Navigation mit eindimensionalen 360°-Bildern

Thomas Röfer
Zentrum für Kognitionswissenschaften, Universität Bremen
Postfach 330440, 28334 Bremen
E-mail: roefer@informatik.uni-bremen.de
WWW: http://www.informatik.uni-bremen.de/~roefer

1 Einleitung

Die Navigation autonomer mobiler Roboter ist noch immer kein vollständig gelöstes Problem. Roboter operieren in einer dreidimensionalen, veränderlichen Welt, in der sie in Echtzeit navigieren müssen. Methoden, die mit dreidimensionalen Weltmodellen arbeiten, sind sehr aufwendig: Das Modell muß erstellt werden, Sensordaten müssen mit dem Modell in Einklang gebracht werden und Veränderungen müssen eingetragen werden. Daher erscheint es sinnvoll, Methoden zu untersuchen, die weit weniger kompliziert sind. Eine Möglichkeit ist die Orientierung an biologischen Vorbildern, da selbst "einfache" Lebewesen hervorragende Navigationsleistungen vollbringen können.

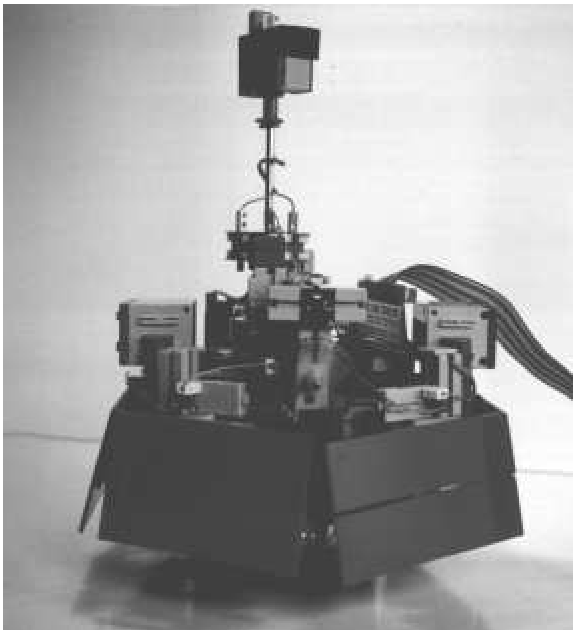

Abbildung 1: Der Versuchsroboter

Cartwright und Collett (1983, 1987) nehmen z.B. an, daß sich Bienen die genaue Position eines Futterplatzes anhand des Bildes merken, das sie an der Futterstelle von ihrer Umgebung wahrnehmen. Wenn sie die Futterstelle ein zweites Mal aufsuchen und sich dabei bereits in ihrer Nähe befinden, können sie aus dem Bild, das sie während des Anflugs sehen, und dem gespeicherten Bild die Richtung zur Futterstelle berechnen.
Diese Methode ist interessant, weil sie es erlaubt, einen Ort wiederzufinden, ohne daß ein Modell der Umwelt existiert. Es wird lediglich ermittelt, wie stark sich Punkte des aktuellen Bildes

gegenüber den Punkten im Zielbild verschoben haben. Aus den Verschiebungen wird dann die notwendige Bewegungsrichtung berechnet.
Das Verfahren wurde auf dem kleinen Versuchsroboter in Abbildung 1 mit einem eindimensionalen "rundum"-Helligkeitssensor implementiert und erfolgreich getestet.

2 Zuordnung von Bildteilen

In Abbildung 2 und Abbildung 3 sind zwei Bilder zu sehen, die mit dem Helligkeitssensor aufgenommen wurden. Das erste Bild wurde an einer Zielposition aufgenommen, das zweite Bild an einer Position, die etwa 50 cm von dieser Zielposition entfernt ist Um nach diesen Bildern navigieren zu können, muß festgestellt werden, welcher Teil des zweiten Bildes einem bestimmten Teil des ersten Bildes entspricht. Hong et al. (1991) entwickelte ein Verfahren, das nach charakteristischen Punkten in einem der Bilder sucht und versucht, diese Punkte in dem zweiten Bild wiederzufinden. Charakteristische Punkte sind Punkte mit maximalem Intensitätswechsel. Während der Suche werden normalisierte Werte innerhalb kleiner Fenster aus beiden Bildern miteinander verglichen. Durch die Normalisierung ergeben sich allerdings viele Verwechslungen. Hong versuchte dem zu begegnen, indem er den Suchbereich nach ähnlichen Punkten sehr stark einschränkte, wodurch nur sehr geringe Verschiebungen im Bild gefunden werden können. Es scheint, daß es nicht ausreicht, die Helligkeit innerhalb kleiner Bereiche miteinander zu vergleichen, um korrespondierende Bereiche in den Bildern zu finden.

