Ein Überblick über SLAM

Nils Schönherr

Hochschule für Angewandte Wissenschaften Hamburg

Zusammenfassung. Diese Arbeit soll einen Überblick über die Simultaneous localization and mapping (SLAM) Thematik geben. Die Betrachtung verschiedener bereits eingesetzter Umsetzungen stellt den Stand der Technik dar. Darüber hinaus werden Schwächen und Stärken der Umsetzungen durch einen Vergleichspaper aufgezeigt.

Schlüsselwörter: SLAM · Simultaneous Localization and Mapping · Lidar

1 Einführung

Simultaneous localization and mapping (SLAM) ist eine der größeren Herausforderungen für autonome Systeme. Die Schwierigkeit besteht darin, dass für eine gute Positionierung das Wissen über die Umgebung genutzt werden muss. Gleichzeitig wird die aktuelle Position des Systems zur Erstellung einer Karte der Umgebung benötigt. Die gegenseitige Abhängigkeit erschwert die Lösung der beiden Probleme.

In dieser Arbeit sollen Grundlagen zum Thema SLAM erarbeitet werden. Dafür werden drei häufig verwendete Umsetzungen für zweidimensionale Karten vorgestellt: GMapping [3], HectorSLAM [5] und Google Cartographer [4]. Anschließend werden zwei aktuellere Arbeiten vorgestellt. Die Erste vergleicht die zuvor beschriebenen Ansätze. Die Zweite stellt eine Optimierung des Ansatzes von GMapping vor. Abschließend wird ein Fazit zu den gewonnenen Erkenntnissen durch die Recherche gezogen.

2 GMapping

GMapping verwendet ein Rao-Blackwellized Partikel Filter zur Lösung des SLAM Problems. Jedes Partikel enthält eine individuelle, komplette Karte der Umgebung. Die Karten werden durch die Beobachtungen und die Trajektorie, welche das Partikel repräsentiert, aufgebaut. Die Trajektorie entwickelt sich anhand der Bewegung des Roboters, weshalb die Proposal Distribution dem probilisitischen Bewegungsmodel der Odometrie entspricht. Es wird ein Sampling Importance Resampling Filter verwendet. Der Algorithmus arbeitet in vier Schritten:

– Sampling: Die nächste Partikelgeneration wird erhalten mithilfe der Proposal Distribution basierend auf der aktuellen Partikelgeneration.

– Importance Weighting: Jedes Partikel erhält ein individuelles Gewicht, welches betrachtet, dass die Proposal Distribution im Allgemeinen nicht der wahren Verteilung der Nachfolgezustände entspricht.
– Resampling: Partikel mit niedrigem Gewicht werden durch Partikel mit hohem Gewicht ersetzt.
– Map Estimation: Für jede Pose wird die korrespondierende Kartenerwartung basierend auf der Trajektorie und den Beobachtungen berechnet.

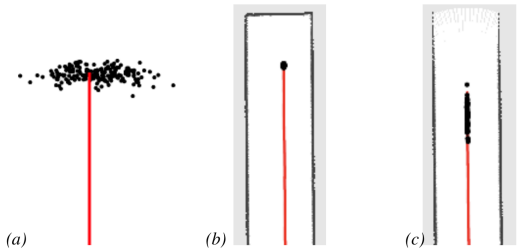


Abb. 1. Proposal Distributions: a) offener Bereich: große Verteilung der Partikel; b) Sackgasse: wenig Unsicherheit in beiden Achsen; c) offener Gang: Unsicherheit entlang des Ganges [Bildquelle [3]]

Die Arbeit beschreibt zwei Optimierungen, welche die Anzahl der benötigten Partikel stark reduziert. Die erste Optimierung ist die Berechnung einer verbesserten Proposal Distribution. Die These ist, dass aufgrund der hohen Genauigkeit der Laserscans, die Ähnlichkeitsfunktion, welche einen Scan mit der bisherigen Karte und der aktuellen Bewegung vergleicht, den größten Einfluss auf die Proposal Distribution hat. Weiter wird diese durch eine Gaußverteilung angenähert. Die Parameter der Gaußverteilung werden durch Auswertung der Ähnlichkeitsfunktion in der Umgebung um das lokale Maximum bestimmt, welches vom Scan-Matcher gefunden wurde. Die daraus folgenden Verteilungen (siehe Abb. 1) zeigen, dass die Anzahl der benötigten Partikel veringert werden kann.

