Vergleich der Pfadverfolgung mit Odometrie und AMCL

Kai Hofmann und Barbara Fischbach
Robotik und Telematik
Universität Würzburg
Am Hubland, D-97074 Würzburg
barbara.fischbach@uni-wuerzburg.de
kai.hofmann@uni-wuerzburg.de

Diese Arbeit ist Bestandteil des Praktikums zur Mess- und Regelungstechnik

Abstract

Die autononome Fortbewegung von Fahrzeugen spielt heutzutage immer eine größere Rolle. Dazu werden verschiedene Algorithmen, zur Lokalisierung, Kartierung, und Pfadverfolgung genutzt. Diese wurden auf einem realen Roboter implementiert und getestet. Nicht nur im Weltall, wo wir unbedingt darauf angewiesen sind, dass Systeme autonom funktionieren, sondern auch auf der Erde, um Systeme sicherer und bequemer für den Benutzer zu machen.

1 Einleitung

Um das beraus komplexe Thema verständlicher zu machen und Algorithmen vorstellbar zu erklären, wird im folgenden ein Fraunhofer-Roboter benutzt, der anhand Odometrie,... sich selbständig in einem bekannten Raum zurechtfinden kann.

2 ROS

Für die Implementierung und Tests der Algorithmen wird das *Robot Operating System*, kurz ROS genutzt. Es ist kein Betriebssystem im eigentlichen Sinn, sondern ein Framework. Es ermöglicht Hardware-Abstraktion, Paket-Management und stellt eine Middleware bereit mit der verschiedene Prozesse kommunizieren können. [2] Die ROS-Software ist in sogenannten Nodes organisiert, die über ROS miteienader kommunizieren können. So können Funktionalitäten wie Plannung, Pfadverfolgung, Sensorik und so weiter getrennt werden. Außerdem können so einfach Nodes von anderen Leuten genutzt werden. Dies ermöglicht uns in kurzer Zeit eine Plattform zum Testen der Algorithmen aufzubauen, und ohne groen Aufwand Algorithmen durch Nodes zu implementieren.

3 Lokalisation

3.1 Odometrie

Die Odometrie beruht auf einer relativen Positionsbestimmung, dabei wird aus der vorher bekannten Position und der zurückgelegten Weg-strecke die neue Position berechnet. Auf kurzen Distanzen liefert die Odometrie sehr genaue Ergebnisse. Mit wachsender Entfernung nehmen auch Fehler durch unterschiedliche Drücke in den Reifen oder Reibung zu. Weitere Fehlerquelle sind höheren Geschwindigkeiten und engeren Kurven, dort neigen die Räder zum Durchdrehen und Wegrutschen. Um das zu zeigen wird der Roboter ähnliche Pfade in langsamer und schneller Geschwindigkeit abfahren. (Verkrppeltes Bild einfgen)

3.2 Adaptive Monte Carlo Localisation [1]

Im folgenden als AMCL abgekürzt, ist ein Algorithmus der mit Hilfe eines Partikelfilters die Position eines Roboters bestimmt. Dazu wird eine Karte der Umgebung benötigt. Der Roboter stellt Hypothesen über seine Pose auf der Karte an. Die anfängliche Verteilung dieser Hypothesen auf der Karte kann verschieden sein. Bei uns ist sie Gauß-verteilt um gegebene anfngliche Pose.

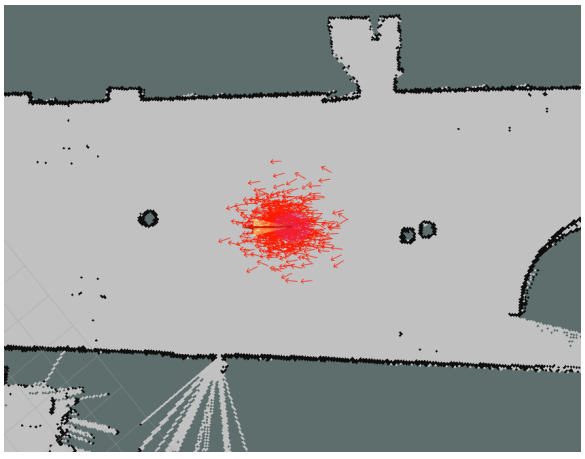


Figure 1: Partikelverteilung in Anfangspose

Die Hypothesen kann man sich als virtuelle Roboter auf der Karte vorstellen. Fährt der

reale Roboter, so fahren auch die virtuellen Roboter, mit den gleichen Steuerungsbefehlen. Die realen Sensorwerte werden mit denen der virtuellen Robotern verglichen. Die virtuellen Roboter gewinnen ihre Sensormesswerte durch die Karte.
Je unstimmiger die Daten des virtuellen Roboters sind, desto unwahrscheinlicher ist die Hypothese dass der reale sich dort befindet.
Unwahrscheinlichere Hypothesen werden gelöscht und neue im Bereich der wahrscheinlicheren aufgestellt, bzw. dort neue virtuelle Roboter erzeugt.