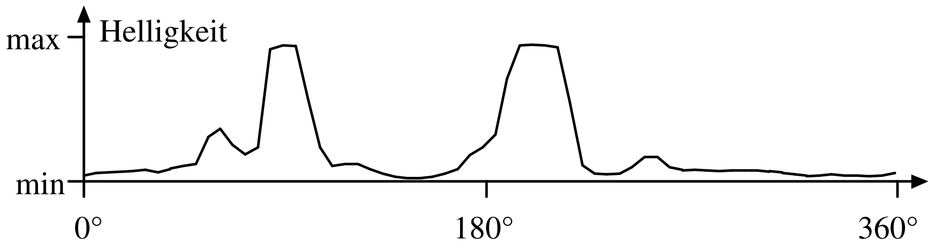

Abbildung 2: Typisches Bild an einer Zielposition

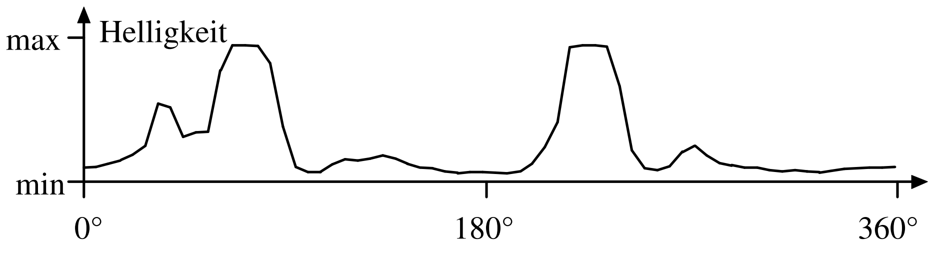

Abbildung 3: Bild einer Position, die etwa 50 cm von der Zielposition entfernt ist

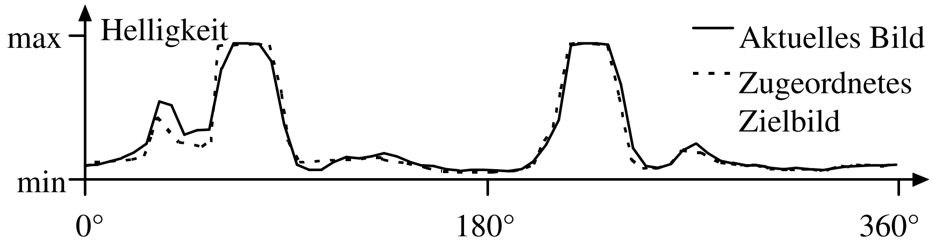

Abbildung 4: Zuordnung der Pixel des Zielbildes zu denen des aktuellen Bildes

Aus diesem Grund wurde von Röfer (1995a, 1995b) eine andere Methode entwickelt, die mehr Information aus den Bildern verwendet und daher zu besseren Ergebnissen kommt. Sie verwendet neben der Helligkeit noch deren erste und zweite Ableitung. Der entscheidende Vorteil gegenüber anderen Methoden ist aber, daß zusätzlich die Topologie der Pixel verwendet wird. Der Algorithmus geht davon aus, daß Punkte, die in einem Bild Nachbarn sind, auch in dem anderen Bild nebeneinander liegen. Diese Einschränkung sorgt dafür, daß die Bilder zuverlässig zur Deckung gebracht werden, solange die Nachbarschaftsbeziehungen in beiden Bildern tatsächlich gleich sind. Das Verfahren basiert auf einer modifizierten Version von Kohonens (1982) selbstorganisierender Merkmalskarte.

3 Berechnen einer Bewegung

Abbildung 5 zeigt eine typische Szene. Der Roboter hat an der Zielposition ein Bild aufgenommen. Ein bestimmter Punkt in der Umgebung wird durch Pixel P_i in dem Bild gesehen. Später befindet sich der Roboter an einer anderen Position und möchte zur Zielposition zurückkehren. Derselbe Punkt in der Umgebung wird nun durch Pixel P_j gesehen. Die Differenz *i-j* geteilt durch die Anzahl der Pixel *p* ist die Verschiebung s_i des Pixels[1]. Das Ziel des Roboters ist es, diese Verschiebung auf Null zu reduzieren. Wenn kein Pixel mehr verschoben ist, hat der Roboter die Zielposition erreicht.

