= (derivation-1 ?prop) 0)
                    (< (derivation-2 ?prop) 0)))
```

Der Meßwert „us-4-val” ist, wie alle Sensorwerte vom Typ „distance” und paßt somit auf die Rollenbeschreibung. Hat der Meßwertverlauf von „us-4-val” ein lokales Maximum, so ist an dieser Stelle die erste Ableitung des Meßwertverlaufes gleich null und die zweite Ableitung kleiner als null. Dadurch wird die Testfunktion an dieser Stelle wahr. Damit wird das Prädikat local-max für den Rollenfüller „us-4-val” erfüllt. Das lokale Maximum ist für diesen Zeitpunkt erkannt.

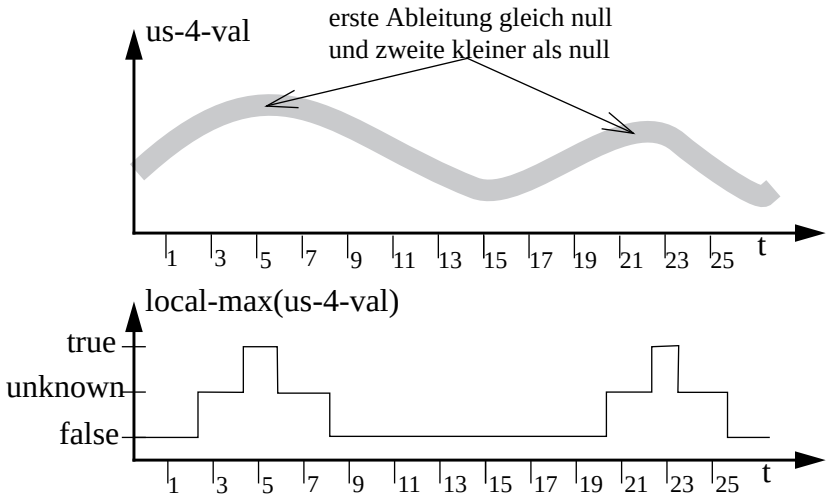

Abbildung 23: Exemplarischer Verlauf von „us-4-val” (oben). Die Dicke der oberen Linie gibt die Werteunsicherheit des Meßwerts an. Unten ist der Wert des Prädikats „local-max” für den entsprechenden Meßwertverlauf angegeben. Dort wo mit Sicherheit die erste Ableitung gleich null und die zweite kleiner null ist, ist der Wert des Prädikats „true”. Die Phasen in denen das Prädikat den Wert „unknown” hat, berücksichtigen neben den Werteunsicherheiten auch noch die zeitlichen Unsicherheiten der Messung.

[1]Da das Prädikat „local-max” für verschiedene Messungen anwendbar ist, ist der Bezug mit einer Variablen (?Prop) definiert. Mit „is-a” wird der Typ der Variablen festgelegt. Hier ist er vom Typ „distance”. Damit wird die Evaluierung des Prädikates für andere Werte der quantitativen Historie (hier: Translation und Rotation) ausgeschlossen. Die Angabe eines Types ist insbesondere in Bezug auf eine Integration verschiedener Sensorsysteme interessant, wie sie in Abschnitt 6.3 auf Seite 79 diskutiert wird.

Mit Prädikaten können Wertebereichsintervalle verschiedener Sensoren mit den Wahrheitswerten „true", „false" oder „unknown" verbunden werden. Der Name des Prädikats gibt dabei die qualitative Bedeutung der Sensorwerte an. Prädikate, die zu „true" ausgewertet werden, nennen wir erfüllt oder gültig.

Die Unsicherheiten der Meßwerte bezüglich des Werts und des Zeitpunkts ihrer Aufnahme können bei Prädikaten durch den Wert „unknown" ausgedrückt werden. Hierbei wird das Prädikat solange auf „unknown" gesetzt, wie ein Teil des möglichen Wertebereichs das Prädikat erfüllen würde und der andere Teil des Intervalls das Prädikat nicht erfüllen würde. Erst wenn der gesamte Wertebereich des Meßwertverlaufes das Prädikat erfüllt, wird der Wert auf „true" gesetzt.

Bei dem in Abbildung 23 gezeigten Meßwertverlauf ist es aufgrund der dargestellten Werteunsicherheit möglich, daß bei t=3 ein lokales Maximum vorliegt. Daher wechselt der Wert des Prädikats auf „unknown". Bei t=5 ist mit Sicherheit die erste Ableitung gleich null und die zweite kleiner als null. Zu diesem Zeitpunkt wechselt der Wert des Prädikats auf „true". Für das Ende des Prädikats gilt das Entsprechende.

Prädikate beschreiben den qualitativen Verlauf von Meßwerten über ein zusammenhängendes Zeitintervall, ohne dabei die Unsicherheiten der Meßwerte (bzgl. Zeit und Wert) zu verwischen. Sie beschreiben den Meßwertverlauf in Form von Konstantheiten; z. B. „die Meßwerte des Sensors 0 sind linear fallend". Assertionen drücken die Gültigkeitsintervalle von Prädikaten aus. Im einfachsten Fall ist dabei das Prädikat durchgehend wahr. Assertionen bilden die Grundlage zur Vorgangserkennung und zur Evaluierung von B-Regionen.

4.2 Vorgangsmodelle

Für die Beschreibung zeitlicher Zusammenhänge im Systemverhalten verwenden wir *Vorgangsmodelle* [Kockskämper, Neumann 94]. Sie stellen den Ausgangspunkt für die Entwicklung von B-Regionenmodellen dar und ordnen einer Menge von dynamischen Vorgängen, die eine bestimmte zeitliche Abfolge haben, eine Bedeutung zu. Vorgangsmodelle erlauben damit, interessante Vorgänge aus Meßwertverläufen herauszuheben. Wir verwenden sie dazu, die Umgebungs- und Bewegungselemente der Roboterdomäne so zu modellieren, wie sie sich im zeitlichen Verlauf der Abstandsmessungen darstellen.

Vorgangsmodelle sind hierarchisch aufgebaut. Sie enthalten eine Proposition, Zeitmarken, die für Beginn und Ende des Vorgangs stehen und eine Menge von Teilvorgängen. Die Teilvorgänge sind konjunktiv verknüpft und durch temporale Relationen geordnet. Ein Vorgang gilt als erkannt, wenn alle seine Teilvorgänge erkannt sind, so daß keine der temporalen Relationen verletzt wird. Die Vorgangsproposition beschreibt die Aussage, die durch die Erkennung des Vorgangs für den Zeitraum zwischen Beginn und Ende gültig ist.

Wir Unterscheiden primitive und zusammengesetzte Vorgangsmodelle. Primitive Vorgangsmodelle definieren die zeitlichen Eigenschaften von Assertionen. Zusammengesetzte Vorgangsmodelle definieren zum einen die zeitlichen Eigenschaften ihrer Teile und zum anderen deren zeitliche Beziehungen untereinander. Durch die Definition eines Vorgangsmodells werden damit Meßwertverläufe spezifiziert die eine Erkennung des Vorgangsmodells ermöglichen.

Wir verwenden für die Darstellung der zeitlichen Zusammenhänge in Vorgangsmodellen sogenannte Zeitnetze. Sie bestehen aus Knoten und gerichteten Kanten. Die Knoten stehen für die Zeitmarken eines Vorganges. Die Zeitmarken sind intervallwertig, um partielle Überlappungen oder andere qualitative zeitliche Zusammenhänge zwischen Teilvorgängen ausdrücken zu können. Die Kanten repräsentieren die zeitlichen Zusammenhänge zwischen den Knoten. Zur Notation der Minimum-Maximum-Intervalle schreiben wir das Minimum unter und das Maximum über die Knoten bzw. die Kanten. Zur Spezifikation der zeitlichen Restriktionen sind bei der Definition von Vorgangsmodellen beschränkte Differenzen zwischen Zeitmarken sowie die Menge der konvexen allenschen Relationen erlaubt.

Unter der *Vorgangserkennung* [Kockskämper, Neumann 94] verstehen wir den Abgleich der Sensordaten mit den beobachteten Vorgangs- oder B-Regionenmodellen. Sie kann als logische Herleitung der entsprechenden Proposition betrachtet werden, bei der die temporalen Restriktionen der Teilvorgänge Berücksichtigung finden. Algorithmisch besteht die Vorgangserkennung aus der sukzessiven Propagierung temporaler Constraints, die sich aus der Spezifikation der Vorgangsmodelle ergeben.

4.2.1 Primitive Vorgangsmodelle

Primitive Vorgangsmodelle bilden die unterste Stufe der Hierarchie von Modellen. Wir können mit ihnen die Gültigkeit eines Prädikats in einem Zeitintervall ausdrücken:

DEFINITION 13: Ein **primitives Vorgangsmodell** ist ein Tripel
VM = (p, ZM, TC), wobei p ein Prädikat ist.
ZM sind die Zeitmarken VM-B und VM-E. Die Zeitmarken sind Intervalle, die die Unsicherheit der Vorgangsmodellgrenzen ausdrücken. TC sind die temporalen Constraints zwischen den Zeitmarken.

Primitive Vorgangsmodelle bestehen aus einem Prädikat und den zugehörigen zeitlichen Restriktionen, den temporalen Constraints. Sie werden zwischen den Zeitmarken VM-B für den Beginn und VM-E für das Ende des Vorgangs definiert. VM-B und VM-E sind zeitliche Intervalle, die die Unsicherheit des Beginn- und Endezeitpunkts ausdrücken. Bei den temporalen Constraints von primitiven Vorgangsmodellen können Intervalle angegeben werden, die den minimalen und maximalen zeitlichen Abstand zwischen Beginn und Ende des Vorgangs angeben. So kann die Dauer eines Vorgangs spezifiziert werden. Wird die Spezifikation ausgelassen, so wird das Standard-Constraint eingefügt. Es besagt, daß der Beginn des Vorgangs vor dem Ende liegen muß. Wir benutzen dazu den kleinstmöglichen, im System vorhandenen, zeitlichen Abstand zwischen zwei Aktionen, den wir mit Δt bezeichnen.[2]

Wie in 2.3.3 beschrieben, ist jeder Meßwert sowohl bezüglich der Zeit als auch des Werts mit einer Unsicherheit behaftet, die bei keiner Messung auszuschließen ist. Die Unsicherheiten werden bei den Prädikaten durch den Wert „unknown” ausgedrückt. In den Vorgangsmodellen werden die Unsicherheiten in Form von zeitli-

[2]Zur Bestimmung von Δt vgl. Kapitel 2.

chen Intervallen für VM-B und VM-E ausgedrückt. Dabei ist der Beginnzeitpunkt der „unknown" Phase des Prädikats das Minimum von VM-B und der Beginnzeitpunkt der „true" Phase das Maximum von VM-B.

BEISPIEL: Primitives Vorgangsmodell Lokales Maximum einer Entfernungsmessung:[3]

```
(defmodel local-max
    :roles ((?dist :is-a distance))
    :label lmx)
```

Bei der Definition des Vorgangsmodells local-max mit „defmodel" werden die Zeitmarken lmx-B für den Beginn und lmx-E für das Ende des Vorgangs angelegt. Es sind keine temporalen Constraints angegeben, da in diesem Fall keine bestimmte Dauer zwischen Beginn und Ende des Vorganges vorgegeben werden soll. Die einzige zeitliche Einschränkung des Vorgangsmodells ist, daß der Beginn vor dem Ende liegen soll. Dies wird von „defmodel" automatisch geleistet und muß daher nicht explizit angegeben werden.

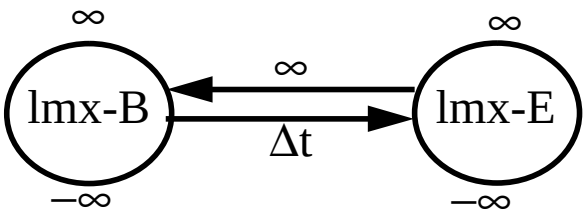

Abbildung 24: Das Zeitnetz des primitiven Vorgangsmodells „local-max" wird nach der Definition mit „defmodel" aus dem vorangegangenen Beispiel automatisch generiert. Die untere Beschriftung der Endeknoten enthält das Minimum des Beginns bzw. des Endes und die obere das entsprechende Maximum. Die Kanten zwischen den beiden Knoten des Zeitnetzes steht für die Relation lmx-B + Δt ≤ lmx-E (untere Kante) und lmx-E ≤ lmx-B + ∞. Weil Δt die kleinste mögliche Zeiteinheit des Systems ist, können wir damit ausdrücken, daß der Beginn vor dem Ende liegen soll.

Die beiden Pfeile des Zeitnetzes in Abbildung 24 stellen den minimalen und maximalen zeitlichen Abstand der beiden Knoten in Form von beschränkten Differenzen dar. Die Beschriftung des unteren Pfeils enthält die minimale Zeitdauer, die zwischen dem linken und rechten Knoten liegen darf. Entsprechend enthält die Beschriftung des oberen Pfeils die maximale Zeitdauer, die zwischen dem linken und rechten Knoten liegen darf. Wenn, wie in Abbildung 24, keine obere Grenze angegeben ist, werden wir den Rückpfeil in der Darstellung von Zeitnetzen weglassen.

Ein primitives Vorgangsmodell wird instantiiert, wenn das Prädikat zu „true" evaluiert wird und die temporalen Constraints des Vorgangsmodells nicht verletzt werden. Dabei werden die Zeitmarken mit den entsprechenden Werten belegt. An

[3]Genau wie bei der Definition des gleichnamigen Prädikates wird hier der Bezug des Vorgangsmodells auf beliebige Entfernungsmessungen (mit „:roles") festgelegt. Daneben ist ein lokal gültiger Kurzname mit „:label" definiert, der zur Erleichterung der Definition von komplexeren Vorgangsmodellen dient. Im Beispiel verwenden wir ihn dazu, die graphische Darstellung zu verkürzen.

den Zeitmarken läßt sich dann ablesen, von wann bis wann das im Vorgangsmodell definierte Prädikat erfüllt war. Ein Vorgangsmodell, dessen Zeitmarken mit konkreten Werten belegt sind, nennen wir instantiiertes Vorgangsmodell oder einen erkannten Vorgang.

Der Übergang von einem Vorgangsmodell, in dem noch keine Zeitmarke mit einem konkreten Wert belegt ist, zu einem erkannten Vorgang, geschieht bei primitiven Vorgangsmodellen in drei Schritten:

- Primitives Vorgangsmodell - keine Zeitmarke instantiiert (vgl. Abbildung 24).
- Teilweise erkannter primitiver Vorgang - VM-B ist mit einem konkreten Zeitwert belegt, und VM-E hat ein aufgrund von VM-B propagiertes Zeitintervall als Wert.
- Erkannter Vorgang: VM-E kann mit einem Wert, der innerhalb des propagierten Intervalls liegt belegt werden, da das Ende des zugehörigen Prädikats erkannt wird.

BEISPIEL: Bei dem in Abbildung 23 gezeigten Verlauf des Meßwertes „us-4-val" kann die Vorgangserkennung zum Zeitpunkt t=7 erkennen, daß die erste Ableitung gleich null war und die zweite größer als null war. Dann kann das Vorgangsmodell „local-max" mit Werten für den Beginn belegt werden. Es ist damit teilweise erkannt. Die Verzögerung bei der Erkennung erklärt sich daraus, daß für die Erzeugung der Ableitung viele Meßpunkte notwendig sind.[4]

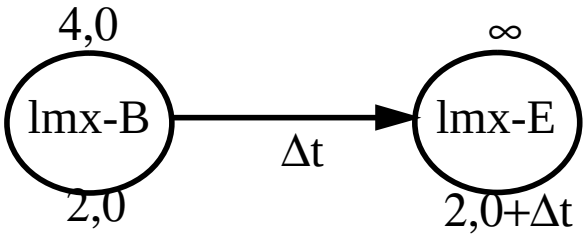

Abbildung 25: Teilweise erkannter primitiver Vorgang „local-max" beim Meßwertverlauf aus Abbildung 23.

Wenn der Beginn des Prädikats erkannt wurde, werden Minimum und Maximum des Beginnknotens mit den entsprechenden Werten belegt. Damit kann der Wert des Minimums des Endeknotens, entsprechend

[4]Die Anzahl der Meßpunkte, die wir brauchen um eine Ableitung an einer bestimmten Stelle zu bestimmen, steigt linear mit dem Grad der Ableitung: Bei der Rekonstruktion des Wertes (nullte Ableitung) benötigen wir zwei Nachbarmessungen. Für die Rekonstruktion der ersten Ableitung vier und für die zweite Ableitung sechs angrenzende Messungen. Damit keine Inkonsistenzen auftreten, muß daher die Vorgangserkennung immer soweit hinter der aktuellen Zeit zurückhängen, daß die höchste spezifizierte Ableitung genügend Werte hat. Die Ableitung zweiten Grades ist in der Roboterdomäne die höchste.

der Definition des Vorgangsmodells, berechnet werden. Das Ende des Vorgangs darf nicht vor dem Beginn stattfinden. Das Minimum des Endeknotens ist also um Δt größer als das Minimum des Beginns.

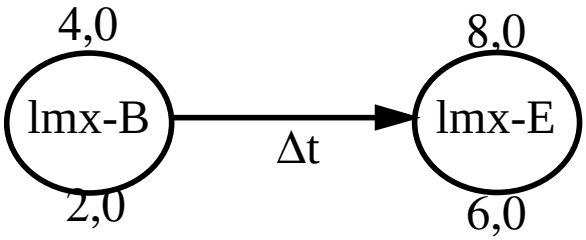

Abbildung 26: Vollständig erkannter primitiver Vorgang „local-max”.

Zum Zeitpunkt t=9 kann dann das Ende des Vorgangs erkannt werden. Da die erkannten Zeitwerte für das Minimum und das Maximum des Endeknotens nicht im Widerspruch zu den propagierten Werten stehen, können die erkannten Zeitwerte am Endeknoten eingetragen werden. Der Vorgang ist damit vollständig erkannt.

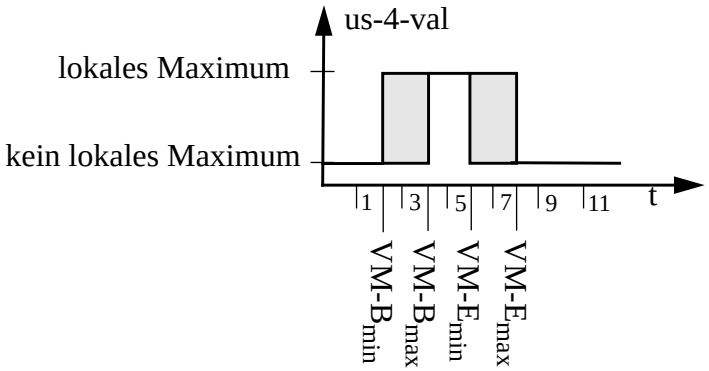

Abbildung 27: Graphische Darstellung des erkannten Vorgangs lokales Maximum mit den Minima und Maxima der Zeitmarken.

Mit primitiven Vorgangsmodellen können einfache zeitliche Zusammenhänge zwischen Beginn und Ende eines Prädikats definiert werden. Vollständig instantiierte Vorgangsmodelle sind qualitative Aussagen über einfache Vorgänge im Systemverhalten. Sie enthalten die konkreten Zeitpunkte, an denen ein definierter Vorgang stattgefunden hat. Alle erkannten Vorgänge werden in der sog. *qualitativen Historie* abgelegt und stehen damit für die Weiterverarbeitung in zusammengesetzten Vorgangsmodellen zur Verfügung.

4.2.2 Zusammengesetzte Vorgangsmodelle

Um komplexe Vorgänge beschreiben zu können, die wir bei der Bewegung durch den Raum für konzeptuell interessant halten, benötigen wir zusammengesetzte Vorgangsmodelle, die als Teile wiederum zusammengesetzte oder primitive Vor-

gänge enthalten können. Durch diese rekursive Form der Modellierung entsteht eine Hierarchie von Vorgangsmodellen, an deren unterem Ende die primitiven Vorgangsmodelle stehen.

DEFINITION 14: Ein **Vorgangsmodell** ist ein Quadrupel **VM** = (P, ZM, VM_i, TC) mit: **P** ist eine Proposition, die die Aussage über das Systemverhalten definiert. **ZM** sind die Zeitmarken des Vorgangsmodells; also der Beginn VM-B und Endzeitpunkt VM-E des Vorganges. Beide werden durch Intervalle [VM-B_{min}, VM-B_{max}] bezeichnet, die die Unsicherheit bzgl. der Zeitpunkte beschreiben. **VM_i** sind die primitiven oder zusammengesetzten Teilvorgänge des Vorgangsmodells (vgl. Definition 13). **TC** sind die temporalen Constraints zwischen den Teilvorgängen VM_i und dem zusammengesetzten Vorgangsmodell VM.

Temporale Constraints zwischen Teilvorgängen können sowohl zwischen den Beginn und Endeknoten der Teile als auch zwischen ganzen Vorgangsmodellen definiert werden. Als Relationen zwischen Zeitpunkten können beschränkte Differenzen spezifiziert werden. Zwischen Intervallen steht die Menge der konvexen Relationen als Teilmenge der allenschen Zeitrelationen zur Verfügung (vgl. [Allen 83], [Nökel 91]). Eine Relation heißt konvex, wenn sich die Bilder der beiden in Relation stehenden Intervalle unter einer Abbildung T stetig ineinander transformieren lassen, so daß alle Zwischenzustände ebenfalls Bilder der beiden Vorgänge unter T sind. Freksa bezeichnet dies als konzeptuelle Nachbarschaften [Freksa 92a] der temporalen Relationen.

BEISPIEL: In der Definition des Vorgangsmodells Geradeausfahrt ist die Bewegungsproposition „(translation . forward)" spezifiziert. Daher bedeutet die Erkennung des Vorgangs, daß sich der Roboter vorwärts bewegt hat, ohne sich zu drehen. Die Umgebungsproposition ist hierbei nicht spezifiziert, da bei dieser Definition über die Umgebung keine Aussage möglich ist.[5]

