Universität Bremen

Fachbereich Informatik

Fachgebiet Bildverarbeitung

Lokalisation mobiler Roboter mit Odometrie und Bildverarbeitung in einer Theaterinstallation

MASTERARBEIT

eingereicht am 03.09.2013

bei

Prof. Dr.-Ing. Udo Frese

Prof. Dr.-Ing. X Y

Universität Bremen

von

Josef F. Hiller

Matr. Nr: 2055491

Osterstr 79

28199 Bremen

Zusammenfassung

Schlagwörter: abc, def, xyz

Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbstständig angefertigt und mich fremder Hilfe nicht bedient habe. Alle Stellen, die wörtlich oder sinngemäß veröffentlichtem oder unveröffentlichtem Schrifttum entnommen sind, habe ich als solche kenntlich gemacht.

Bremen, xx.09.2013

Josef F. Hiller

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

In dieser Arbeit soll, mit Hilfe einer Simulation, untersucht werden, wie gut sich Kameras bei der Standortbestimmung in mobilen Roboter einsetzten lassen. Anlass dieser Arbeit war eine Anfrage der Künstlergruppe *Beobachter der Bediener von Maschinen* (BBM), die schon bei mehreren Performances[1] mobile Roboter eingesetzt haben. Dabei interessierten sie sich für eine Lokalisierungslösung welche möglichst ohne weitere spezial Hardware und geringem Installationsaufwand vor Ort auskommt. Der Ansatz der daraus entstand war: die Kameras, die bereits an jedem der Roboter verbaut waren, zu nutzen um markante Muster in der Bühneninstallation zu erkennen und gemeinsam mit den Odometrie-Daten zur Positionsberechnung zu verwenden. Dabei sollte ein Partikelfilter als Zustandsschätzer verwendet werden. Zu der Bühneninstallation gehören große Lichtwände, zu sehen auf Abbildung 1.1 und 1.2. Auf diese Lichtwände könnte ein hell/dunkel Bit-Muster angezeigt werden das es mit Hilfe geeigneter Bildverarbeitungsalgorithmen und mit den Kameras zu erkennen gilt.

1.1 Ziel und Aufgabenstellung der Arbeit

Im Rahmen dieser Arbeit soll eine Simulationsumgebung mit Hilfe geeigneter 3D-Visualisierungs Bibliotheken erstellt werden. Die in der Lage ist eine 3D-Szene der Bühne zu simulieren und Kamerabilder sowie Odometrie-Daten einer Roboterfahrt zu erzeugen. Außerdem soll ein Lokalisationsalgorithmus entwickelt werden, der auf Grundlage dieser Daten die Position und Orientierung des Roboters auf der Bühne schätzen kann. Anschließend soll die Qualität dieser geschätzten Position beurteilt werden und mögliche Fehlerquellen diskutiert werden.

[1] unter Anderem:
2000 Themenpark "Wissen"der Expo 2000 Hannover
2010 Joybots in der BMW-Welt
2012 EPKOT Experimental Prototype Killers of Tomorrow , Hannover
siehe auch `http://www.bbm.cfnt3.de`

