Verbesserte Effizienz der Monte-Carlo-Lokalisierung im RoboCup

Patrick Heinemann, Jürgen Haase, Andreas Zell

Wilhelm-Schickard-Institut der Universität Tübingen,
Lehrstuhl Rechnerarchitektur,
Sand 1, 72074 Tübingen
{heinemann, jhaase, zell}@informatik.uni-tuebingen.de

Zusammenfassung. Aktuelle Implementierungen der Monte-Carlo-Lokalisierung benötigen mindestens 100 Samples, was in zeitkritischen Roboter-Systemen, wie z.B. einem RoboCup-Roboter, zu einem Ressourcen-Engpass führen kann. Dieser Artikel beschreibt einen neuen Ansatz für Monte-Carlo-Lokalisierung, bei dem die Anzahl der benötigten Samples adaptiv bis auf ein Minimum von nur einem Sample sinkt, wenn die Positionsschätzung ausreichend exakt ist. Experimente zeigen, dass der vorgestellte Algorithmus sehr schnell in diesen effizienten 'Tracking-Modus' übergeht. Durch eine iterative Verbesserung der Positionsschätzung kann sogar eine höhere Genauigkeit der Lokalisierung erreicht werden, als dies mit bisherigen Ansätzen möglich ist.

1 Einleitung

Die Lokalisierung in einem globalen Koordinatensystem ist eine wichtige Aufgabe mobiler Roboter, besonders wenn sie ihr Ziel nur durch Kooperation oder in Konkurrenz zu anderen Robotern erreichen können. Im RoboCup [7]wurden daher in den letzten Jahren viele Ansätze zur Positionsschätzung entwickelt ([4], [6], [8] und [1]). Die aktuelleren Ansätze basieren alle auf der Monte-Carlo-Lokalisierung ([2], [12], [10], [11] und [9]). Diese Ansätze unterscheiden sich hauptsächlich in der Anzahl der benötigten Samples und damit in der Effizienz der Lokalisierung. Der in diesem Artikel beschriebene Ansatz implementiert eine verbesserte Monte-Carlo-Lokalisierung anhand der Spielfeldlinien eines RoboCup-Spielfelds. Dabei wird die Samplezahl abhängig von der Güte der Positionsschätzung adaptiert. Durch eine anschließende iterative Verbesserung dieser Schätzung ist es möglich, den Roboter auch noch mit einem einzigen Sample zu lokalisieren.

2 Verbesserte Monte-Carlo-Lokalisierung

Monte-Carlo-Lokalisierung (MCL) [3] ist eine effiziente, auf Sampling basierende Approximation der kontinuierlichen Wahrscheinlichkeitsverteilung über alle möglichen Positionen eines Roboters, wobei sich die Verteilung der Samples nach

der Wahrscheinlichkeit richtet, dass sich der Roboter tatsächlich an der betrachteten Position befindet (Sampling/Importance Resampling).

Der vorgestellte Algorithmus verfügt zu Beginn über kein Vorwissen, obwohl dies möglich wäre, beispielsweise wenn der Roboter immer an der selben Position des Feldes starten würde. Daher wird zunächst die maximale Anzahl von Samples $N_{\max}$ generiert und gleichmäßig über den Zustandsraum, hier das befahrbare Feld und alle Ausrichtungen, verteilt.

$$S = \{s_i | s_i = ((x_i, y_i, \theta_i), w_i)\} \tag{1}$$

repräsentiert die aktuelle Menge der Samples s_i, wobei jedes s_i eine Positionsschätzung $(x_i, y_i, \theta_i) = l_i$ und ein Gewicht w_i beinhaltet, das die Wahrscheinlichkeit ausdrückt, dass sich der Roboter an Position l_i befindet.