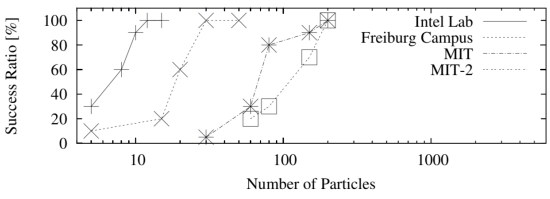


Abb. 2. Erfolgschance von 10 Durchläufen in verschiedenen Umgebungen. Bei MIT-2 wurde adaptives Resampling ausgeschaltet. [Bildquelle [3]]

Die zweite Optimierung geschieht durch selektives Resampling. Dafür wird die Anzahl an sogenannten effektiven Partikeln bestimmt. Wenn die Anzahl der effektiven Partikel weniger als die Hälfte aller Partikel ist, dann wird ein Resamplingprozess gestartet. Dadurch soll verhindert werden, dass durch zu häufiges Resampling gute Partikel gelöscht werden. Die durchgeführten Tests zeigen, dass der Algorithmus akkurate Karten erstellen kann. Die Anzahl der benötigten Partikel konnte gesenkt werden. In Abb. 2 kann man sehen, dass das adaptive Resampling mit weniger Partikeln auskommt.

3 Hector SLAM

Hector SLAM nutzt anders als GMapping keine Odometrieinformationen. Bei diesem Ansatz werden 2D Lidarscans mittels Inertial Measurement Unit (IMU) in eine Ebenenkarte transformiert. Dadurch ist es nicht nötig, dass der Lidar stets waagerecht orientiert ist. Dies ermöglicht den Einsatz des Systems auf Fahrzeugen, welche sich nicht ausschließlich auf einer Ebene bewegen.

Zur Darstellung der Umwelt wird eine Occupancy Grid Map verwendet. Der transformierte Lidarscan wird gefiltert, sodass ausschließlich Endpunkte innerhalb eines gewissen Höhenintervals für das Scan-Matching verwendet werden.

Während des Scan-Machtings werden die gefundenen Endpunkte des Lidarscans bestmöglich in der bereits gelernten Karte orientiert. Dabei wird eine Transformation gesucht, welche die Ausrichtung und Position des Roboters beschreibt.

Abb. 3. Multiresolution: 20cm, 10cm und 5cm Raster [Bildquelle [5]]

Die Optimierung der Transformation wird mit dem Gauß-Newton-Verfahren erreicht. Um zu vermeiden, dass bei der Optimierung nur ein lokales Minimum erreicht wird, werden mehrere Karten mit jeweils halber Auflösung angelegt (siehe Abb. 3).

Zur Bestimmung der aktuellen Position und Ausrichtung des Roboters wird ein Extended Kalman Filter verwendet. Die Geschwindigkeit und Position, welche durch Integration ermittelt werden, würden ohne Korrektur driften, weshalb die 2D Position und Rotation des Scan-Matchers zur Korrektur genutzt werden. Ebenso werden die Position und Ausrichtung des Kalman Filters in die 2D Ebene projiziert und dienen als Ausgangspunkt für den Scan-Matcher.

