Maximilian Eibl, Martin Gaedke. (Hrsg.): INFORMATIK 2017,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2017 503

Hardwaregestützte Positionsschätzung mit Bayes'schen Filtern auf Basis 3-dimensionaler Umgebungsmodelle für den Innenbereich

Christian Schott, Daniel Froß, Marko Rößler und Ulrich Heinkel[1]

Abstract: Die Lage und Position von mobilen Geräten spielt eine zunehmend wichtigere Rolle bei Anwendungen die sowohl im Konsumbereich als auch in Produktion und Logistik liegen. Im Außenbereich bilden globale Positionierungssysteme auf Basis geostationärer Satelliten die Datengrundlage, welche durch weitere Sensorik verfeinert wird. Im Innen- und Nahbereich sind Signallaufzeitmessungen von Radio- oder Radarwellen mit nachgelagerter Triangulation eine wesentliche Grundlage für eine grobe Positionierung und Stand der Technik. Aufgrund der gegebenen Unsicherheit dieser Messverfahren bedarf es robuster Auswertealgorithmen für eine zuverlässige Positionierung, beispielsweise mittels Bayes'scher Filter. Der vorliegende Beitrag befasst sich mit der Erweiterung einer hardware-orientierten Umsetzung eines Partikelfilters, die es erlaubt apriori bekanntes Wissen aus einem 3-dimensionalen Umgebungsmodell mit den Entfernungsmessdaten in Echtzeit zu fusionieren. Vorgestellt wird eine Hardware/Software-Systemarchitektur für einen effektiven Zugriff auf die Modelldaten sowie deren Auswertung innerhalb einer Pipeline-Struktur des Partikelfilters auf Register-Transfer-Ebene. Die prototypenhafte Implementierung erfolgte für einen FPGA-Schaltkreis der ZYNQ-Familie.

Keywords: FPGA, Partikelfilter, 3D-Umgebungsmodell, Indoor-Lokalisierung

1 Einleitung

Standort- und Lageinformationen spielen im Zusammenhang mit drahtlosen mobilen Geräten und Sensoren eine zunehmend wichtige Rolle. Vielfältige praktische Anwendungen benötigen heute Positionsinformationen und Sensordaten bekommen erst im Zusammenhang mit dem Ort der Erhebung einen auswertbaren Kontext. Im Allgemeinen werden zur Lokalisierung im globalen Kontext spezifische Hardwarekomponenten, z.B. GPS-Empfänger, zur Auswertung von Signal-Laufzeitinformationen zu geostationären Satelliten eingesetzt. Deren Verwendung ist weitgehend auf den Außenbereich beschränkt und die Qualität sinkt in bebauten innerstädtischen Bereichen. Um diese Beschränkung aufzuheben bedarf es weiterer Ansätze.

Zum einen lassen sich Radiosignale von funkbasierten Kommunikationssystemen in ähnlicher Weise zur Informationsgewinnung in Form von Messdaten mit Bezug zur Position

[1] alle: Technische Universität Chemnitz, Fakultät für Informations- und Elekrotechnik, Professur Schaltkreis- und Systementwurf, Reichenhainer Straße 70, 09126 Chemnitz, Deutschland, {*christian.schott*, *daniel.fross*, *marko.roessler*, *ulrich.heinkel*} @etit.tu-chemnitz.de

doi:10.18420/in2017_46

nutzen. Dies gilt sowohl für zellulare Mobilfunktechniken als auch für nicht-zellulare Fern- und Nahbereichsnetze. Grundlage sind dabei Signalstärke-, Laufzeit- oder Winkelmessungen der hochfrequenten Funksignale zwischen kommunizierenden Teilnehmern und stationären Referenzpunkten. Sensoren auf Basis von Optik, Radar oder Schall lassen sich ebenso zur Distanzmessung zu Referenzpunkten mit spezifischen Vor- und Nachteilen einsetzen [Ma12]. Zum anderen helfen bei der Positionierung Informationen die direkt aus bekanntem Wissen über die Eigenschaften des zu lokalisierenden Objektes und dessen Umgebung abgeleitet werden können. Denkbar sind hier Bewegungsmuster oder Raum- und Kartendaten, die zur Modellierung genutzt und schließlich mit den Messdaten fusioniert in einer Position bzw. Lokalisierung resultieren.

Aufgrund der gegebenen Unsicherheit der Messverfahren, die zum Beispiel durch Signalausbreitungseffekte begründet sind und der begrenzten Genauigkeit der erweiterten Datenquellen, bedarf es robuster Auswertealgorithmen zur Positionsermittlung. Eine verbreitete und allgemein anerkannte Methode den Systemzustand (hier Position oder Bewegung) auf Grundlage von unscharfen oder verrauschten Beobachtungen (hier Messungen) zu bestimmen, sind Bayes'sche Filter, bei denen sowohl der Zustand als auch die Beobachtungen als Funktion der Wahrscheinlichkeit abgebildet werden.

