Farbbild-basierte Monte-Carlo-Selbstlokalisation für Mobile Roboter *

Schröter, C., König, A., Groß, H.-M.
Technische Universität Ilmenau, Fachgebiet Neuroinformatik
christof.schroeter@tu-ilmenau.de

Zusammenfassung

Wir stellen ein Lokalisationsverfahren für autonome mobile Roboter vor, welches zur Erfassung seiner Umwelt Panoramafarbbilder der Umgebung verwendet. Die entwickelte Lösung beruht auf der Monte-Carlo-Lokalisation (MCL), ein partikelbasiertes Verfahren zur Zustandschätzung in dynamischen Wahrscheinlichkeitsverteilungen. Das bekannte Verfahren, welches abstandsmessenden Sensoren verwendet, führt in unserer realen Umgebung zu Mehrdeutigkeiten und wurde daher so modifiziert, dass es mit Bildern einer omnidirektionalen Kamera arbeitet. Wir präsentieren experimentelle Ergebnisse, die mit einem mobilen Roboter gewonnen wurden, welcher als interaktiver Shopping-Assistent in einer Baumarkt-Umgebung agiert. Unsere Resultate zeigen, dass das vorgestellte Verfahren eine genaue und robuste Selbstlokalisation für autonome Roboter ermöglicht.

1 Einführung und Motivation

In unserem PERSES/SerRoKon-Projekt [3] entwickeln wir einen Serviceroboter, welcher als interaktiver Einkaufs-Assistent innerhalb einer Markt-Umgebung dienen soll. Typische Dienstleistungen, die angeboten werden sollen, beinhalten beispielsweise das Führen des Kunden zu einem gesuchten Artikel oder das Begleiten als mobiles Informationsterminal. Die Akzeptanz eines solchen Systmems hängt dabei entscheidend von einer möglichst intuitiven Interaktion mit den Kunden ab. Zur Lösung dieser Aufgabe werden insbesondere visuell basierte Methoden eingesetzt, da diese einerseits in der Lage sind, viele Informationen über die Umgebung zu vermitteln, die mit anderen Sensore nicht wahrgenommen werden können und andererseits zu einer möglichst natürlich wirkenden Kommunikation mit menschlichen Nutzern führen sollen. Zu diesem Zweck ist der Roboter unter anderem mit einer omnidirektionalen Kamera ausgestattet, welche aus einer digitalen Farbkamera und einem sphärischen Spiegel besteht (Abb. 4). Neben

*gefördert durch das Thüringer Ministerium für Wissenschaft, Forschung und Kunst, Projekt SERROKON(B609-01017)

der Interaktion müssen eine Reihe von Navigationsaufgaben gelöst werden. Der Roboter muss eine Repräsentaton seiner Einsatzumgebung kennen und in der Lage sein, bestimmte Punkte darin selbstständig anzufahren. Dazu muss er zu jedem Zeitpunkt Kenntnis von seiner eigenen Position innerhalb der Umwelt haben (Selbstlokalisation).