Die Tests der Autoren zeigen, dass das System in verschiedenen Situationen erfolgreich eingesetzt werden kann. Zum Beispiel wurde ein unebenes Terrain des RoboCup 2011 zu Fuß erkundet oder mit einem Boot auf einem Fluß gefahren. Die generierten Karten stimmen gut mit Grundrissen oder Satellitenbildern überein. Hervorzuheben ist, dass kein expliziter Loop-Closure ausgeführt wird.

Die Berechnungen für diesen Algorithmus lassen sich auch auf Leistungsschwachen Computern in Echtzeit berechnen.

4 Google Cartographer

Der Ansatz von Cartographer verwendet Submaps, um Laserscans einzufügen. Das Scan-Matching wird nur auf der aktuellen Submap ausgeführt, welche als ausreichend akkurat für kurze Zeiträume angenommen wird. Wenn eine Submap abgeschlossen ist, werden keine weiteren Scans hinzugefügt. Abgeschlossene Submaps werden verwendet, um Loop-Closure mittels Scan-Matching zu machen. Dabei werden Submaps in der Nähe der geschätzten Roboter Position verglichen mit dem aktuellen Scan. Passt der Scan gut in die Submap, so wird sie verwendet, bei der Loop-Closure Optimierung.

Die Submaps bestehen aus einem Raster mit Abstand r, in dem jeder Pixel die Wahrscheinlichkeit enthält, dass er belegt ist. Beim Einfügen eines neuen Scans werden alle Pixel, welche von einem Scanpunkt getroffen werden in eine Liste der Treffer geschrieben und alle Pixel, welche von einem Strahl vom Scan-Ursprung zum Scanpunkt geschnitten werden, in eine Liste der Fehltreffer geschrieben. Nun werden die Wahrscheinlichkeiten in der Submap aktualisiert.

Bevor ein neuer Scan in die aktuelle Submap eingefügt wird, muss die Scan-Pose optimiert werden. Dafür wird der Ceres Solver [1] verwendet. Während der Optimierung sollen die Wahrscheinlichkeiten an den Scanpunkten maximiert werden. Ähnlich wie bei Hector SLAM ist auch bei diesem Verfahren ein guter erster Schätzwert für die Scan-Pose wichtig, da lokale Optimierungen ausgeführt werden. Über eine IMU kann die Rotation der Scan-Pose geschätzt werden.

Für Loop-Closure wird ebenfalls Ceres verwendet um das Optimierungsproblem zu lösen. Dabei werden sowohl die Submap-Posen sowie die Scan-Posen durch Einschränkungsbedingungen optimiert. Die Bedingungen beziehen jeweils für eine Submap und für einen Scan die relative Pose des Scans in der Submap und die zugehörige Kovarianzmatrix ein.

Um Scans in Submaps zu matchen wird Branch-and-bound Scan-Matching verwendet. Bei dieser Methode wird die Pose innerhalb eines Suchfensters um eine Startpose herum gesucht. Innerhalb des Suchfensters werden eine ganzzahlige Anzahl an Posen ermittelt. Dabei wird die Pose um die Rastergröße r verschoben oder so gedreht, dass sich der am weitesten entfernte Punkt des Scans maximal um r verschiebt.

Anstatt alle möglichen Posen im Suchfenster auszuprobieren um die beste zu finden, wird eine Baumstruktur aufgebaut, in der die Wurzel alle Posen im Suchfenster repräsentiert. Alle Kindknoten eines Knotens im Baum ergeben eine Partition der möglichen Posen des Elternknotens. Blattknoten repräsentieren

eine einzelne Pose. Wichtig hierbei ist, dass der Wert eines Knotens eine obere Grenze für die Werte aller seiner Kindknoten ist. Damit die Berechnung für die obere Grenze eines jeden Knotens schnell ablaufen kann, werden die Submaps für jede Baumebene mit einer Max-funktion auf einem $2^h \times 2^h$ Kern für Baumebene h vorberechnet (siehe Abb. 4).