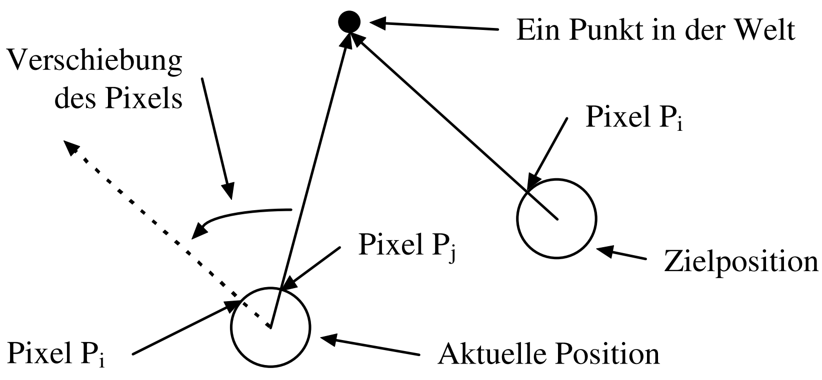

Abbildung 5: Die Peilungen eines Punktes der Umwelt in der aktuellen Position und der Zielposition

[1] Der Kohonen-Algorithmus liefert Zuordnungen, die genauer als ein Pixel sind, d.h. *i-j* ist reellwertig.

3.1 Berechnen der Eigenrotation

Bevor der Roboter die Richtung zum Ziel aus den Verschiebungen berechnen kann, muß eine eventuell vorhandene Eigenrotation kompensiert werden. Dazu eine einfache Überlegung: Wenn das aktuelle Bild nicht gegenüber dem Zielbild verdreht ist, zeigen alle Verschiebungen von der Zielrichtung weg. Daher ist die Hälfte der Verschiebungen positiv, die andere Hälfte negativ. Die Summe über alle Verschiebungen ist in diesem Fall Null. Wenn eine Verdrehung existiert, ist sie in jeder Verschiebung enthalten. Deshalb ergibt die Summe über alle Verschiebungen die Verdrehung multipliziert mit der Anzahl der Verschiebungen p. Daher kann die Eigenrotation bestimmt werden, indem der Durchschnitt über alle Verschiebungen gebildet wird. Diese wird verwendet, um die bereinigten Verschiebungen s_i' zu berechnen:

$$s_{\mathrm{i}}' = s_{\mathrm{i}} - \frac{1}{p} \sum_{\mathrm{j}} s_{\mathrm{j}} \tag{1}$$

3.2 Berechnen der Richtung zur Zielposition

Abbildung 6 illustriert den Algorithmus, mit dem die Richtung zum Ziel für jede einzelne Verschiebung berechnet wird. Wie in Hongs Bericht wird angenommen, daß der Roboter die Verschiebung eines Pixels am schnellsten verringern kann, wenn er die Mitte zwischen der Position des Pixels im aktuellen und im Zielbild bestimmt und sich dann senkrecht dazu bewegt. Da die Entfernung zu den gesehenen Objekten unbekannt ist, wird die Größe der Verschiebung als Länge für den dabei entstehenden Richtungsvektor verwendet. Dieser Vektor wird für jedes Pixel berechnet. Die einzelnen Vektoren werden dann zu dem endgültigen Fahrtrichtungsvektor zusammenaddiert. Dieser gibt dann die Richtung zur Zielposition an:

$$\beta = \tan^{-1} \frac{\sum_{\mathrm{i}} s_{\mathrm{i}}' \sin 2\pi \left(\frac{i}{p} - \frac{s_{\mathrm{i}}'}{2} + \frac{1}{4} \right)}{\sum_{\mathrm{i}} s_{\mathrm{i}}' \cos 2\pi \left(\frac{i}{p} - \frac{s_{\mathrm{i}}'}{2} + \frac{1}{4} \right)} \tag{2}$$

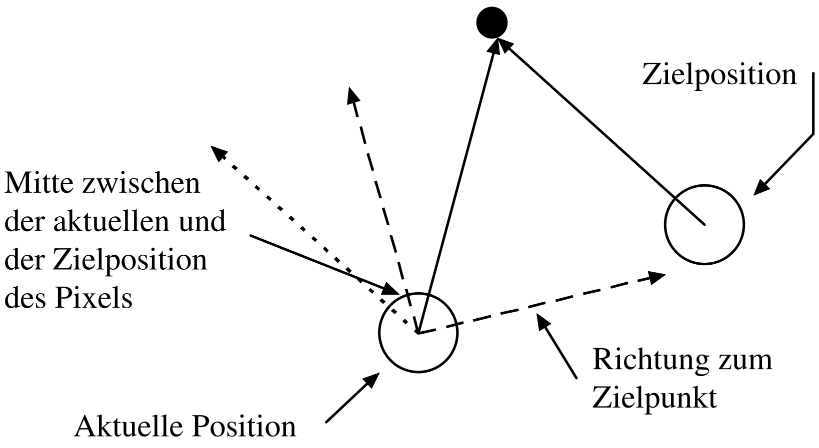

Abbildung 6: Bestimmen der Richtung zum Zielpunkt

3.3 Festlegen eines Abbruchkriteriums

Da der Roboter die Entfernung zur Zielposition nicht berechnen kann, arbeitet der Algorithmus inkrementell: Der Roboter berechnet die Richtung zum Ziel, fährt ein Stück und berechnet wieder die Richtung usw. Daher benötigt der Roboter ein Abbruchkriterium, das ihm sagt, wann er die Zielposition erreicht hat. Außerdem muß er wissen, wie weit er pro Schritt fahren soll. Für beides wird die Länge γ des unter 3.2 berechneten Vektors verwendet:

$$\gamma = \sqrt{\left(\sum_{\mathrm{i}} s_{\mathrm{i}}' \sin 2\pi\left(\frac{i}{p} - \frac{s_{\mathrm{i}}'}{2} + \frac{1}{4}\right)\right)^2 + \left(\sum_{\mathrm{i}} s_{\mathrm{i}}' \cos 2\pi\left(\frac{i}{p} - \frac{s_{\mathrm{i}}'}{2} + \frac{1}{4}\right)\right)^2} \qquad (3)$$

Um die Strecke zu berechnen, die der Roboter in einem Schritt zurücklegt, wird γ mit einem konstanten Faktor multipliziert. Der Roboter hat das Ziel erreicht, wenn γ kleiner als ein vorgegebener Wert ist.

4 Lernen von Trajektorien

Die Entfernungen, die sich mit dieser Methode der Navigation überwinden lassen, hängen von der Distanz zu den wahrgenommenen Objekten ab:

- Sind die Objekte der Umgebung weit entfernt, können große Distanzen überwunden werden, da sich das Bild, das der Roboter von seiner Umgebung aufnimmt, bei Bewegung nur langsam ändert. Allerdings kann die Zielposition auch nur ungenau wiedergefunden werden, weil sich das Bild der Umgebung innerhalb mehrerer zurückgelegter Meter kaum ändert.

- Ist der Abstand zur Umgebung gering, können gelernte Positionen präzise wieder aufgesucht werden, allerdings können auch nur geringe Entfernungen überwunden werden, da sich das Bild der Umgebung bei Bewegung sehr schnell verändert und schon nach kurzer Fahrt keine Gemeinsamkeiten zwischen dem gelernten und dem aktuellen Bild mehr festzustellen sind.

Für die Biene trifft eher der erste Fall zu. Daher reicht ihr ein Bild, um die gewünschte Position wiederzufinden. Für einen Roboter, der im Innenraum navigieren soll, gilt der zweite Fall, d.h. er kann mit der bisher vorgestellten Methode keine großen Distanzen überwinden. Daher ist es notwendig, das Verfahren zu erweitern. Deshalb wird nicht mehr nur eine Zielposition gelernt, sondern eine ganze Folge von Positionen. Diese wählt der Roboter während einer Trainingsfahrt selbständig aus. Der Ablauf ist folgendermaßen:

1. Der Roboter wird an eine Startposition gebracht. Diese Position wird benannt, damit der Benutzer dem Roboter später mitteilen kann, wohin er fahren soll. Der Roboter nimmt ein Bild dieser Position auf.

2. Mit einem Joystick wird der Roboter entlang einer zu lernenden Trajektorie gesteuert. Während er fährt, bestimmt er mit dem in Abschnitt 3 vorgestellten Algorithmus die Richtung zurück zur letzten gelernten Position, anfangs also zur Startposition. Liegt die so berechnete

Richtung genau entgegengesetzt zur aktuellen Fahrtrichtung (mit einer Toleranz von +/- 15°), so ist das Zurückfinden zur letzten gelernten Position noch gewährleistet. Weicht die Richtung allerdings ab, hat sich das Bild der Umgebung so stark verändert, daß die Rückkehrrichtung nicht mehr korrekt bestimmt werden kann. In diesem Fall lernt der Roboter das Bild der vorletzten Position[2], an der die Rückkehrrichtung noch bestimmt werden konnte. Dadurch entsteht eine Folge von Bildern, bei der sichergestellt ist, daß der Roboter von einer Position, die durch ein Bild in dieser Folge repräsentiert wird, zu der Position fahren kann, deren Bild sich unmittelbar davor in der Folge befindet. Positionen, an denen der Roboter seine Fahrrichtung ändert, werden generell gelernt.

3. Dieser Vorgang wird so lange wiederholt, bis der Benutzer dem Roboter signalisiert, daß er die Endposition dieser Trajektorie erreicht hat. Das Bild dieser Position wird ebenfalls gelernt. Auch die Endposition wird benannt. Dadurch entsteht ein Datensatz, der die Bildfolge enthält sowie die Information, das diese Folge z.B. den Weg vom Wohnzimmertisch zur Wohnzimmertür darstellt.