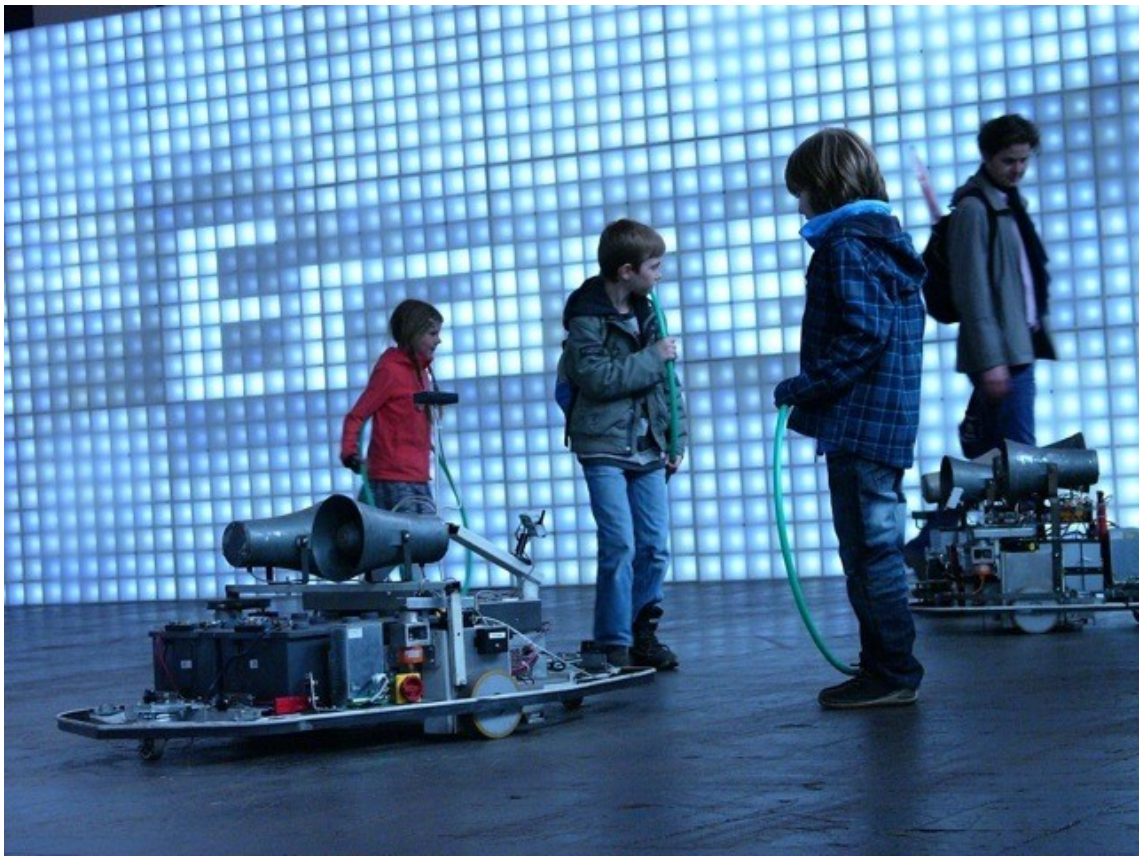

Abbildung 1.1: Roboter und Besucher auf der EPKOT Quelle: http://www.bbm.cfnt3.de

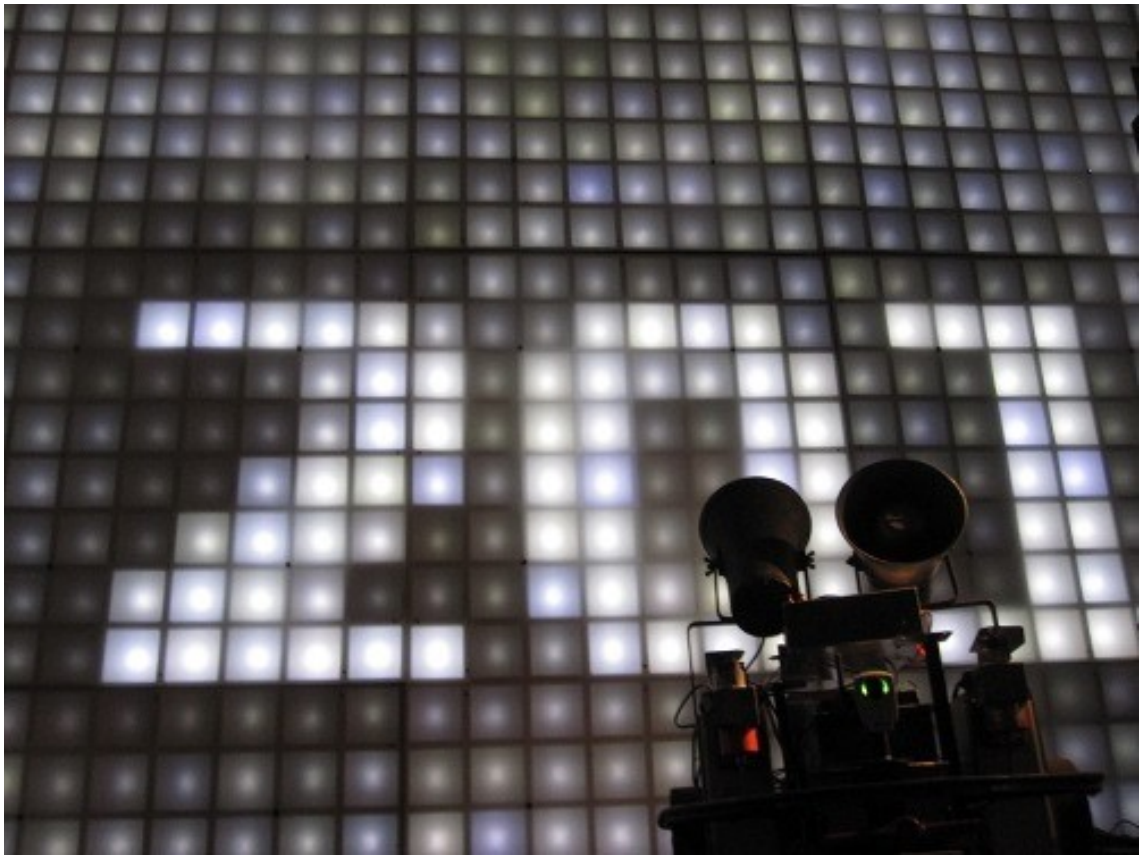

Abbildung 1.2: Roboter vor Lichtwand auf der EPKOT Quelle: http://www.bbm.cfnt3.de

1.2 Gliederung

Die Arbeit wird im folgenden Kapitel eine kurze Übersicht gängiger Lokalisationsverfahren sowie ihre Vor- und Nachteile geben. Außerdem wird in die Grundlagen eingeführt, welche zum Verständnis der folgenden Kapitel notwendig sind. Das Kapitel *Lokalisierung mittels Bildverarbeitung* beschreibt wie die Methoden aus den Grundlagen an das gestellte Problem angepasst wurden und wie das vorgestellte Verfahren funktioniert. Anschließend wird die Simulations- und Lokalisationssoftware vorgestellt und deren Implementation erklärt und begründet. Insbesondere wird darauf eingegangen, wie realistisch die Simulation ist. In Kapitel 5 beginnt die Beschreibung verschiedener Versuche die zum Beurteilen der Lokalisationsergebnisse durchgeführt wurden. Ergebnispräsentation und Diskussion erfolgen jeweils im Anschluss der Beschreibungen. Abschließend wird ein Ausblick zur möglichen Anwendung dieses Verfahrens gegeben, sowie mögliche Fehlerquellen und Probleme dabei. Im letzten Kapitel wird ein Fazit zu den Erkenntnissen dieser Arbeit gezogen.