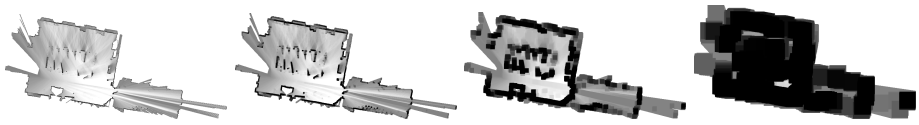


Abb. 4. Vorberechnete Submaps (Größe 1, 4, 16, 64). [Bildquelle [4]]

Dies wird erst dann gemacht, wenn die jeweilige Submap abgeschlossen ist und keine weiteren Scans eingefügt werden. Somit muss die Vorberechnung für die Baumebenen nur einmalig ausgeführt werden. Dadurch kann die obere Grenze eines Knotens im Baum mit linearem Aufwand zu der Anzahl der Scanpunkte berechnet werden.

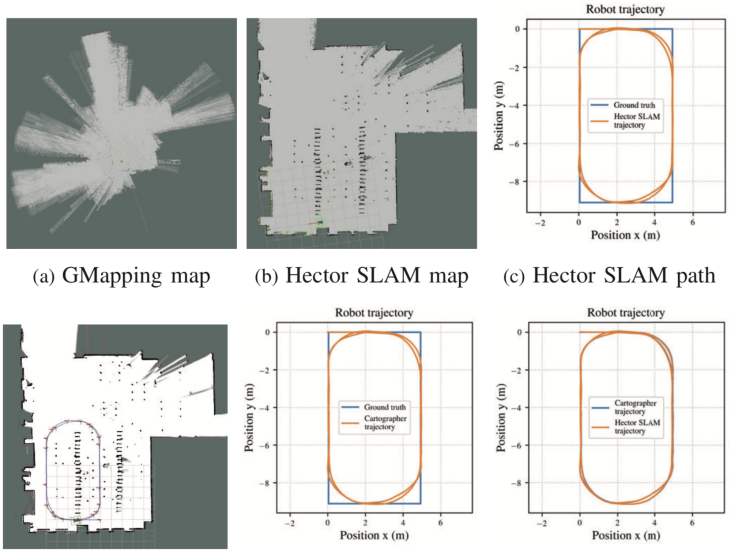
Abb. 5. Durch Cartographer generierte Karte mit Revo LDS Sensordaten. [Bildquelle [4]]

Die Autoren zeigen die Resultate des vorgestellen Algorithmus zum Beispiel für größere Areale (2. Stock des Deutschen Museums) oder zeigen, dass auch mit kostengünstiger Hardware eines Staubsaugerroboters akkurate Ergebnisse erreicht werden können (siehe Abb. 5). In letzterem Versuch wurde die erzeugte Karten exemplarisch mit handgemessenen Kanten verglichen und erreichen relative Fehler von unter einem Prozent.

Weitere Vergleiche zu anderen Algorithmen (Graph Mapping und Graph FLIRT) werden mithilfe des Radish Datensatzes gemacht. Der Algorithmus kann in Echtzeit auf durchschnittlicher Hardware ausgeführt werden und liefert teils bessere und teils schlechtere Ergebnisse als die zu vergleichenden Algorithmen, wobei auch die schlechteren Ergebnisse weiterhin im Erwartungshorizont der Autoren liegen.

5 Vergleichspaper

Die Autoren von [2] vergleichen in ihrer Arbeit verschiedene SLAM Algorithmen. Abgesehen von den kamerabasierten Algorithmen wurden auch GMapping, Hector SLAM und Cartographer in der Kategorie Lidar SLAM verglichen. Das Testszenario ist ein offener Büroraum in dem die Testplattform zwei Runden auf einem $5m \times 9m$ Rechteck fährt. Die Plattform wird manuell gesteuert. Die Auswertung der Algorithmen erfolgt auf den aufgenommenen Daten (Rosbag) der Testfahrt.