4. Während der Trainingsfahrt wurde dem Roboter gezeigt, wie er von Punkt A zu einem Punkt B gelangt. Gelernt hat er aber das genaue Gegenteil: er kann nämlich bisher nur von Punkt B zurück zu Punkt A fahren. Um auch die andere Richtung zu lernen, fährt der Roboter von sich aus zu Punkt A zurück und lernt währenddessen den Weg von A nach B. Die dafür benötigten Bilder werden nicht separat gespeichert, sondern in die bereits bestehende Bildfolge eingefügt. Im Idealfall muß der Roboter auf dem Rückweg keine neuen Bilder mehr lernen.

Wird der Roboter mit vielen Trajektorien trainiert, spannen diese einen Graphen auf, in dem der Roboter Wege von einem Ort zu einem anderen finden kann.

5 Hindernisvermeidung

Der Roboter fährt die Bahn zwischen zwei gelernten Punkten nicht immer auf einer Geraden ab, sondern bewegt sich teilweise auch in einem leichten Bogen. Daher ist es notwendig, ihn mit einer Hindernisvermeidung auszustatten, die dafür sorgt, daß er z.B. den Wänden nicht zu nahe kommt. Wie auch das Navigationsverfahren kommt der hier vorgestellte Ansatz ohne ein Modell der Umwelt aus.
Der angenommene Roboter ist tonnenförmig und rundum mit entfernungsgebenden Sensoren ausgestattet. Jeder Sensor hat eine Reichweite von 25 cm. Da die Sensoren rundum montiert sind, „blickt" jeder Sensor in eine Richtung α_i relativ zum robotereigenen Koordinatensystem. Die von jedem Sensor gemessene Distanz wird als d_i bezeichnet.
Die Ausweichstrategie beruht auf einer einfachen Überlegung: Wenn ein Hindernis in den Wahrnehmungsbereich der Sensoren tritt, so sollte der Roboter eine Aktion ausführen, die dazu führt, daß er sich von dem Hindernis entfernt. Betrachtet man nur einen einzelnen Entfernungssensor, so ist es die beste Aktion, in die Richtung zu fahren, die dem Wahrnehmungsbereich des Sensors genau gegenüberliegt, d.h. geradewegs vom Hindernis weg in Richtung $\alpha_i + \pi$. Als Fahrtstrecke reicht eine Distanz, die das Hindernis aus seinem Sensorbereich befördert,

[2] Die vorletzte Position wird gelernt, um auf der sicheren Seite zu sein.

also die Sensorreichweite abzüglich der gemessenen Entfernung d_i. Berechnet man diesen Bewegungsvektor für alle Sensoren und addiert die Vektoren, erhält man einen Richtungsvektor h, der den besten Kompromiß zwischen den Ausweichrichtungen darstellt.

$$h = \sum_i (25 - d_i) \begin{pmatrix} \cos(\alpha_i + \pi) \\ \sin(\alpha_i + \pi) \end{pmatrix} \tag{4}$$

Da der Roboter bei der Vermeidung von Hindernissen weiterhin seine trainierte Trajektorie abfahren soll, wird der Vektor h gewichtet und zur normalen Fahrtrichtung hinzuaddiert. Dadurch wird die normale Fahrtrichtung modifiziert und der Roboter „gleitet“ an Hindernissen vorbei.
Diese Art der Hindernisvermeidung sorgt zum Beispiel beim Durchfahren einer Tür dafür, daß der Roboter diese in der Mitte passiert. Außerdem bremst der Roboter vor der Tür leicht ab, erreicht zwischen den Türpfosten wieder seine normale Geschwindigkeit, verläßt die Tür mit etwas höherer Geschwindigkeit und kehrt zu seiner normalen Geschwindigkeit zurück, sobald er den Türbereich verlassen hat.