Kalman Filter sind eine effiziente und populäre Implementierung des Bayes'schen Filter Algorithmus. Zustand und Beobachtung werden dabei als unimodale Gauß-Verteilung modelliert und das Filter-Update lässt sich in Form einer geschlossenen Aktualisierungsformel für den Mittelwert und die Kovarianzen abbilden. Nachteilig ist die damit einhergehende Einschränkung auf Gaußsche-Fehlerverteilungen und Systeme mit linearen Zustandsübergängen und Beobachtungsmodellen. Beliebige Verteilungen lassen sich mit einer anderen Implementierung des Bayes'schen Filters, den Partikelfiltern behandeln, bei dem zur näherungshaften Bestimmung des Systemzustandes ein Satz zufälliger Stichproben (Partikel) genutzt wird. Die Genauigkeit der Näherung ist dabei von einer größtmöglichen Anzahl von Partikeln abhängig. Der daraus resultierende Berechnungs- und Speicherbedarf ist erheblich und im Zusammenhang mit einer Echtzeitberechnung der Position von bewegten Objekten für eine Berechnung in eingebetteten Systemen oder Mikrocontrollern ungeeignet. In diesen Fällen ist es erforderlich die berechnungsintensiven Anteile in eine dedizierte Beschleunigungshardware zur parallelisierten Abarbeitung auszulagern. Eine grundlegende Partikelfilter-Implementierung als digitale Schaltung für einen FPGA wurde bereits in [Fr10] gezeigt.

Im vorliegenden Beitrag wird diese Hardware-Implementierung um ein 3-dimensionales Umgebungsmodell erweitert, auf dessen Basis die Schätzung der Position ergänzt und verbessert werden kann. Dabei wird unter Berücksichtigung von Belegtheitsinformationen der Systemzustand validiert und angepasst. Das Design wurde in VHDL auf RT-Ebene implementiert und auf ein Hardware/Software-System in Form der ZYNQ-Architektur abgebildet und synthetisiert. Die Implementierung ist über den AXI-Systembus an die Kontroll-Software innerhalb des FPGA-Schaltkreises angebunden.

Der Beitrag gliedert sich wie folgt. Nach einer kurzen Einführung in die generelle Struktur des Partikelfilter-Algorithmus in Abschnitt 2, wird im darauf folgenden Abschnitt 3 zunächst auf verschiedene 3-dimensionale Umgebungsmodelle eingegangen und letztlich gezeigt, wie diese in den Schätzprozess des Partikelfilters integriert werden können. Anschließend wird in Abschnitt 4 die eigentliche Erweiterung der bereits bestehenden Hardware-Implementierung im Detail vorgestellt. Abschnitt 5 erläutert die Testumgebung mit dem Referenzmodell, welche zur Evaluierung der Umsetzung erstellt wurde. Die Resultate der erweiterten Positionsschätzung sowie der Bedarf an FPGA Ressourcen werden dann in Abschnitt 6 dargestellt und diskutiert, gefolgt von einer kurzen abschließenden Zusammenfassung dieses Beitrags.

2 Partikelfilter zur Positionsschätzung

Partikelfilter stellen eine Implementierungsvariante von Bayes-Filtern dar, bei der die Aufenthaltswahrscheinlichkeit des zu lokalisierenden Objektes durch einen Satz von Zufallswerten, sog. Partikel, repräsentiert wird. Jedes Partikel entspricht dabei einer hypothetischen Position des Objektes. Je höher die Partikeldichte, desto höher die Aufenthaltswahrscheinlichkeit. Der Vorteil dieser Implementierungsvariante liegt in der Abbildbarkeit beliebiger, d.h. auch nicht Gauß-verteilter oder multimodaler Verteilungen. Dem gegenüber steht ein erhöhter Berechnungsaufwand, der sich jedoch durch Möglichkeiten der Hardware-Beschleunigung mittels FPGA oder DSP optimieren lässt. Detaillierte Betrachtungen zu Bayes-Filtern und den verschiedenen Implementierungsvarianten finden sich in [TBF05]. Für ein Filter-Update sind folgende Schritte auszuführen:

Zustandsvorhersage In diesem Schritt erfolgt die Vorhersage der neuen Partikelpositionen für den aktuell betrachteten Zeitpunkt unter Einsatz eines anwendungsspezifischen Bewegungsmodells. Dieses Modell beschreibt die angenommene Zufallsbewegung des zu ortenden Objektes. Ausgehend von der alten Position wird damit für jedes Partikel eine neue hypothetische Position errechnet, die im weiteren Filterverlauf unter Auswertung von Sensordaten validiert oder verworfen wird.
In der vorliegenden Implementierung wird eine ungerichtete Zufallsbewegung des zu ortenden Objektes angenommen, bei der sich die neue Partikelposition aus der alten ergibt, die für jede der drei Raumachsen X,Y und Z mit Zufallswerten aus drei voneinander unabhängigen Gleichverteilungen mit einstellbarem Wertebereich beaufschlagt wird.