```
(defmodel straight-forward
  :movement '(translation . forward)
  :roles ((?robot :is-a robot))
  :auxilliary-roles
    ( (?front   :is-a front-middle-sensor)
      (?d-front :is-a distance)
      (?rear    :is-a rear-sensor)
      (?d-rear  :is-a distance)
```

[5]Bei der Definition zusammengesetzter Vorgangsmodelle müssen, neben dem Bezug des Vorgangsmodells, noch die Bezüge der Teilmodelle festgelegt werden, dies geschieht mithilfe von „:auxilliary-roles". Dabei werden die Typen für die verwendeten Variablen (wie bei „:roles") festgelegt. Durch die Angabe der Beziehungen „proprty-of" (Eigenschaft von) und „part-of" (Teil von) unter „:constraints", können wir die Sensoren des Roboters in sinnvolle Gruppen einteilen. In den Bezügen spiegelt sich auch die unterschieliche Natur der Teilmodelle „decreasing", „increasing" und „linear" gegenüber „forward" und „no-rotation" wider. Die ersten drei beziehen sich auf die Werte von Sensoren, während sich die letzteren auf den Roboter selbst beziehen.

```
:constraints ((property-of  ?d-front  ?front)
              (property-of  ?d-rear   ?rear)
              (part-of      ?front    ?robot)
              (part-of      ?rear     ?robot))
:submodels ((decreasing (?d-front):label dec)
            (increasing (?d-rear) :label inc)
            (linear     (?d-front):label lf)
            (forward    (?robot)  :label fw)
            (no-rotation(?robot)  :label nr) )
:label sf
:time-restrictions( (sf-b = inc-b)
                    (sf-e = inc-e)
                    (dec (equal) inc)
                    (inc (equal) lf)
                    (sf (during) fw)
                    (sf (during) nr)))
```

Um die Vorgangsmodelle mit den eingehenden Messungen zu vergleichen, wird die Hierarchie bis auf die Ebene der primitiven Vorgangsmodelle aufgelöst. Auf dieser Ebene können dann die Testfunktionen der Prädikate mit den Meßwerten verglichen werden. Ist ein Prädikat erfüllt, so wird es in den entsprechenden Vorgangsmodellen mit den konkreten Zeitpunkten für Beginn und Ende eingetragen. Danach werden diese Zeitpunkte innerhalb des Vorgangsmodells mit den definierten temporalen Constraints propagiert.

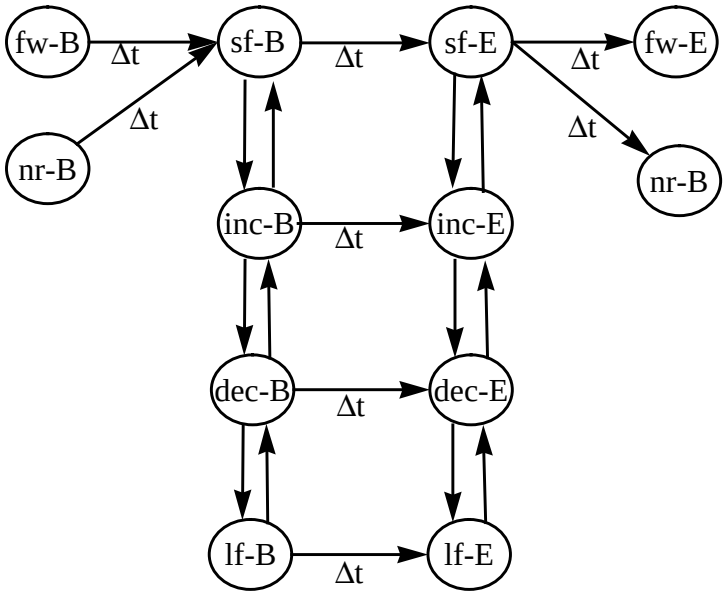

Abbildung 28: Das Zeitnetz des im vorigen Beispiel definierten Vorgangsmodells. Die Pfeile ohne Beschriftung drücken kleiner-gleich aus ($a + 0 \leq b$ für $a \longrightarrow b$).

Können alle Teilvorgänge des Vorgangsmodells VM_i mit Zeitmarken instantiiert werden, die die temporalen Constraints nicht verletzen, so ist VM_i instantiiert und der durch die Proposition des Vorgangsmodell benannte Vorgang ist erkannt. Alle

Vorgänge, die während der Laufzeit erkannt werden, legen wir in der *qualitativen Historie* ab. Sie stellt damit eine qualitative Beschreibung der Folge von Vorgängen dar.

4.2.3 Unsicherheitsintervalle

In vielen Fällen ist eine sichere Bestimmung des Beginns und des Endes eines Vorganges aufgrund von Werteunsicherheiten und zeitlichen Unsicherheiten nicht möglich. Daher müssen wir auf allen Ebenen der Repräsentation mit diesen Unsicherheitsintervallen umgehen. Auf der Ebene der Vorgangsmodelle bedeuten die Intervalle für Beginn- und Endezeit des Vorgangs Zeiträume, in denen der Beginn bzw. das Ende des Vorgangs möglicherweise stattgefunden hat. Wenn wir nun die zeitliche Relation zweier Vorgänge betrachteten, die beide mit Unsicherheiten bezüglich ihrer Beginn- und Endezeitpunkte behaftet sind, ist es nicht immer möglich eindeutig festzustellen, ob die Vorgänge seriell oder parallel stattfinden.

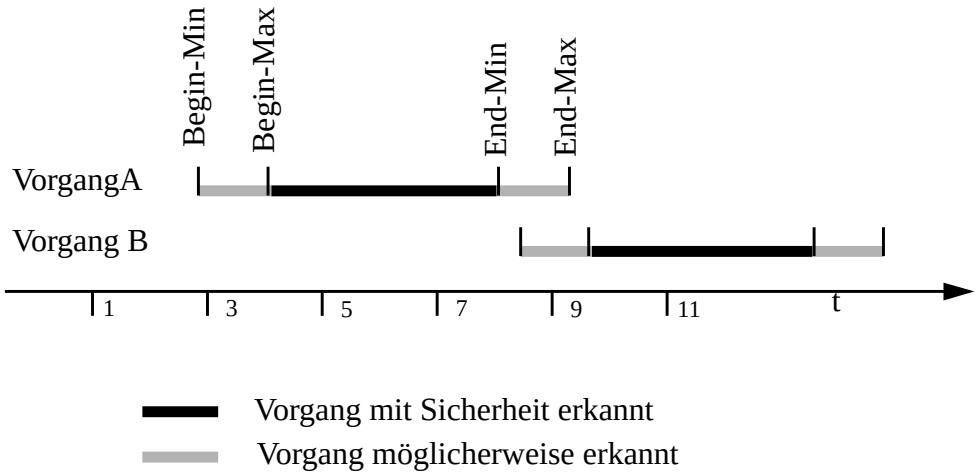

Abbildung 29: Für die zeitliche Reihenfolge der beiden Vorgänge A und B gibt es mehrere Interpretationen, je nachdem welche tatsächlichen Ende- bzw. Beginnzeitpunkte angenommen werden. A und B können in der Relation „after", „meets" oder „overlaps" stehen.

Abbildung 29 zeigt zwei, in ihren Unsicherheitsbereichen überlappende Vorgänge A und B. Sie können seriell sein, wenn das tatsächliche Ende von A kleiner oder gleich dem tatsächlichen Beginn von B ist. Sie können parallel stattfinden, wenn das tatsächliche Ende von A größer als der tatsächliche Beginn von B. Wir müssen beide mögliche Interpretationen in die Betrachtung einfließen lassen. Bei der Modellierung von Vorgängen können solche Beziehungen zwichen zwei Vorgängen durch die konvexen allensche Relation ausgedrückt werden. Diese werden in Ungleichungsbeziehungen zwischen den Minima und den Maxima der Unsicherheitsintervalle übersetzt. Bei der Propagierung von Zeitwerten werden dabei die Minima entlang der Kanten und die Maxima entgegen der Kanten propagiert.

4.3 Repräsentation von B-Regionen

Wir haben bisher Prädikate, primitive Vorgangsmodelle und zusammengesetzte Vorgangsmodelle als Modellierungsmittel für die Vorgangserkennung kennengelernt. Wir verwenden sie dazu, die Domäne so zu strukturieren, daß möglichst viele Teilvorgänge in verschiedenen Modellen wiederverwendbar sind. Die Vorgangs-

modelle bilden eine Hierarchie, deren einfachste Einheiten die primitiven Vorgangsmodelle bilden. Die komplexesten Einheiten der Hierarchie sind die sog. *B-Regionenmodelle*. Sie enthalten primitive oder zusammengesetzte Vorgangsmodelle als Teilvorgänge und können als erweiterte Form von Vorgangsmodellen aufgefaßt werden.

4.3.1 B-Regionenmodelle

Mit B-Regionenmodellen werden Raumausschnitte so definiert, wie sie sich dem Roboter in seinen Meßwerten darstellen. Sie geben damit den Raumausschnitten eine sowohl für den Roboter als auch für den Beobachter interpretierbare Bedeutung. Erkannte B-Regionenmodelle sind die Bausteine aus denen der Roboter die Route, die er für seine Navigation benutzt, zusammensetzt.

DEFINITION 15: Ein **Regionenmodell** ist ein Sextupel **RM** := (N, UP, BP, ZM, VM_i, TC) mit: **N** ist der eindeutige Name des Regionenmodells. **UP** ist die Umgebungsproposition, sie beschreibt die Umgebung, die mit der Erkennung des Regionenmodells erkannt wird. **BP** ist die Bewegungsproposition, sie ist in einen Translations- und einen Rotationsanteil unterteilt und beschreibt die Bewegung, die mit der Erkennung des Regionenmodells erkannt wird. **ZM** sind die Zeitmarken des Vorgangsmodells; also der Beginn RM-B und Endzeitpunkt RM-E des Vorganges. Beide werden durch Intervalle z. B. [$RM\text{-}B_{min}$, $RM\text{-}B_{max}$] bezeichnet, die die Unsicherheit bzgl. der Zeitpunkte beschreiben. **VM_i** sind die Teilvorgänge des Regionenmodells, die gemäß Definition 14 definiert sind. **TC** sind die temporalen Constraints zwischen den Teilvorgängen VM_i und dem Regionenmodell.

Wir nennen ein Regionenmodell erfüllt, wenn alle seine Teilvorgänge in Zeiträumen erkannt sind, die die temporalen Constraints nicht verletzen. Ist ein Regionenmodell erfüllt, so nennen wir die durch die Umgebungs- und Bewegungsproposition definierte B-Region erkannt.

BEISPIEL: Wir definieren die B-Region, die man umgangssprachlich mit „Der Roboter läuft geradeaus durch einen Korridor” beschreiben kann. Die Definition erfolgt mithilfe von „defregion”. Der Name des Modells „left-and-right-wall-p-f0” ist nur von technischer Bedeutung. Dagegen geben die Umgebungsproposition „corridor” und die Bewegungsproposition „(nil . forward)” die Formalisierung der umgangssprachlichen Beschreibung der B-Region an. Hierbei bedeutet „(nil . forward)”, daß der Roboter keine Rotation ausführt und vorwärts fährt. Danach folgen die von der Definition der Vorgangsmodelle mit „defmodel” bekannten Rollenbeschreibungen und Constraints. Die Teilmodelle „parallel-to-wall-f0”, „close-to-wall-f0 ” und „straight-forward” geben die Meßwertverläufe für die B-Region Korridor und Geradeausfahrt an.[6]

[6]Im Unterschied zur Definition von Vorgangsmodellen ist die Angabe der Umgebungsproposition („:environment”) und der Bewegungsproposition („:movement”) bei der Definition von B-Regionen zwingend erforderlich.