2 Monte-Carlo-Lokalisation

Eine relativ neue Methode zur Selbstlokalisation ist die Monte-Carlo-Lokalisation (MCL) [1], [2]. Sie beruht auf der Schätzung der Wahrscheinlichkeitsverteilung für die tatsächliche Position des Roboters mittels eines Partikel-Filters. Jedes Partikel (Sample) gibt durch den ihm zugeordneten *Importance Factor* für eine bestimmte Position im Raum die Wahrscheinlichkeit an, dass sich der Roboter dort befindet. Bei initial unbekannter Position entspricht die zu approximierende Wahrscheinlichkeitsverteilung einer Gleichverteilung über alle möglichen Zustände (Zustand = x,y,Orientierung des Roboters). Die Samples werden daher am Anfang zufällig verteilt und erhalten alle den gleichen Importance Factor. Während der Bewegung des Roboters werden die Positionen der Samples und ihr zugehöriger Importance Factor schrittweise aus der vorherigen Verteilung neu berechnet. Dazu werden zunächst Samples mit niedriger importance entfernt und in der Umgebung von Samples mit hoher importance neue Samples eingefügt. Die Sample-Verteilung adaptiert sich also derart, dass an den Positionen, die mit hoher Wahrscheinlichkeit als tatsächliche Position eingeschätzt werden, die Dichte der Samples erhöht wird. Als nächstes wird die gemessene Eigenbewegung des Roboters auf die Samples übertragen. Dabei kommt ein probabilistisches Bewegungsmodell zur Anwendung, welches die Ungenauigkeit der Bewegungsmessung berücksichtigt. Zuletzt wird der Importance Factor jedes Samples neu berechnet, indem der aktuelle Sensoreindruck y_t aus der Umgebung mit dem für jedes Sample i zu erwarteten Sensoreindruck $y_t^{(i)}$ verglichen wird. Dazu wird ein Umgebungsmodell und ein Sensormodell benötigt. Das Umgebungsmodell beschreibt die dem Roboter bekannte Umwelt und enthält zu jeder möglichen Position die erwartete Beobachtung (Sensoreindruck). Das Sensormodell gibt an, mit welcher Wahrscheinlichkeit an der selben Position der selbe sensorische Input erzeugt wird. Es berücksichtigt dabei die Ungenauigkeit der Beobachtungsmessung. Die Ungenauigkeit wird im MCL-Algorithmus verwendet. Diese Berechnungen werden zyklisch wiederholt. Die aktuelle Positionsschätzung ergibt sich aus dem mit dem Importance Factor gewichteten Positionen aller bzw. der besten Samples (z.B 10%). Durch Berücksichtigung von Eigenbewegung und Umweltbeobachtung wird, ausgehend von einer völlig unbekannten Position, innerhalb weniger Schritte die tatsächliche Position ermittelt. Die Verwendung des Partikel-Filters verringert den Zeitbedarf gegenüber anderen Verfahren und ermöglicht gleichzeitig durch die hohe Auflösung an interessanten Punkten eine sehr hohe Genauigkeit.

Bisherige Implementierungen der MCL benutzten zur Wahrnehmung der Umgebung abstandsmessende Sensoren wie Ultraschall- oder Laserentfernungsmesser. In unserem Szenario sind diese jedoch nur bedingt geeignet, da die Einsatzumgebung, ein Baumarkt, viele gleichartige lange Gänge aufweist, was zu sehr viele gleichen Beob-

achtungen an unterschiedlichen Orten (sensorische Mehrdeutigkeiten) und damit zu Unsicherheiten bei der Positionsbestimmung führt. Abbildung 1 zeigt den Grundriss des Baumarktes. Der Roboter erhält an vielen Stellen den gleichen Sonarscan und kann seine tatsächliche Position nur sehr schwer bestimmen. Durch die unterschiedliche Bestückung der Regalreihen sind diese allerdings farblich relativ gut unterscheidbar. Unser Ansatz besteht daher darin, visuelle Merkmale der Umwelt für die Lokalisation zu verwenden. Durch die Verwendung von Farbmerkmalen, welche aus dem Bild der Omnikamera extrahiert werden, sollte eine deutlich robustere Lokalisation erreicht werden.