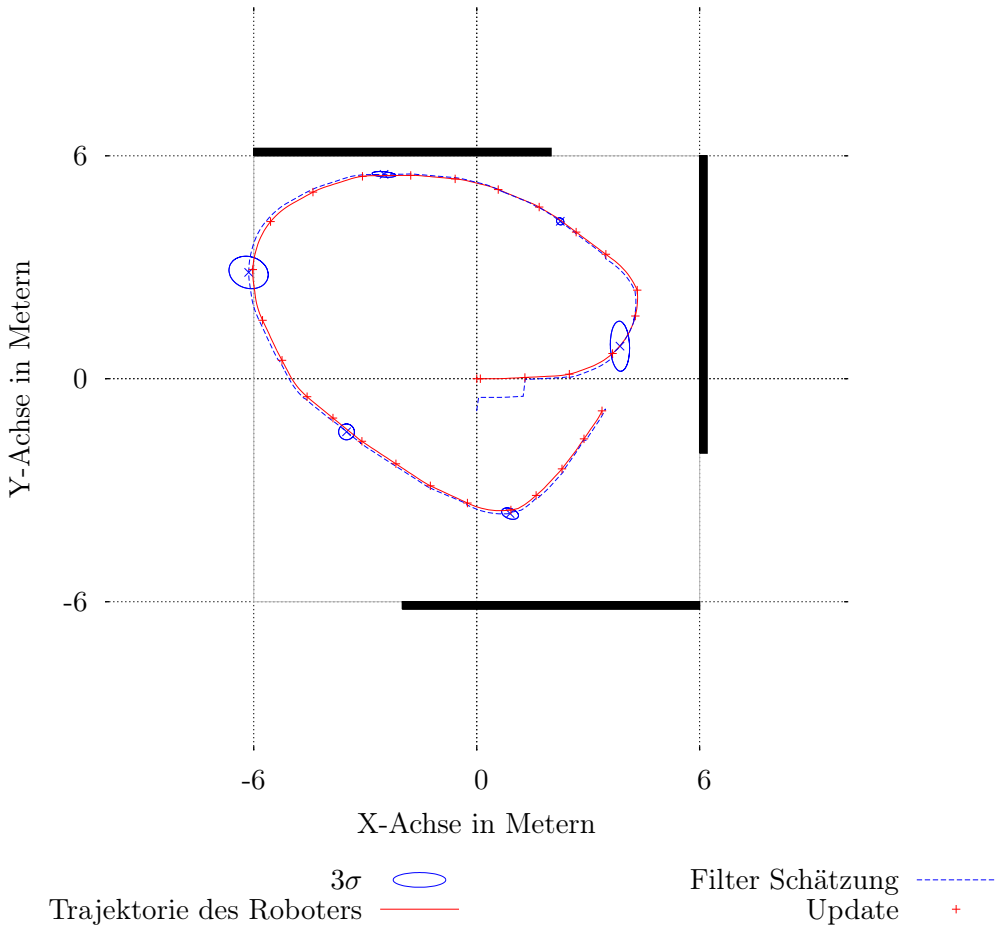

Abbildung 1.3: every1000mspics

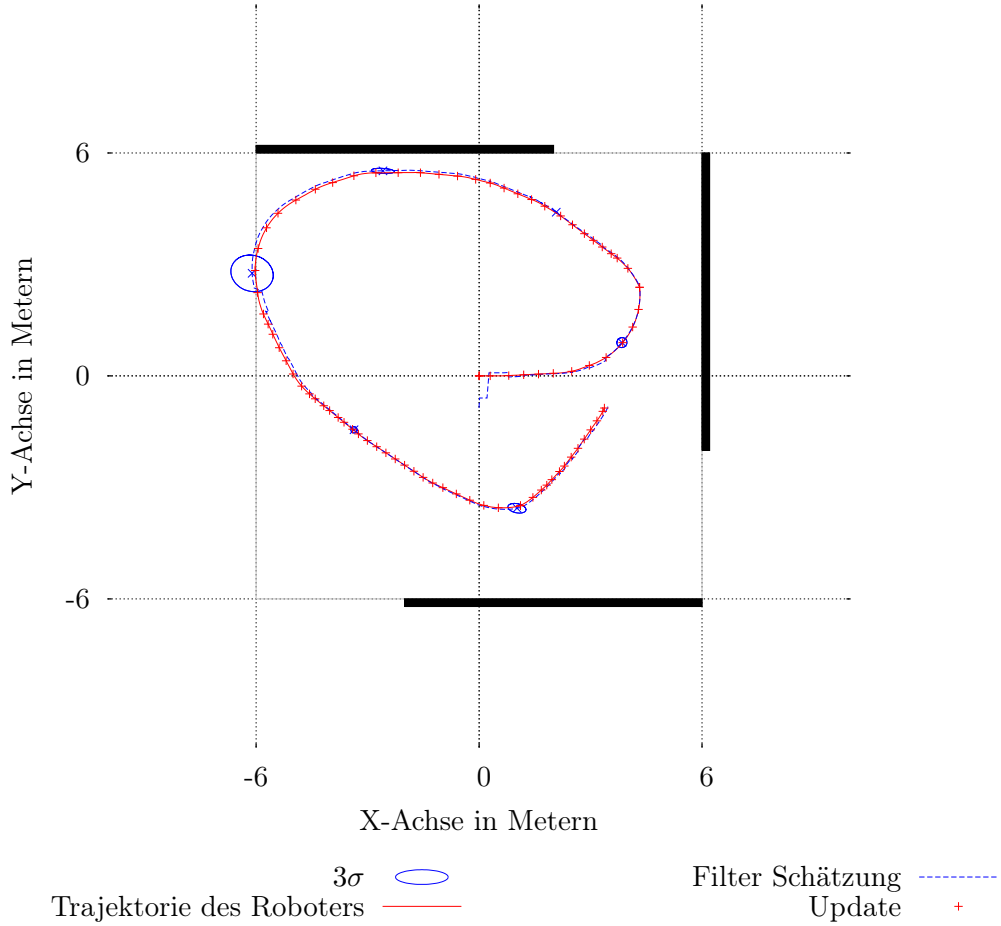

Abbildung 1.4: every500mspics

2 Grundlagen

2.1 Lokalisation

Die Lokalisation ist eines der Grundprobleme das beim Einsatz von mobilen Robotern auftritt. In [Thrun et al., 2006, Seite 193] wird die sie in drei Teilprobleme zerlegt:

Position Tracking bezeichnet den Vorgang, bei dem mit bekannter Ausgangsposition diese mit Hilfe von Sensordaten bei Bewegungen verfolgt werden kann. Dabei spielt das Dynamikmodell des Roboters sowie darin modellierte Unsicherheiten eine wichtige Rolle. Denn bewegt sich der Roboter von einer bekannten Position aus, wird mit dem Dynamikmodell seine neue Position geschätzt. Die Unsicherheiten des Modells erzeugen dabei eine Wahrscheinlichkeitsverteilung um diese neue Position, in der sich die wahre Position befinden sollte. Ohne Messungen von weiteren Sensoren die Rückschlüsse auf die Umgebung erlauben, würde die Positionsschätzung mit der Zeit immer ungenauer. Mit Hilfe eines Messmodells lässt sich beurteilen, ob eine Messung an einer Bestimmten Position wahrscheinlich erscheint, oder nicht. Dadurch lässt sich die Wahrscheinlichkeitsverteilung der Position nach einer Bewegung durch eine Messung wieder eingrenzen. Auf den Bereich, in dem der Messwert des Sensors am Wahrscheinlichsten ist.

Global Localization ist das finden der Anfangsposition des Roboters unter allen Möglichen Posen die im Szenario vorkommen können. Im Vergleich zum *Position Tracking*, bei dem es genügte die Unsicherheit um die geschätzte neue Pose zu berücksichtigen, umfasst hier der Raum möglicher Posen ein erheblich größeres Volumen. Ein Ansatz wäre alle möglichen Posen mit der selben Wahrscheinlichkeit anzunehmen, und mit den ersten Messungen und dem Messmodell diese einzugrenzen. Auf Bereiche in denen diese Messungen mit hoher Wahrscheinlichkeit auftritt.

Kidnapped Robot Problem ist eine verschärfte Variante des *Global Localization* Problems. Dabei geht man davon aus, das sich der Roboter spontan an einem anderen Ort aufhält als vom Roboter angenommen. Nun müsste die Lokalisation des Roboters wieder im gesamten möglichen Raum erfolgen. Nur das der Roboter diesen Zustand nicht feststellen kann.