```
(defregion left-and-right-wall-p-f0
  :environment  'corridor
  :movement     '(nil . forward)
  :roles        ((?robot :is-a robot))
  :auxilliary-roles
    ( (?left    :is-a left-side
      (?right   :is-a right-side))
  :constraints ((part-of ?left  ?robot)
                (part-of ?right ?robot))
  :submodels ((parallel-to-wall  (?left)
                 :label wl)
              (close-to-wall    (?left)
                :label cl)
              (parallel-to-wall (?right)
                :label wr)
              (close-to-wall    (?right)
                  :label cr)
              (straight-forward (?robot)
                  :label sf)           )
  :label k
  :time-restrictions( (wr (equal) sf)
                      (cr (equal) sf)
                      (wl (equal) sf)
                      (cl (equal) sf)
                      (k  (equal) sf)     ))
```

B-Regionenmodelle sind die räumlichen Grundbausteine aus denen die Route aufgebaut wird. Sie enthalten Angaben über die Bewegung und die Umgebung des Roboters, die zur Erkennung der B-Region notwendig ist.

4.3.2 B-Regionen

Die Definition von B-Regionenmodellen für die Roboterdomäne ordnet bestimmten Meßwertverläufen eine Bedeutung zu. Damit heben wir diese aus dem Meßraum heraus und geben ihnen einen Namen. Dies führt zu einer Aufteilung des Meßraums in solche Meßwertverläufe denen eine Bedeutung zugeordnet ist und solche bei denen das nicht der Fall ist. Die Messungen sind über die Sensorik des Roboters mit den Bewegungsvorgängen verbunden, bei denen sie aufgezeichnet werden. Dadurch werden auch im Konfigurationenraum des Roboters bestimmte Trajektorien, auf denen diese bedeutungstragenden Meßwertverläufe erzeugt werden, ausgezeichnet. Da die Modellierung nur die für die Bedeutung wichtigen Angaben festlegt, kann der Roboter auf verschiedenen Trajektorien einen Meßwertverlauf erzeugen, der das B-Regionenmodell erfüllt. Alle diese Trajektorien gehören zu der durch das Modell definierten Äquivalenzklasse. - Sie gehören zur gleichen B-Region.

DEFINITION 16: Eine **B-Region** $R := \{P \mid P \in [RM]\}$, wobei P eine Trajektorie (vgl. Definition 8) ist und RM ein B-Regionenmodell (vgl. Definition 15). RM definiert eine Äquivalenzklasse [RM] von Trajektorien P, die es erfüllen ($P \in [RM]$). Die Menge der Trajektorien, die Element einer B-Region sind, nennen wir äquivalent.

Verschiedene Trajektorien P können so ähnlich sein, daß es keinen konzeptuellen Unterschied macht, ob P_1 oder P_2 gewählt wird.[7] Vergleichbar ist dies mit einer Straße, die auf der linken oder auf der rechten Seite befahren werden kann. Solange kein Gegenverkehr existiert, macht es keinen Unterschied, welche Trajektorie befahren wird; beide gehören zur selben B-Region. Eine B-Region ist also die Menge von Trajektorien, die ein B-Regionenmodell erfüllen.

Von der Definition her sind B-Regionen somit Äquivalenzklassen sechsdimensionaler Punktfolgen. Allerdings wäre es bei der Modellierung höchst ungeschickt und kompliziert, solche B-Regionenmodelle zu entwerfen, die eine ständige Veränderung in allen sechs Dimensionen des K-Raums für deren Erkennung vorraussetzen. Die Veränderungen in den drei Dimensionen der Lage, sind für ein bewegtes Objekt unvermeidbar und in unserem Sinne erwünscht. Aber die Variation der beiden Bewegungsdimensionen ist beeinflußbar. Unsere Versuche haben gezeigt, das es modellierungstechnisch einfacher ist, B-Regionen zu definieren, die möglichst wenige Wechsel der Bewegungsform für die Erkennung vorraussetzen.

Für die im nächsten Kapitel vorgestellte Navigation setzen wir voraus, daß jedes B-Regionenmodell genau eine gleichförmige Bewegung des Roboters spezifiziert. Dies ist syntaktisch bereits durch die Angabe einer Bewegungsproposition vorgegeben. Ein Wechsel der Bewegung führt demnach zwangsläufig zu einer neuen Region.[8]

BEISPIEL: Abbildung 30 zeigt die räumliche Extension des B-Regionenmodells ECKE-RECHTS-DURCHFAHREN. Die schattiert dargestellte B-Region wird durch die verschiedenen Möglichkeiten für Trajektorien, auf denen das Vorgangsmodell erkannt wird, aufgespannt. Ein auf der gestrichelt gezeichneten Trajektorie fahrender Roboter würde das Vorgangsmodell nicht erkennen. Daher gehört diese nicht zur B-Region, obwohl Anfang und Ende auf Positionskoordinaten liegen, auf denen auch Trajektorien der B-Region liegen.

[7] Wir verwenden hier T als Abkürzung für Trajektorie, um eine Verwechslung mit der Abkürzung für die Zeit T zu vermeiden. Als mnemotechnische Hilfe können wir „Pfad" verwenden.

[8] In der graphischen Darstellung von Trajektorien kann es aber trotzdem vorkommen, daß die gezeichnete Linie viele verschiedene Kurven zeigt. Damit sind nicht verschiedene Motoransteuerungen gemeint, sondern solche Abweichungen von der beabsichtigten Bewegung, die durch Schlupf entstehen (vgl. Seite 16).

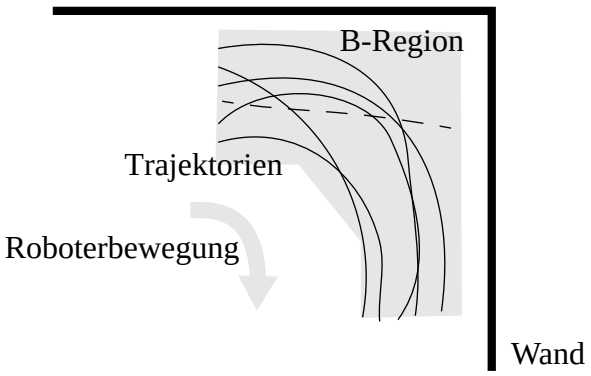

Abbildung 30: Fünf äquivalente Trajektorien, die alle zu einer B-Region gehören, die durch das Modell ECKE-RECHTS-DURCHFAHREN definiert wird. Die gestrichelt gezeichnete Trajektorie ist nicht mit den fünf anderen äquivalent und damit auch nicht Element der B-Region.

Bei der graphischen Darstellung von Trajektorien ist es nicht möglich, alle fünf Dimensionen des K-Raums darzustellen. Man muß daher beachten, daß die Geschwindigkeit und die Orientierung des Roboters, obwohl nicht dargestellt, mit zur Trajektorie und somit auch zur B-Region gehören. Die Verwendung von B-Regionenmodellen zur Definition von B-Regionen bringt es mit sich, daß eine bestimmte Positionskoordinate in verschiedenen B-Regionen liegen kann, die alle unterschiedliche Orientierungen oder Geschwindigkeiten vorraussetzen. Dies widerspricht leider der intuitiven Vorstellung einer B-Region als der Beschreibung eines Teils der Grundfläche. Denn wenn man umgangssprachlich von einer B-Region spricht, so meint man meist eine Menge von Positionskoordinaten, die eine gemeinsame Eigenschaft haben; z. B. in der Nähe der Wand zu sein. In Abschnitt 4.3.4 werden wir eine Möglichkeit skizzieren, die verschiedenen, vom Roboter erkannten B-Regionen auf eine, der intuitiven Auffassung von „B-Region" näher kommende, Form der Repräsentation abzubilden.

Wir verwenden B-Regionenmodelle dazu, den Raum, bei einer gegebenen Fahrt des Roboters, in bedeutungstragende B-Regionen zu partitionieren, indem Teile des Raums, die in einer bestimmten Weise durchfahren werden, mit Namen belegt werden. Diese Namen drücken die Bedeutung aus, die ein bestimmter Meßwertverlauf hat. Die B-Regionenmodelle übersetzen damit die Meßverläufe des Roboters in qualitative Aussagen, die für den Beobachter verständlich sind. Es ist zwar denkbar, daß ein Beobachter die Meßdaten der quantitativen Historie mithilfe seiner Erfahrung ebenfalls interpretieren könnte, aber aufgrund der Datenmenge wäre der Aufwand dafür doch enorm.

Abbildung 31 zeigt die Benennung von erkennbaren B-Regionen in einem einfachen Raum. Typischerweise überlappen angrenzende B-Regionen sowohl zeitlich als auch räumlich, denn sie enthalten teilweise gleiche Positionskoordinaten. Z. B. enthalten die B-Regionenmodelle mit der Umgebungsproposition ECKE ein

Wandstück vor und nach der Ecke, welches ebenfalls in einem B-Regionenmodell mit der Umgebungsproposition WAND enthalten ist.

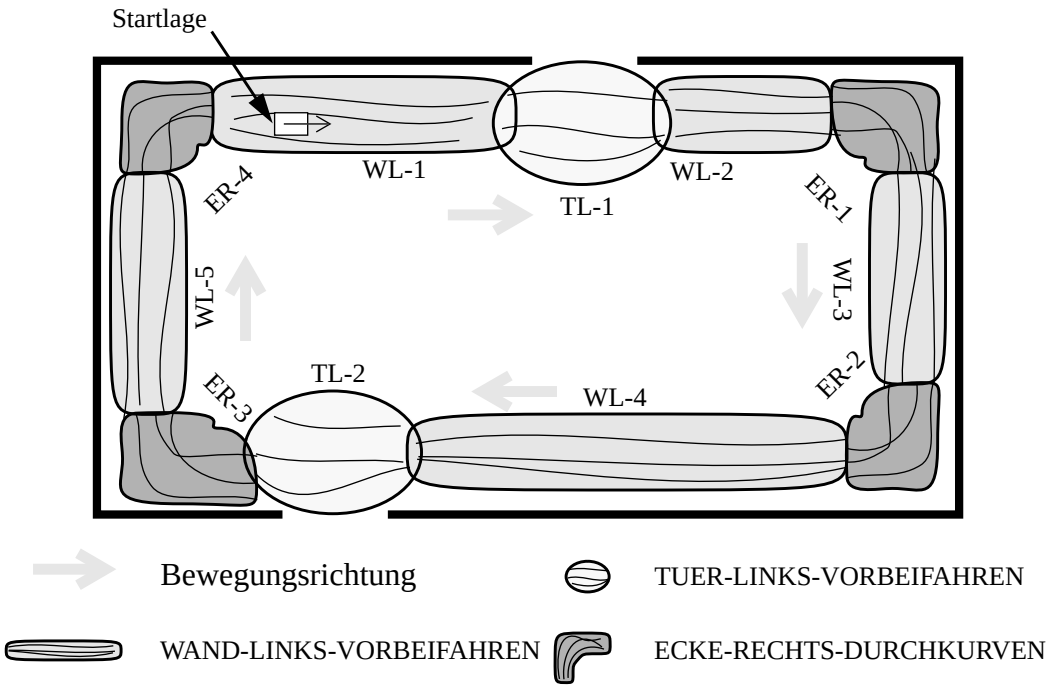

Abbildung 31: Die Vorgangsmodelle WAND-LINKS-VERFOLGEN, TUER-LINKS-VORBEIFAHREN und ECKE-RECHTS-DURCHKURVEN teilen die Grundfläche des Raumes in elf einander teilweise überlappende B-Regionen auf. Erkannte B-Regionen werden mit einem eindeutigen Namen benannt.

Wenn eine B-Region erkannt wird, erhält sie einen für diese Route eindeutigen Namen, der aus dem Namen des B-Regionenmodells und einer Zahl generiert wird. Die Zahl gibt an, die wievielte Instanz eines B-Regionenmodells in der aktuellen Roboterfahrt erkannt wurde. In Abbildung 31 sind die Namen der Instanzen abgekürzt dargestellt.

Während der Fahrt gleicht die Vorgangserkennung die definierten Modelle mit den eingehenden Meßverläufen ab und versucht möglichst viele Vorgangs- und B-Regionenmodelle mit konkreten Zeitwerten zu instantiieren. Wenn in der Modellierung der Domäne Mehrdeutigkeiten enthalten sind, so werden alle auf die aktuelle Messung passenden Modelle instantiiert. Daher muß bei der Definition der B-Regionenmodelle darauf geachtet werden, daß sie möglichst genau das Verhalten der Meßwerte beschreiben, das mit der qualitativen Beschreibung gemeint ist.

BEISPIEL: Wenn bei den B-Regionenmodellen WAND-VERFOLGEN (vgl. Abbildung 31) keine Obergrenze für den Abstand zur Wand definiert wäre, so würde neben WAND-LINKS-VERFOLGEN auch eine B-Region WAND-RECHTS-VERFOLGEN erkannt werden, da die untere Wand ebenfalls innerhalb der Reichweite der Sensoren liegt. Dies würde dazu führen, daß die verschiedenen B-Regionen so sehr überlappen, daß die Ortsbestimmung des Roboters, anhand der

B-Region in der sich befindet, nicht mehr sinnvoll wäre.

Bei der Modellierung der B-Regionen muß demnach darauf geachtet werden, daß die Modelle genügend präzise formuliert sind. Ist dies nicht der Fall, so wird die Lokalisierung des Roboters anhand der B-Region in der er sich gerade befindet zu ungenau, um daraus Schlüsse für die weitere Steuerung zu ziehen.

Bei der Erkennung von B-Regionen können Vorgänge wie WAND-LINKS-VERFOLGEN oder ECKE-RECHTS-DURCHFAHREN in einem bestimmten Zeitintervall erkannt werden. Dadurch werden die Modelle mit konkreten Zeitmarken instantiiert.[9] Aus dem instantiierten B-Regionenmodell können die Propositionen bzgl. der Umgebung und Bewegung des Roboters abgeleitet werden. Durch die Erkennung einer B-Region erhalten wir die drei folgenden Informationen:

- Die Dauer des Vorgangs, sowie die absoluten Anfangs- und Endzeitpunkte.
- Die Bewegung, die der Roboter während des Vorgangs ausführte, mit ihrem Translations- und Rotationsanteil.
- Die Umgebung, durch die der Roboter fuhr.

Diese Informationen werden in der Route abgelegt, um sie in der Reproduktionsphase abzuarbeiten. Dabei wird die Bewegung ausgeführt und die Umgebung sowie die Dauer des Vorgangs mit der gelernten verglichen. Wir kommen darauf im Abschnitt 5.3 auf Seite 67 zurück.

4.3.3 Ausdehnung von B-Regionen

Da eine Bewegung des Roboters immer mit einer konstanten Geschwindigkeit vorrausgesetzt wird, können wir leicht aus der Dauer eines Vorgangs auf die Ausdehnung des entsprechenden Ortes im Konfigurationenraum (siehe Definition 3) schließen, wenn der Vorgang nicht mit der speziellen Bewegungsform Stillstand verbunden ist: Je länger ein Vorgang gedauert hat, desto mehr räumliche Veränderungen müssen während des Vorgangs stattgefunden haben. Die Ausdehnung des Vorgangs im K-Raum kann durch die Dauer des Vorganges als zusätzliche Information zum Ort in der Route verwendet werden. Um einen Ort dann in der Route wiederzuerkennen, darf die eingetragene Dauer nicht zu stark von der beim zweiten Besuch dieses Ortes differieren.

BEISPIEL: Die in Abbildung 32 gezeigten Umgebungen werden durch das gleiche Vorgangsmodell beschrieben, da sie beim Durchfahren die gleichen Sensorverläufe und Bewegungsmuster erzeugen. Bei beiden Fahrten bleiben die Abstände links und rechts weitgehend konstant und der Roboter fährt eine Rechtskurve, die nach einem Rotationsanteil von

[9] Vgl. Seite 43.

90^o endet.

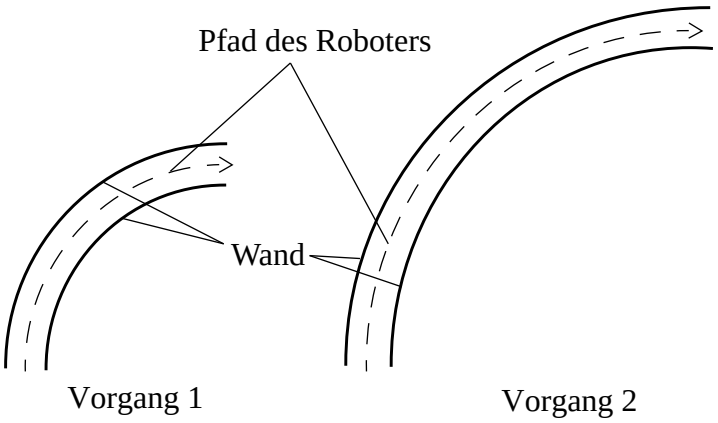

Abbildung 32: Zwei Vorgänge, die durch die Dauer zwischen Beginn und Ende unterschieden werden.

4.3.4 Typisierung von B-Regionen

Die Benennung der B-Regionenmodelle ist auf den ersten Blick vollkommen bedeutungslos, denn für den Roboter ist es egal, ob das Modell AN-DER-TÜR-VORBEIFAHREN oder HUGO heißt, denn für den Roboter sind nur die Beziehungen zwischen den Teilvorgänge von Bedeutung. Wenn wir aber nun davon ausgehen, daß die Modellierung so genau erfolgt ist, daß die B-Regionenmodelle eindeutige Beschreibungen der Umgebungen sind, die in der Umgebungsproposition angegeben sind, so haben wir eine brauchbare Typisierung der B-Regionenmodelle zur Verfügung. Damit können wir dann die verschiedenen B-Regionen, wie z. B. AN-DER-TÜR-VORBEIFAHREN und TÜR-DURCHFAHREN, über die gemeinsame Umgebungsproposition TÜR miteinander vergleichen. Wir könnten dann eine Abbildung von einer Menge von B-Regionenmodellen auf eine lokal eindeutige Beschreibung eines Positionsbereiches definieren. Sie entspricht einer Disjunktion von B-Regionenmodellen, die die gleiche Umgebungsproposition enthalten. Die Menge von Positionskoordinaten die in der Nähe der Tür liegen, könnte dann durch die Disjunktion von B-Regionenmodellen beschrieben werden, die die Umgebungsproposition TÜR enthalten.

Eine Menge von Positionskoordinaten eindeutig durch eine Disjunktion von B-Regionenmodellen zu beschreiben, ist aufgrund der Vielzahl von Bewegungsmöglichkeiten nicht möglich. Aber bei einer geschickten Einschränkung des Roboterverhaltens ist es denkbar, eine Beschreibung von Räumen in dieser Weise zu schaffen. Mit einer solchen Beschreibung wäre dann die Grundlage dafür gelegt, räumliche Inferenzen zu generieren. Durch einen Abgleich von räumlich symmetrischen B-Regionen wie z. B. in AN-TÜR-LINKS-VORBEIFAHREN oder AN-TÜR-RECHTS-VORBEIFAHREN könnte ein Rückweg für eine gelernte Route generiert werden. Dieser Ansatz wurde in der vorliegenden Arbeit nicht weiter verfolgt, da dies für das hier behandelte Anwendungsszenario nicht erforderlich war.

4.4 Zusammenfassung

Der Roboter wird in der Simulation von einer unabhängigen Steuerung durch seine Umgebung gefahren. Die dabei aufgezeichneten Messungen seiner Sensoren werden in der quantitativen Historie mit einem Zeitstempel versehen abgespeichert. Hier wäre z. B. ein Eintrag wie ([2.4 2.5] us-5-val [2310 2426]) möglich, wobei [2.4 2.5] das Zeitintervall der Messung, „us-5-val" den Sensor und [2310 2426] das Intervall des Meßwertes bezeichnet.

Dies ist die Eingabe für die Vorgangserkennung. Um Meßwerte zu beliebigen Zeitpunkten überprüfen zu können, werden mithilfe der Rekonstruktoren bei Bedarf kontinuierliche Meßwertverläufe generiert. Im einfachsten Fall besteht die Rekonstruktion aus einer linearen Interpolation der dem angeforderten Zeitpunkt benachbarten Messungen. Es sind aber auch komplexere Interpolationen vorstellbar.[10]

Die qualitative Historie besteht aus den Prädikaten, die aufgrund der Meßwertverläufe erkannt werden konnten. Sie stellt damit die erste Abstraktion der Messungen in qualitativen Termini dar. Hierin sind z. B. Aussagen wie "us-1-val ist im Intervall [10...20] fallend" enthalten. Dies ist gegenüber der Darstellung in der quantitativen Historie bereits eine deutliche Datenreduktion, die die wesentlichen Informationen erhält.

Mit den Vorgangsmodellen werden komplexe Vorgänge spezifiziert. Die Vorgangsmodelle, die mit der qualitativen Historie in Einklang gebracht werden können, werden zu erkannten Vorgängen instantiiert. Ein erkannter Vorgang könnte z. B. „(Ecke, rotation-rechts, 100, 200)" sein. Hierbei bezeichnet „Ecke" die erkannte Umgebung, „rotation-rechts" die erkannte Bewegung des Roboters und 100, 200 bezeichnet das Zeitintervall, in dem der Vorgang stattgefunden hat.

Die drei Informationen werden nun im Raummodul getrennt weiterverarbeitet. Die Umgebungsinformation „Ecke" wird mit vorher erkannten, und in der Karte abgelegten, Umgebungsinformationen in die Relation gesetzt, die sich durch die räumliche Relation „rotation-rechts" ergibt. Dem erkannten Objekt wird dazu ein eindeutiger Name im System zugeordnet. Danach wird geprüft, ob es sich dabei um einen in der Karte bereits vorhandenen Ort handeln könnte. Dies leistet der Inferenzmechanismus des Raummoduls, dessen Datenstruktur die qualitative Karte ist.

[10] Mehr dazu in [Lahres & Trowe 94].

5 Navigation

Im vorangegangenen Kapitel haben wir eine Repräsentationsform kennengelernt, die es ermöglicht „Bausteine" der Roboterumgebung zu beschreiben. Dabei werden nur die wesentlichen Eigenschaften einer Region spezifiziert, so daß die gleiche B-Region für verschiedene Trajektorien des Roboters erkannt werden kann, solange die Abweichungen nicht den Charakter der B-Region verändern. Wir werden im Folgenden davon ausgehen, daß ein geeigneter Satz von B-Regionenmodellen vorliegt, der die Bausteine der betrachteten Welt für die gegebene Navigationsaufgabe ausreichend differenziert beschreibt. Der Roboter ist dadurch in der Lage die Bewegungsregion, in der er sich gerade befindet, während der Fahrt (sozusagen im Vorbeigehen) zu erkennen.

B-Regionenmodelle sind allgemeine, wiederverwendbare Beschreibungen von Umgebungselementen des Roboters. Der Ort des Roboters bezüglich seiner Umgebung ist durch die erkannte B-Region i. A. nicht eindeutig bestimmt (es sei denn es gibt jedes Umgebungselement nur ein einziges Mal in der Roboterwelt). Die B-Regionen sind somit Marken, die den Typ der Umgebungsform bestimmen. Zur Navigation benötigen wir aber eine eindeutige Identifikation der Region in der sich der Roboter gerade befindet. Die Route leistet diese Identifikation, indem sie die B-Regionen in der Reihenfolge ihrer Erkennung mit ihren zeitlichen Beziehungen anordnet. Die B-Region wird zu einer Landmarke, die den Ort des Roboters eindeutig beschreibt. Die Ortsbestimmung ist qualitativ, in dem Sinne, daß nicht die genaue Lage des Roboters, aber die, für die Fortsetzung der Aufgabe wesentliche, Information bekannt ist.

