Proceedings of SOAVE 2000, Ilmenau, Germany, VDI-verlag Nr. 643, pp 31-38, 2000.

Visuell basierte Monte-Carlo Lokalisation für mobile Roboter mit omnidirektionalen Kameras

Alexander König, Jürgen Key & Horst-Michael Gross

Zusammenfassung

Es wird ein Ansatz zur Selbstlokalisation eines autonomen Roboters vorgestellt. Das Finden und Verfolgen der Position erfolgt mittels der Monte-Carlo Methode. Die Umwelt wird extrem spärlich durch ein sogenanntes View-Grid repräsentiert. Die Merkmale werden aus Aufnahmen einer in der Rotationsachse des Roboters montierten omnidirektionalen Kamera gewonnen. Die Position des Roboters kann mit dem vorgestellten Ansatz durch Interpolation im Zustands- und im Merkmalsraum mit einer Genauigkeit, die unter der Maschenweite des View-Grid liegt, ermittelt werden.

1 Weshalb visuell basierte Selbstlokalisation?

Der vorliegende Ansatz wurde im Rahmen des PERSES-Projektes[1] realisiert. In diesem Projekt geht es um die Entwicklung eines mobilen Shoppingassistenten, der in realen Umwelten zur Selbstlokalisation, zur Interaktion mit Kunden und zur Navigation fähig sein soll.

Navigation und Selbstlokalisation sollen dabei im wesentlichen über visuelle Sensoren erfolgen. Dies hat mehrere Gründe: Einmal arbeiten verfügbare Entfernungssensoren wie Ultraschall, Laser oder Infrarot nur in einer Ebene. Das bedeutet, daß das System damit die nicht in der Sensorebene liegenden Hindernisse nicht wahrnehmen kann. Genau solche Hindernisse treten aber zum Beispiel als aus den Regalen herausragende Gegenstände sehr häufig in unserem Szenario auf.

Ein weiterer Grund für den die Verwendung visueller Selbstlokalisations- und Navigationsverfahren liegt in der Tatsache, daß die Interaktion zwischen Roboter und Nutzer ebenfalls visuell stattfindet (visuelles Tracken der Position des Interaktionsparters...). Mit der Verwendung visueller Informationen soll die Interaktions- und die Navigationsebene konsistent gehalten werden.

Im hier vorgestellten Ansatz werden probabilistische Verfahren mit visuellen Merkmalen kombiniert, um eine robuste Selbstlokalisation zu gewährleisten.

[1]Gefördert durch das Thüringer Ministerium für Wissenschaft, Forschung und Kunst (B611-98041).

2 Selbstlokalisation mittels der Monte-Carlo Methode

Die Monte-Carlo Methode wurde bereits verschiedentlich zur Lokalisation von Robotern und und zum Tracken von Objekten eingesetzt. Der Unterschied zu vielen anderen Arbeiten (zum Beispiel [6] und [1]) ist die Tatsache, daß bei der Lokalisation eines autonomen Systems dieses nicht von außen beobachtet werden kann. Die Objektverfolgung erfolgt hier vielmehr mit Beobachtungen *vom Standort des zu verfolgenden Objektes aus*[3][4].

Die Monte-Carlo-Methode ist in der Lage, eine *multimodale* Verteilung im Zustandsraum abbilden und auch über die Zeit propagieren zu können.

Bei vielen Trackingaufgaben ist es so, daß Mehrdeutigkeiten bei der Abbildung der Sensorinformationen auf die Umweltrepräsentation auftreten. Diese verschiedenen Hypothesen über den Aufenthaltsort des zu trackenden Objekts bilden lokale Maxima in der Verteilung. Bei der Monte-Carlo Methode werden diese Maxima nur durch aufeinanderfolgende Beobachtungen ausgeschlossen und nicht durch systemimmanente Beschränkungen, wie zum Beispiel bei Kalman-Filtern, die man nur bei unimodalen Verteilungen benutzen kann.