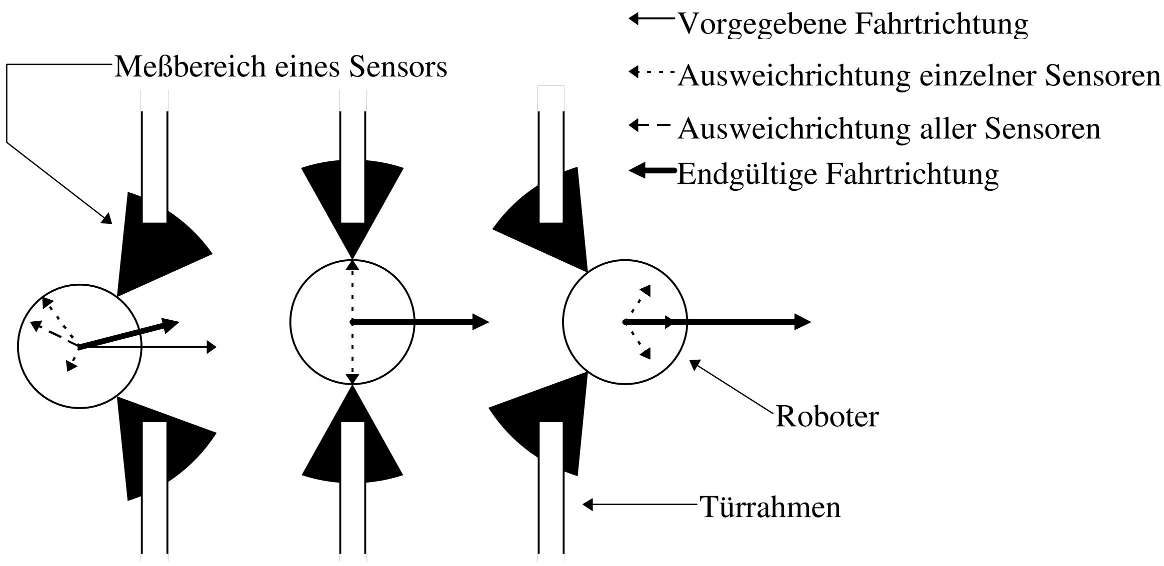

Abbildung 7: Schematische Darstellung des Durchfahrens eines Türrahmens

6 Simulationsergebnisse

Das hier vorgestellte Verfahren wurde in einer Simulation getestet. Diese wurde mit dem Robotik-Simulator *SimRobot* erstellt, der von Siems et al. (1994) entwickelt wurde[3]. Als Umgebung für den simulierten Roboter wurde eine Dachwohnung gewählt, die einem realen Vorbild nachempfunden und in Abbildung 9 zu sehen ist[4].

[3] SimRobot ist per WWW unter http://www.informatik.uni-bremen.de/~simrobot für verschiedene Plattformen erhältlich.

[4] Ein MPEG-Video der simulierten Wohnung kann unter der WWW-Adresse des Autors abgerufen werden.

Der simulierte Roboter verfügt über eine eindimensionale 360°-Farbkamera mit einer Auflösung von 64 Pixeln und über 16 Entfernungssensoren. Ein typisches Bild der Kamera ist in Abbildung 8 dargestellt. Gemessen an der Sensorausstattung des Roboters ist der Detaillierungsgrad der simulierten Umgebung ausreichend, um realistische Ergebnisse zu liefern.
Für die Zuordnungsberechnung der Farbbilder werden - im Gegensatz zu den Experimenten mit dem realen Roboter - hier keine Ableitungen der Helligkeit verwendet, sondern auf sieben Stufen verschieden stark tiefpaßgefilterte Bilder. Da drei Farbkanäle existieren, zieht der Algorithmus daher 21 verschiedene Informationen pro Pixel heran, um die Verschiebungen zwischen den zwei Bildern zu berechnen. Dieses Verfahren liefert in der gesamten simulierten Wohnung zufriedenstellende Ergebnisse.