In jedem Zyklus bekommt der Algorithmus neue Odometrie-Daten a und ein Bild s der omnidirektionalen Kamera des RoboCup-Roboters. Die Odometrie-Information wird im *Bewegungsmodell* verarbeitet. Dabei werden zunächst alle Samples von S nach der gemessenen Bewegung a verschoben und rotiert. Zusätzlich werden die Samples noch durch ein Gaußsches Rauschen proportional zur Messungenaugkeit der Odometrie verrauscht.

Die Daten der Kamera werden im sogenannten *Sensormodell* verarbeitet. Dieses gibt die bedingte Wahrscheinlichkeit $P(s|l_i)$ an, dass der Roboter das Bild s erhält unter der Voraussetzung, dass er sich an der Position l_i befindet. Zur effizienten Berechnung von $P(s|l_i)$ werden zunächst weiße Pixel, die die Feldlinien des Spielfelds darstellen, aus dem Kamerabild extrahiert und daraus durch inverse perspektivische Transformation die Position der Linienpunkte im Roboter-Koordinatensystem errechnet ([5]). Diese Linienpunkte werden dann an die Positionsschätzung l_i der Samples transformiert . $P(s|l_i)$ kann, wie von Röfer *et al.* [10,11] und Hundelshausen *et al.* [12] vorgeschlagen, als Summe der quadratischen Distanzen der Linienpunkte zu ihrer nächstgelegenen Linie im Modell der Feldlinien berechnet werden. Da diese Distanzen nur von der Position auf dem Feld abhängig sind, kann man sie vorherberechnen und für schnellen Zugriff in einer zweidimensionalen Tabelle ablegen. Um die Symmetrie, die sich in der Positionsschätzung durch die Linien eines RoboCup-Spielfelds ergibt, aufzulösen, benutzt der Algorithmus zusätzlich noch die Winkel zu den beiden unterschiedlich farbig markierten Toren. Diese Winkel werden mit den erwarteten Torwinkeln an der Position l_i verglichen und ergeben ein weiteres Distanzmaß. Die Gesamtdistanz wird schließlich durch eine Linearkombination der beiden Distanzen errechnet und mit dieser die Gewichte der Samples aktualisiert.

Jetzt kann aus den besten n Samples ein gewichtetes Mittel als Positionsschätzung generiert werden

$$\widehat{p} = (x, y, \theta) = \sum_n w_n l_n. \tag{2}$$

Diese Positionsschätzung wird nun verbessert, indem iterativ eine Kraft F_k und ein Drehmoment M_k, die das Linienmodell auf die Linienpunkte ausübt, berechnet und auf die Schätzung angewendet werden. Die Kraft ergibt sich aus

der Summe der Abstandsvektoren zwischen gemessenen Linienpunkten und der jeweils nächstgelegenen Modelllinie. Das Drehmoment errechnet sich aus dem Kreuzprodukt der Abstandsvektoren mit den Vektoren vom Roboter zum jeweiligen Linienpunkt. Fügt man nun einen Bruchteil der Kraft F_k und des Drehmoments M_k zur Positionsschätzung $\widehat{p}$ hinzu, erhält man in jeder Iteration k eine neue Schätzung

$$(x_k, y_k) = (x_{k-1}, y_{k-1}) + \mu F_{k-1} \tag{3}$$

$$\theta_k = \theta_{k-1} + \nu M_{k-1}. \tag{4}$$

Die Iterationen werden fortgesetzt, bis die Verbesserung einen Schwellwert unterschritten hat oder eine maximale Anzahl Iterationen ausgeführt wurde. Diese Idee zur iterativen Positionsverbesserung wurde bereits von Hundelshausen *et al.* [12] für ein Dead-Reckoning Verfahren zur Lokalisierung benutzt.

Ausgehend von der Gesamtdistanz D der endgültigen Positionsschätzung wird nun noch die Anzahl der Samples für den nächsten Zyklus berechnet.

$$N_{t+1} = \begin{cases} N_{\max} & : \text{ if } \xi D \geq N_{\max} \\ \xi D & : \text{ if } 1 < \xi D < N_{\max} \\ 1 & : \text{ if } \xi D \leq 1 \end{cases}, \tag{5}$$

wobei ξ regelt, wie schnell N_{t+1} auf ein Sample reduziert wird. Zum Abschluss eines Zyklus werden die Samples abhängig von ihrem Gewicht neu gesampled. Hierbei werden schlechte Samples aussterben und gute mehrfach neu erzeugt.