Die Propagation einer Wahrscheinlichkeitsdichte über die Zeit besteht aus immer wiederkehrenden Zyklen, die sich jeweils in drei Phasen einteilen lassen. Diese Phasen beinhalten bei der konkreten Aufgabe der Verfolgung eines Objektes:

Drift: Änderung der Wahrscheinlichkeitsverteilung durch die deterministische Komponente der Objektdynamik. Bei einem mobilen autonomen System wäre das die Berechnung der hypothetischen nächsten Position abhängig von der Kinematik des Systems und der jeweiligen augenblicklichen Position.

Diffusion: Einbeziehung der nichtdeterministischen Komponente der Objektdynamik.

Measure: *Reinforcement* durch Beobachtung. Entsprechend der aktuellen Sensoreindrücke des Systems werden die Werte der Wahrscheinlichkeit aktualisiert.

In [5] wird mit *conditional density propagation* ein mathematischer Apparat vorgestellt, der es auf effiziente Art erlaubt, eine multimodale Wahrscheinlichkeitsdichte über die Zeit zu propagieren. Dieser als CONDENSATION bekannte Algorithmus wird auch in der vorliegenden Arbeit verwendet.

3 Visuell basierte Selbstlokalisation

Im vorliegenden Ansatz werden visuelle Merkmale benutzt, um die Karte zu gewinnen und den Roboter darin zu lokalisieren.

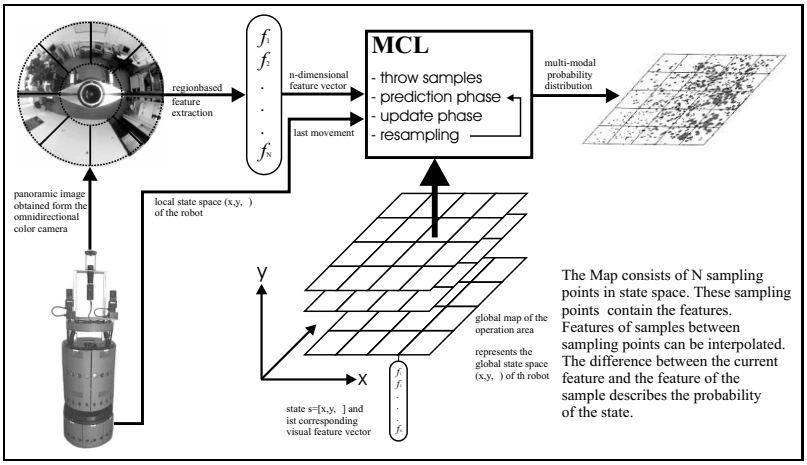


Abbildung 1: Funktionsprinzip der Monte Carlo Lokalisation mit visuellem Input

Der Ansatz einer visuellen Monte-Carlo Lokalisation (MCL) wurde unabhängig von uns zum Beispiel bereits durch Dellaert u.a. [2] untersucht. Wir gehen hier aber einen anderen Weg der Merkmalsgewinnung: Während Dellaert eine explizite, sehr detaillierte Karte der Decke der Umwelt anfertigt konzentrieren wir uns auf eine spärliche Repräsentation der Umwelt durch ein sogenanntes *View-Grid*. Dabei wird die Umwelt durch mittels einer omnidirektionalen Kamera gewonnene Panoramaansichten an den Knoten dieses Grids repräsentiert. Diese Knoten sind äquidistant angeordnet und haben einen Abstand von bis zu $0.5m$ zueinander.

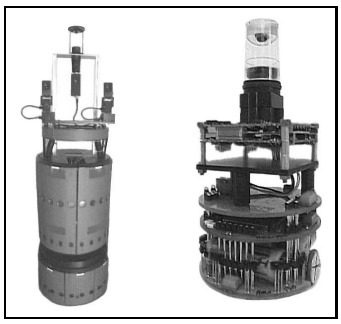
Der Ansatz beruht auf einer Repräsentation der Umwelt, die durch den Designer vorgegeben wird. Zum Aufbau des View-Grids werden an den Knotenpunkten Bilder aufgenommen und daraus Merkmalsvektoren berechnet. Diese sogenannten *Referenzvektoren* bilden dann die Wissensbasis des Systems.