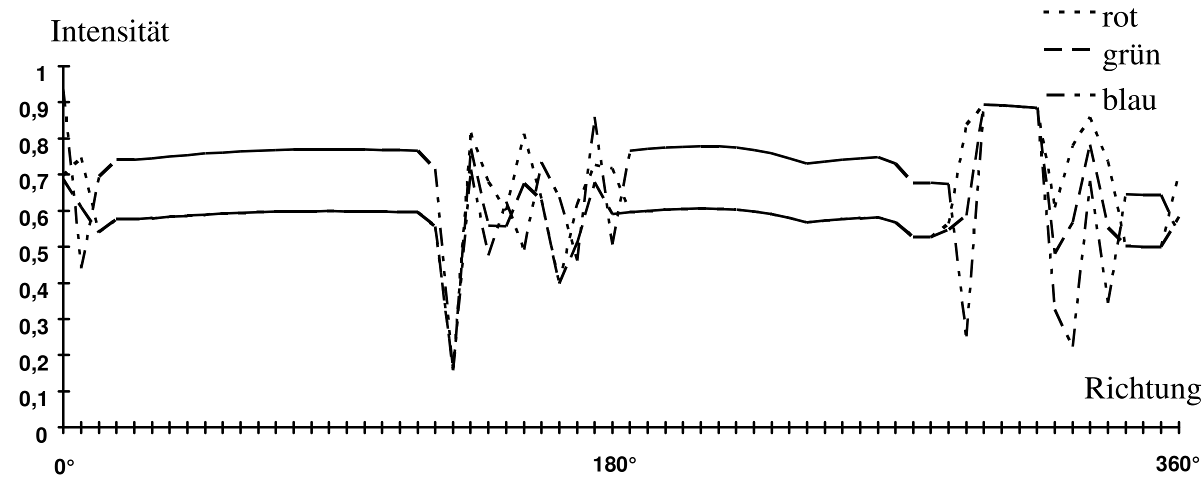

Abbildung 8: Ein simuliertes 360°-Farbbild

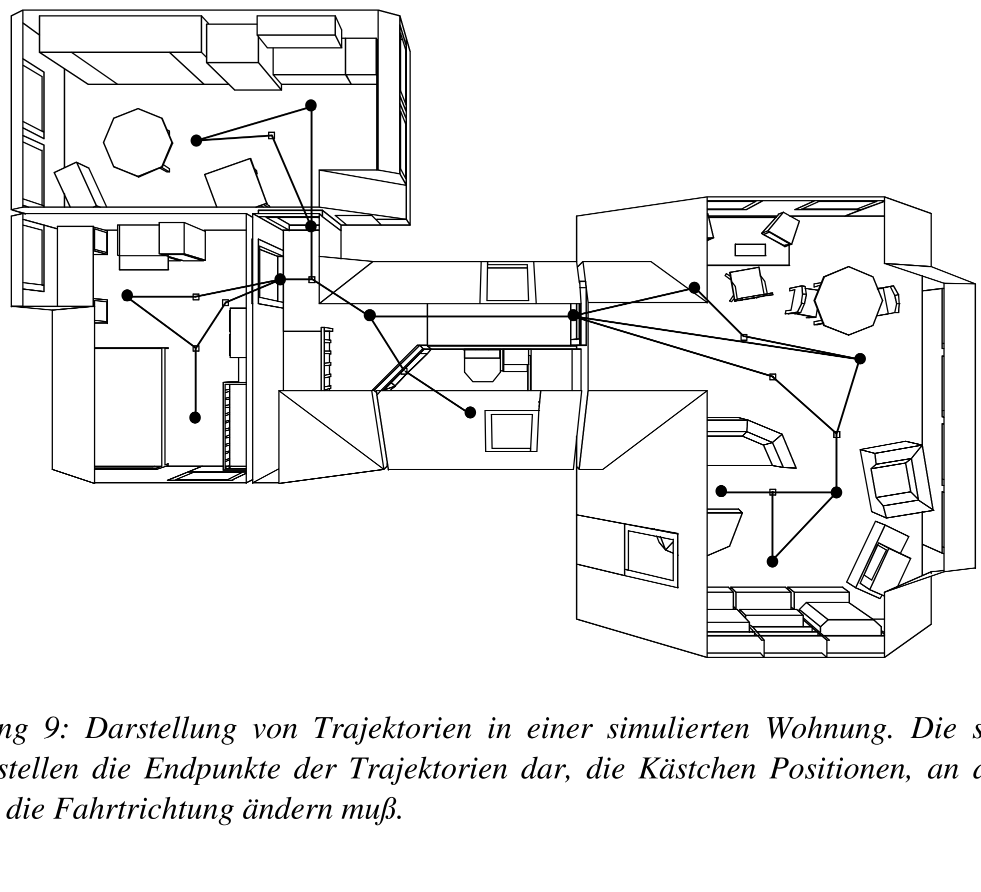

Abbildung 9: Darstellung von Trajektorien in einer simulierten Wohnung. Die schwarzen Punkte stellen die Endpunkte der Trajektorien dar, die Kästchen Positionen, an denen der Roboter die Fahrtrichtung ändern muß.

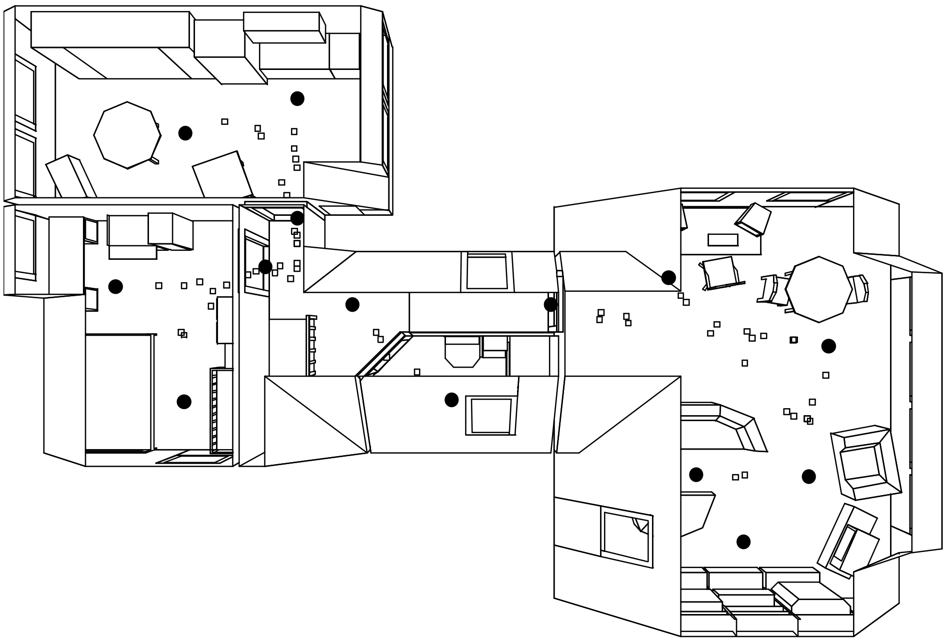

Abbildung 10: Darstellung der Positionen, an denen der Roboter ein Bild der Umgebung gelernt hat. Die schwarzen Punkte stellen die Endpunkte der Trajektorien dar, die Kästchen weitere Orte, die er gelernt hat.