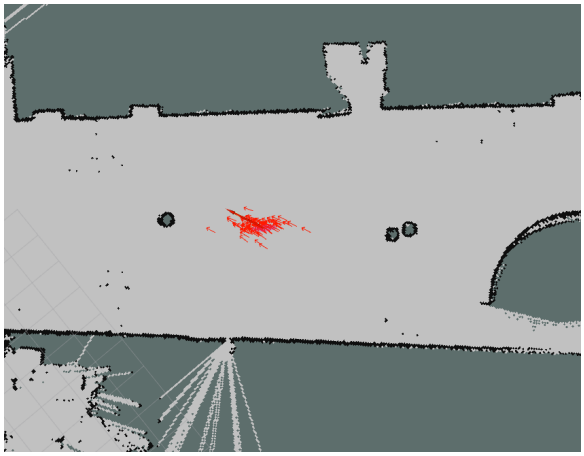
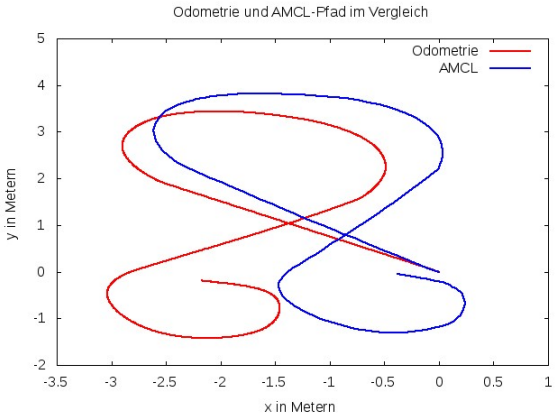


Figure 2: Partikelverteilung nach kurzer Neuorientrierung durch Bewegung

Der virtuelle Roboter mit besten den bereinstimmungen, ist die beste Estimation der Pose. Je näher die wahrscheinliche Hypothesen bei einander liegen, desto sicherer ist der Roboter sich seiner Pose. Dann kann die Anzahl der Hypothesen reduziert werden, daher dass A wie Adaptive aus AMCL. Dies reduziert die CPU Auslastung und den Speicher Partikel und Gewicht, virtuelle Roboter nur in einem Satz erwhnen.

3.3 Odometrie und AMCL im Vergleich

Um die Lokalisation durch AMCL und Odometrie miteinander zu vergleichen fährt der Roboter einen "Acht"-förmigen Pfad ab. Die Steuerung erfolgt manuell und wird zweimal in verschiedenen Geschwindigkeiten durchgeführt. Ein Problem entsteht am Anfang, da AMCL nur korrekt arbeitet, wenn die Startpose vorher durch den Benutzer auf den Punkt genau geschätzt wird oder durch Bewegung des Roboters spezifiziert wird. Durch die Bewegung verliert aber die Odometrie an Genauigkeit und startet nicht mehr im Ursprung (0/0). Um Genauigkeit zu garantieren, wird ein Algorithmus verwendet, der die Koordinatensysteme nach der Kalibrierung transformiert.



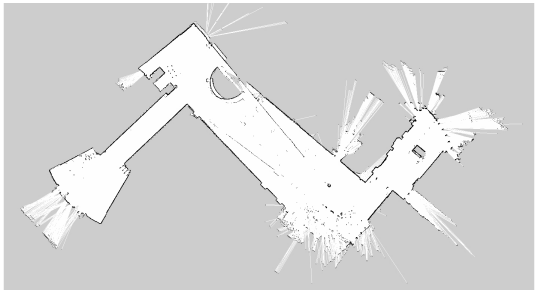
(a) Mit einer Geschwindigkeit $\leq$ 2km/h

Figure 3: Odometrie und AMCL im Vergleich.

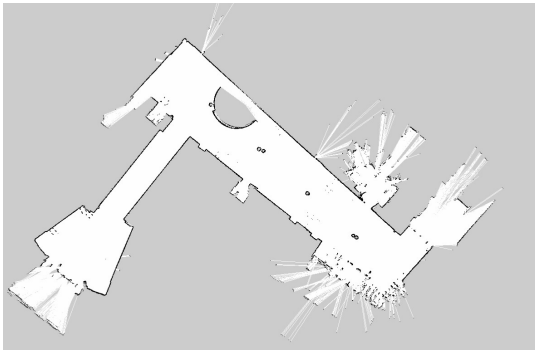
Man erkennt folgende Unterschiede ...
Im Experiment startet der Roboter im Ursprung und schon am Anfang kennzeichnet sich ab, dass der Odometriepfad fehlerhaft ist. Bis der Roboter wieder an der Startpose ist summieren sich die Fehler auf und die Zielpose weicht bis zu 2 m von der Sartpose ab. Der AMCL Pfad dagegen kehrt bis auf wenige Zentimeter zur Startpose zurück. Zusätzlich muss die menschliche Ungenauigkeit in diesem Versuch beachtet werden. Aus dem Vergleich ist ersichtlich, dass der Fehler bei AMCL auch in der Distanz, also bei langen Pfaden, nicht gröer ist, während bei der Odometrie bei langer Laufzeit die Qualität stetig abnimmt.
Nachteil AMCL: bentigt Karte, Pfad kann nicht in unbekannter Umgebung abgefahren werden
Vorteil: Odometrie ist häufiger und schneller verfügbar da der Rechenaufwand geringer ist.