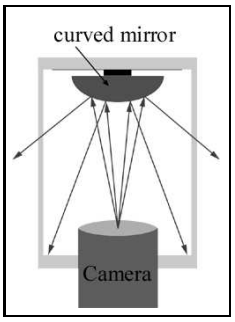
Zur Selbstlokalisation wird die Monte-Carlo Methode benutzt. Zur Erzielung einer Subrastergenauigkeit bezogen auf die Maschenweite des View-Grids werden Interpolationsverfahren im Ort angewendet, zum Vergleich der aktuellen Sensorinformation mit Referenzvektoren kommen Interpolationsverfahren im Merkmalsraum zum Einsatz.

Zur Selbstlokalisation werden anfänglich gleichverteilt Samples im Zustandsraum verteilt. Jedes Sample steht dabei für eine Position und eine Orientierung im Zustandsraum und verkörpert damit praktisch eine Hypothese über den Zustand[2] des Roboters.

Im Verlauf der Lokalisation bilden sich nun verschiedene Maxima in der Verteilung, die mehrdeutige Sensorsituationen darstellen. Nach einigen Iterationen

[2]Der Zustand des Systems in der Umwelt wird durch die kartesischen Ortskoordinaten x und y und die Orientierung oder „Blickrichtung“ φ bestimmt.





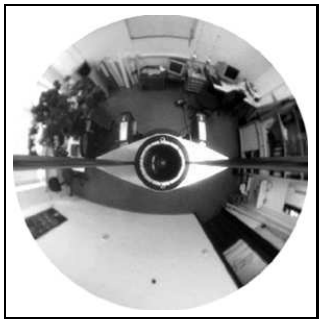


Abbildung 2: Unsere Versuchsplattformen B21 und Khepera (links), das Prinzip einer omnidirektionalen Kamera (Mitte) und eine mit einer solchen Kamera gewonnene Aufnahme (vom B21, rechts).

des Verfahrens existiert nur noch ein Maximum, welches dann die Position des Roboters darstellt.

Abb. 1 zeigt die prinzipielle Struktur des Systems. Auf den genauen Verlauf der Propagierung der Dichte wird im folgenden eingegangen:

3.1 Sensorische Erfassung der Umwelt

In Abb. 2 ist das mittels einer Omnikamera aus der Umwelt gewonnene Bild dargestellt. Die Umgebung des Roboters wird dabei durch die Spiegelform bedingt verzerrt und in einem Kreis abgebildet. Zur Weiterverarbeitung wird das Bild in N Segmente unterteilt (siehe Abb. 3). Der innere Kreis, in dem der Roboter selbst zu sehen ist, bleibt dabei unberücksichtigt. Die kreisförmige Anordnung der Segmente ist für die weitere Verarbeitung vorteilhaft. Für jedes Segment wird der Mittelwert aller Intensitäten in einem Segment bestimmt. Bei Grauwertbildern wäre das die mittlere Helligkeit. Bei Farbbildern geschieht diese Berechnung für jedes Farbband. Dieser Mittelwert stellt die *Aktivierung* des Segments dar. Damit erhält man einen Merkmalsvektor $\mathbf{f}$ der Dimension N wie folgt:

$$\mathbf{f} = (f_1, ..., f_N) \qquad Merkmalsvektor\ \mathbf{f}$$

$$f_i = \sum_{r=0}^{r_{max}} \sum_{\varphi=0}^{\varphi_{max}} B(r,\varphi)/|M_j| \quad | \ (r,\varphi) \in M_j$$

Dabei ist M_j die Menge aller Helligkeitswerte $B(r,\varphi)$ im Segment j und $|M_j|$ die Mächtigkeit dieser Menge. Bei Farbbildern mit C Bändern hat der Merkmalsvektor entsprechend die Dimension $C \cdot N$.