Um eine Ladung auf einem gelernten Weg zu transportieren, muß der Roboter in der Lage sein, eine Bewegung zu lernen und sie fehlertolerant nachzuvollziehen. Die Navigation zerfällt dabei in verschiedene Phasen, die wir mit *Akquisitionsphase*, *Reproduktionsphase* und *Recovery* bezeichnen.

In der *Akquisitionsphase* steuert der Trainer den Roboter auf einer Beispieltrajektorie vom Start zum Ziel. Dabei gleicht der Roboter die aufgenommenen Meßwerte inkrementell mit allen definierten B-Regionenmodellen ab und versucht sie zu instantiieren. Dabei können mehrere Modelle gleichzeitig erkannt werden, die dann parallel in der Route abgelegt werden. Der Roboter generiert einen eindeutigen Namen für die erkannten B-Regionen und baut sie mit ihren zeitlichen Relationen zu den bisher erkannten Instanzen von B-Regionenmodellen in die Route ein. Dabei werden die Umgebungs- und Bewegungsproposition sowie die Zeitdauerangaben gespeichert.

Der Abgleich der Meßwerte mit den Definitionen der B-Regionenmodelle erfolgt schritthaltend mit der Bewegung des Roboters. Dies ist nicht unbedingt notwendig, denn man könnte sich auch vorstellen, daß der Roboter zunächst bis zum Ziel gesteuert wird, dabei die Daten aufzeichnet, um dann die entsprechenden Modelle retrospektiv zu erkennen. Da in diesem Fall die Meßdaten für Beginn und Ende einer Region gleichzeitig vorliegen, könnte der Aufwand für die Verwaltung verschiedener Interpretationen der Meßwerte (in Form von Modellen mit gleichem Beginn aber unterschiedlichem Ende) erheblich reduziert werden. Der entscheidende Nachteil liegt bei diesem Vorgehen darin, daß der Trainer erst am Ende der Route erfahren würde, ob alle B-Regionen korrekt erkannt wurden. Wir haben uns deshalb für eine inkrementelle Erkennung entschlossen, die ein direktes Feedback ermöglicht.

Die *Route* ist ein gerichteter Graph mit einem eindeutigen Startknoten „Start” und einem eindeutigen Endeknoten „Ziel”. Die restlichen Knoten sind Beschreibungen von B-Regionen, die in der Reihenfolge ihrer minimalen Beginnzeitpunkte im Graphen angeordnet werden. Die Kanten geben die zeitlichen Relationen zwischen zwei aufeinanderfolgenden B-Regionen an. Dabei sind die Relationen „overlaps”, „meets” und „after” sowie deren disjunktive Kombinationen möglich. Ein Knoten der Route kann mehrere Nachfolger und Vorgänger haben. Überlappende B-Regionen können dabei keine widersprüchliche Bewegungsproposition besitzen, da immer wenn eine neue Bewegung vom Trainer eingeleitet wird, eine neue Region beginnt. Diese steht in der Relation „meets” zu den bisher aktiven B-Regionen. B-Regionen können demnach nur dann überlappen, wenn sie die gleiche Bewegungsproposition haben.

Zu Beginn der *Reproduktionsphase* wird der Roboter in der gleichen B-Region positioniert, die bei der Akquisition der Route Ausgangspunkt war. Nach dem Start des Roboters reproduziert dieser die Bewegung, die in der Bewegungsproposition des aktuellen Knotens abgespeichert ist. Dabei gleicht er die Meßwerte der Sensoren mit der Spezifikation des entsprechenden B-Regionenmodells ab. Solange das entsprechende B-Regionenmodell erfüllt werden kann, ist der Roboter in der richtigen B-Region. Werden Messungen aufgenommen, die nicht mehr mit dem aktuellen B-Regionenmodell in Einklang gebracht werden können, so wird eine Fehlerkorrektur, das Recovery, eingeleitet. Genauso wird verfahren, wenn das Durchfahren der B-Region zeitlich wesentlich von der aufgezeichneten B-Region abweicht.

5.1 Route

Im Alltag verwenden wir Routen als Wegbeschreibungen, in denen wir Landmarken miteinander durch Bewegungsanweisungen verbinden. Landmarken können dabei sowohl punktförmige (z. B. eine Straßenecke) als auch ausgedehnte Objekte (z. B. eine Allee) sein, die einen Ort eindeutig identifizieren.[1] Auch wenn wir viele in Routen verwendete Objekte als punktförmig ansehen, so sind sie bei genauerer Betrachtung meist trotzdem ausgedehnt. Es kommt hierbei, wie so oft, auf die Auflösung bei der Betrachtung an. Wir werden in den Beispielen dieser Arbeit nur Landmarken verwenden, die eine Ausdehnung haben, weil wir so eine größere Abstraktionsleistung erreichen. Allerdings kann der Formalismus ebenfalls mit punktförmigen Landmarken umgehen, die in bestimmten Fällen sinnvoll einsetzbar sind.

In Abschnitt 4.3 haben wir die Modellierung von B-Regionen kennengelernt. Erkannte B-Regionen enthalten Informationen über die Umgebung, Bewegung und die zeitliche Ausdehnung bei der Erkennung. Wir werden sie als Landmarken verwenden, die wir zu einer Route zusammensetzen.

[1] Die Differenzierung zwischen Landmarken und (nicht eindeutigen) Marken ist [Atiya 95] entnommen.

DEFINITION 17: Sei R eine Menge von B-Regionen, L := {o, om, oma, m, ma, a} eine Menge von Kantenbeschriftungen, a, b ∈ R und l, l' ∈ L.
Eine **Route** ist ein Graph (R, L, K, Start, Ziel) mit $K \subseteq R \cup \{Start\} \times L \times R \cup \{Ziel\}$, ist die Menge der Kanten mit der Eigenschaft: $(a, l, b) \in K \Rightarrow (b, l', a) \notin K$ und ∃ Weg (offener, einfacher Kantenzug) vom Start- zum Zielknoten.

Eine Route ist ein gerichteter, zyklenfreier Graph mit einem ausgezeichneten Startknoten, der mit Start gekennzeichnet ist und einem ausgezeichneten Endknoten, der mit Ziel gekennzeichnet ist. Alle Knoten der Route außer Start und Ziel bezeichnen B-Regionen. Sie sind alle Nachfolger von Start und Vorgänger von Ziel. Die Knoten werden mit einem eindeutigen Namen benannt, der auf das zugehörige B-Regionenmodell verweist. Die Kantenbeschriftungen geben die zeitliche Relation zwischen Vorgänger- und Nachfolgerregion an. Dabei steht *o* für „overlaps", *m* für „meets" und *a* für „after". Die Beschriftungen *om*, *oma* und *ma* stehen für Disjunktionen der Einzelbeschriftungen. Sie sind notwendig, um die zeitlichen Unsicherheiten bei Beginn und Ende von erkannten Regionen in der Route auszudrücken. Die Knoten sind im Graphen nach der zeitlichen Reihenfolge ihrer Beginnzeitpunkte angeordnet.

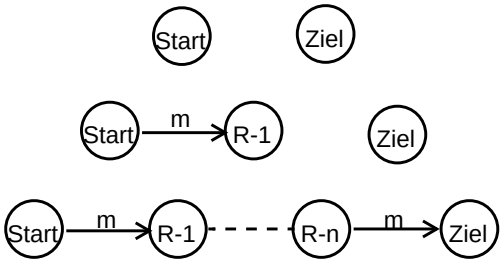

Abbildung 33: Initiale Route mit den Knoten Start und Ziel (oben). Das B-Regionenmodell R wird erkannt und mit der Relation zu Start in die Route eingefügt (mitte). Der Namen des Knotens ist für die Route eindeutig generiert und verweist auf das B-Regionenmodell R. Die zeitliche Relation zwischen R-1 und dem Ziel ist noch nicht bekannt. Beim Endekommando wird die Relation zwischen den zuletzt erkannten B-Regionen mit dem Zielknoten in den Graphen eingegangen (unten).

Die Route hat initial die Form (∅, L, ∅, Start, Ziel). Sie enthält nur die Knoten Start und Ziel. Die erkannten B-Regionen werden dann in der Akquisitionsphase sukzessive mit den zeitlichen Relationen zu ihren Nachbarknoten in den Graphen eingefügt. Es ist durchaus möglich, daß ein Knoten mehrere Nachfolgerknoten besitzt, da die Vorgangserkennung alle B-Regionenmodelle instantiiert, die auf die Meßverläufe passen. Dies führt zu alternativen Kantenzügen. Durch die Art der Generierung in der Akquisitionsphase enthalten B-Regionen, die hinter einer Gabelung liegen, immer dieselbe Bewegungsproposition, so daß Gabelungen nicht zu Mehrdeutigkeiten in der Reproduktionsphase führen:

SATZ 2: Sei (R, L, K, Start, Ziel) eine Route mit $l \in L$;
$a, a_1, ..., a_n, b_1, ..., b_m \in R$; $i \in \{1, ..., n\}$; $j \in \{1, ..., m\}$ und K* die transitive Hülle von K.
$\forall (a, l, a_i), (a, l, b_j) \in K^*$ und $(a_i, l, b_j) \notin K^*$ gilt:
$BP(a_i) = BP(b_j)$.

Umgekehrt führt ein Wechsel der Roboterbewegung dazu, daß alle B-Regionen mit der alten Bewegungsproposition zuende gehen. Vom Weltmodell ist zu fordern, daß bei einem Wechsel der Geschwindigkeit sofort mindestens eine neue B-Region erkannt werden kann. Dadurch stehen die neu generierten Knoten in der Relation „meets" mit den bisher letzten Knoten des Graphen:

SATZ 3: Sei (R, L, K, Start, Ziel) eine Route mit $l \in L$, $a, b, c, d \in R$, und K* die transitive Hülle von K.
$\forall (a, l, c) \notin K^*$ mit $BP(a) = BP(b)$ und $BP(b) \neq BP(d)$ gilt:
(a, meets, b), (c, meets, d),
(a, meets, d) und (c, meets, b) $\in$ K.

Da die Bewegungsproposition Teil der Definition der B-Regionenmodelle ist, muß das bisherige B-Regionenmodell zuende gehen, wenn ein Wechsel der Geschwindigkeit stattfindet. Es könnte aber sein, daß kein B-Regionenmodell existiert, das auf die neue Geschwindigkeit in der aktuellen Umgebung paßt. Dies führt dazu, daß der nächste Knoten in der Relation after zum letzten stehen würde. Dann „übersieht" der Roboter den Wechsel der Geschwindigkeit und verpaßt ihn bei der Reproduktion der Route. Solche Modellierungsfehler führen zu einer Unterbrechung der Akquisition.

5.2 Akquisitionsphase

Bevor der Roboter eine Route lernen kann, muß die Modellierung der Welt mit Prädikaten, Vorgangsmodellen und B-Regionenmodellen soweit abgeschlossen sein, daß der Roboter in möglichst jedem der, auf der geplanten Route erreichten, Bewegungszustände eine B-Region erkennen kann. Denn wenn es Zustände gibt, in denen der Roboter nicht weiß in welcher B-Region er sich gerade befindet, so kann er nicht prüfen, ob er sich noch auf dem richtigen Weg befindet. Jede Lücke im Weltmodell führt dazu, daß der Roboter, wenn er auf eine solche Lücke trifft, orientierungslos ist. Je größer eine solche Lücke ist, desto schwerwiegender für die Reproduzierbarkeit der Route.

Die Akquisition einer Route wird durch den Trainer eingeleitet, der den Roboter auf seine Startlage bringt und dies dem Roboter durch das Startkommando bekannt gibt. Die Vorgangserkennung vergleicht nun zu jedem Zeitschritt die rekonstruierten Meßverläufe mit den im Weltmodell definierten Prädikaten, Vorgangsmodellen und B-Regionenmodellen. Die Messungen sind typischerweise zunächst konstant, da der Roboter zu Beginn der Programmierung stillsteht. Alle B-Regionenmodelle, die auf den Meßverlauf passen, werden als teilweise erkannt markiert. Erst wenn das Ende eines Modells festgestellt wurde, gilt die entsprechende B-Region als erkannt, denn wir versuchen mit der Vorgangserkennung maximal lange Instanzen der Modelle zu produzieren.

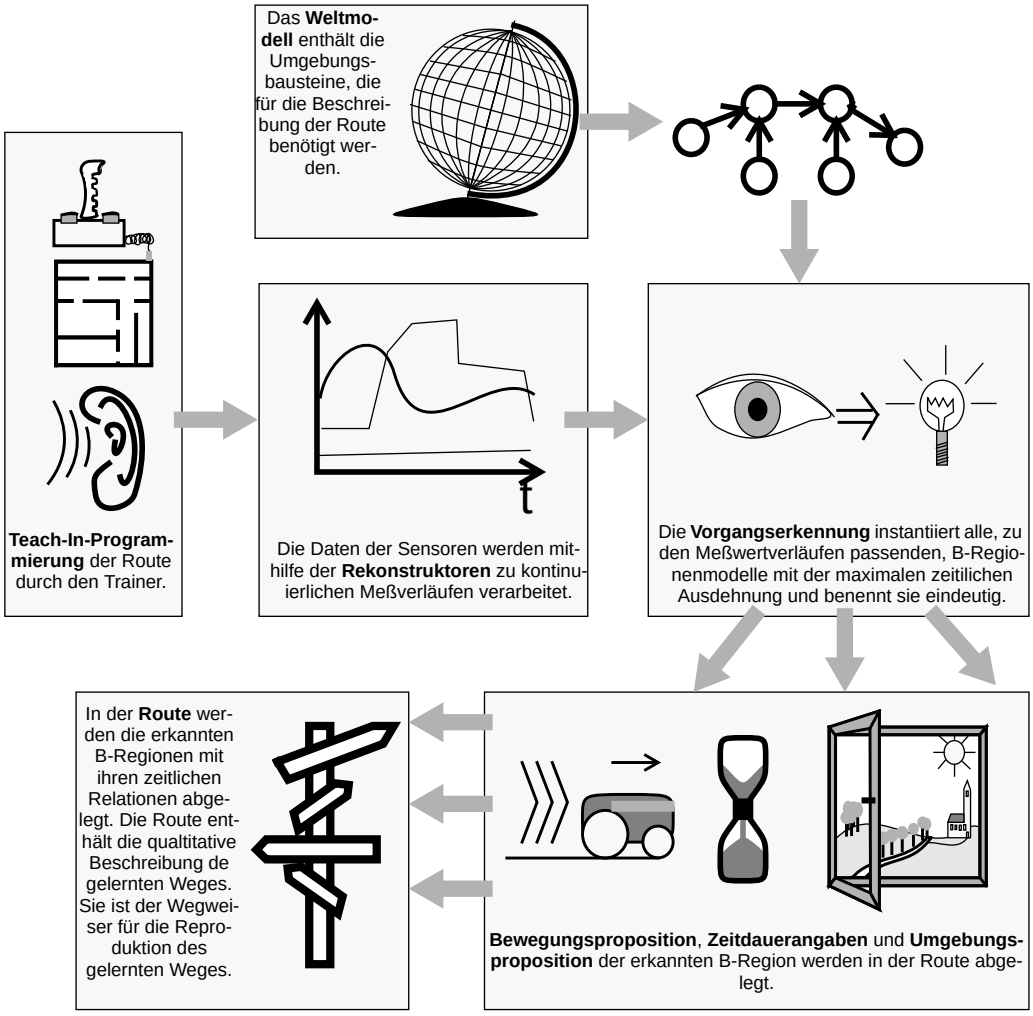

Abbildung 34: Überblick über die Akquisitionsphase der Route: Der Roboter wird durch den Trainer auf dem Weg gefahren, den er lernen soll. Die Sensoren des Roboters erzeugen dann den Input für die Vorgangserkennung, die die Meßwertverläufe mit den Definitionen der Prädikate, Vorgangs- und B-Regionenmodelle abgleicht, und instantiiert die erkannten B-Regionenmodelle mit den entsprechenden Zeitmarken ihrer Erkennung. Aus den Instanzen der B-Regionenmodelle kann dann die Bewegungs-, Umgebungs- und Dauerinformation abgelesen werden. Die Informationen werden dann in der Route mit den bisher erkannten B-Regionen in die entsprechende zeitliche Relation gesetzt und mit einem eindeutigen Namen in den Graphen eingetragen.

Wenn der Roboter vom Trainer losgefahren wird, verändert sich die Geschwindigkeit des Roboters. Da diese in den Modellen mit spezifiziert ist, können alle Modelle, die Stillstand als Bewegungsproposition vorraussetzen, nicht mehr erfüllt sein. Ihr Ende wird erkannt. Für alle erkannten B-Regionenmodelle werden nun eindeutige Namen generiert, die mit den jeweiligen Propositionen und Zeitdauerangaben in die Route eingetragen werden. Die Beginn- und Endezeitpunkte werden mit den bisher zuletzt erkannten B-Regionen in Relation gesetzt und mit den entsprechenden Kantenbeschriftungen als Knoten in die Route eingefügt. Da wir uns im Augenblick noch am Anfang des zu lernenden Weges befinden, müssen die B-Regionen nur mit dem Start- und dem Zielknoten verbunden werden.[2] Neu

[2]Vergleiche Abbildung 33.

erkannte B-Regionen werden dann entsprechend nach den bisher erkannten in den Graphen eingefügt, bis der Trainer das Endekommando gibt. Das teilt dem Roboter mit, daß die Route abgeschlossen ist und die Relation zwischen dem letzten erkannten Vorgang und dem Zielknoten eingetragen werden kann. Die Route ist damit abgeschlossen.

Betrachten wir nun die Routengenerierung an einem Beispiel. Wir haben es zum Teil schon auf Seite 6 kennengelernt.

BEISPIEL: Abbildung 35 zeigt die Trajektorie auf der der Roboter die Route vom Start zum Ziel lernen soll.

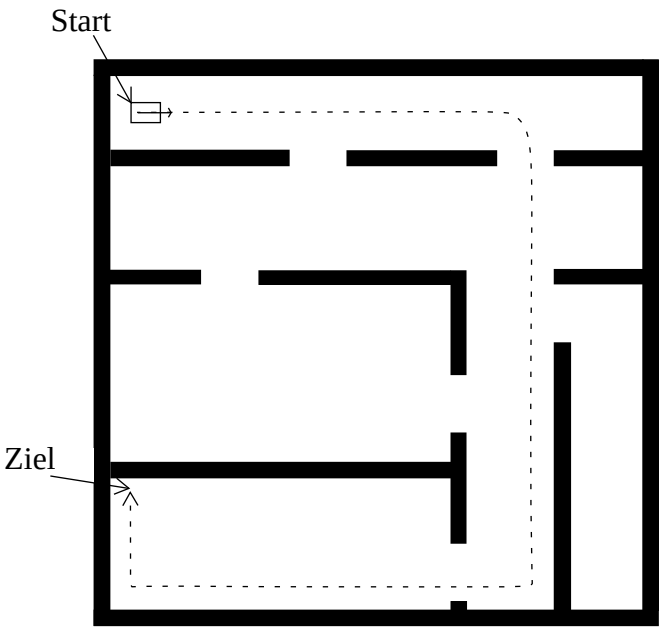

Abbildung 35: Der Roboter wird vom Trainer auf der gestrichelt eingezeichnete Trajektorie von der Startlage zur Ziellage gesteuert.

Vorraussetzung für den Algorithmus zur Vorgangserkennung ist die Modellierung der Prädikate, Vorgangs- und B-Regionenmodelle. Die Modelle sind zu einem großen Teil von dem Weg, den der Roboter lernen soll abhängig. Es empfiehlt sich möglichst nur die, für den zu lernenden Weg benötigten, Modelle im Weltmodell zur Überprüfung auszuwählen. Denn alle nicht benötigten Modelle bedeuten unnötigen

Rechenaufwand für den Algorithmus. Für den in Abbildung 35 dargestellten Weg haben wir die folgenden B-Regionen aus dem Weltmodell ausgewählt:

Abkürzung	Name des Vorgangsmodells	Umgebung	Translation	Rotation
CR-f0	corner-right-f0	Ecke rechts	vorwärts	
CR-0r	corner-right-0r	Ecke rechts		rechts
DEb-00	front-free-rest-blokked-m-00	Sackgassenbeginn		
DEb-f0	front-free-rest-blokked-m-f0	Sackgassenbeginn	vorwärts	
FC-f0	left-right-wall-p-f0	Korridor	vorwärts	
FCDr-f0	follow-corr-r-open-f0	Tür rechts	vorwärts	
FCWEr-f0	follow-corr-du-r-f0	Wandende rechts	vorwärts	
FCWEr-f0	du-r-f0	Wandende rechts	vorwärts	
FCWEr-fr	du-r-fr	Wandende rechts	vorwärts	rechts
FlW-f0	follow-left-wall-f0	Wand links	vorwärts	
FlW-f0	follow-right-wall-f0	Wand rechts	vorwärts	
FlWDr-f0	follow-l-r-open-f0	Tür rechts	vorwärts	
FlWEr-f0	follow-l-du-r-f0	Wandende rechts	vorwärts	
PlW-f0	parallel-left-wall-f0	Wand links	vorwärts	
PrW-f0	parallel-right-wall-f0	Wand rechts	vorwärts	
VbW-00	vert-rear-wall-00	Wand		
VbW-f0	vert-rear-wall-f0	Wand	vorwärts	

Table 3: Auswahl von B-Regionenmodellen aus dem Weltmodell für die Akquisition der Route in Abbildung 36.

Die Aufzeichnung der Route beginnt damit, daß der Trainer das Startkommando gibt. Jede erkannte B-Region wird benannt und in die Route aufgenommen. Die folgende Abbildung zeigt die dabei gene-

rierte Route. Dabei sind die Namen der Knoten aus den Abkürzungen in der Modelle aus der obigen Tabelle generiert. Sie sind damit ein Verweis auf die Definition des B-Regionenmodells.

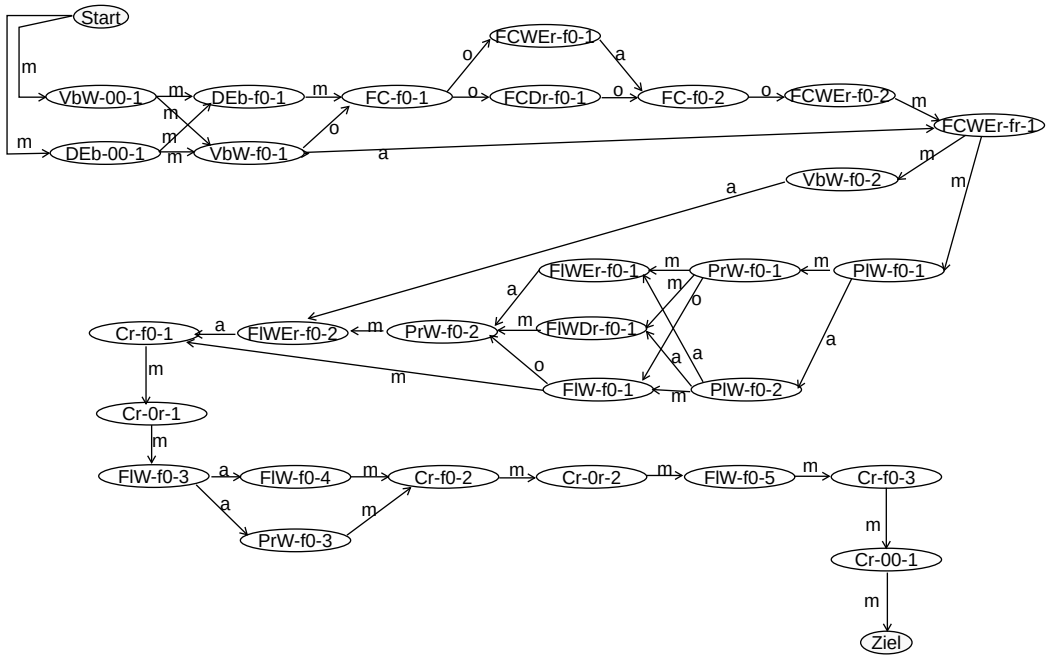

Abbildung 36: Route zur Trajektorie in Abbildung 35.

5.2.1 Algorithmus zur Routenerstellung

Die Route ist die Repräsentation der Bewegung des Roboters durch seine Umgebung. Sie wird inkrementell während der, durch den Trainer gesteuerten, Fahrt des Roboters aufgebaut. Dabei werden die Prädikate, Vorgangs- und B-Regionenmodelle von der Vorgangserkennung mit den Meßwertverläufen der Sensoren sowie mit der Bewegung des Roboters abgeglichen. Dabei können die Beobachtungen zu folgenden Ereignissen der Vorgangserkennung führen:

- Eine neue Instanz eines Modells wird instantiiert: Dabei wird zumindest der Beginn des Modells mit der Zeitmarke belegt, an dem die Beobachtung stattfand. Die restlichen (nicht beobachteten) Zeitwerte sind gemäß der Modelldefinition propagiert.
- Die Unsicherheitsintervalle einer Instanz werden eingeschränkt: Beobachtungen von Teilvorgängen führen dazu, daß propagierte Werte konkretisiert werden können.
- Ein Modell wird vollständig erkannt: Beginn- und Endeintervalle sind mit beobachteten Zeitpunkten belegt. Außerdem sind alle Teilvorgänge ebenfalls vollständig erkannt.
- Eine Instanz wird verworfen: Aufgrund von Verletzungen zeitlicher Constraints innerhalb des Vorgangsmodells, wird die vollständige Instantiierung des teilweise Instantiierten Modells unmöglich.

Für die Route sind allein die Zeitintervalle für den Beginn und das Ende einer B-Region interessant, da hier von der internen Definition der Modelle abstrahiert wird. Daher muß die Routenerstellung nur dann reagieren, wenn sich diese Intervalle ändern. Dementsprechend reagiert die Routenerstellung auf die folgenden Nachrichten der Vorgangserkennung:

- Neue Instanz
- Unsicherheitsintervalle für Beginn oder Ende einer Instanz werden eingeschränkt.
- Instanz wird verworfen.

In allen Fällen wird dazu neben dem Nachrichtentyp ein Verweis auf die Instanz übergeben. Mit dem Verweis auf die Instanz kann dann auch auf die aktuellen Beginn- und Endeintervalle zugegriffen werden. Wir müssen nicht unterscheiden, ob eine B-Region vollständig erkannt wurde, oder ob sich nur die Werte des Endeintervalls durch Propagierung verändert haben. Beides führt einfach zu einer Aktualisierung der Intervallgrenzen.

Die Relation zwischen zwei erkannten B-Regionen kann mit einem Element der Menge {[overlaps], [overlaps meets], [overlaps, meets after], [meets after], [after]} beschrieben werden (vgl. Definition 17). In der Implementierung haben wir dies durch Intervalle mit einem Minimum und einem Maximum ausgedrückt, da diese Repräsentation äquivalent zur Definition, aber für die Programmierung einfacher zu handhaben ist.

Die Funktion event-loop ruft die Unterprogramme mit den Parametern auf, wenn die entsprechende Nachricht in der Ereignisliste vorliegt.