(a) GMapping map (b) Hector SLAM map (c) Hector SLAM path

Abb. 6. Vergleich von GMapping, Hector SLAM und Cartographer [Bildquelle [2]]

In Abb. 6 kann man sowohl die erzeugten Karten, sowie die sich daraus ergebenen Trajektorien sehen. GMapping kann aus den Daten keine sinnvolle Karte erstellen, weshalb auch keine Trajektorie ermittelt werden kann. Hector SLAM und Cartographer haben sehr ähnliche Karten und Trajektorien ermittelt. Die Autoren empfehlen jedoch Cartographer als bessere Lösung, weil dort eine globale Kartenoptimierung für Loop-Closure im Gegensatz zu Hector SLAM gemacht wird.

Die Ergebnisse der Autoren bezüglich kamerabasierter Algorithmen möchte ich in dieser Arbeit nicht weiter betrachten.

6 Optimierung für eingebettete Systeme

Das Paper [6] beschäftigt sich mit der Implementation eines Rao-Blackwellized Partikel Filters, welches speziell für eingebettete Systeme optimiert ist. Eine Beschleunigung soll erreicht werden durch algorithmische Optimierung und Parallelisierung. Im Gegensatz zu GMapping, wo das Gradientenverfahren zum Scan-Matching angewendet wird, verwenden die Autoren zur groben Schätzung eine Brute-Force-Methode aus [7], um nicht in lokalen Minima gefangen zu bleiben. Anschließend wird die Schätzung mit dem Gradientenverfahren verfeinert. Die Brute-Force-Methode wird beschleunigt durch die Verwendung einer Lookup-Tabelle, welche die rechenintensive Suche nach dem nächsten Hindernispunkt ersetzt.

Drei Multithreadingmodelle werden vorgestellt: Boost Thread, Threading Building Blocks und Open Multi-Processing. Diese sollen die acht Kerne der Testhardware (ODROID-XU4) ausnutzen, um die Verarbeitung der Partikel zu beschleunigen. Zum Vergleich wird ein Lenovo T540p Laptop verwendet.

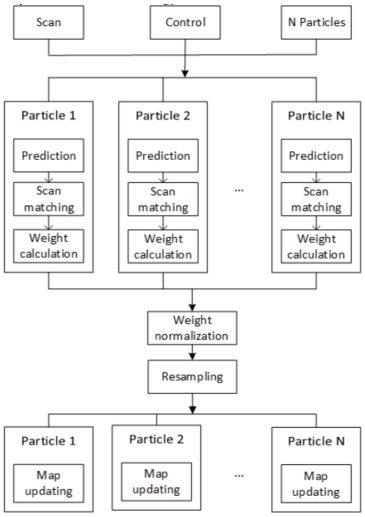


Abb. 7. Processing Loop des parallelisierten Rao-Blackwellized Partikelfilters [Bildquelle [6]]

In Abb. 7 sind die Veränderungen durch Parallelisierung sichtbar: Die Operationen auf Partikeln (Prediction, Scan-Matching, Weight Calculation und Map Updating) laufen parallel statt seriell ab. Dies sind die Operationen, welche bei

der vorherigen Analyse mehr als 94% der Rechenzeit in Anspruch genommen haben.

Zur Auswertung werden die zwei Datensätze Intel und ACES verwendet. Die durchschnittliche Rechenzeit für das Partikelupdate ist 9.9 mal so schnell wie bei GMapping auf dem Odroid. Auf dem Lenovo Laptop ist es nur ein Faktor von 1.4. Zu beachten ist, dass die Optimierung durch Lookup-Tabelle mehr Speicher bedarf. Auch die verschiedenen Modelle zur Parallelisierung benötigen mehr Speicher als die serielle Variante. Die Parallelisierung des seriellen Ansatzes durch BoostThread und TBB ist besser als durch OpenMP und liegt im Bereich um Faktor 4 (OpenMP etwas unter 4). Es wird darauf hingewiesen, dass der Odroid 4 starke Kerne und 4 schwache Kerne hat, wohingegen der Laptop 4 Kerne (8 Threads) hat, weshalb unter Anderem die theoretische Beschleunigung mit Faktor 8 (Anzahl der Threads) nicht erreicht wird. Der Laptop ist im Durchschnitt 4.63 mal so schnell wie der Odroid.