Solche Merkmalsvektoren bilden die *Referenzvektoren* im View-Grid. Die Merkmalsvektoren, die aus dem jeweils aktuellen Kamerabild berechnet berechnet werden, werden als *Situationsvektoren* bezeichnet.

Zum Vergleich zweier Merkmalsvektoren f_1 und f_2 wird ein Fehlermaß E eingeführt. Um die Vektoren auch bei unterschiedlichen Helligkeiten des Bildes vergleichen zu können wird E lediglich als Winkel zwischen zwei Merkmalsvektoren definiert. Dabei geht man von folgender Vereinfachung aus: Ändert sich die Umgebungshelligkeit, so geschieht dies gleichmäßig in *jedem* Segment. Das bedeutet, daß sich nur die Länge, nicht aber die Richtung der Vektoren, ändert und somit der Winkel zwischen ihnen gleich bleibt. Damit ergibt sich als Fehlermaß:

$$E = \arccos\left(\frac{\mathbf{f}_1^T \cdot \mathbf{f}_2}{|\mathbf{f}_1| \cdot |\mathbf{f}_2|}\right)$$

Diese Annahme funktioniert hinreichend genau, wie empirsche Untersuchungen in verschiedenen Szenarien zeigten. Durch die im Monte-Carlo Vefahren enthaltene Probabilistik sind die dennoch entstehenden Fehler gut beherrschbar.

Weitere Störungen können durch bewegliche Hindernisse (z.B Personen) hervorgerufen werden. Durch die verzerrte Abbildung werden diese Objekte im Bild jedoch relativ klein dargestellt. Die Störung in einem Segment und der daraus resultierende Fehler ist vergleichsweise gering (siehe Abb. 4). Hier wird die Winkelabweichung zwischen den Merkmalsvektoren zweier Aufnahmen in identischer Position mit teilweiser Abdeckung des Sichtfeldes dargestellt. Objekte, die mehr als 10% des Sichtfeldes verdecken, müßten so groß sein, daß ein Auftreten in realen Umwelten und speziell in unserem Szenario sehr unwahrscheinlich erscheint.

3.2 Interpolation zur Gewinnung der Merkmalsvektoren

Zur Erzeugung der Umweltrepräsentation werden an den Knoten die Zustände $\mathbf{z} = (x, y, \varphi)$ und die dazu gehörenden Referenzvektoren $\mathbf{r}(x, y, \varphi)$ gespeichert. Bei einem solchen dreidimensionalen Zustandsraum wird die Karte auch bei spärlicher Repräsentation der Umwelt vergleichsweise umfangreich. Aufgrund der speziellen Eigenschaften einer in der Rotationsachse montierten Kamera kann man die Größe der Karte reduzieren, indem der Winkel nicht explizit in die Karte aufgenommen wird. Damit wird die Karte nun zu einer zweidimensionalen Repräsentation. Es werden nur die Zustände $\mathbf{z} = (x, y, 0°)$ und die Referenzvektoren $\mathbf{f}(x, y, 0°)$ gespeichert. Die Merkmale für alle anderen Orientierungen sind leicht zu berechnen. Man muß dazu nur die Elemente des jeweiligen Referenzvektors entsprechend rotieren. Angenommen, ein Grauwertbild hätte 100 Segmente (Winkelauflösung 3.6°) und man möchte den Referenzvektor $\mathbf{r}(x_0, y_0, 40°)$ berechnen. Bei einem Referenzvektor

$$\mathbf{r}(x_0, y_0, 0°) = (m_0, m_1, m_2, \ldots, m_{97}, m_{98}, m_{99})$$

ergibt sich dann

$$\mathbf{r}(x_0, y_0, 40°) = (m_{11}, m_{12}, m_{13}, \ldots, m_8, m_9, m_{10}).$$