```
route-akquisition:
INPUT:     Event-Type:
           RegionInst:
begin
     Goal-Command := 'false
     Route := initialize-route()
     while (not Goal-Command) do
           switch Event-Type
           case 'new:           Route := new-region(RegionInst, Route)
           case 'update:        Route := update-intervals(RegionInst,
                                                          Route     )
           case 'discard:       Route := discard-region(RegionInst, Route)
           case 'goal-reached:  Goal-Command := 'true
                                Route := new-region (Route.Goal, Route)
           endswitch
     endwhile
end
```

Bei der Initialisierung der Route werden die beiden ausgezeichneten Knoten „Start" und „Goal" erzeugt, die den Beginn und das Ende der Route kennzeichnen. Die Kantenmenge ist zu Beginn leer:

```
initialise-route:
INPUT:
OUTPUT:    Route:
begin
     Route.Start := make-vertex("Start", 0, 0, ∞, ∞)
     Route.Goal := make-vertex("Goal", ∞, ∞, ∞, ∞)
     Route.Vertices := '(Route.Start, Route.Goal)
     Route.Edges := '()
end
```

Die Funktion „new-region” wird aufgerufen, wenn eine neue Instanz eines B-Regionenmodells erzeugt wurde. Sie ist die zentrale Funktion für den Aufbau des Graphen der Route. Hier werden die neuen Knoten inkrementell an die vorhandene Graphstruktur angehängt.

Für die Reproduktionsphase ist es von Vorteil, wenn der Graph der Route eine möglichst minimale Beschreibung der zeitlichen Abfolge erkannter B-Regionen ist. Das heißt, daß in dem Graph nur die direkten Präzedenzen zwischen zwei Knoten als Kanten abgespeichert sind. Indirekte Beziehungen, also solche, die sich transitiv über eine Folge von Kanten erschließen lassen, sollen nicht als Kanten abgelegt sein. Dadurch kann der Graph in der Reproduktionsphase dazu benutzt werden, die erwarteten Nachfolgeregionen zu der aktuellen B-Region zu bestimmen. Denn die Kanten geben die direkten Nachfolger des aktuellen Knotens an.

Auf der anderen Seite soll die Route aber überlappende B-Regionen, durch parallele Kantenwege ausdrücken können (siehe Abbildung 36 z. B. gleich nach dem Startknoten). Eine Schwierigkeit stellen hierbei die Unsicherheitsintervalle für Beginn und Ende eines Vorganges dar, die verschiedene Interpretationen zwischen zwei B-Regionen erlauben (vgl. Abschnitt 4.2.3). Denn durch die Intervalle kann es verschiedene Interpretationen der Relation zweier B-Regionen geben. Sie können einerseits seriell sein oder parallel. Wenn sie seriell sind, so soll das durch eine Kante im Graphen ausgedrückt werden. Wenn sie parallel sind, so soll es möglich sein, einen Kantenweg durch den Graphen zu finden, der nur genau einen der beiden B-Regionen berührt. Die Unterscheidung der Fälle 1 bis 3 im folgenden Algorithmus berücksichtigt diesen Umstand.

```
new-region:
INPUT:     Region:    Verweis auf die Instanz eines Regionenmodells bei dem
                      die Vorgangserkennung den Beginnknoten des Zeitnetzes
                      mit beobachteten Zeitwerten belegt hat. Die rest-
                      lichen Werte sind aufgrund der Modelldefinition
                      propagiert.
           Route:     Beschreibung der zeitlichen Zusammenhänge der bisher
                      erkannten Regionen.
OUTPUT:    Route:     Route mit in den Graphen eingefügter neuer Region.
begin
     New-Vertex := make-vertex(generate-name (Region.name), Region
                                Region.begin.min, Region.begin.max, ∞, ∞)
     forall Vertex in Route.Vertices do
(1)        if (       (Vertex.end.min ≤ New-Vertex.begin.max)
                      // Es gibt eine Interpretation der
                      // Unsicherheitsintervalle,
                      // bei der Die Knoten hintereinander liegen können.
                      // Und
(2)              and  (     (New-Vertex.begin.min ≤ Vertex.end.max)
                            // Es liegt eine Überlappung der
                            // Unsicherheitsintervalle vor.
                            // Oder
(3)                     or  (notany-in-between-p (Route,
                                                  Vertex.end.min,
                                                  New-Vertex.begin.max))
                            // Es gibt keinen anderen Knoten im Graphen, der
                            // zeitlich vollständig zwischen den beiden
                            // liegt.
                      ))
           then Route.Edges := add-edge(Route.Edges, Vertex, New-Vertex)
     endfor
     Route.Vertices := push(Route.Vertices, New-Vertex)
     return Route
end
```

Eine neue B-Region muß immer dann mit einer anderen im Graphen verbunden werden, wenn es eine Interpretation der Unsicherheitsintervalle gibt, so daß die B-Regionen seriell sein können, es aber keine andere B-Region gibt, die zwischen den beiden liegt; denn wenn eine dazwischen liegt, dann wäre die neue Kante bereits durch die Transitivität ausgedrückt.

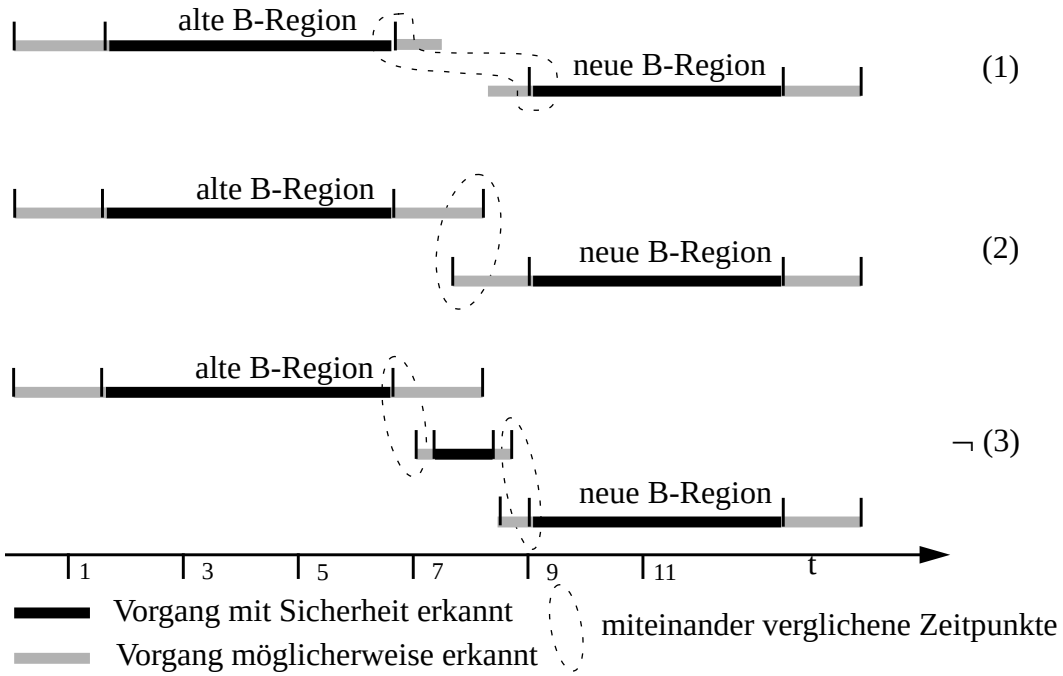

Abbildung 37: Graphische Darstellung der Bedingungen aus dem Algorithmus „new-region". Die obere Graphik stellt Situation der Bedingung (1) im Algorithmus dar, bei der geprüft wird, ob es eine Interpretation gibt, bei der die beiden B-Regionen seriell sein können. Dazu werden das sichere Ende der alten B-Region mit dem sicheren Beginn der neuen B-Region verglichen. Die mittlere Graphik stellt die Situation der Bedingung (2) dar, bei der, wenn (1) gilt, eine Kante auf jeden Fall eingefügt wird, da es eine Interpretation gibt, bei der die beiden B-Regionen direkte Nachfolger sind, egal welchen Wert (3) liefert. Die untere Graphik stellt die Negation der Bedingung (3) dar, aufgrund der immer dann eine Kante eingefügt wird, wenn keine andere B-Region zwischen zwei möglicherweise seriellen B-Regionen liegt.

Die Funktion „notany-in-between-p" prüft, ob es einen Knoten im Graphen gibt, der vollständig zwischen zwei Zeitwerten liegt. Vollständig heißt hier, daß auch die Unsicherheitsintervalle für Beginn und Ende innerhalb der Grenzen liegen. Wenn es einem solchen Knoten gibt, ist dieser links schon mit dem alten Knoten verbunden und wird rechts (vgl. Abbildung 37 unten) mit der neuen verbunden werden, wenn dies nicht schon geschehen ist. Die Kante zwischen dem alten und dem neuen Knoten darf dann nicht eingefügt werden, weil sie sonst eine transitive Relation zwischen dem alten und dem neuen Knoten abkürzen würde.

```
notany-in-between-p:
INPUT:     Route:          (siehe oben)
           OldEndMin:      Minimum vom Endeintervall des alten Knotens, der
                           schon im Graphen hängt.
           NewBegMax:      Maximum vom Beginnintervall des neuen Knotens,
                           der in den Graphen eingehängt werden soll.
OUTPUT:    Truth-Value:    Wahr: es gibt keinen Knoten zwischen alt und
                                 neu.
                           Falsch: es existiert mindestens einer
                                 dazwischen.
begin
     Truth-Value := 'true
     forall V in Route.Vertices do
          if (       (V.end.max ≤ NewBegMax)
               and   (V.begin.min ≥ OldEndMin))
          // Jede Interpretation von V liegt zwischen den beiden
          // Zeitpunkten.
          then begin
                    Truth-Value := 'false
                    return Truth-Value
               end
          endif
     endfor
     return Truth-Value
end
```

Da sich die Werte der Intervalle sowohl bei der endgültigen Erkennung einer B-Region als auch während der Vorgangserkennung durch Propagierung der Werte von Teilvorgängen, ändern können, gibt es die Nachricht „update". Sie löst den Aufruf der Funktion „update-intervals" aus. Hier werden die Kantenbeschriftungen des Graphen aktualisiert. Die Länge der Intervalle kann sich hierbei nur verkürzen. Die Unsicherheitsintervalle werden konkreter.