3 Umwelterfassung und Merkmalsextraktion

In dieser Arbeit wurde eine Version der Monte-Carlo-Lokalisation entwickelt, die eine omnidirektionale Kamera zur Erfassung der Umwelt nutzt. Sie ermöglicht uns eine 360°-Panoramaansicht der Umgebung. Für eine vollständige Ansicht müssen keine Einzelbilder aneinander gefügt werden, wie es bei Frontalkameras der Fall ist. Der sensorische Umwelteindruck (Kamerabild) muss zunächst in einer geeigneten Weise vorverarbeitet werden, da eine Nutzung des eigentlichen Kamerabildes zur Lokalisation aus Komplexitätsgründe nicht sinnvoll und möglich ist. Aus dem Kamerabild werden zunächst bildbeschreibende Merkmale berechnet, um so die Dimension des Eingaberaums zu verringern. Diese Merkmale charakterisieren die Beobachtung $\underline{\mathbf{y}}_t$, die der Roboter erhält, wenn er sich im Zustand $\underline{\mathbf{x}}_t$ befindet. Mit Hilfe der Merkmale $\underline{\mathbf{y}}_t$ schätzt die Monte-Carlo-Lokalisation, wie in Abschnitt 2 beschrieben, den aktuellen Zustand (Position und Orientierung) des Roboters.

3.1 Merkmalsextraktion

Zunächst wird das kreisförmige omnidirektionale Kamerabild polartransformiert. Für jeden Punkt $P(x, y)$ des Originalbildes wird der Radius r und der Winkel φ bezüglich des Mittelpunktes des Bildes bestimmt (Mittelpunkt des Bildes = Rotationsachse des Roboters). Die polaren Koordinaten des Eingangsbildes werden anschließend in die kartesischen Koordinaten des Ergebnisbildes transformiert. Die Abb. 3 zeigt das Ausgangsbild und das Ergebnis der Transformation. Das Ergebnisbild wird anschließend für die Merkmalsextraktion in n vertikale Segemente unterteilt. Die Segmentierung des Ergebnisbildes entspricht einer Unterteilung des Ausgangsbildes in Kreissegemente (Abb. 3).

Aus jedem der n Segmente können jetzt bildbeschreibende Merkmale berechnet werden. Für die Merkmalsextraktion wurde ein sehr einfacher Ansatz gewählt. Die Merkmale berechnen sich aus der mittleren Intensität pro Farbkanal und pro Segment (hier RGB-Farbraum $\Rightarrow 3 \cdot n$ Merkmale). Mit diesen Merkmalen soll die farbige Vielfalt der Baumarktumgebung erfaßt werden. Der daraus entstandene Merkmalsvektor beschreibt die aktuelle Beobachtung $\underline{\mathbf{y}}_t$ des Roboters, welche er in einem bestimmten Zustand $\underline{\mathbf{x}}_t$ erhalten hat. Mit diesem Merkmalsvektor wird die räumliche Verteilung von

Farbe in der lokalen Umgebung erfasst, um damit einen Rückschluß auf die Position und die Orientierung des Roboter zu erhalten. Hier wird eine sehr einfache Merkmalsextraktion verwendet, um den Berechnungsaufwand für die Merkmalsgewinnung, sowie für die Weiterverarbeitung der Merkmale so gering wie möglich zu halten, da das Lokalisationsverfahren unter Echtzeitbedingungen eingesetzt werden soll.

3.2 Beobachtungsschätzung

Mit den oben beschriebenen Merkmalen wird eine Umweltkarte aufgebaut, indem Referenzbeobachtungen aus der Umgebung gesammelt werden. Hierfür ist eine Teach-Fahrt des Roboters nötig. Die Position und die Orientierung der Beobachtung, sowie die Beobachtung selbst werden im Modell (Graphenrepräsentation) gespeichert (Abb. 2). Für den MCL-Algorithmus ist es wichtig für jeden möglichen Zustand $\underline{\mathbf{x}}_t$ des Roboters die zu erwartende Beobachtung $\underline{\mathbf{y}}_t$ zu kennen. Dafür wird im Umgebungsmodell der Referenzpunkt $\underline{\mathbf{x}}_r$ mit dem kleinsten Abstand zu dem gesuchten Zustand $\underline{\mathbf{x}}_t$ berechnet. Diese Referenzbeobachtung $\underline{\mathbf{y}}_r$ ist annähernd gleich der Beobachtung $\underline{\mathbf{y}}_t$ im Zustand $\underline{\mathbf{x}}_t$, wenn ein bestimmter Abstand zwischen dem Referenzpunkt $\underline{\mathbf{x}}_r^{(i)}$ und dem Zustand $\underline{\mathbf{x}}_t$ nicht überschritten wird. Enthält das Umgebungsmodell genügend viele Referenzpunkte mit entsprechend geringem Abstand zueinander, so kann für jeden möglichen Zustand eine zu erwartende Beobachtung über das Umgebungsmodell bestimmt werden (siehe Abb. 2).