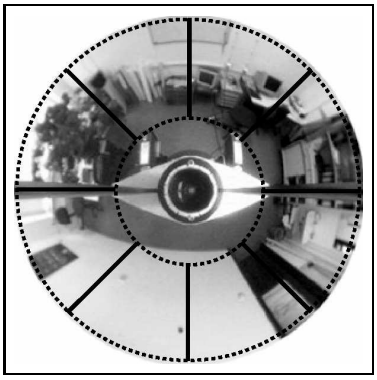


Abbildung 3: Segmentiertes Kamerabild, die Mitte ist ausgespart

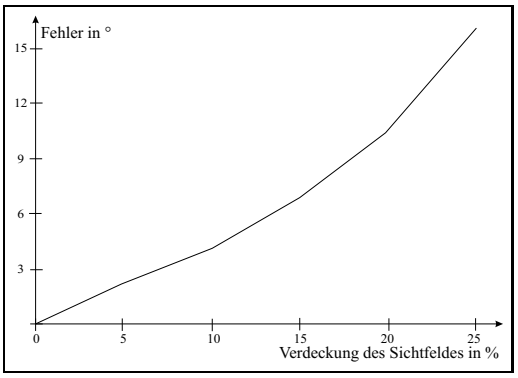


Abbildung 4: Der resultierende Fehler bei Verdeckung durch Hindernisse

Da nur diskrete Winkel direkt darstellbar sind, muß man für andere Werte interpolieren. Die bei der Rotation und der Interpolation entstehenden Fehler fallen im Vergleich zum Einfluß der Helligkeit gering aus ($Fehler < 3.0°$ für $N > 10$). Je größer die Anzahl der Segmente N ist, desto geringer wird der Interpolationsfehler.

Wenn man den sensorischen Input für eine beliebige Position $\mathbf{p} = (x, y)$ aus den sensorischen Inputs nahe benachbarter Positionen berechnen könnte, wäre es möglich, die Größe der Karte weiter zu verringern. Die grundlegende Idee dazu ist die Benutzung einer rasterförmigen Unterteilung der Umwelt. An allen Gitterpunkten des Rasters werden die Merkmalsvektoren bestimmt und dienen nun als Grid von Referenzvektoren (siehe Abb. 7).

Da sich das Kamerabild bei einer Bewegung in einer statischen Umwelt nicht sprunghaft ändert, kann man davon ausgehen, daß sich die *Merkmale* zwischen zwei Punkten linear interpolieren lassen. Diese These wurde durch experimentelle Untersuchungen bestätigt.

In Abb. 5 sieht man, daß sich die Aktivierung in den einzelnen Segmenten tatsächlich stetig und monoton ändert. Mit steigender Entfernung steigt die Wahrscheinlichkeit für einen nichtmonotonen Verlauf allerdings, was die Rasterweite des Gitters eingrenzt. Damit können Merkmalsvektoren für beliebige Punkte $\mathbf{p}$ durch eine lineare Interpolation zwischen den drei $\mathbf{p}$ am nächsten liegenden Referenzvektoren berechnet werden.

Dazu wird aus den drei nächsten Referenzvektoren $\mathbf{f}_p(x, y, 0°)$ linear interpoliert und anschließend entsprechend der Orientierung φ wie oben beschrieben rotiert. Dadurch erhält man den geschätzten Referenzvektor $\tilde{\mathbf{r}}_p(x, y, \varphi)$. Dieser Vektor entspricht den Merkmalen für den sensorischen Input im Zustand $\mathbf{z}_i = (x, y, \varphi)$.