```
update-intervals:
INPUT:     Vertex:    Knoten mit den neuen Werten für das Beginn- und
                      Endeintervall
           Route:     (siehe oben)
OUTPUT:    Route:     aktualisierte Route
begin
     forall Edge in Route.Edges do
          if Edge.next = Vertex then
               // Edge ist der Vorgängerknoten von Vertex
               if Edge.max > Vertex.begin.max - Edge.pred.end.min
                    // Neues Maximum ist kleiner als das alte
                    //⇒ aktualisieren
                    Edge.max := Vertex.begin.max - Edge.pred.end.min
               endif
          else if Edge.pred = Vertex then
               // Edge ist der Nachfolgerknoten von Vertex
               if Edge.min < Vertex.end.max - Edge.next.begin.min
                    // Neues Minimum ist größer als das alte
                    // ⇒ aktualisieren
                    Edge.min := Vertex.end.max - Edge.next.begin.min
               endif
          endif
     endfor
     return Route
end
```

Wird eine B-Region durch neue Beobachtungen verworfen, müssen wir sie aus dem Graphen entfernen, ohne die Struktur des Graphen an dieser Stelle zu zerreißen. Wir müssen alle Vorgänger des Knotens mit allen Nachfolgerknoten durch eine Kante verbinden, falls diese Kante nicht bereits existiert.

```
region-discarded:
INPUT:     Region:    Die Region, die aus dem Graphen gelöscht werden soll.
           Route:     (siehe oben)
OUTPUT:    Route:     Route, aus der die verworfene Region gelöscht wurde,
                      ohne den Zusammenhang des Graphen zu zerstören.
begin
     Before := find-all-preciding-vertices (Region, Route)
     After := find-all-following-vertices (Region, Route)
     forall Edge in Route.Edges
          if Edge.next.Region = Region then
               // Vorgängerknoten gefunden
               // ⇒ Verbinde sie mit allen Nachfolgern
               forall A in After do
                    // füge neue Kante ein, wenn noch keine existiert
                    add-edge-if-new (Edge.pred, A)
               endfor
          else if Edge.pred.Region = Region then
               // Nachfolgeregion gefunden
               // ⇒ Verbinde sie mit allen Vorgängern
               forall B in Before do
                    // füge neue Kante ein, wenn noch keine existiert
                    add-edge-if-new (B, Edge.next)
               endfor
          endif
end
```

Die Funktion „add-edge-if-new” fügt eine Kante in den Graphen zwischen zwei Knoten ein, falls diese Kante noch nicht existiert.

```
add-edge-if-new:
INPUT:     Route:     (siehe oben)
           Left:      Vorgängerknoten
           Right:     Nachfolgerknoten
OUTPUT:    Route      Wenn die Kante noch nicht exisitierte, wird sie ein-
gefügt.
begin
     EdgeExists := ’false
     forall Edge in Route.Edges do
          if (       (Edge.pred = Left)
               and   (Edge.next = Right)) then
               EdgeExists := ’true
          endif
     endfor
     if (not EdgeExists) then
          Route.Edges := add-edge(Route.Edges, Left, Right)
     endif
     return Route
end
```

Die Funktion „add-edge” erzeugt ein neues Kantenobjekt und fügt es in den Graphen ein:

```
add-edge:
INPUT:     Edges:     Kantenmenge des Graphen
           Old:       Knoten, der bereits im Graphen hängt.
           New:       Neuer Knoten
OUTPUT:    Edges:     Neue Kantenmenge des Graphen
begin
     NewEdge := make-edge()
     NewEdge := update-edge(NewEdge, Old, New)
     Edges := push(Edges, NewEdge)
     return Edges
end
```

Die Funktion „update-edge” aktualisiert die Kantenbeschriftung gemäß der Unsicherheitsintervalle des Vorgängerendes und des Nachfolgerbeginns:

```
update-edge:
INPUT:     Edge:      Zu aktualisierende Kante
           Pred:      Vorgängerknoten
           Next:      Nachfolgerknoten
OUTPUT:    Edge:      Aktualisierte Kante von Pred nach Next.
begin
      Edge.min := Next.min - Pred.max
      Edge.max := Next.max - Pred.min
      return Edge
end
```

Die Abbildung der, in der Implementierung verwendeten, Intervalle auf die, in der Definition verwendeten, qualitativen Relationen ist der folgenden Tabelle 4 zu entnehmen:

Intervall [(B.begin.min - A.end.max), (B.begin.max - A.end.min)]	qualitative Relation
A [-x, -y] B	A {overlaps} B
A [-x, 0] B	A {overlaps, meets} B
A [0, 0] B	A {meets} B
A [-x, y] B	A {overlaps, meets, after} B
A [0, x] B	A {meets, after} B
A [x, y] B	A {after} B

Tabelle 4: Übersetzung der Intervalle zwischen zwei aufeinanderfolgenden Knoten des Graphen in die qualitativen Relationen; dabei sind x, y $\in \mathrm{IR}^{+}$.

Wir haben in diesem Abschnitt kennengelernt, wie der Roboter aus dem Abgleich der Sensorverläufe mit den im Weltmodell definierten B-Regionenmodellen einen Graphen aus instantiierten B-Regionen aufbaut, der die Route des Roboters beschreibt. Durch die vorgegebene Menge von B-Regionenmodellen werden die Bewegungsbausteine für die folgende Fahrt mit deren Toleranzen festgelegt. Der Trainer steuert den Roboter auf einer exemplarischen Trajektorie durch den Raum vom Start zum Ziel. Dabei baut der Roboter die Route aus den erkannten B-Regionenmodellen auf. Die erkannten B-Regionen werden mir ihren Beginn- und Endezeitpunkten, die ab der Startzeit ermittelt werden, abgelegt. Dabei drücken parallele Kantenzüge in der Route gleichzeitig erkannte B-Regionen aus. Eine Kante drückt aus, daß zwei B-Regionen aufeinanderfolgen.

Die Route ist eine qualitative Beschreibungen einer Menge von Trajektorien, die alle vom Start zum Ziel führen. In der Reproduktionsphase kann der Roboter dann anhand der, in den B-Regionen gespeicherten, Bewegungsinformation eine dieser Trajektorien produzieren und die einzelnen B-Regionen in der Reihenfolge, wie sie in der Route abgelegt wurden, wiedererkennen. Die Route ist ein Modell des Weges vom Start zum Ziel.

5.3 Reproduktionsphase

In der Reproduktionsphase wird das Wissen, das der Roboter während der Akquisitionsphase gelernt hat, angewandt. Der Roboter kann anhand der Route den Weg vom Start zum Ziel nachvollziehen. Wir gehen dabei davon aus, daß der Roboter in der gleichen B-Region gestartet wird, die auch zu Beginn der Akquisitionsphase eingenommen wurde. Nachdem die Route vom Operateur ausgewählt wurde, wird der Roboter gestartet. Der aktuelle Knoten der Route ist der Knoten „Start". Der Roboter bestimmt nun den oder die Nachfolgerknoten des aktuellen Knotens. Aufgrund der Generierung der Route müssen die B-Regionen aller alternativen Knoten dieselbe Bewegungsproposition haben. Die Bewegung ist eindeutig. Bevor der Roboter die Bewegung ausführt, setzt er die Zeit auf null und nimmt die B-Regionenmodelle des Weltmodells in den Beobachtungsfokus, die auf die Bewegungsprosition und die Umgebungspropositionen der Nachfolgerknoten passen. Die Nachfolgerknoten werden zu den aktuellen Knoten der Route. Die Bewegung der aktuellen Knoten wird ausgeführt und die Meßwerte der Sensoren werden mit den Modellen des Beobachtungsfokusses abgeglichen. Können die B-Regionen wieder mit Zeitwerten instantiiert werden, die nicht zu stark von den Instantiierungen in der Route, abweichen, so nimmt der Roboter an, daß er sich noch in der gelernten B-Region befindet. Schon jetzt müssen die nächsten Nachfolgerknoten in den Beobachtungsfokus mit einbezogen werden, wenn sie mit dem aktuellen Knoten in einer Relation stehen, die overlaps enthält, da der Roboter sonst den Beginn der nächsten B-Region nicht beobachten könnte. Dies wird solange wiederholt bis entweder der Zielknoten erreicht wird oder ein Fehler auftritt. Bei der Reproduktion der gelernten Route ist der Roboter durch die Form der Repräsentation nicht exakt an die gelernte Trajektorie gebunden, denn die B-Regionen definieren eine Menge verschiedener, aber konzeptuell gleicher Trajektorien. Erst wenn die Trajektorie die Beschreibung der erwarteten B-Region verletzt, tritt das Recovery in Aktion, das den Fehler behandelt, wenn ein entsprechendes Recovery-Werkzeug zur Verfügung steht. Wir werden das Recovery von Fehlern im Abschnitt 5.4 kennenlernen.

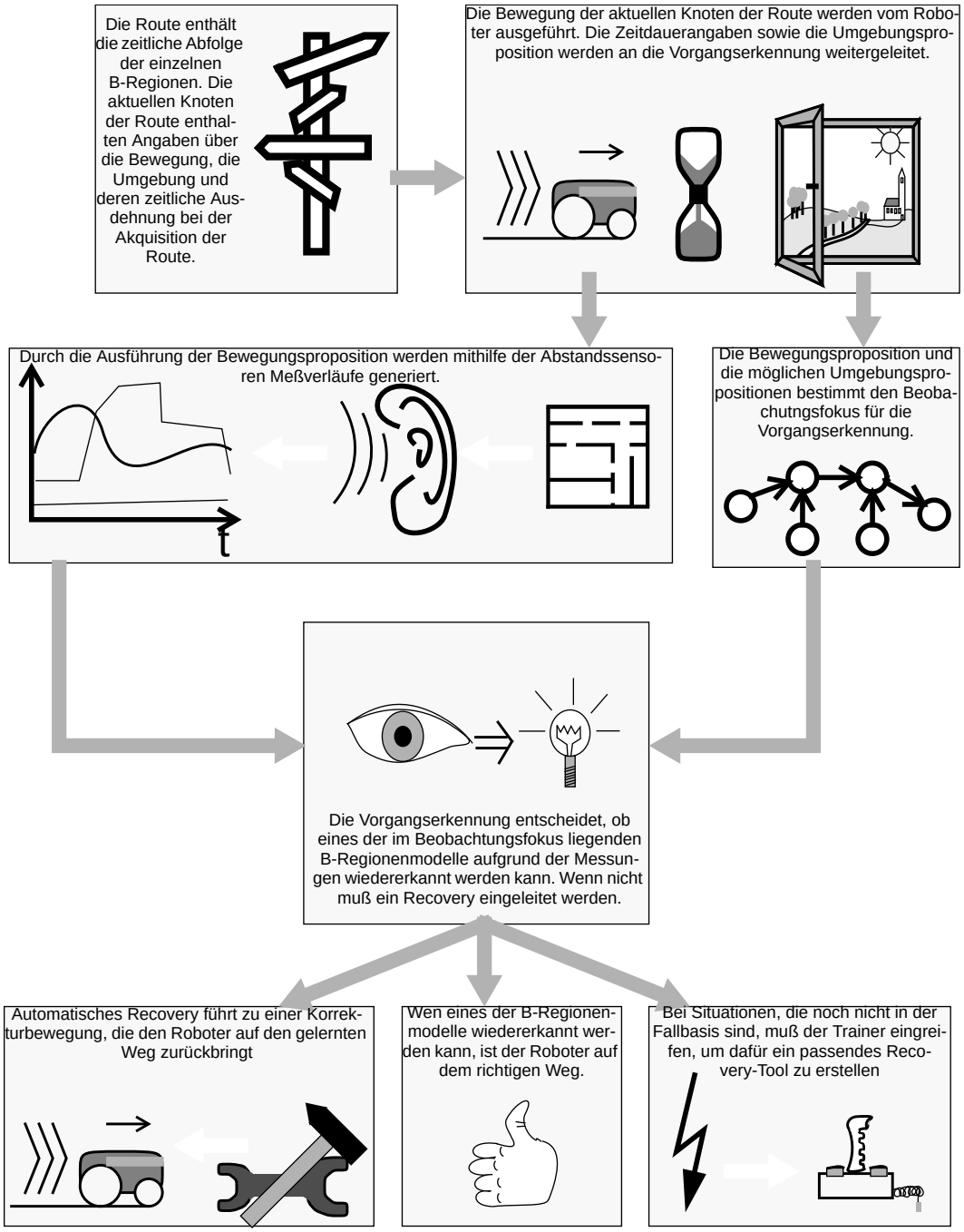

Abbildung 38: Überblick über die Reproduktionsphase: Die Route enthält Angaben über die zeitliche Abfolge der einzelnen B-Regionen. Hier wird bestimmt, welche B-Regionen gerade aktuell sind. Zu Beginn einer neuen B-Region wird deren Bewegungsproposition vom Roboter ausgeführt. Dies erzeugt Meßwertverläufe, die von der Vorgangserkennung verarbeitet werden. Aufgrund der Umgebungs- und der Bewegungsproposition wird der Beobachtungsfokus für die Vorgangserkennung festgelegt. Hier wird nun versucht, die im Fokus befindlichen B-Regionenmodelle mit den neuen Messungen zu in Einklang zu bringen („matching"). Passen die wiedererkannten B-Regionen zeitlich mit denen der Route zusammen, so weiß der Roboter wo er sich gerade befindet. Kann keine B-Region wiedererkannt werden, so müssen Recovery-Maßnahmen eingeleitet werden.

5.3.1 Algorithmus zur Reproduktion von Routen

Die Reproduktion der gelernten Route wird durch die Funktion „reproduction" geleistet. Dazu wird in der Route für jeden Zeitschritt überprüft, welche Knoten im Graphen aktuell sind. Diese Knoten bestimmen durch ihre zugeordneten B-Regionenmodelle, welche Bewegung auszuführen ist und welche Umgebung dabei erwartet wird. Entsprechend werden dann B-Regionenmodelle zur Wiedererkennung an die Vorgangserkennung übergeben, die diese in den Fokus der Beobachtung aufnimmt. Weiterhin wird der Roboter so gesteuert, wie dies in den aktuellen Knoten der Route spezifiziert ist. Widersprüchliche Bewegungspropositionen sind dabei aufgrund der Spezifikation der B-Regionenmodelle unmöglich.

```
reproduction:
INPUT:     Route:     Die in der Akquisitionsphase aufgenommene Route
OUTPUT:
SIDE-
EFFECTS:   Roboterbewegung
           Änderung des Beobachtungsfokusses der Vorgangserkennung
LOCALES:   Environment:    Aufmerksamkeitsfokus für die Beobachtung der
                           Umgebung
           RepState:       Reproduktionsstatus, enthält die aktuelle Zeit
                           und die aktuellen Knoten der Route
begin
     RepState.Time := Route.start.begin.min
     Actual := actual-nodes(Time, Route.start, Route)
     while (not find-goal(Actual))
          // Setzen des Beobachtungsfokuses auf die Regionen der
          // aktuellen Knoten der Route
          set-attention(Actual)
          // Bewegen des Roboters gemäß einer beliebigen Region der
          // aktuellen Knoten
          move-robot(Actual)
          RepState.NewActual := ∅
          forall V in Actual
               switch state-of-region(RepState.Time, V.Region)
               case       'recognized
                    or    'recognizable:
                    RepState.NewActual :=
                              RepState.NewActual
                         ∪    actual-nodes(RepState.Time, V, Route)

               case 'unexpected-end
                    // V wurde bereits erkannt, aber das Ende wurde
                    // früher als erwartet beobachtet
                    RepState := recovery('unexpected-end, RepState.Time,
                                                 V, Actual, Route)
               case 'expected:
                    // V wird zum Zeitpunkt Time erwartet, kann aber
                    // noch nicht wiedererkannt werden
                    RepState := recovery('expected, RepState.Time,
                                              V, Actual, Route)
               case 'expired:
                    // V dauert länger, als in der Routenbeschreibung
                    // vorgesehen
                    RepState := recovery('expired, RepState.Time,
                                              V, Actual, Route)
               endswitch
          endfor
          if (NewActual = ∅) then return 'fatal-error
          else Actual := NewActual
          // Gran ist di,e durch die Abtastrate der Sensorwerte vorgebene,
          // zeitliche Rasterung des Systems.
          Time := Time + Gran
     endwhile
end
```

Solange der Zielknoten nicht in der Menge der aktuellen Knoten enthalten ist, wird zu jedem Zeitschritt der Beobachtungsfokus auf die aktuellen Knoten gesetzt und der Roboter bewegt. Dabei werden alle B-Regionen der Knoten einzeln auf ihren Zustand überprüft. Solange eine B-Region erkennbar („recognizable”) oder erkannt („recognized”) ist, befindet sich der Roboter auf dem durch die Route vorgegebenen Weg. Der Roboter kann seinen Weg fortsetzen. Alle anderen Zustände für B-Regionen werden zur Unterscheidung verschiedener Recovery-Mechanismen benötigt.

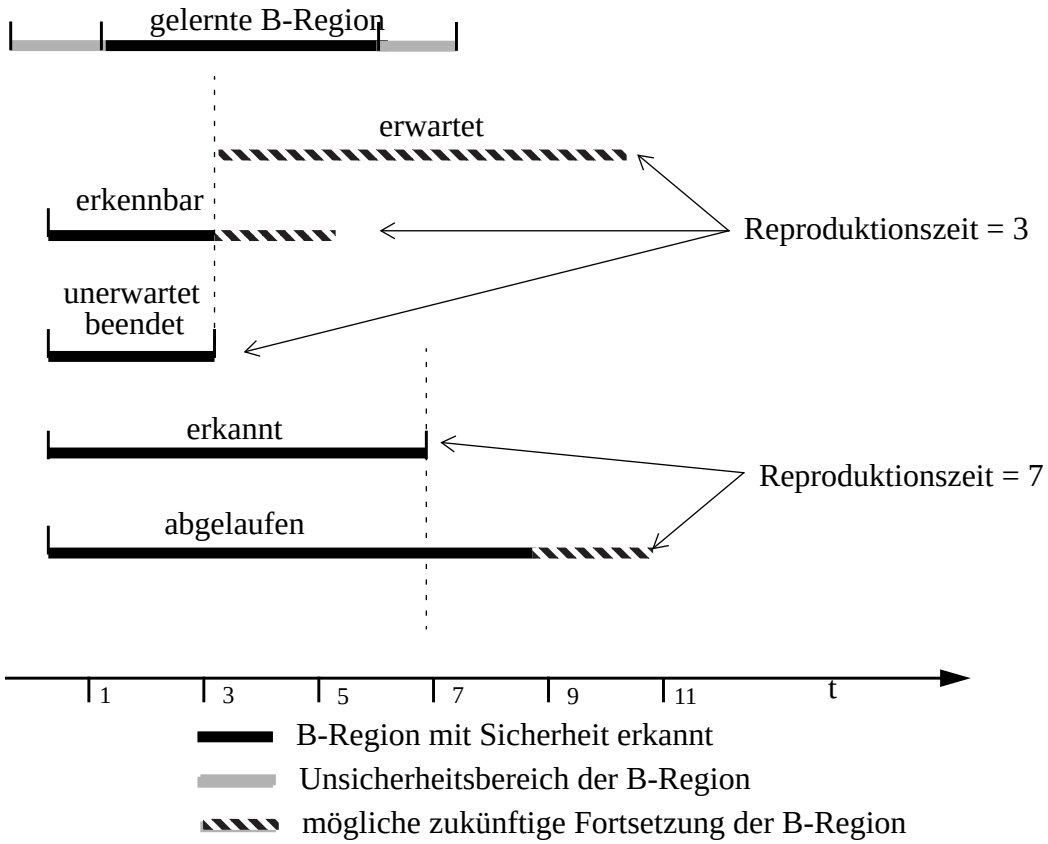

Abbildung 39: Mögliche Zustände von B-Regionen in der Reproduktionsphase. Oben ist die gelernte Route mit ihren Beobachtungszeitpunkten aus der Route aufgezeichnet. Unten sind die verschiedenen zeitlichen Möglichkeiten für die Wiedererkennung der B-Region dargestellt. Die Unsicherheitsintervalle für die wiedererkannten B-Regionen sind zur Vereinfachung weggelassen worden, da bei der Wiedererkennung nur jeweils eine Grenze des Unsicherheitsbereiches relevant wird.

Die Funktion „state-of-region” bestimmt die verschiedenen Stati einer B-Region, die die folgenden Bedeutungen haben:

- erwartet („expected”): Die B-Region ist in der Route zum aktuellen Zeitpunkt erwartet, kann aber aufgrund der Messungen nicht wiedererkannt werden.
- erkennbar („recognizable”): Der Beginn der B-Region ist bereits wiedererkannt worden und sie kann in der Zukunft noch vollständig wiedererkannt werden.
- unerwartet beendet („unexpected-end”): Das Ende der B-Region wird früher beobachtet, als dies in der Route vorgesehen ist.

- erkannt („recognized”): Beginn und Ende der Region konnten konsistent zur Spezifikation in der Route wiedererkannt werden.
- Abgelaufen („expired”): Die Region kann nicht mehr konsistent wiedererkannt werden, da die aktuelle Zeit zu weit fortgeschritten ist.