3 Ergebnisse

Die Effizienz des vorgestellten MCL-Algorithmus und die Qualität der errechneten Positionsschätzung wurde in zwei Experimenten untersucht. Die dabei verwendeten Parameter können Tabelle 1 entnommen werden.

n	$N_{\max}$	ξ	μ	ν
N	200	2500	0.001	0.0003

Tabelle 1. Die für die Experimente verwendeten Parameter

Für die Berechnung des gewichteten Mittels aus den besten n Samples wurden alle N Samples herangezogen, um ein aufwändiges Sortieren der Samples zu vermeiden. Die maximale Anzahl von Samples N_{max} wurde so gewählt, dass bei einer schlechten Positionsschätzung gerade so viele Samples generiert werden, wie in einem Standard MCL-Verfahren. Die Wahl von ξ bilded einen guten Kompromiss zwischen einer Explosion der Sample-Zahl bei schlecher Schätzung und einer schnellen Relokalisierung beim 'Kidnapped-Robot'-Problem. Die Regelparameter μ und ν wurden empirisch so ermittelt, dass durch die Iterationen

kein Schwingen der Schätzung um das Minimum auftritt, aber gleichzeitig eine möglichst schnelle Konvergenz erzielt werden kann. Im ersten Experiment wurde der Algorithmus mit Bildern eines stehenden Roboters getestet. Die Bilder 1 bis 98 wurden an Position $p_1 = (1.04, 1.07, 2.1)$ aufgenommen, während die Bilder 99 bis 196 an einer anderen Position $p_2 = (1.64, 2.68, 0.0)$ aufgenommen wurden, um das 'Kidnapped-Robot'-Problem zu simulieren. Dieses Experiment wurde sowohl mit dem hier vorgestellten verbesserten MCL-Verfahren, als auch mit Standard MCL-Verfahren ([3]) mit fester Sample-Zahl von $N = 50$, $N = 100$ und $N = 200$ Samples durchgeführt, um die Laufzeit und die Qualität der Schätzung mit bestehenden Verfahren zu vergleichen. Abbildung 1 zeigt die

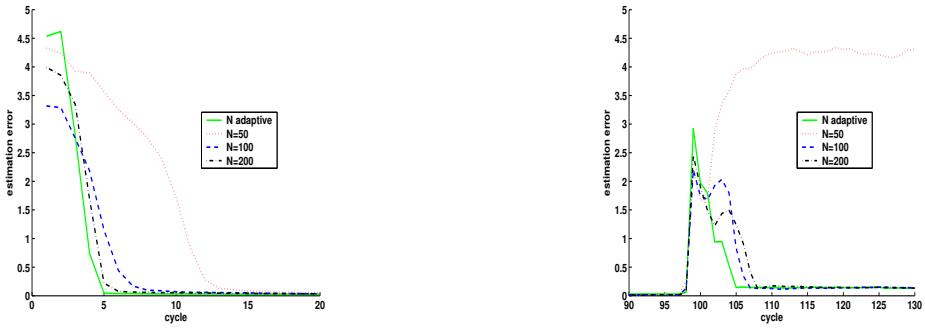

Abb. 1. Comparison of our algorithm to MCL with fixed numbers of samples.