2.1.1 Übersicht gängiger Verfahren

Die meisten Lösungsansätze verwenden im Grunde eine Variante des Bayes-Filters wie z.B. in [Thrun et al., 2006, Seite 26] beschrieben. Abhängig vom Einsatzumfeld und der verwendeten Hardware gibt es aber noch Andere Verfahren.

In [Seco et al., 2009] zum Beispiel werden, neben dem Bayes-Filter, drei weitere genannt: *Geometry-Based Methods*, *Minimization of the cost function* und *Fingerprint Methods* die sich in Gebäuden unter Nutzung von Elektromagnetischen Signale verwenden lassen. Je nach Sensorausstattung werden verschiedene Karten der Umgebung verwendet. Auf die sich der Roboter durch Auswertung von Sensormessungen lokalisieren muss. Dabei gibt es Karten die vorher angefertigt wurden, und dem Roboter bereits zur Verfügung stehen. Und es gibt Karten die der Roboter selbstständig während der Lokalisation erstellen muss. Letzteres wird als *Simultaneous Localization and Mapping* (SLAM) Problem bezeichnet zu dem es bereits viele Veröffentlichungen gibt (u.a.) [Thrun et al., 2006, Seite 309], [Thrun, 2002], [Hertzberg et al., 2012, Seite 229]

Der in Kapitel 1 erwähnte Partikel Filter ist eine Variante des Bayes-Filters. Er ist auch als *Monte Carlo Localization* (MCL) bekannt. Alternativ dazu gibt es zum Beispiel den Kalman Filter, der mit Normalverteilungen die Wahrscheinlichkeiten abbildet. Er ist in seiner Anwendbarkeit auf lineare Probleme beschränkt oder ist auf Linearisierung angewiesen. Dafür ist er aber eines der ältesten Verfahren und damit am besten untersucht. In [Ha et al., 2012] wird er bei einem Lokalisationsproblem verwendet. Außerdem gibt es noch so genannte Grid-Based-Filter oder Histogram Filter. Diese teilen den Zustandsraum in so genannte Grids auf in denen ein Wahrscheinlichkeitswert jeweils den Zustand der Region abbildet. Für bestimmte Probleme mit nur zwei Zuständen, kann ein Binary Bayes Filter zum Einsatz kommen. Da in dieser Arbeit ein Partikel Filter verwendet wird, soll seine Funktionsweise kurz erklärt werden.