4 Kartierung mit Gmapping

[3] Die Lokalisation eines Roboters benötigt eine Karte. Eine unbekannte Umgebung wird aus den Daten eines Laserscanners und den aktuellen Posedaten erfasst und von dem Algorithmus GMapping verarbeitet. Die Posedaten publiziert die Odometrie an das ROS Paket gmapping. Die Herausforderung liegt dabei in der gleichzeitigen Lokalisierung und Kartierung, dem *simultaneous localization and mapping* Problem kurz SLAM. Denn beide bedingen sich gegenseitig. Um zum Beispiel zwei Laserscans zu einer Karte zusammenzufügen, müssen die relativen Posen der Aufnahmen bekannt sein. Also ein Lokalisierungsproblem. Und um sich mit dem Laserscanner zu lokalisieren, benötigt man wiederum Kartendaten.
g von Gmapping erklren!



(a) fehlerhafte Karte



(b) korrekte Karte

Figure 4: Karten aufgezeichnet mit Gmapping

Das Untergeschoss des Informatikinstituts Würzburg ist mit einem Frauenhofer-Roboter, der mit einem Sick-Laserscanner ausgestattet ist, in einer 2D-Karte kartiert. Um diese Karte korrekt aufzunehmen, werden nur so und soviel Partikel benötigt. Die Aufnahme ist bis auf einen 1 cm genau und zeigt keine signifikanten Fehler. Bewegte Objekte wie Menschen werden erkannt und nicht in der Karte verzeichnet. Dagegen sorgt helles Licht, das durch die Fensterfronten scheint für eine Ungenauigkeit und kann nicht als klare Begrenzung festgestellt werden. Für klare Linie wie Wände ist es wichtig, das Gelände mit einem Geschwindgkeitslimit von 1 m/s abzufahren.
Deutliche Unterschiede sind in den Karten von Figure 4 zu erkennen. Bild (a) zeigt eine Karte, die im ersten Versuch aufgenommen ist und aus Unwissenheit mit zu hoher Geschwindigkeit und nicht oft genug abgefahren ist. Im Vergleich dazu ist die Karte (b) durch langsameres und stetigeres Abfahren detailgetreuer und hat klare Linien.
eigene Karte einfgen. AMCL erwhnen.

5 Pfadverfolgung mit Giovanni Indiveri

[4]

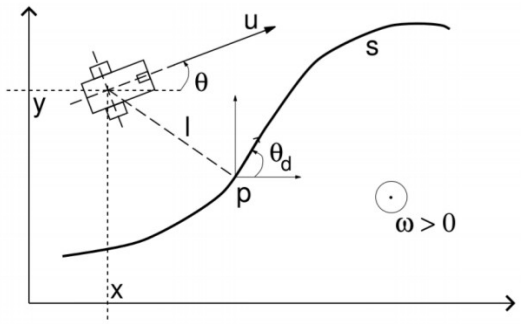


Figure 5: Modell der Pfadverfolgung

Der nicht-lineare Regler des Giovanni Indiveri und der Maria L. Corradini wird zur Pfadverfolgung verwendet. Die Implementierung garantiert nach Lyapunov, für einen beschränkten, nicht-linearen Pfad, asymptotische Konvergenz und asymptotisch stabile Fehlerdynamik. Dabei muss sowohl die maximale Geschwindigkeit, als auch der minimale Wendekreis des Roboters beachtet werden. Schrittweise neue Berechnungen der Regler-

parameter führen zu einer schnelleren Konvergenz. Hierzu verwendet der Algorithmus eine orthogonale Projektion auf den Roboter selbst. Der Pfad wird zur Vereinfachungen in lineare Teilschnitte genähert, wodurch sich die Gleichungen vereinfachen. Die Formel für die Winkelgeschwindigkeit ω

$$\omega = \frac{u\kappa(s)cos(\tilde{\theta})}{1 - l\kappa(s)} - hul\frac{sin(\tilde{\theta})}{\tilde{\theta}} - \gamma\tilde{\theta} : h\gamma > 0 \tag{1}$$

vereinfacht sich im linearen Fall $\kappa(s) = 0$ zu

$$\omega = -huy\frac{sin(\theta)}{\theta} - \gamma\theta : h\gamma > 0 \tag{2}$$

Durch Koordinatentransformation, dient die x-Achse als die Fahrtrichtung des Robotors und y (der Roboter-Pfad-Abstand) übernimmt die Rolle des l. Der Winkel Theta wird durch das übereinanderlegen zu null.

6 Zusammenfassung und Ausblick

References

[1] Monte carlo localization. Wikipedia.

[2] Robot operating system. Wikipedia.

[3] Wolfram Burgard Giorgio Grisetti, Cyrill Stachniss. Improved techniques for grid mapping with rao-blackwellized particle filters. *IEEE Transactions on Robotics*, 23:34–46, 2007.

[4] Giovanni Indiveri Maria Letizia Corradini. Switching linear path following for bounded curvature car-like vehicles. 2004.