Euklidische Distanz der Positionsschätzungen zur tatsächlichen Position für alle Verfahren. Unabhängig von der verwendeten Sample-Zahl haben alle Verfahren nach spätestens 20 Zyklen die Position auf etwa 10cm genau geschätzt. Der vorgestellte Algorithmus reduziert seine Sample-Zahl von $N = N_{max} = 200$ auf $N = 1$ in nur 6 Zyklen und erreicht dabei schneller als alle anderen Verfahren einen Positionsfehler von unter 10cm. Bis zur Änderung der Position des Roboters in Zyklus 99 bleibt der Algorithmus bei nur einem Sample und generiert dann sofort wieder $N = N_{max}$ Samples, um auf den großen Schätzfehler zu reagieren. Abgesehen von dem Standard-Verfahren mit 50 Samples, das nach dem Positionssprung nicht mehr auf die neue Position konvergiert, sind alle Algorithmen in der Lage, das 'Kidnapped-Robot'-Problem zu lösen. Der gemittelte Positionsfehler über alle Zyklen liegt bei diesen Verfahren bei etwa 20cm, wobei der vorgestellte Algorithmus sogar noch unter diesem Wert bleibt. Zusätzlich reduziert der vorgestellte Algorithmus allerdings in 92.87% aller Zyklen die Samplezahl auf ein einziges Sample, wodurch ein erheblicher Geschwindigkeitsvorteil in der Berechnung der Positionsschätzung erreicht wird. Tabelle 2 fasst diese Ergebnisse zusammen.

Um zu zeigen, dass das vorgestellte Verfahren auch bei einem fahrenden Roboter gute Ergebnisse liefert und dennoch in Echtzeit arbeitet, wurde mit einem Testroboter per Fernsteuerung ein bestimmtes Rechteck abgefahren.

	Verbesserte MCL	Standard MCL	Standard MCL	Standard MCL
	$N = adaptiv$	$N = 50$	$N = 100$	$N = 200$
Mittlere Rechenzeit	1.7632ms	3.6426ms	6.8223ms	14.2508ms
Mittlerer Fehler	0.1936m	2.2515m	0.2075m	0.2057m

Tabelle 2. Ergebnisse von 196 Zyklen mit verschiedener Samplezahl N.

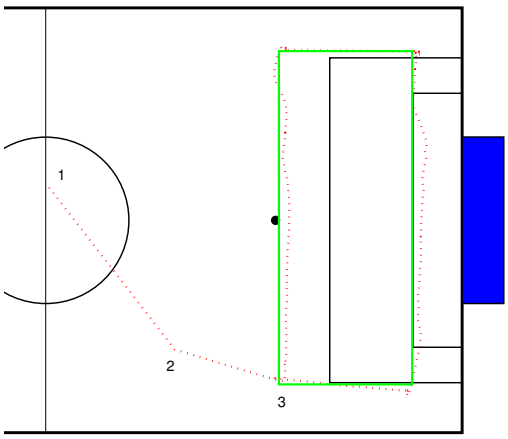

Abb. 2. Die durch den vorgestellten Algorithmus in Echtzeit generierte Positionsschätzung des entlang des schwarzen Rechtecks ferngesteuerten Roboters. Der Roboter steht zu Beginn an der mit einem schwarzen Kreis markierten Position.

Abbildung 2 zeigt die Ergebnisse dieses Experiments. Im ersten MCL-Zyklus sind die Samples noch gleichmäßig über das Feld verteilt, so dass sich durch das gewichtete Mittel die Positionsschätzung (1) in der Mitte des Feldes ergibt. Bereits ab dem dritten Zyklus befindet sich allerdings die Schätzung schon auf der korrekten Startposition (3) im Rechteck. Über die gesamte Restdauer von 502 Zyklen des Experiments wird die Position des Roboters korrekt geschätzt, wobei in 96.21% aller Zyklen nur ein einziges Sample berechnet werden muss. Damit ergab sich eine mittlere Berechnungszeit für einen Zyklus von 0.5413ms auf einem Athlon XP 1800+ System. Leider lässt sich nicht feststellen, ob die gelegentlichen Abweichungen vom perfekten Rechteck durch Ungenauigkeit in der Schätzung oder durch ungenaues Steuern des Roboters hervorgerufen wurden, da ein noch präziseres unabhängiges Lokalisierungssystem nicht zur Verfügung stand.