3.3 Ablauf der visuell-basierten MCL

Die von uns benutzte visuelle MCL mit CONDENSATION ist in Abb. 1 schematisch dargestellt. In einer Menge von M Samples stellt jedes Sample S_k einen Zustand $\mathbf{z}_k = (x, y, \varphi)$ des Roboters dar. Jedes Sample beinhaltet außerdem die Wahrscheinlichkeit dafür, daß der Roboter sich tatsächlich in diesem Zustand befindet. Die Gesamtheit aller Samples repräsentiert die Wahrscheinlichkeitsverteilung über dem Zustandsraum. Der Algorithmus der MCL läuft wie folgt ab:

Initialization Der Roboter steht an einer für ihn unbekannten Position in der Karte. Die M Samples werden zufällig und gleichverteilt im Zustandsraum ausgestreut. Das bedeutet, daß jedem Sample S_k ein Zustand $\mathbf{z}_k = [x, y, \varphi]$ zugewiesen wird. Die Samples dürfen dabei nicht außerhalb des View-Grid landen, da keine Extrapolation der Merkmale außerhalb der Karte möglich ist.

Für jedes Sample S_k wird der interpolierte Referenzvektor $\tilde{\mathbf{r}}_k(x, y, \varphi)$ interpoliert und die Wahrscheinlichkeit dafür bestimmt, daß sich der Roboter an der durch Sample k repräsentierten Position befindet. Dazu wird das Fehlermaß E aus dem interpolierten Referenzvektor $\tilde{\mathbf{r}}_k(x, y, \varphi)$ und den aus dem aktuellem Sensorinput ermittelten Merkmalen $\mathbf{f}_{input}$ wie folgt bestimmt:

$$E_k = \arccos\left(\frac{\tilde{\mathbf{r}}_k^T(x, y, \varphi) \cdot \mathbf{f}_{input}}{|\tilde{\mathbf{r}}_k(x, y, \varphi)| \cdot |\mathbf{f}_{input}|}\right)$$

Die Wahrscheinlichkeit, daß sich das System tatsächlich an der durch das Sample definierten Stelle befindet wird durch

$$p_k^* = 1.0 - \alpha \cdot E_k$$

angegeben. Der Faktor α normiert den Fehler auf 1. Die Summe der Wahrscheinlichkeiten aller Samples muß 1 ergeben:

$$p_i = \frac{p_i^*}{\sum_{l=0}^{M} p_l^*}$$

Motion Der Roboter führt einen Bewegungsschritt $\Delta\mathbf{z} = (\Delta x, \Delta y, \Delta\varphi)$ aus. Diese Bewegung wird auf die Samples entsprechend ihrer Position im dreidimensionalen Zustandsraum übertragen. Daher erfährt jedes Sample eine Ortsveränderung entsprechend seiner Position und Orientierung. Jedes Sample S_k erhält dadurch einen neuen Zustand $\mathbf{z}_k = (x, y, \varphi)$ (siehe Abb 6). Die Wahrscheinlichkeit für Samples, die sich nach dieser Bewegung außerhalb der Karte befinden, muß auf $p_k = 0$ gesetzt werden. Nun müssen die interpolierten Referenzvektoren $\tilde{\mathbf{r}}_k(x, y, \varphi)$ für die neuen Zustände berechnet werden.

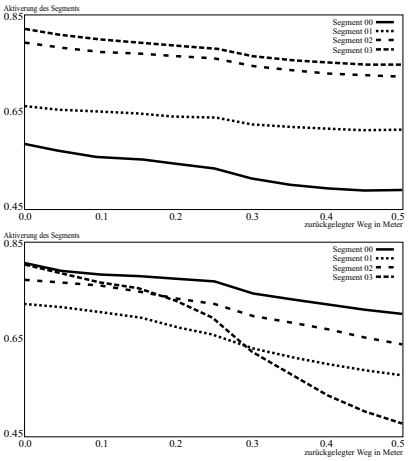


Abbildung 5: Die stetige Änderung der Aktivierung einzelner Segmente bei einer Translationsbewegung des Roboters.