```
state-of-region
INPUT:     Time:      Reproduktionszeit
           Region:    Region der Routenbeschreibung
OUTPUT:    'recognized|'recognizable|'unexpected-end|'expected|'expired
LOCALES:   PartRec:   Beginn der Region konnte bereits wiedererkannt werden
           Rec:       Beginn und Ende der Region konnten wiedererkannt
                      werden
begin
     // Die Funktion has-matching-member prüft, ob sich in der Menge, die
     // als zweites Argument übergeben wird, eine Region findet, die
     // auf die im ersten Argument übergebene Region, bezüglich der
     // Umgebungs- und Bewegungsproposition paßt.
     PartRec    := has-matching-member
                      (Region, partially-recognized-events(Time))
     Rec        := has-matching-member
                      (Region, recognized-events(Time))
     if (       not (PartRec)
          and   not (Rec)
          and   (Time > Region.begin.max))
     // Region sollte jetzt wiedererkannt werden - sie wir erwartet
     then return 'expected
     if (       PartRec
          and   not (Rec)
          and   (Time ≤ Region.end.max))
     // Region kann noch wiedererkannt werden
     then return 'recognizable
     if (       PartRec
          and   Rec
          and   (Time < Region.end.min))
     // Region endet vor dem in der Route gespeicherten Zeitpunkt
     then return 'unexpected-end
     if (       Rec
          and   (Time ≥ Region.end.min)
          and   (Time ≤ Region.end.max))
     // Region wurde zum passenden Zeitpunkt wiedererkannt
     then return 'recognized
     if (       PartRec
                not (Rec)
          and   (Time > Region.end.max))
     // Die Zeit Time ist zu weit fortgeschritten um die Region
     // wiederzuerkennen
     then return 'expired
end
```

Die Funktion „acutal-nodes” bestimmt die aktuellen Knoten der Route, die Nachfolger eines bestimmten Knotens „V” sind. Aktuell sind dabei die Knoten, deren minimaler Beginnzeitpunkt kleiner oder gleich der aktuellen Reproduktionszeit ist und deren maximaler Endezeitpunkt größer oder gleich der aktuellen Reproduktionszeit ist. Ist der betrachtete Knoten aktuell, so wird er zusammen mit seinen aktuellen Nachfolgern zurückgegeben. Der rekursive Abstieg endet, wenn ein Knoten betrachtet wird, der nicht aktuell ist (dann können seine Nachfolger auch nicht aktuell sein) oder wenn ein Knoten betrachtet wird, der keine Nachfolger im Graphen hat.

```
actual-nodes:
INPUT:     Time:          Aktuelle Reproduktionszeit
           Vertex:        Aktueller Knoten der Route
           Route:         Beschreibung der Route
OUTPUT:    Succ:          Menge aller zum aktuellen Zeitpunkt möglichen
                          Nachfolger von Node
begin
     Succ := ∅
     // Wenn Time nicht zwischen min und max von Vertex liegt, ist Vertex
     // zum Zeitpunkt Time nicht aktuell
     if (       (Vertex.begin.min    > Time)
          or    (Vertex.end.max      < Time)))
     then return ∅
     // wenn keine Nachfolger mehr vorhanden sind, ist Vertex
     // der einzige aktuelle Knoten
     else if (successors(Vertex) = ∅) then return {Vertex}
     else forall S in successors(Vertex)
          // rekursiver Abstieg
          // bestimmen aller aktuellen Nachfolgerknoten von S
          Succ := actual-nodes(Time, S, Route) ∪ Succ
     endfor
end
```

Wir haben in diesem Abschnitt die Reproduktion gelernter Routen kennengelernt und ihren algorithmischen Ablauf untersucht. Sie ist eigenständig, solange die Abweichungen des Roboters, die durch Schlupf der Räder und durch Ungenauigkeiten bei der Abstandsmessung unvermeidbar sind, keine qualitativen Abweichungen der reproduzierten Trajektorie gegenüber der Beschreibung in der Route bedeuten. Wenn der Roboter qualitativ etwas anderes tut, als dies in der Spezifikation der B-Regionen angegeben ist, so werden Recovery-Maßnahmen eingeleitet, die den Roboter wieder zurück in eine B-Region bringen, die er in der Route gespeichert hat. Aus der Sicht der Reproduktion sind dabei drei verschiedene Recovery-Arten aufgrund verschiedener zeitlicher Konstellationen zwischen gelernter B-Region aus der Route und der wiedererkannten B-Region zu unterscheiden.

5.4 Recovery

Aufgrund von Ungenauigkeiten in der Ausführung der Bewegung kann es vorkommen, daß der Roboter die Orientierung verliert. Entweder kann der Roboter dann den Verlauf der Messungen nicht mehr mit einer aktuellen B-Region in der Route in Übereinstimmung bringen oder die wiedererkannte B-Region weicht in ihrer zeitlichen Ausdehnung von der, in der Akquisitionsphase wiedererkannten, B-Region ab. Der Roboter erreicht die nächste B-Region nicht oder er „verläßt" die gelernte B-Region.

Zur Beschreibung solcher Fehlerfälle und der entsprechenden Reaktion darauf benutzen wir Recovery-Tools. Sie liegen in Form von (Teil-) Routenbeschreibungen vor, die vom Roboter verfolgt werden, bis er sich wieder auf der gelernten Route befindet.

DEFINITION 18: Ein **Recovery-Tool** RT ist eine spezielle Form von Route (R, L, K, Beginn, Ende). Mit einer Regionenmenge R, einer Menge von Kantenbeschriftungen L, einer Menge von Kanten und zwei Markierungen Beginn und Ende, die den ersten bzw. letzten Knoten der Teilroute kennzeichnen.

Recovery-Tools sind Routen mit ausgezeichneten Markierungen Beginn und Ende, die kennzeichnen mit welchen Knoten das Tool zu einer existierenden Route eingehängt werden kann. Um ein Recovery einsetzen zu können, müssen die Beginn- und Endeknoten des Tools mit B-Regionen einer normalen Route zur Deckung gebracht werden können. Beginn und Ende sind demnach Beschreibungen von fehlerlosem Roboterverhalten. Die folgenden Knoten sind Beschreibungen des Fehlers und die entsprechende Reaktion zur Korrektur.

BEISPIEL: Abbildung 40 zeigt die aktuelle B-Region R1 einer Route (hier: parallel zur Wand fahren). Angenommen der Roboter befindet sich in R1 und die Sensorwerte steigen, statt, wie spezifiziert, konstant zu bleiben. Der Roboter verläßt damit die B-Region R1 früher, als das in der Route spezifiziert ist. Nehmen wir nun an, daß ein Recovery-Tool der Form $R1 \rightarrow R' \rightarrow R'' \rightarrow R1$ existiert, wobei R' das Ansteigen der Sensorwerte beinhaltet und R'' eine Rechtsdrehung spezifiziert, die den Roboter wieder parallel zur Wand ausrichtet. Durch die Anwendung des Recovery-Tools wird die B-Region R1 wieder erreicht und der Roboter kann seine Route fortsetzen.

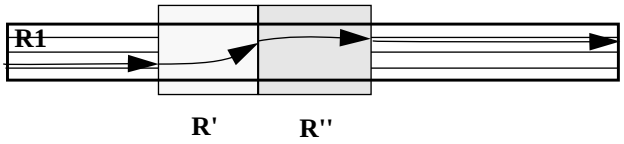

Abbildung 40: Beispiel für die Korrektur einer Routenabweichung.

Recovery-Routen können als Ergänzungen bestimmter Routenbeschreibungen oder als allgemeine Reparaturverfahren bereitgestellt werden. In beiden Fällen wird bei der Durchführung des Recovery, die Route um das Recovery-Tool erweitert. Diese Erweiterung existiert nur zur Reproduktionszeit; sie wird nicht in die gespeicherte Route aufgenommen, da wir davon ausgehen, daß die Abweichungen von der gelernten Route zufällig auftreten und nicht Ausdruck einer sich ändernden Umgebung sind.

Die Definition von Recovery-Verfahren erfolgt ähnlich wie die Definition von B-Regionenmodellen. Der Unterschied ist, daß Recovery-Tools Routenbeschreibungen aus mehreren B-Regionenmodellen sind, die zueinander in „meets"-Relation stehen. Der Spezifikationsrahmen für Recovery-Tools lautet wie folgt:

$$<\hat{R}\,(\text{overlaps})\,R';\ R'(\text{meets})R_{i1;}\ \{R_{i1}(\text{meets})...(\text{meets})R_{in}\}^{*}\ R_{in}\ (\text{overlaps})\ \tilde{R}>$$

$\hat{R}$ und $\tilde{R}$ sind B-Regionen der gewünschten Route. Bei R' handelt es sich um die B-Region, in die der Roboter abgedriftet ist und die R_is stehen für die Gegenmaßnahmen, die ergriffen werden müssen. Die allgemeine Form für eine solche Recovery-Routenbeschreibung ist in Abb. 41 zu sehen.

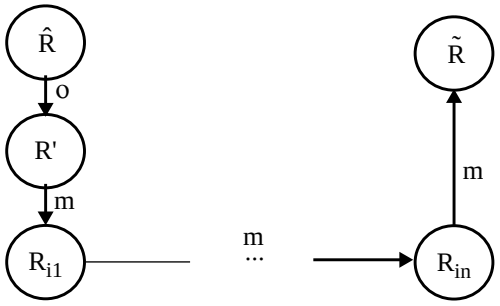

Abb. 41: Die allgemeine Recovery-Route

Wie im vorangegangenen Beispiel können $\hat{R}$ und $\tilde{R}$ gleich sein, wenn durch das Recovery die vorhergehende B-Region wieder erreicht wird.

5.4.1 Algorithmus des Recovery

Die Reparatur von Abweichungen während der Reproduktion der Route wird durch die Funktion „recovery" geleistet. Hier wird zwischen den verschiedenen, in der Funktion „reproduction" auftretenden Recovery-Situationen unterschieden.

```
recovery:
INPUT:     Situation:      Situation, in der das Recovery notwendig wurde
           Time:           aktuelle Reproduktionszeit
           Node:           Wiedererkannte Region, in der die Abweichung
                           auftritt
           ActualNodes:    Menge der augenblicklich zu beobachteten
                           Regionen
           Route:          Routenbeschreibung
OUTPUT:    RepState:       Korrigierte Reproduktionszeit
                           Menge der aktuellen Knoten nach dem
                           Recovery
begin
     switch (Situation)
     case 'unexpected-end:
          if (is-in-time-tolerance (Node, RepState.Time)) then
               RepState.ActualNodes := ActualNodes ∪ successors(Node)
                                       / Node
               RepState.Time := Node.end.min
               return RepState
          else
               // Suche ein Tool, daß die zu früh beendete Region
               // fortsetzt.
               Tool := tool-search(Node, Node, RepState)
               RepState := move-robot(Tool, ActualNodes)
               return RepState
     case 'expected:
          if (is-in-time-tolerance (Node, RepState.Time)) then
               // Weiterfahren und abwarten, ob die Region noch
               // innerhalb der Toleranz erkannt werden kann
               return RepState
          else
               // Suche nach einem Tool, daß den Vorgänger der
               // aktuellen Region mit der aktuellen Region verbindet
               // und mit den Daten der Historie zusammenpaßt
               Tool = tool-search(predecessor(Node), Node, RepState)
               RepState := move-robot(Tool, ActualNodes)
          endif
```