4 Zusammenfassung

In diesem Artikel wurde eine effiziente Verbesserung der Monte-Carlo Lokalisation vorgestellt. Dieser Algorithmus nutzt einen schnellen Tabellenzugriff um die Fitness der Samples zu bestimmen und eine iterative Verbesserung der Positionsschätzung. Damit kann die Zahl der benötigten Samples bis auf ein einzi-

ges Sample reduziert werden. In Experimenten konnte gezeigt werden, dass der Algorithmus dank der variablen Sample-Zahl in der Lage ist, das 'Kidnapped-Robot'-Problem zu lösen und die Position eines fahrenden Roboters mit hoher Genauigkeit zu verfolgen. Die dazu benötigte Rechenzeit lässt ausreichend viel Platz für weitere wichtige Berechnungen des Roboters, wie z.B. Objekterkennung und -verfolgung und Pfadplanung. Als weitere Untersuchung der Genauigkeit des Algorithmus sind Experimente mit einem Laser-Scanner geplant, der die exakte Position des Roboters zum quantitativen Vergleich der Ergebnisse des Verfahrens bestimmt.

Literaturverzeichnis

1. F. de Jong, J. Caarls, R. Bartelds, and P. Jonker. A Two-Tiered Approach to Self-Localization. In *RoboCup 2001: Robot Soccer World Cup V*, volume 2377 of *Lecture Notes in Computer Science*, pages 405–410. Springer Verlag, 2002.
2. S. Enderle, M. Ritter, D. Fox, S. Sablatnög, G. Kraetzschmar, and G. Palm. Vision-based Localization in RoboCup Environments. In *RoboCup 2000: Robot Soccer World Cup IV*, volume 2019 of *Lecture Notes in Computer Science*, pages 291–296. Springer Verlag, 2001.
3. D. Fox, W. Burgard, F. Dellaert, and S. Thrun. Monte Carlo Localization: Efficient Position Estimation for Mobile Robots. In *Proceedings of the National Conference on Artificial Intelligence*, pages 343–349, 1999.
4. S. Gutmann, T. Weigel, and B. Nebel. Fast, Accurate, and Robust Self-Localization in Polygonal Environments. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '99)*, 1999.
5. P. Heinemann, T. Rückstieß, and A. Zell. Fast and Accurate Environment Modelling using Omnidirectional Vision. In *Dynamic Perception*, pages 9–14. Infix Verlag, 2004.
6. L. Iocchi and D. Nardi. Self-Localization in the RoboCup Environment. In *RoboCup-99: Robot Soccer World Cup III*, volume 1856 of *Lecture Notes in Computer Science*, pages 318–330. Springer Verlag, 2000.
7. H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, and E. Osawa. RoboCup: The Robot World Cup Initiative. In *AGENTS '97: Proceedings of the first international conference on Autonomous agents*, pages 340–347. ACM Press, 1997.
8. C. Marques and P. Lima. A Localization Method for a Soccer Robot Using a Vision-Based Omni-Directional Sensor. In *Proceedings of EuRoboCup Workshop 2000*, 2000.
9. E. Menegatti, A. Pretto, and E. Pagello. A New Omnidirectional Vision Sensor for Monte-Carlo Localization. In *RoboCup 2004: Robot Soccer World Cup VIII*. Springer Verlag, 2005.
10. T. Röfer and M. Jüngel. Vision-Based Fast and Reactive Monte-Carlo Localization. In *Proceedings of the 2003 IEEE International Conference on Robotics & Automation*, pages 856–861, 2003.
11. T. Röfer and M. Jüngel. Fast and Robust Edge-Based Localization in the Sony Four-Legged Robot League. In *RoboCup-2003: Robot Soccer World Cup VII*, Lecture Notes in Computer Science. Springer Verlag, 2004.
12. F. von Hundelshausen, M. Schreiber, F. Wiesel, A. Liers, and R. Rojas. MATRIX: A force field pattern matching method for mobile robots. Technical Report B-08-03, Free University of Berlin, 2003.