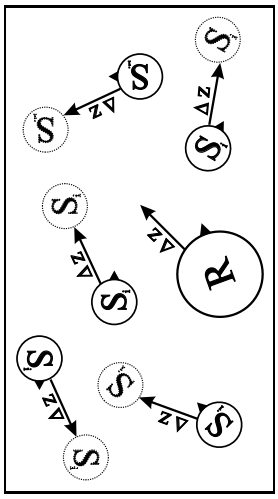


Abbildung 6: Die spezifische Bewegung ausgewählter Samples um $\Delta \mathbf{z}$

Die Verrechnung mit den aus dem aktuellen sensorischen Input errechneten Merkmalen ergibt die neuen Wahrscheinlichkeiten für die Samples.

Es findet ebenfalls eine Normierung der Gesamtwahrscheinlichkeit aller Samples auf 1 statt. Die neuen Wahrscheinlichkeiten werden mit den vorhergehenden multipliziert und das ergibt die eigentliche neue Wahrscheinlichkeitsverteilung über dem Zustandsraum. Anschließend wird die Gesamtwahrscheinlichkeit nochmals auf 1 normiert.

Resampling Hierbei werden die Samples mit geringer Wahrscheinlichkeit entfernt. Die dadurch freigewordenen Samples werden in der Umgebung von Samples mit hoher Wahrscheinlichkeit eingefügt. Dadurch wird gewährleistet, daß dort, wo sich der Roboter am wahrscheinlichsten befindet, mit einer höheren Genauigkeit gearbeitet werden kann.

Das Verfahren wird, beginnend bei Schritt 2, zyklisch wiederholt. Nach einigen Schritten werden alle Mehrdeutigkeiten aufgelöst und die Samples konzentrieren sich an der tatsächlichen Position des Roboters.

4 Experimente

Das vorgestellte Verfahren wurde zunächst auf der Experimentalplattform Khepera getestet. Die Testumgebung für den Khepera war ein künstliches Labyrinth,