Dies gilt aber nur für die Position und nicht für die Orientierung des Roboters. Damit wir die Beobachtung für jede mögliche Orientierung bestimmen können, wird der Vorteil der omnidirektionalen Kamera genutzt, dass sich bei der Rotation lediglich der Bildinhalt dreht. Für das transformierte Bild führt dies zu einer horizontalen Verschiebung der Bildpixel. Da im Umgebungsmodell nur die Merkmalsvektoren enthalten sind, hat dies zur Folge, dass nur die Elemente des Merkmalsvektors verschoben werden müssen (siehe Abb. 5). Bei Rotationen die zu Verschiebung von nicht ganzzahligen Elementen führen, müssen die Elemente durch eine lineare Interpolation bestimmt werden. Damit kann für jede Position und Orientierung mit einem geringen Fehler eine erwartete Beobachtung geschätzt werden.

3.3 Sensormodell

Um das oben angesprochene Sensormodell zu bestimmen, wurde eine Häufigkeitsanalyse [4] durchgeführt. Hier wurde gemessen, wie stark die Beobachtung $\underline{\mathbf{y}}$ in einem festen Zustand $\underline{\mathbf{x}}$ variieren kann. Dies führt zu einer Verteilung wie sie in Abb. 6 dargestellt ist. Diese kann durch eine Gaußkurve approximiert werden und ermöglicht uns, die Wahrscheinlichkeit zu bestimmen, dass sich der Roboter bei der Messung einer Beobachtung $\underline{\mathbf{y}}_t$ im Zustand $\underline{\mathbf{x}}_t$ befand. Mit dem Sensormodell werden die Fehler bei der Messung einer Beobachtung erfasst und können so bei der Lokalisation berücksichtigt werden.

3.4 Einfluss von Beleuchtungsschwankung

Bei der Verwendung einer Kamera zur Erfassung der Umwelt ergeben sich neue Probleme. Dies trifft im besonderen auf die Schwankung der Beleuchtung zu. Auch bei unterschiedlich starker Beleuchtung der Umgebung müssen die Beobachtungen miteinander vergleichbar sein. Damit dies möglich ist, wird die Blende der Kamera meist automatisch geregelt. In unserem Ansatz wird auf die kameraeigene Blendenregelung verzichtet, da diese mit der Verwendung des Omni-Spiegels zu Regelungsproblemen führt (Bild wird zu hell geregelt). Stattdessen wurde eine eigenen Regelung implementiert, welche versucht, die Blende so zu steuern, dass das innere des interessierenden Kreisringes eine bestimmte mittlere Helligkeit aufweist. Diese Helligkeit wird im Kreisring in sechs verschiedenen rezeptiven Feldern gemessen. Mit der Abweichung der aktuellen von der gewünschten Helligkeit des Bildes wird die Blende gesteuert (siehe Abb. 7). Damit können sehr gut globale Beleuchtungsschwankungen (z.B. durch Oberlichter verursacht) kompensiert werden, so dass Beobachtungen auch unter unterschiedlichen Beleuchtungssituationen vergleichbar sind. Problematisch sind jedoch lokale Beleuchtungsänderungen, welche sich nur auf einen Teil des Bildes beschränken (z.B. Fensterfassaden). Sie können durch diese Regelung nicht ausgeglichen werden und führen so zu unterschiedlichen Beobachtungen am selben Ort. Diese Schwankungen können derzeit nur durch die Robustheit der Positionsschätzung mittels Partikelfilter kompensiert werden.