2.1.2 Der Partikel Filter

Zustandsraum

Der Partikel Filter bildet die Wahrscheinlichkeitsverteilung des vermuteten Zustandes durch eine diskrete Menge von so genannten Partikeln ab. Ein Partikel ist dabei ein Punkt im Zustandsraum zu dem noch ein Gewichtungswert gehört. Angenommen, der Zustand wäre die Position in einem dreidimensionalen Koordinatensystem ist, dann wäre der Zustandsraum alle möglichen Punkte in diesem Koordinatensystem. Ein Partikel in diesem Raum hätte also vier Attribute: einen X-Wert, einen Y-Wert, einen Z-Wert und einen Gewichtungswert. Im Grundzustand, ist der Gewichtungswert bei allen Partikeln gleich und die räumliche Verteilung oder Häufung repräsentiert die Wahrscheinlichkeitsverteilung des vermuteten Zustandes (hier die Position im Koordinatensystem). Häuften sich die Partikel um eine Position, so wäre bei einer zufälligen Ziehung aus den Partikeln, die Wahrscheinlichkeit höher ein Partikel von dieser Position zu ziehen. Mittelt man nun über alle Partikel im Zustandsraum, so erhält man eine Position die dem vermuteten Zustand entspricht. Dabei ist die Varianz über alle Partikel ein Maß dafür, wie zuverlässig diese Position ist.

Dynamikmodell

Partikel Filter nutzen für die Zustandsschätzung ein Dynamikmodell. In diesem Modell wird abgebildet, wie sich der Zustand über die Zeit verändert, z.B. durch Physikalische Gesetzmäßigkeiten oder durch eine Hilfsgröße die sich Messen lässt (z.B. zurück gelegter Weg). Damit erlaubt das Dynamikmodell eine Prognose über den nächsten Zustand abzugeben. Um das Beispiel weiter zu führen stelle man sich vor, man will die Position eines Satelliten der um die Erde kreist bestimmten. Dabei soll lediglich die Position im Raum und nicht seine Orientierung betrachtet werden. Der Zustandsraum besteht damit schon einmal aus X, Y, und Z-Koordinaten. Für das Dynamikmodell könnten hier die Keplerschen Gesetzte zur Bahnberechnung verwendet werden. Allerdings ist es dafür nötig, die Geschwindigkeit des Objektes zu kennen. Dazu wird der Zustandsraum um die Geschwindigkeiten in X, Y und Z-Richtung erweitert. Nun ist es bei einem Gegebenen Zustand möglich einen Folgezustand nach einer verstrichenen Zeit zu errechnen. Leider wird diese Prognose mit der Zeit immer ungenauer, denn in der Realität gibt es immer Störgrößen, die sich nicht vorhersagen lassen. Häufig jedoch lassen sich Aussagen über die Art und Stärke

der Störung machen. Dieses Wissen kann in Form von einer Wahrscheinlichkeitsverteilung um die Prognose verwendet werden. Bei dem Satelliten wäre als Störung der Einfluss von Sonnenwind denkbar der ihn von der Sonne weg beschleunigt. Weiß man wie stark der Einfluss werden kann, könnte man ihn als Statistische Größe in die Prognose einfließen lassen.

Messmodell

Das Messmodell wird benötigt, um entscheiden zu können, wie plausibel eine Messung bei gegebenem Zustand (Im Beispiel die Position) ist. Ein Sensor, über den das System verfügt, liefert einen Messwert. Mit dem Messmodell kann jetzt zu einem Beliebigen Zustand eine Aussage darüber gemacht werden, wie Wahrscheinlich es wäre diesen Messwert zu messen. Im Beispiel des Satelliten könnte der Abstand zu einer Bodenstation gemessen werden. Im Messmodell würde dann aus einem gegebenen Zustand, also Position und Geschwindigkeit, zusammen mit weiteren Informationen wie Position der Bodenstation auf der Erde, Datum und Uhrzeit der Messung errechnet werden wie groß der Abstand sein müsste. Stimmt dieser erwartete Abstand nicht mit dem gemessenen überein, so ist die Messung bei diesem Zustand (Position und Geschwindigkeit) unwahrscheinlich. Zusätzlich ist natürlich jeder Messwert mit einer Ungenauigkeit behaftet, die vom Messprinzip und Sensortyp abhängt. Aber auch diese lässt sich Statistisch angeben und auf den Messwert aufschlagen bevor er ins Messmodell gegeben wird. So lassen sich Zustände im Zustandsraum hervorheben, für die eine gegebene Messung wahrscheinlich sind.