In der Simulation wurden zwanzig Trajektorien trainiert. Beim Training dieser Wege lernt der Roboter 104 Bilder. Die Positionen, an denen die Bilder aufgenommen wurden, sind in Abbildung 10 zu sehen. 53 der gelernten Bilder sind durch Start- und Endpunkte sowie Punkte, an denen die Fahrtrichtung geändert wird, bereits fest vorgegeben. Die übrigen 51 Bilder hat der Roboter selbst ausgewählt. Der Speicherplatz für die 104 Bilder beläuft sich auf nur etwa 20 KB[5] - nicht viel für ein „Umweltmodell".

7 Zukünftige Arbeiten

Folgende Weiterentwicklungen sind für die Zukunft geplant:

- Der momentan in der Simulation verwendete Algorithmus arbeitet nur bei konstanten Beleuchtungsverhältnissen. Daher soll untersucht werden, inwieweit das Verfahren gegenüber Beleuchtungsschwankungen resistent gemacht werden kann. Hier ist z.B. an einen Wechsel des Farbmodells von RGB hin zu Farbe/Helligkeit/Sättigung gedacht. Bei entsprechender Gewichtung der Komponenten könnte eine Unabhängigkeit von der Beleuchtung erreicht werden.

- Da nichts den Test in der Realität ersetzen kann, soll das Verfahren auf einem realen Roboter implementiert werden.

[5] 104 Bilder mal 64 Pixel mal 3 Farbkanäle

8 Zusammenfassung

Ein Umweltmodell-freies Navigationsverfahrens wurde vorgestellt. Aus zwei eindimensionalen 360°-Bildern - eines an einer Zielposition aufgenommen, das andere an der gegenwärtigen Position - wurde die Fahrtrichtung zur Zielposition bestimmt. Diese aus der Biologie bekannte Methode wurde zu einem Navigationsverfahren für Roboter ausgebaut. Dabei navigierte ein Roboter ausschließlich anhand einer Liste von Bildern. Beim Abfahren einer Trajektorie „hangelte" er sich an dieser Liste von Bild zu Bild. Die dazu notwendigen Bilder wählte der Roboter während einer Trainingsfahrt selbständig aus. Das Verfahren wurde um eine einfache Hindernisvermeidung ergänzt.

Das Zurückkehren zu einer Zielposition anhand zweier Bilder wurde auf einem realen Roboter getestet. Das gesamte Navigationsverfahren wurde erfolgreich in einer Simulation implementiert. Die Übertragung der gesamten Arbeit auf einen realen Roboter ist geplant.

Danksagungen

T. Röfer wurde von der Deutsche Forschungsgemeinschaft und der Freien Hansestadt Bremen durch das Graduiertenkolleg „Raumorientierung und Handlungsorganisation autonomer Systeme" gefördert.

Literatur

Cartwright, B. A. / Collett, T. S.: Landmark Learning in Bees, Journal of Comparative Physiology 151, 521-543 (1983)

Cartwright, B. A. / Collett, T. S.: Landmark Maps for Honeybees, Biological Cybernetics 57, 85-93 (1987)

Hong, J., Tan, X. / Pinette, B. / Weiss, R. / Riseman, E. M.: Image-based Homing, Proc. of the 1991 IEEE International Conference on Robotics and Automation, 620-625 (1991)

Kohonen, T.: Self-organized Formation of Topologically Correct Feature Maps, Biological Cybernetics 43, 59-69 (1982)

Röfer, T.: Controlling a robot with image based homing, in: Krieg-Brückner, Roth, Schwegler (Hrsg.): ZKW Bericht Nr. 3/95, Universität Bremen (1995)

Röfer, T.: Image based homing using a self-organizing feature map, in: Proc. Int. Conf. Artificial Neural Networks, erscheint im Oktober (1995)

Siems, U. / Herwig, C. / Röfer, T.: SimRobot, ein System zur Simulation sensorbestückter Agenten in einer dreidimensionalen Umwelt, Krieg-Brückner, Roth, Schwegler (Hrsg.): ZKW Bericht Nr. 1/94, Universität Bremen (1994)