Bezogen auf den Positions- und Rotationsfehler erzielen GMapping und der neuvorgestellte Algorithmus vergleichbare Ergebnisse. Der neue Ansatz ist jedoch mit allen Optimierungen 12 mal so schnell wie GMapping.

7 Fazit

Die drei vorgestellten Umsetzungen von SLAM umfassen verschiedene Ansätze: Partikelfilter, lokales Scan-Matching ohne Loop-Closure und kombiniertes lokales Scan Matching mit globaler Optimierung für Loop-Closure.

Der Vergleich der Autoren von [2] lieferte für GMapping keine Ergebnisse, weshalb kein Vergleich von Hector SLAM und Cartographer zu GMapping gemacht werden konnte. Das ausgewählte Testszenario enthielt nur wenig Merkmale und erstreckte sich über einen kleinen Bereich. Die wesentlichen Unterschiede zwischen Cartographer und Hector SLAM wären wohl erst in komplexeren Testfeldern erkennbar geworden. Dort kämen die Vorteile der globalen Optimierung von Cartographer zum Vorschein. Ein Vergleich zwischen den Umsetzungen in Bezug auf die benötigte Rechenleistung wäre aus meiner Sicht ebenfalls interessant gewesen.

Besonders spannend finde ich die Optimierungen für eingebettete Systeme, welche in [6] vorgestellt wurden. Die erzielten Verbesserungen sind erheblich und zeigen, dass durch weitere Forschung in diese Richtung auch kleinere Systeme mit wenig Rechenleistung in die Lage versetzt werden können, mittels SLAM Karten aufzubauen und sich darin zu orientieren.

In weiterführenden Arbeiten könnte die Anwendbarkeit dieser Algorithmen in Systemen mit stärker eingeschränkten Ressourcen erforscht werden.

Literatur

1. Agarwal, S., Mierle, K., Others: Ceres solver. http://ceres-solver.org
2. Filipenko, M., Afanasyev, I.: Comparison of various slam systems for mobile robot in an indoor environment. In: 2018 International Conference on Intelligent Systems (IS). pp. 400–407 (Sep 2018). https://doi.org/10.1109/IS.2018.8710464
3. Grisettiyz, G., Stachniss, C., Burgard, W.: Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling. In: Proceedings of the 2005 IEEE International Conference on Robotics and Automation. pp. 2432–2437 (April 2005). https://doi.org/10.1109/ROBOT.2005.1570477
4. Hess, W., Kohler, D., Rapp, H., Andor, D.: Real-time loop closure in 2d lidar slam. In: 2016 IEEE International Conference on Robotics and Automation (ICRA). pp. 1271–1278 (2016)
5. Kohlbrecher, S., Meyer, J., von Stryk, O., Klingauf, U.: A flexible and scalable slam system with full 3d motion estimation. In: Proc. IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR). IEEE (November 2011)
6. Li, Q., Rauschenbach, T., Wenzel, A., Mueller, F.: Emb-slam: An embedded efficient implementation of rao-blackwellized particle filter based slam. In: 2018 3rd International Conference on Control, Robotics and Cybernetics (CRC). pp. 88–93 (Sep 2018). https://doi.org/10.1109/CRC.2018.00026
7. Olson, E.B.: Real-time correlative scan matching. In: 2009 IEEE International Conference on Robotics and Automation. pp. 4387–4393 (May 2009). https://doi.org/10.1109/ROBOT.2009.5152375