4 Experimentelle Ergebnisse

Die hier vorgestellte visuell basierte Monte-Carlo-Lokalisation wurde in einer realen Baumarktumgebung getest. Dafür wurde zunächst ein Umgebungsmodell mit den zugehörigen Merkmalsvektoren aufgebaut. Die Größe der Umgebung betrug $45 \cdot 50$ Meter (≈ 2000 m^2). Der Roboter hatte die Aufgabe, sich aus einer unbekannten Situation (eigene Position ist nicht bekannt) heraus selbst zu lokalisieren. In Abb. 10 ist die Anfangssituation dargestellt. Die Samples (rote Punkte) sind gleichmäßig in der Umgebung verteilt. Der blaue, mit „robot“ bezeichnete, Punkt ist die tatsächliche Position, welche bei den Experimenten mitgeschrieben wurde. Der Mittelpunkt und der Radius des Kreises zeigen die geschätzte Position und die Sicherheit der Schäzung an. Der Roboter führt nun Bewegungsschritte aus und aktualisiert seine Schätzung mit Hilfe der erhaltenen Beobachtungen. Man erkennt (Abb. 11, nach vier Schätzungen = vier Bewegungsschritte), dass sich die Samples an wahrscheinlicheren Position konzentrieren und zum Schluß an der tatsächlichen zusammenlaufen. In Abb. 12 ist die Situation nach neun Schätzungen dargestellt. Hier hat sich der Roboter korrekt lokalisiert. Die geschätzte Position des Roboters ergibt sich aus dem Schwerpunkt der Samplewolke. In weiteren Bewegungsschritten muss die korrekt geschätzte Position nur noch verfolgt werden (Positionsverfolgung). Der durchschnittliche Lokalisationsfehler beträgt dabei $\approx$ 25 - 50 cm. In der Abb. 8 ist der Zusammenhang zwischen der Anzahl der Segmente des Bildes und dem Lokalisationsfehler zu sehen. Die besten Ergebnisse erreicht

man mit 10 Segmenten. Ein weiteres sehr häufiges Problem in realen Umgebungen ist die Verdeckung des Bildes durch Personen. Bei der hier verwendeten Omni-Kamera kommen realistische Verdeckungen im Bereich von 0 - 15 % vor. Die Abb. 9 zeigt den Einfluß auf die Lokalisationsgenauigkeit. Für typisch auftretende Verdeckungen erhöht sich der Fehler nur gering.

5 Zusammenfassung

Mit dem hier vorgestellte Verfahren sind mobile autonome Roboter in der Lage sich in großen, realen Umgebungen selbst zu lokalisieren. Dies wird durch die bekannte Monte-Carlo-Lokalisation erreicht, welche so verändert wurde, dass sie Farbkamerabilder zur Wahrnehmung der Umwelt nutzen kann. Die visuelle Erfassung der Umgebung erhöht die Lokalisationsgschwindigkeit und -genauigkeit, besonders bei sehr großen Einsatzumgebungen, da sehr viel weniger sensorische Mehrdeutigkeiten im Vergleich zu herkömmlichen entfernungsmessende Sensoren auftreten. Die korrekt Lokalisation erfolgt auch unter unterschiedlichen Beleuchtungen (wird durch Blendenregelung kompensiert) und Verdeckungen durch Personen (Roboustheit der Partikelfiltersysteme). Die Lokalisationsgenauigkeit nimmt dabei etwas ab, aber dennoch erfolgt eine korrekte Positionsbestimmung. Für den Einsatz als Shoppingassitent ist eine Genauigkeit von 25 - 75 cm völlig ausreichend, um bestimmte Regalreihen bzw. Gangabschnitt exakt anzufahren. Das System arbeitet in Echtzeit. Dies wird durch die einfach Merkmalsextraktion und den Sampleansatz erreicht.