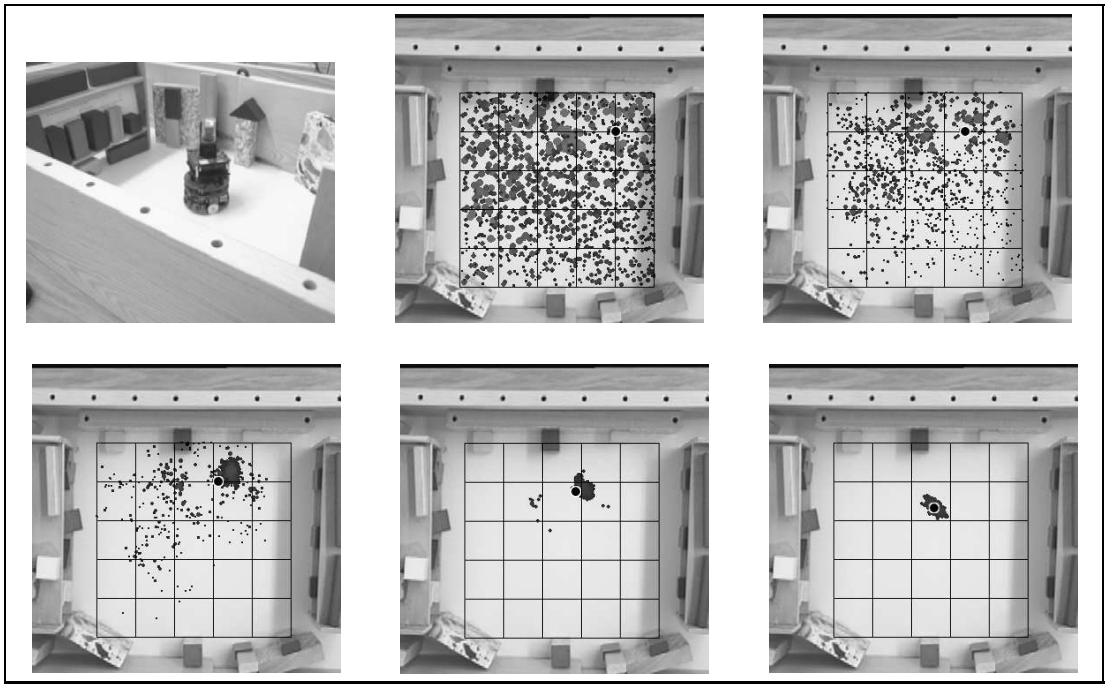


Abbildung 7: o.l.: Khepera in einer Testumgebung. o.m.: Testumgebung von oben gesehen, das Gitter repräsentiert die erstellt Karte, im Initialzustand sind alle Samples gleichmäßig in der Karte verteilt (große Kreise = hohe Wahrscheinlichkeit). o.r: Die erste Bewegung und das Resampling sind erfolgt, die Samples häufen sich an Stellen mit hoher Wahrscheinlichkeit. u.l.: Die Verteilung der Samples nach einer weiteren Bewegung des Roboters. u.m.: Eine bestimmte Vermutung festigt sich. u.r.: Alle Samples sind an einer Stelle im Zustandsraum zusammengelaufen - das Kreuz markiert die tatsächliche Position des Roboters

in dem zur nötigen Unterscheidbarkeit verschiedene Gegenstände verteilt waren. Die Rasterweite des View-Grid betrug ungefähr einen Roboterdurchmesser ($4cm$, siehe Abb. 7). Die erreichten Ergebnisse bestätigten eindrucksvoll die Fähigkeiten der visuellen MCL. Nach nur wenigen (< 6) Iterationen hatte sich der Roboter lokalisiert. Die Genauigkeit der Positionsschätzung lag dabei deutlich unter der Rasterweite. Der Verlauf eines Experiments ist in Abb. 7 dargestellt.

5 Ausblick

Als nächstes ist der Aufbau einer Karte für den Roboter B21 in einer realen Baumarktumgebung geplant.

Weiterhin soll das Modell dahingehend erweitert werden, sich nach einer, zum Beispiel durch teilweise Verdeckung des Sichtfeldes hervorgerufenen, Fehllokalisa-

tion wieder relokalisieren zu können. Dazu müssen einige der wegen zu geringer Wahrscheinlichkeit entfernten Samples erneut gleichverteilt im Zustandsraum ausgestreut werden. Damit wird der eigentliche Vortei des Monte-Carlo Verfahrens ausgenutzt, beliebig viele Maxima der Verteilung gleichzeitig verfolgen zu können.

Außerdem stehen Untersuchungen zur maximalen Rasterweite sowohl für Khepera wie B21 an.

Des weiteren müssen die tatsächlichen Auswirkungen von Helligkeitsschwankungen und allgemein den Beleuchtungsbedingungen auf die Rasterweite und damit die Robustheit des Verfahrens untersucht werden.

Literatur

[1] Andrew Blake, Benedict Bascle, Michael Acheson Isard, and John MacCormick. Statistical models of visual shape and motion. *Proc. Roy. Soc. Lond. A*, 356:1283–1302, 1998.

[2] Frank Dellaert, Wolfram Burgard, Dieter Fox, and Sebastian Thrun. Using the condensation algorithm for robust, vision-based mobile robot localization. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'99)*, June 1999.

[3] Frank Dellaert, Dieter Fox, Wolfram Burgard, and Sebastian Thrun. Monte carlo localization for mobile robots. In *Proceedings of ICRA-99*, 1999.

[4] Dieter Fox, Wolfram Burgard, Frank Dellaert, and Sebastian Thrun. Monte carlo localization: Efficient position estimation for mobile robots. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-99)*, 1999.

[5] Michael Acheson Isard. *Visual Motion Analysis by Probabilistic Propagation of Density.* PhD thesis, University of Oxford, September 1998.

[6] Michael Acheson Isard and Andrew Blake. Condensation – conditional density propagation for visual tracking. *Int. J. Computer Vision*, 29:5–28, 1998.