Angewendet auf den Filter

Wenn der Zustandsraum alle möglichen Zustände eines Systems beschreibt, so sind die Partikel des Partikel Filters eine Untermenge davon, die in ihrer räumlichen Häufung den tatsächlichen Zustand des Systems beschreiben. Während des Betriebs gibt es zwei Ereignisse auf die der Filter reagiert:

Das **Dynamik-Update** berechnet auf Grundlage des Dynamikmodells eine neue Prognose. Konkret wird im Partikel Filter dafür zu jedem Partikel eine eigene Prognose gestellt. Dabei wird eventuelles Systemrauschen durch eine Ziehung pro Partikel aus dessen Wahrscheinlichkeitsverteilung (häufig Gaußverteilung) abgebildet. Die Prognosen aller Partikel bilden dann den neuen Systemzustand.

Der **Observe** Aufruf verarbeitet eine Messung mit Hilfe des Messmodells. Konkret

wird für jedes Partikel, und damit Zustand des es repräsentiert, geprüft wie wahrscheinlich eine solche Messung wäre. Bei einer hohen Wahrscheinlichkeit, wird der Gewichtungswert im Partikel größer. Bei geringer Wahrscheinlichkeit für eine solche Messung sinkt der Gewichtungswert. Nun repräsentieren diese Gewichte der Partikel den neuen Systemzustand.

Das **Resampling** überführt die Information in den Partikel Gewichten wieder in die Partikel Häufung. Es wird normaler weise nach jedem Observe vorgenommen. Dies geschieht durch Ziehen von Partikeln der alten Partikelmenge mit einer Wahrscheinlichkeit die proportional zum Gewicht eines Partikels ist. Somit werden Partikel die ein hohes Gewicht hatten häufiger gezogen als jede, die nur ein geringes Gewicht hatten. Damit "überleben"das Resampling die Partikel dessen Zustand von einer vorgenommenen Messung bestätigt wird. In ... wird ein Problem beim Resampling beschrieben, welches bei unabhängigen Ziehungen auftritt. Es wird ... genannt und kann dazu führen, das Partikel auch mit hohem Gewicht zufällig gar nicht gezogen werden. Dies würde einen Informationsverlust bedeuten, der im Extremfall den wahren Zustand "vergessen"lässt.

[Thrun, 2002]

2.2 Bildverarbeitung

2.2.1 Projektion aus dem Raum auf die Bildebene

3 Lokalisierung mittels Bildverarbeitung

3.0.2 Partikel Filter

Der **Zustandsraum** der Partikel setzt sich aus der Position und der Pose des Roboters zusammen. Da er sich ausschließlich auf einer ebenen Bühne befindet, führt dies zur Reduktion der Freiheitsgrade von sechs auf drei: X, Y und ψ. Wieviele Partikel, und warum? Wie sieht der Partikelraum aus? Wie ist das Messmodell aufgebaut? Besonderheiten im Messmodell: erst an 10 Punken wird ausgewertet, geringer Kontrast führt zu Abwertung. Wie wird initialisiert? Wie ist das Dynamikmodell aufgebaut? Wie ist das Muster aufgebaut?

Wie wird die "geschätzte"Position berechnet?

4 Software

Die Software die im Rahmen dieser Arbeit entwickelt wurde, ist in C++ geschrieben. Als Entwicklungsumgebung wurde Eclipse mit den *C/C++ Development Tools* (CDT) auf einem Linux[1] Betriebssystem verwendet.

4.1 Simulation

Grundlage der Simulation ist das 3D Grafiktoolkit *Open Scene Graph* (OCG)[2]. Damit lässt sich eine 3D Szene in Form eines Graphen aufbauen und mit einem Viewer darstellen. Um den Ablauf kontrollieren zu können, lässt sich die Render-Schleife manuell aufrufen um jeden Frame einzeln berechnen zu lassen. Dies wurde als Simulationsschritt gewählt in dem alle nötigen Berechnungen durchgeführt werden können. Da die Geschwindigkeit mit der die Simulation im manuellen Modus abläuft nicht begrenzt wird, wurde eine Mindestbearbeitungszeit integriert. Denn die Geschwindigkeiten von Objekten in der Simulation, wie dem Roboter, werden durch eine zurückgelegte Strecke pro Simulationsschritt festgelegt. Bei sehr schneller Hardware ergab dies eine zu hohe Bewegungsrate um den Roboter noch manuell steuern zu können. Für automatisierte Simulationsläufe mit festgelegten Fahrprofilen könnte man diese Begrenzung wieder lösen um Zeit zu sparen.

4.1.1 Die Szene

Die Szene in der Simulation ist aus mehreren Komponenten aufgebaut, die im Folgenden näher beschrieben werden. Dabei wird ein Vergleich zu den echten Elementen auf der Bühne gezogen, um zu erläutern wie deren Attribute in der Simulation abgebildet werden können. In Abbildung 4.1 kann man alle Elemente der Szene erkennen.

[1] Ubuntu 12.04 LTS

[2] `http://www.openscenegraph.org/`

Die Grundfläche (1) der Bühne misst 12 x 12 m. Sie wird als einfach weiße Fläche in der Szene dargestellt. Da die Bildverarbeitung nur auf den Oberen Teil des Bildes beschränkt ist, spielen Farbe und Helligkeit keine Rolle bei der Erkennung des Musters im Bild.