Literatur

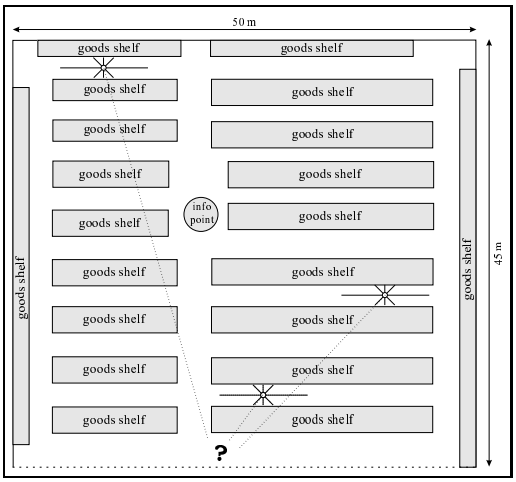
[1] Doucet, A., Godsill, S.J., Andrieu, C., On Sequential Monte Carlo Sampling Methods for Bayesian Filtering, Statistics and Computing, vol. 10, no. 3, pp. 197-208, 2000

[2] Fox, D., Burgard, W., Delleart, F. and Thrun, S. (1999b) Monte carlo localization: Efficent position estimation for mobiles robots, In Proc. of the AAAI, National Conference on Artificial Intelligence, AAAI, Orlando, Fl.

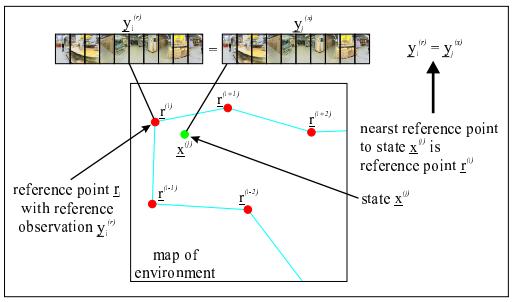
[3] Gross, H.-M., Boehme, H.-J. A contribution to vision-based localization, tracking and navigation methods for an interactive mobile service-robot. in: 2001 IEEE Intern. Conference on Systems, Man and Cybernetics (IEEE-SMC 2001), Tucson (USA), pp. 672-677, IEEE Press, 2001

[4] A. König, Automatischer Kartenaufbau für die visuelle Monte-Carlo-Selbstlokalisation mobiler Serviceroboter in Baumarktumgebungen, Diplomarbeit, TU Ilmenau 2002

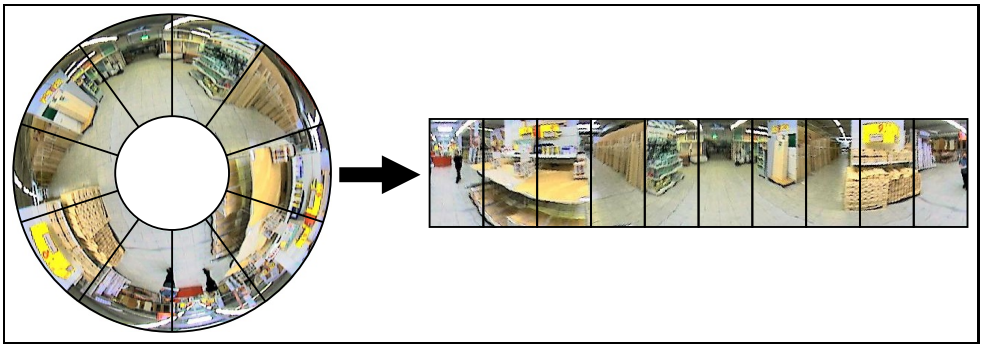
1)


50 m
goods shelf
info point
45 m
?