```
      case 'expired:
           if (is-in-time-tolerance (Node, RepState.Time) then
                // Solange die Reproduktionszeit noch innerhalb der
                // Toleranz liegt fahren wir weiter und warten darauf,
                // daß das Ende noch erkannt werden kann
                return RepState
           else
                // Suche ein Tool, daß den aktuellen Knoten mit dessen
                // Nachfolger verbindet
                Tool = tool-search(Node, successor(Node), RepState)
                RepState := move-robot(Tool, ActualNodes)
           endif
      endswitch
end
```

Solange die zeitlichen Differenzen der gelernten Regionen zu den reproduzierten Regionen klein sind, wird der Roboter mit der aktuellen Bewegung weitergefahren, bis entweder das erwartete Ereignis eintritt oder die zeitliche Toleranz überschritten wird. Wenn das der Fall ist, wird mit der Funktion „tool-search" ein entsprechendes Recovery-Tool aus der Fallbasis herausgesucht, welches die Regionen der beiden übergebenen Knoten miteinander verbindet.

Die Funktion „tool-search" sucht das passende Werkzeug nach folgenden Kriterien aus der Fallbasis aus:

1. Passende Umgebungs- und Bewegungspropositionen für die beiden übergebenen Regionen, wobei der erste Parameter auf $\hat{R}$, und der zweite Parameter auf $\tilde{R}$ des Recovery-Tools passen müssen.[3] Existiert kein solches Werkzeug, so kann der Roboter nicht auf die Fehlersituation reagieren und meldet dies. Der Trainer muß dann ein für diese Situation passendes Recovery-Tool entwerfen. Existieren mehrere solcher Werkzeuge, so entscheidet das folgende Kriterium.

2. Auf der Basis der vom Roboter bisher aufgenommenen Daten, die in der Historie abgelegt sind, startet das Recovery eine retrospektive Vorgangserkennung. Dabei wird der Beobachtungsfokus um die Recovery-Tools erweitert. Alle Werkzeuge, die dabei auf die Daten passen, können für die Fehlersituation angewendet werden. Kann kein Werkzeug mit den Daten zur Deckung gebracht werden, liegt wieder ein Fall vor, bei dem der Trainer eingreifen muß.

Nach der Ausführung des Recovery-Tools mittels „move-robot" wird die Reproduktionszeit entsprechend gesetzt und die Reproduktion der Route wiederaufgenommen.

Wir haben in diesem Abschnitt einer formalen Rahmen für die Formulierung von Recovery-Tools vorgestellt, der auf den gleichen Mechanismen beruht wie die Routenverfolgung. Damit verfügen wir über einen homogenen Rahmen, der zur Beschreibung von Normalverhalten des Roboters (in Form von automatisch generierten Routen) genauso geeignet ist, wie zur Beschreibung von Ausnahmebehandlungen (in Form von Recovery-Tools).

[3] Vgl. Kommentar zur Funktion „has-matching-member" auf Seite 68.

6 Schlußbetrachtung

Abschließend vergleichen wir zunächst den vorgestellten Ansatz mit verwandten Arbeiten. Danach untersuchen wir, inwieweit die in der Einleitung definierten Ziele der Arbeit erreicht wurden. Im Ausblick stellen wir mögliche Erweiterungen vor, die uns für die weitere Entwicklung dieses Ansatzes sinnvoll erscheinen.

6.1 Verwandte Arbeiten

In der vorliegenden Arbeit wurden Methoden aus dem Monitoring und der Diagnose technischer Systeme auf die Roboterdomäne übertragen. Dabei fassen wir die Messungen der Abstandssensoren als beobachtbare Entitäten auf, deren Verhalten wir qualitativ in Form von B-Regionenmodellen beschreiben. Der Formalismus zur Beschreibung des Sensorverhaltens, sowie die Erkennung bedeutungstragender Regionen ist dabei aus der in [Kockskämper, Neumann 94] vorgestellten inkrementellen Vorgangserkennung abgeleitet. Erweitert wurde die Semantik der Vorgangsmodelle um Propositionen bzgl. der Bewegung und der Umgebung des Roboters, so daß sie eine räumliche Interpretation der Sensordaten darstellen. Durch die Benutzung von zeitlich ausgedehnten Vorgängen beschreiben wir die Bewegung des Roboters in ihrer räumlichen und zeitlichen Ausdehnung, ohne dabei auf Punktfolgen, mit denen die Bewegung eines autonomen mobilen Roboters meist repräsentiert wird, angewiesen zu sein.

Mit den Arbeiten über den NX Roboter an der Universität von Texas in Austin ([Kuipers, Byun 87], [Kuipers, Byun 88], [Kuipers, Byun 90] und [Kuipers, Levitt 90]) wird ein Szenario beschrieben, daß dem Unsrigen sehr nahe kommt, denn NX ist ein simulierter Roboter, der mit Abstandssensoren ausgestattet ist und der physischen Robotern in seiner Funktionalität nachempfunden ist. Die vom Roboter aufgebauten Karten sind ebenfalls qualitativer Natur. Die Aufgabe des NX Roboters unterscheidet sich von unserer: NX baut eine vollständige Karte in einem unbekannten Raum aufgrund von vordefinierten subsumptionsbasierten Behaviors auf. Dazu werden punktförmige Landmarken verwendet, die sich durch verschiedene fest verdrahtete Vergleiche zwischen Messungen verschiedener Sensoren oder zwischen aufeinanderfolgenden Messungen ergeben. Die Landmarken werden über die numerischen Werte aller Sensoren an dieser Stelle identifiziert und als Knoten in einen Graphen eingetragen. Die Kanten des Graphen werden durch die brooksschen Bewegungs-Behavior gebildet. Zur Reproduktion eines bestimmten Weges ist der Roboter gezwungen per hill-climbing genau auf die in der Akquisitionsphase aufgenommenen Punkte zu steuern. Dies ist eine geringere Stufe qualitativer Abstraktion als dies in unserem Ansatz vorgestellt wird.

In [Brooks 86] werden subsumptionsbasierte Behavior dazu benutzt, Verhalten ohne interne Repräsentation zu erzeugen. Brooks spricht hier von „emergent behavior". Er vertritt in [Brooks 91] die vielversprechende These, daß intelligentes Verhalten in vielen Anwendungen auch ohne Repräsentation auskommen kann. Die Frage, die sich dabei unweigerlich stellt ist die, ob nicht auch die subsumptionsbasierte Programmierung eine Form von (prozeduraler) Repräsentation darstellt. Spätestens, wenn der Roboter Kenntnisse über seine aktuelle Position haben soll, die ihn in die Lage versetzen, zu navigieren, kommt er nicht mehr ohne eine, über die Behaviors hinausgehende, Form der Raumrepräsentation aus. In [Matarik 92] wird eine Raumrepräsentation für einen physischen Roboter aufgebaut, der mit einer Steuerung ausgestattet ist, die auf Behaviors basiert. Matarik verwendet dazu vier Typen von Landmarken („left walls", „right walls", „corridors" und „irregular

walls"), die sich aufgrund der Behavior für das Verfolgen von Wänden erkennen lassen. Der Roboter exploriert seine Umgebung und baut dabei eine qualitative Karte in Form eines Graphen auf. Die Landmarken werden zusammen mit ihrem Typ, ihrer zeitlichen Ausdehnung[1] und ihrer, durch einen 16-wertigen Kompaß ermittelten, globalen Ausrichtung als Knoten in den Graphen eingetragen. Über diese Angaben wird ein Ähnlichkeitsmaß definiert, das zur Positionsbestimmung benutzt wird. Insgesamt ergibt sich ein Verfahren, daß die wahrscheinlichste Region des Roboters feststellt. Die Kartenkomponente verwendet nur das Behavior zur Wandverfolgung. Alle anderen Bewegungen des Roboters werden nicht in die Karte aufgenommen. Die generierten Karten sind dadurch nicht auf die jeweiligen Anforderungen der Umgebung oder die Aufgabe des Roboters anpaßbar, wie das mit der Definition von Regionenmodellen der Fall ist. In diesem Zusammenhang kann man die Verwendung von Regionenmodellen, zusammen mit den passenden Recovery-Maßnahmen, als Alternative zur Steuerung durch Behavior auffassen.

Arbeiten die sich am Aufbau menschlicher kognitiver Karten orientieren, haben unsere Idee einer adäquaten Raumrepräsentation stark beeinflußt. Hierbei sind die Systeme Tour [Kuipers 78] und SPAM [McDermott, Davis 84], sowie die Arbeit von Yeap [Yeap 81] die wichtigsten. Die Repräsentation des Raumes in Form von ausgedehnten Regionen die sich zu Routen zusammensetzen, kann im Sinne Yeaps als egozentrisches, schwach strukturiertes Modell der Welt angesehen werden, das für bestimmte Entwicklungsstufen des Menschen als adäquate Beschreibung angesehen wird.

Ebenso einflußreich, wenn auch in der vorliegenden Fassung der Arbeit weniger sichtbar, waren Überlegungen zu den Arbeiten von Freksa und Zimmermann. In [Freksa 92] und [Freksa, Zimmermann 92] entwickeln sie ein Kalkül, daß die relative Lage von Orten qualitativ Repräsentiert. Sie vertreten die These, daß die Orientierung im Raum allein auf der Basis von 15 qualitativen Richtungsrelationen zwischen einem Punkt und einem Vektor möglich ist. Die Richtungsrelationen sind kognitiv motiviert. Leider konnte der ursprüngliche Anspruch der vorliegenden Arbeit nicht umgesetzt werden, die B-Regionenmodelle mithilfe dieser Richtungsrelationen zu einer qualitativen Karte der Roboterumgebung zusammenzusetzen. Es erwies sich als unmöglich, die räumlich ausgedehnten B-Regionen mit dem auf Punkten basierenden Kalkül zu einem Gesamtsystem zu integrieren.

Zum Abschluß der Literaturbetrachtung möchte ich noch auf zwei Veröffentlichungen hinweisen, die als Zusammenfassung dieser Diplomarbeit herangezogen werden können: [Kockskämper, Vogel 95] entstand als „extended abstract" in englischer Sprache und kann als solcher nur das Thema umreißen. Die zehnseitige Arbeit [Kockskämper, Neumann, Vogel 95] ist deutschsprachig und bietet einen guten Überblick über das Thema.

6.2 Diskussion

Wir haben in der vorliegenden Arbeit einen Formalismus entwickelt, der die Vorteile einer qualitativen Repräsentation mit denen einer sensorbasierten verbindet. Sensorbasierte Ansätze werden oft in technisch-reaktiven Systemen benutzt, in denen die Regeln des Roboters, wie er auf die Sensordaten reagiert, meist nicht explizit modelliert werden. Sie haben den Vorteil, daß sie robust gegen Abwei-

[1] Auch hier wird die die Zeit als Entfernungsmaß benutzt, indem der Roboter immer mit konstanter Geschwindigkeit fährt.

chungen des Roboters durch Schlupf o. ä. sind, da sie die Reaktion aufgrund der Sensordaten bestimmen. Der Vorteil einer qualitativen Repräsentation ist die kompakte Darstellung und die daraus folgende Verständlichkeit. Sie ermöglicht es, bestimmten Entwicklungen von Sensordaten (Meßwertverläufe) eine Bedeutung zuzuordnen. So wird auf jeder Stufe der Repräsentationshierarchie die Bedeutung der aufgenommenen Messungen expliziert. Insgesamt wird durch die Verwendung von B-Regionenmodellen die Reaktion des Roboters (in Form einer bestimmten Bewegung) durchschaubarer. Die gelernte Route kann auf verschiedenen Abstraktionsstufen betrachtet werden, was eine manuelle Verbesserung von Ungenauigkeiten der Steuerung, die in der Akquisitionsphase durch Bedienungsfehler auftreten können, ermöglicht.

Durch die Repräsentation der Roboterbewegung in Form von B-Regionenmodellen kann die Bewegung des Roboters qualitativ modelliert werden. Zeit und Raum werden im Gegensatz zu schnappschußbasierten Ansätzen (die zwischen Orientierungspunkten und Bewegungen unterscheiden müssen) als Einheit repräsentiert. Dadurch kann die Bewegung selbst qualitativ modelliert werden. Das hat den Vorteil, daß wir in der Lage sind, genau die Merkmale einer Roboterfahrt zu spezifizieren, die für die gewählte Aufgabe wesentlich sind. Unwichtige Merkmale können weggelassen werden, so daß die Repräsentation inhärent robust gegen geringe Abweichungen des Roboters von der gelernten Trajektorie ist. Unwichtige Abweichungen vom gelernten Weg führen nicht zwangsläufig dazu, daß die B-Region verlassen wird. Weicht der Roboter tatsächlich einmal von der gelernten B-Region ab, so werden geeignete Recovery-Aktionen eingeleitet, die den Roboter wieder auf „den richtigen Weg“ zurückbringen.

Die hierarchische Form der Modellierung bringt neben der Wiederverwendbarkeit der einzelnen Modelle den Vorteil mit sich, daß zahlreiche heterogene Sensorsignale leicht in den Vorgangsmodellen zusammengeführt werden können: Indem für die verschiedenen Sensordaten jeweils Prädikate definiert werden, können ihre zeitlichen Beziehungen untereinander in Vorgangsmodellen oder B-Regionenmodellen beschrieben werden und für die Navigation genutzt werden.

B-Regionen haben eine gewisse Ähnlichkeit mit Trajetorienbeschreibungen im K-Raum. Dabei wird der Begriff „Konfiguration” im allgemeinen dazu verwendet, die Freiheitsgrade der Roboterbewegung zu beschreiben. Trajektorienbeschreibungen definieren damit bestimmte Bewegungsmuster des Roboters. Die von B-Regionen beschriebenen Trajektorien gehen aber einen entscheidenden Schritt weiter: Neben den Bewegungsdimensionen umfaßt der durch B-Regionen beschriebene Raum hier ebenfalls die Sensordimensionen. Dadurch wird zusätzlich zur Bewegung die Umgebung des Roboters durch die B-Region abgebildet. B-Regionen sind Teilräume des kombinierten Bewegungs- und Sensorraumes.

Aus der Sicht der Kognitionswissenschaft können wir die B-Regionen als Modellierung eines subjektiven Raumempfindens werten, das für bestimmte Phasen der menschlichen Entwicklung diskutiert wird. Der umgebende Raum wird hier mithilfe der subjektiven zeitlichen Muster der Aufnahme von Sensorwerten „erfahren” und zusammen mit der Bewegung des Roboters abgebildet. Für viele Anwendungsbereiche der Navigation ist diese Form der Umgebungsrepräsentation bereits ausreichend. Um aber von dem Erfahrungswissen der Routen auf den invarianten Teil des Raumes zu schließen (wie es z. B. für räumliche Inferenzen notwendig ist), bedarf es der Übersetzung des zeitlichen Constraint-Systems in ein räumliches. Dazu können die Bewegungsdaten der B-Regionen unter Berücksichtigung

der Zeit in Positions- und Orientierungsdaten relativ zur Startposition überführt werden. Eine tiefergehende Betrachtung dieses Sachverhalts ist zur Zeit noch nicht abgeschlossen.

6.3 Ausblick

Im Sinne der Aufgabenstellung dieser Diplomarbeit, die wir in der Einleitung formuliert haben, wurde das Ziel erreicht, einen Formalismus zu entwickeln, mit dem Roboternavigation aufgrund von räumlich und zeitlich ausgedehnten B-Regionenmodellen prinzipiell möglich ist. Die algorithmische Struktur für die Akquisition und Reproduktion von Routen sowie für das Recovery wurden detailliert entwikkelt. Leider erwies sich die Implementierung des Gesamtsystems aufgrund technischer Probleme als zu langwierig, um sie in der Bearbeitungsfrist der Diplomarbeit fertigzustellen. Daher wurde nach der Entwicklung von Fragmenten, zugunsten der Theorieentwicklung darauf verzichtet, die Implementierung zu vervollständigen. Die Arbeit muß daher den Beweis der Machbarkeit durch ein integriertes Anwendungsbeispiel schuldig bleiben. Implementiert wurde dagegen die Modellierung der Roboterdomäne mit ihren B-Regionenmodellen, sowie deren inkrementelle Erkennung. Die weitere Verarbeitung in der Route und beim Recovery wurde nicht realisiert. Die im Kapitel 5 formulierten Algorithmen sind aber soweit ausgearbeitet, daß an der Übertragbarkeit des Pseudocodes in eine Programmiersprache kein Zweifel besteht. Wir wollen im Folgenden, trotz des dargestellten Status der Implementierung, einen Blick auf mögliche Erweiterungen der Konzepte werfen, da diese uns im Hinblick auf die wissenschaftliche Entwicklung wichtiger sind.

Die auf den ersten Blick einfachste Erweiterung betrifft die Geschwindigkeit des Roboters. Wir haben bisher angenommen, daß der Roboter immer mit einer konstanten Geschwindigkeit fährt, sowohl für die Translation als auch für die Rotation. Eine Umrechnung der Zeitangaben in den Modellen auf Entfernungsangaben kann leicht durch die Parametrierung mit der Geschwindigkeit erfolgen. Dies setzt allerdings voraus, daß die geplante Geschwindigkeit des Roboters auch immer erreicht wird. Ohne diese, im Hinblick auf den Schlupf recht gewagte Annahme, ergeben sich aber Probleme bei der Vorgangserkennung, da diese von einer konstanten Samplingrate für die einzelnen Sensoren ausgeht. Bei einer variablen Geschwindigkeit beziehen sich aufeinanderfolgende Meßwerte aber auf verschieden weit entfernte Lagen. Eine Lösung dieses Problems müßte somit eine Erweiterung der Vorgangserkennung in diesem Punkt einschließen.

Die Aufgabe einer konstanten Geschwindigkeit deutet schon an, daß wir die Überführung des bisher zeitlichen Constraint-Systems in ein räumliches für sinnvoll halten. Zur Entwicklung einer Abbildung die dies leistet, sollte zunächst ein Inferenzmechanismus entwickelt werden, der die Umgebungspropositionen verschiedener B-Regionen miteinander in Beziehung setzen kann (z. B. Abbildung von „Wand links verfolgen” auf „Wand rechts verfolgen” nach einer Drehung um 180^{o}). Dadurch könnte als erster Schritt der Rückweg aus der gelernten Route automatisch generiert werden.

Weiterhin erscheint eine Integration des Recovery in die Modellierung der B-Regionenmodelle sinnvoll zu sein. Denn bei deren Erkennung werden die vom spezifizierten Verlauf abweichenden Beobachtungen bereits dazu verwendet, das Modell zu verwerfen. Das Recovery könnte, wenn es integraler Bestandteil des B-Regionenmodells wäre, auf die bereits erkannten Teile des Modells zurückgreifen. Dies erlaubt zusätzlich eine weitere Einschränkung des Beobachtungsfokus-

ses. Dadurch wird die Funktion der B-Regionenmodelle entscheidend erweitert: Sind sie bisher „nur" Modelle, die mit den eingehenden Meßdaten abgeglichen werden, so wären sie dann Aktionsmodelle, die die Meßwerte eigenständig beeinflussen können, wenn sich deren Verlauf nicht so verhält wie dies spezifiziert ist.

In [Kockskämper, Neumann, Vogel 95] schlagen wir vor, die Routen in Form von Zeitnetzen abzulegen, um die Propagierungsfunktionen der Vorgangserkennung ebenfalls zur Ermittlung der aktuellen B-Region(en) zu verwenden. Darauf aufbauend, ist eine Aggregation von B-Regionenmodellen denkbar, die eine Strukturierung des Raumes und die Verschmelzung von B-Regionenmodellen mit verschiedenen Bewegungspropositionen erlaubt. Allerdings ist dabei das Problem einer adäquaten Typisierung noch in der Diskussion.

Um das Nichtwissen zu wissen, ist Stärke.
Das Wissen zu ignorieren, ist Dummheit.
(Laotse)

7 Literatur

[Allen 83]
J.F. Allen: Maintaining Knowledge About Temporal Intervals, in: Communications of the ACM 26(11), p. 832-843, 1983.

[Atiya 95]
S. Atiya: Nagation von mobilen Robotern mit Hilfe bildgebender Senssoren: Ein Mengenbasierter Ansatz. VDI Fortschrittsbericht Reihe 8 Nr. 456.

[Brooks 86]
R. A. Brooks: A Robust Layered Control System for a Movile robot; In: IEEE Journal of Robotics and Automation S.14-23

[Brooks 91]
R. A. Brooks: Intelligence Without Representation; In: Artificial Intelligence Vol. 47(2), S. 139-160.

[Davis 90]
E. Davis: Representations of commonsense Knowledge; Morgan Kaufmann Series in Representation and Reasoning.
(Occupancy Grid) Chapter 6 - Space

[Freksa 92]
Ch. Freksa: Using Orientation Information for Qualtitative Spatial Reasoning; in: Proc. International Conference on Theories and Methods of Spatio-Temporal Reasoning in Geographic Space, Pisa, Italy, S. 162-178, 1992.

[Freksa 92a]
Ch. Freksa: Temporal Reasoning based on Semi-Intervals; Artificial Intelligence Vol. 54, S. 199-227.

[Freksa, Zimmermann 92]
Ch. Freksa, K. Zimmermann: On the Utilization of Spatial Structures for Cognitively Plausible and Efficient Reasoning; in: Proc. IEEE International Conference on Systems, Man and Cybernetics, Chicago, IL, S. 261-266, 1992.

[Kockskämper et al. 93]
S. Kockkämper, B. Neumann, M. Schick: Using Event Models to Cope with Time-dependent Phenomena in Dynamical Systems; behavior memo 04-93, Universität Hamburg, 1993.

[Kockskämper et al. 94]
S. Kockkämper, B. Neumann, M. Schick: Extening Process Monitoring by Event Recognition. Proc. ISE-94, 1994.

[Kockskämper, Neumann 94]
S. Kockskämper, B. Neumann: Vorgangserkennung - ein wissensbasiertes Verfahren zur Überwachung technischer Prozesse; in: KI 2/94, S. 19 - 27, 1994.

[Kockskämper, Vogel 95]
S. Kockskämper, U. Vogel: Using Regions as Landmarks for Robot Navigation; in: KI-95 Activities, L. Dreschler-Fischer, S. Pribbenow (Eds.), p. 109-110.

[Kockskämper, Neumann, Vogel 95]
S. Kockskämper, B. Neumann, U. Vogel: Raum-zeitlich ausgedehnte Regionen als Orientierungsmittel bei der Roboternavigation; erscheint in: Proc. Autonome Mobile Systeme-95; Dillmann, Rembold (Hrsg.); Informatik Aktuell, Springer Verlag.

[Kuipers 78]
B. J. Kuipers: Modelling Spatial Knowledge, Cognitive Science, 2(2), 1978.

[Kuipers, Byun 87]
B. J. Kuipers, Y. T. Byun: A Qualitative Approach to Robot Exploration and Map-Learning; in: Proc. of the Workshop on Spatial Reasoning and Multi-Sensor-Fusion, Los Altos, CA, S. 390-404, 1987.

[Kuipers, Byun 88]
B. J. Kuipers, Y. T. Byun: A Robust Qualitative Method for Robot Exploration and Map-Learning. Technical Report AI88-73, University of Texas at Austin.

[Kuipers, Byun 90]
B. J. Kuipers, Y. T. Byun: A Robot Exploration and Mapping Strategy Based on a Semantic hierarchy of Spatial Representations. Technical Report AI90-120, University of Texas at Austin, 1990.

[Kuipers, Levitt 90]
B. J. Kuipers and T. S.Levitt. Navigation and Mapping in Large-Scale Space. AI Magazine, 9(2), 1988.

[Lahres, Trowe 93]
B. Lahres, M. Trowe: Erzeutung qualitativer Information über physikalische Systeme aus numerischen Meßdaten; Studienarbeit, Universität Hamburg, 1993.

[Matarik 90]
M. K. Mataric: A Distributed Model for Mobile Robot Environment Learning and Navigation; Technical Report AI-TR 1228, MIT Artificial Intelligence Laboratory, 1990.

[Matarik 92]
M. K. Mataric: Integration of Representation Into Goal-Driven Behavior-Based Robots; IEEE Transactions onRobotics and Automation, Vol. 8, No.3, June 1992.

[McDermott, Davis 84]
Planning Routes through Uncertain Territory, Artificial Intelligence, 22(2), 1984.

[Nökel 91]
K. Nökel: Temporally Distributed Symptoms in Technical Diagnoses, in: Lecture Notes in Artificial Intelligence, Springer-Verlag, 1991.

[Yeap 81]
W. K. Yeap: Cognitive Map, University of Essex, Cognitive Studies Centre, CSCM-4.