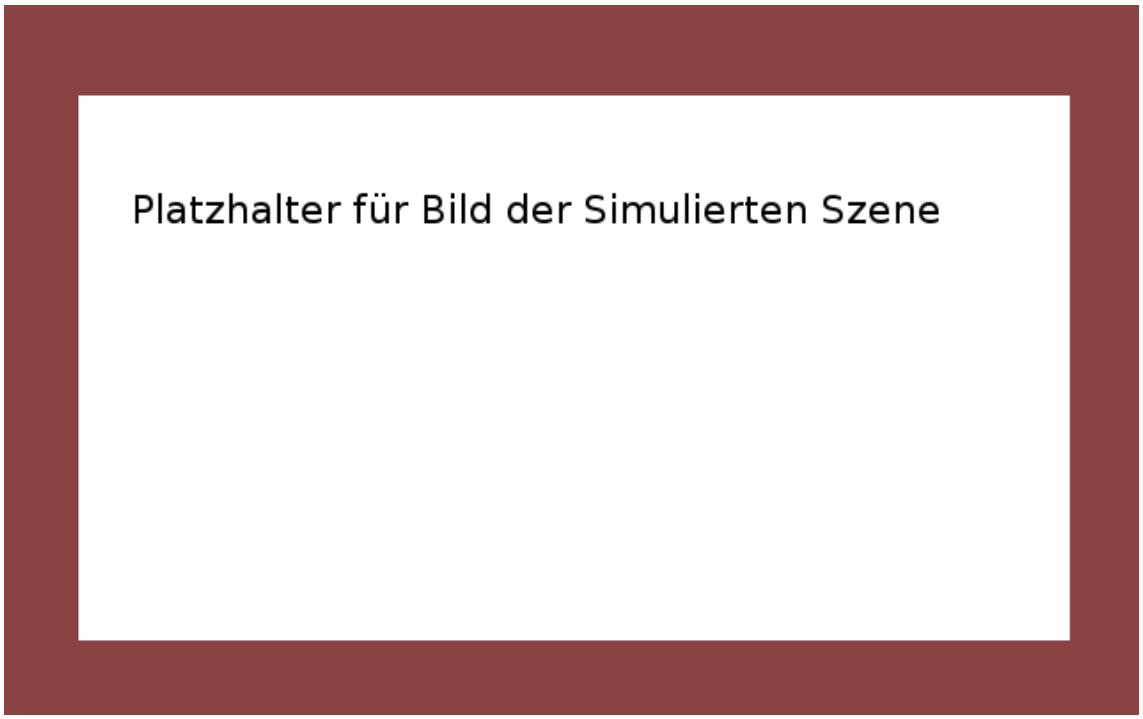

Abbildung 4.1: Bild Simulierte Szene

Die Lichtwände (2,3) sind zu drei Seiten der Grundfläche aufgestellt. Sie messen 3m in der Höhe und 8m in der Länge. Wie bereits in @@@@@@@@ beschrieben soll das Muster in der obersten Zeile der Lichtwände dargestellt werden. Hierzu können verschiedene Texturen geladen werden die das Bitmuster enthalten. In der Realität sind die Lichtwände auf der Bühne sind aus 1 x 1 Meter großen Segmenten zusammen gesetzt, dies wird in der Simulation nicht abgebildet. Es wäre jedoch denkbar, dies in den Aufbau der Texturen einfließen zu lassen. Also leichte Abstands Änderungen der Bits in der Textur. Auf jedem Segment sind 8 x 8 Bits unter gebracht. Das macht eine Kantenlänge von 125 mm bei jedem dieser Pixel und 64 Pixel über die ganze Länge einer Wand. Ein solches Segment besteht aus einer Milchigen Plexiglasplatte, die auf eine Struktur geschraubt wurde die für jedes Pixel ein Leuchtmittel vorsieht. Dessen Helligkeit lässt sich einstellen, wäre aber für das Bitmuster auf die Zustände: dunkel(ausgeschaltet) oder hell(ein, mit größter Helligkeit) einzustellen. Die Plexiglasplatte wird also pro Pixel von Hinten durchleuchtet. Dies führt dazu, das die Helligkeitsverteilung in einem beleuchteten Pixle inhomogen ist. In der Mitte ist die größte Helligkeit, während sie radial nach Außen etwas

abnimmt. Die Ecken der Pixle sind die dunkelsten Stellen. Auf Abbildung 1.2 auf Seite 2 ist dieser Effekt gut zu sehen. Er wird besonders stark, wenn die Leuchtmittel mit niedriger Helligkeit betrieben werden. Bei hoher Helligkeit ist der Effekt noch wahrnehmbar, aber nicht mehr so ausgeprägt. Und noch etwas fällt auf wenn man dieses Bild betrachtet, die Lichtfarbe und Helligkeit variiert leicht von Pixel zu Pixel. All diese Effekte können in der Simulation nur durch verändern der Texturen abgebildet werden.

Der Roboter (4) @@@@@@@@@@@@@@@ Platzhalter @@@@@@@@@@@@ Er lässt sich mit den Tasten W, A, S, D (vorwärts, links, rückwärts, rechts) grob verfahren oder mit einem Pad mit Analogsticks auch präziser steuern. Die Steuerbefehle für den Roboter werden in Geschwindigkeit (gerade aus) und Drehrate interpretiert. Diese führen dann, in jedem Simulationsschritt zu einer Positionsänderung in Richtung der Orientierung und einer anschließenden Änderung der Orientierung um die Drehrate. <——(Implementierung möglicherweise noch ändern???) .