2)

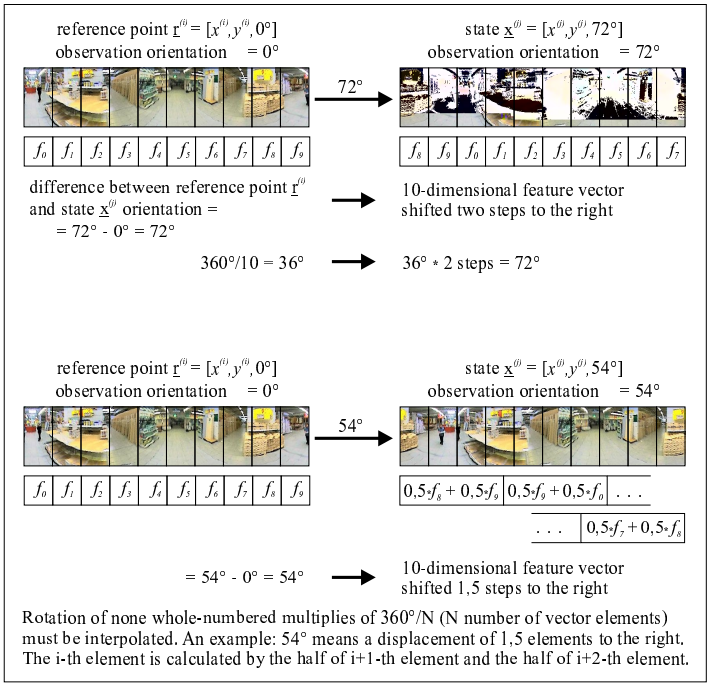

reference point r with reference observation y
map of environment
state x
nearst reference point to state x is reference point r

3)

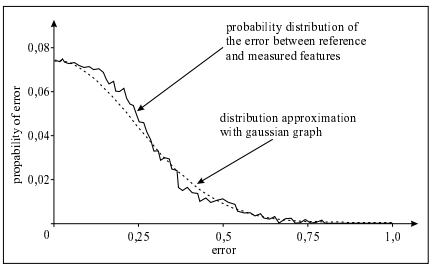


4)

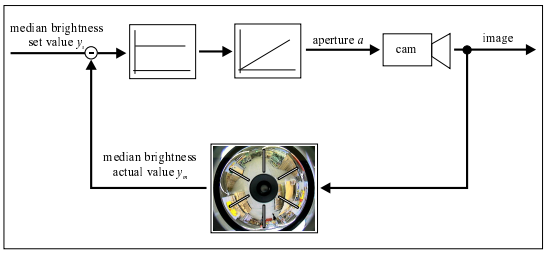
5)


reference point r = [x, y, 0°]
observation orientation = 0°
state x = [x, y, 72°]
observation orientation = 72°
72°
difference between reference point r and state x orientation =
= 72° - 0° = 72°
10-dimensional feature vector shifted two steps to the right
360°/10 = 36°
36° * 2 steps = 72°
reference point r = [x, y, 0°]
observation orientation = 0°
state x = [x, y, 54°]
observation orientation = 54°
54°
= 54° - 0° = 54°
10-dimensional feature vector shifted 1,5 steps to the right
Rotation of none whole-numbered multiplies of 360°/N (N number of vector elements) must be interpolated. An example: 54° means a displacement of 1,5 elements to the right. The i-th element is calculated by the half of i+1-th element and the half of i+2-th element.

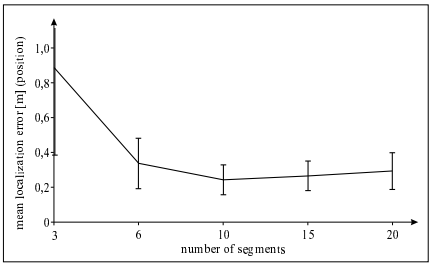
6)


probability distribution of the error between reference and measured features
distribution approximation with gaussian graph
propability of error
0,08
0,06
0,04
0,02
0
0,25
0,5
0,75
1,0
error

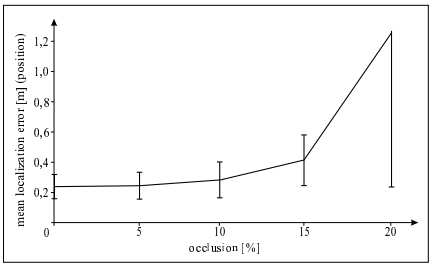
7)


median brightness set value y_s
aperture a
cam
image
median brightness actual value y_m

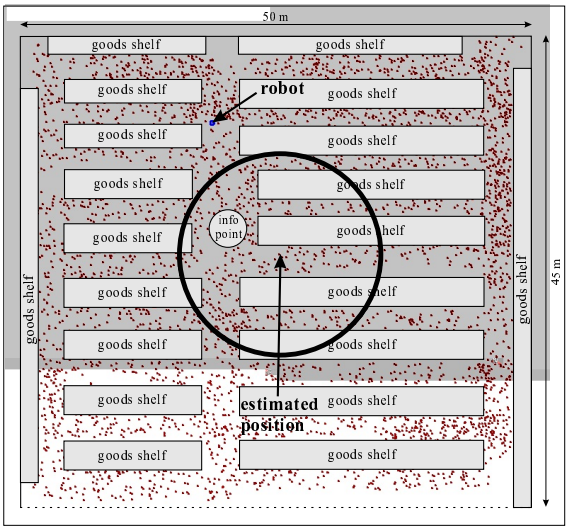
8)


mean localization error [m] (position)
1,0
0,8
0,6
0,4
0,2
0
3
6
10
15
20
number of segments

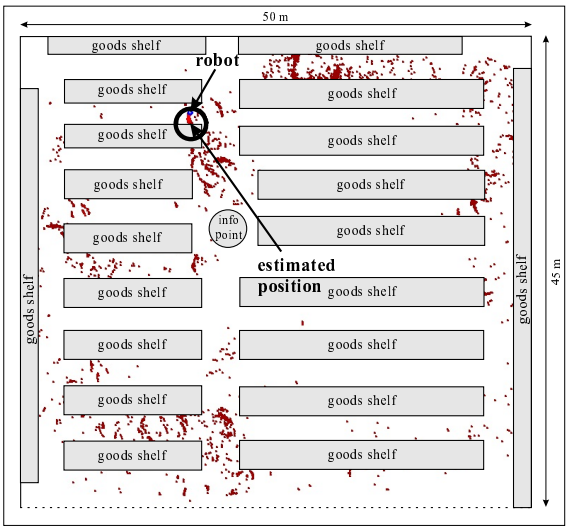
9)


mean localization error [m] (position)
1,2
1,0
0,8
0,6
0,4
0,2
0
5
10
15
20
occlusion [%]

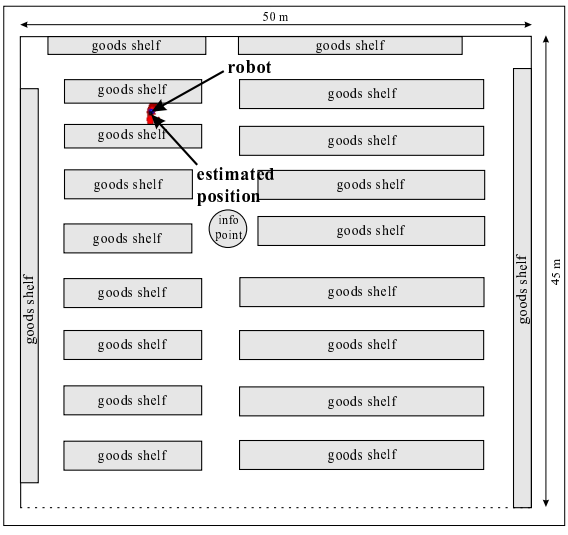
10)


50 m
goods shelf
robot
info point
estimated position
45 m

11)


50 m
goods shelf
robot
info point
estimated position
45 m

12)


50 m
goods shelf
robot
estimated position
info point
45 m