Passanten oder Besucher (5) @@@@@@@@@@@ Platzhalter @@@@@@@@@@@

Partikel (6) werden zur Veranschaulichung und zu debugging Zwecken visualisiert. Sie werden durch rote kleine spitze Dreiecke dargestellt. Der spitze Winkel zeigt dabei die Orientierung an. Sie befinden sich nur auf dem Boden und beeinträchtigen die Bildverarbeitung deshalb nicht.

4.1.2 Messen in der Simulation

Die Simulation soll, neben der Visualisierung, Messungen liefern um den Lokalisationsalgorythmus testen zu können. Anders als bei Messungen an realen Experimenten hat man in der Simulation den Vorteil, alle das Messergebnis beeinflussenden Faktoren unter Kontrolle zu haben. Möchte man also, dass Messungen eine systematische Abweichung aufweisen, muss diese in der Simulation definiert werden. Genau so verhält es sich mit statistischen Abweichungen. Es lassen sich also schnell die Messunsicherheiten der Simulierten Vorgänge anpassen. Beim Debugging und Funktionstest des Lokalisationsalgorythmus wurde die Unsicherheit zum Beispiel zeitweise entfernt.

Messwerte der Inkrementalgeber

Wie beim Robotermodell schon beschrieben, wird in der Simulation dessen X/Y-Koordinate sowie der Winkel zur X-Achse als Repräsentation der Pose verwendet. Aus den Positions- (Δs) und Orientierungsänderungen ($\Delta\psi$), in jedem Simulationsschritt, werden Drehwinkeländerungen ($\Delta\alpha$) der Beiden Räder berechnet:

$$\Delta\alpha_{l/r} = \underbrace{\Delta s \pm \Delta\psi \cdot \frac{L_{achse}}{2}}_{zurckgelegte\ Strecke\ pro\ Rad} \cdot \frac{\gamma}{U_{rad} \cdot g}$$

mit Länge der Achse: L_{achse}, Radumfang: U_{rad}, Getriebeübersetzung: g und Inkrementalgeber Auflösung: γ (pro Umdrehung)

Über alle Schritte akkumuliert, ist der Drehwinkel der Räder ($\alpha_{l/r}$) der wahre Zustand der Odometrie in der Simulation. Aus diesen Winkelstellungen[3] der Räder wird ein Wert für die Inkrementalgeber abgeleitet und auf ganze Inkremente gerundet.

Bilder der Kamera

noch mehr test text zum testen.

4.2 Lokalisation

Die Lokalisation ist als eine Klasse implementiert

[3] Die Winkelstellung der Räder wird nur numerisch simuliert und ist am Modell nicht sichtbar.

5 Versuchsdurchführung

5.1 Theoretische Vorüberlegungen

Ideen zu Fragestellungen die Versuche nötig machen:

- Wie viele Bilder pro Zeit/Stecke sind nötig? Wie genau will man dabei noch sein? Evtl. Bildfrequenz zu Mittlerer Unsicherheit Darstellen?
- Wie wirkt es sich aus, wenn teile des Musters verdeckt sind? Ist eine Lokalisation auch bei ständiger Verdeckung möglich?
- Wie muss das Muster aufgebaut sein? Was für Fehler könne entstehen, wenn ein ungeeignetes Muster verwendet wird?
- Wie gut kommt der Filter mit Systematischen Fehlern zurecht?

5.2 Versuchsaufbau

5.3 Versuchsvorbereitung

5.4 Praktische Versuchsdurchführung

5.5 Versuchsergebnisse

5.6 Diskussion der Ergebnisse

6 Mögliche Anwendung und Ausblick

7 Fazit

A Abkürzungsverzeichnis

CDT *C/C++ Development Tools*

OCG *Open Scene Graph*

BBM *Beobachter der Bediener von Maschinen*

SLAM *Simultaneous Localization and Mapping*

MCL *Monte Carlo Localization*

B Literaturverzeichnis

[Ha et al., 2012] Ha, X. V., Ha, C., and Lee, J. (2012). *Intelligent Computing Technology: 8th International Conference, ICIC 2012, Huangshan, China, July 25-29, 2012. Proceedings: Trajectory Estimation of a Tracked Mobile Robot Using the Sigma-Point Kalman Filter with an IMU and Optical Encoder*, volume 7389 of *Lecture Notes in Computer Science*. Berlin, Heidelberg.

[Hertzberg et al., 2012] Hertzberg, J., Lingemann, K., and Nüchter, A. (2012). *Mobile Roboter: eine Einfuührung aus Sicht der Informatik*. eXamen.press. Springer Vieweg, Berlin [u.a.]. X, 389 S. : zahlr. Ill. und graph. Darst.

[Seco et al., 2009] Seco, F., Jiménez, A. R., Prieto, C., Roa, J., and Koutsou, K. (2009). A survey of mathematical methods for indoor localization. In *Intelligent Signal Processing, 2009. WISP 2009. IEEE International Symposium on*, pages 9–14. IEEE.

[Thrun, 2002] Thrun, S. (2002). Particle filters in robotics. In *Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence*, UAI'02, pages 511–518, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

[Thrun et al., 2006] Thrun, S., Fox, D., and Burgard, W. (2006). *Probabilistic robotics*. Intelligent robotics and autonomous agents. MIT Press, Cambridge, Mass. [u.a.]. XX, 647 S. : Ill., graph. Darst.