Gewichtsberechnung Ausgehend von dem vorhergesagten Partikelsatz erfolgt in diesem Schritt die Berechnung eines Gewichts für jedes Partikel. Dieses Gewicht repräsentiert die Wahrscheinlichkeit, dass eine eingehende Messung zur vorhergesagten Partikelposition passt. Die Gewichtsberechnung erfolgt unter Verwendung eines sensorspezifischen Messmodells. In der vorliegenden Implementierung wird ein funkbasiertes Laufzeitmesssystem eingesetzt,

das Entfernungsinformationen zwischen einem zu ortenden Mobilknoten und mehreren ortsbekannten Infrastrukturknoten (Anker) liefert. Mindestens vier Ankerknoten sind für eine eindeutige Positionsbestimmung im 3-dimensionalen Raum nötig. Über Signallaufzeitmessungen HW-bestätigter Funknachrichten zwischen dem Mobil- und den Ankerknoten wird die jeweilige Entfernung zum Ankerknoten bestimmt. Das eingesetzte Messmodell berechnet für jedes Partikel dessen euklidische Distanz zum Ankerknoten und liefert ein Gewicht in Abhängigkeit der Differenz zwischen gemessener und erwarteter euklidischer Entfernung. Aufgrund des Mehrwegeempfangs der Funknachrichten und der begrenzten Auflösung des Messsystems treten positive Distanzfehler häufiger auf als negative. Dieser Umstand wird in der Berechnung durch eine stärkere Gewichtung längerer Messungen berücksichtigt.

Resampling Im letzten Schritt erfolgt ein Resampling des Partikelsatzes in Abhängigkeit der einzelnen Partikelgewichte. Dabei werden Partikel, deren Gewicht das durchschnittliche Gewicht aller Partikel übersteigt tendenziell vervielfältigt, während Partikel mit kleinerem Gewicht wegfallen. Auf diese Weise wird sichergestellt, dass sich im Laufe des Schätzprozesses die Partikel an den durch die Messung validierten Positionen konzentrieren. In der Implementierung wird ein sogenannter Low-Variance Resampler eingesetzt, der sich durch seine Ressourcen- und Laufzeiteffizienz auszeichnet.

Eine detaillierte Beschreibung der oben genannten Filterabläufe ist in [Fr10] zu finden.

3 Erweiterung des Partikelfilter durch Umgebungsmodell

In Abbildung 1 ist die hardware-orientierte Umsetzung des Partikelfilter in Form einer Pipeline gezeigt. Der *Zustand* eines Partikel ist durch die Werte X,Y und Z als Position im Raum gegeben. Im Block *Occupancy Check (OCC)* wird der Zustand eines Partikels gegen ein Umgebungsmodell geprüft und validiert. Die Form der Umgebungsmodellierung hat dabei wesentlichen Einfluss auf die Zugriffsgeschwindigkeit zu den Modelldaten und deren Umfang bzw. Größe. Traditionell wird zwischen geometrischen und wahrscheinlichkeitsbasierten Ansätzen unterschieden.

Beim geometrischen Ansatz werden die Modelldaten als *Belegtheitsinformation* interpretiert, die über geometrischen Koordinaten der von den Objekten im Raum belegten Bereiche in einer definierten Auflösung gespeichert werden. Die Daten liegen als Punktwolke (engl. *Point Cloud*) vor und können beispielsweise direkt über Laserscanner ermittelt werden. Entsprechende Modelle finden vor allem in der Robotertechnik bzw. Robotics Anwendung und spielen dort bei der (Selbst-)Lokalisierung oder dem Motion Planning des Roboters eine wichtige Rolle. Die Vorteile dieses Ansatzes liegen in der hohen Genauigkeit, die bei der Modellierung der Umgebung in Abhängigkeit der Sensorauflösung erreicht werden kann, sowie des direkten Zugriffs mittels geometrischer Koordinaten. Nachteilig ist der hohe Speicherbedarf, der sich proportional zur Anzahl zu speichernder Punkte ergibt.

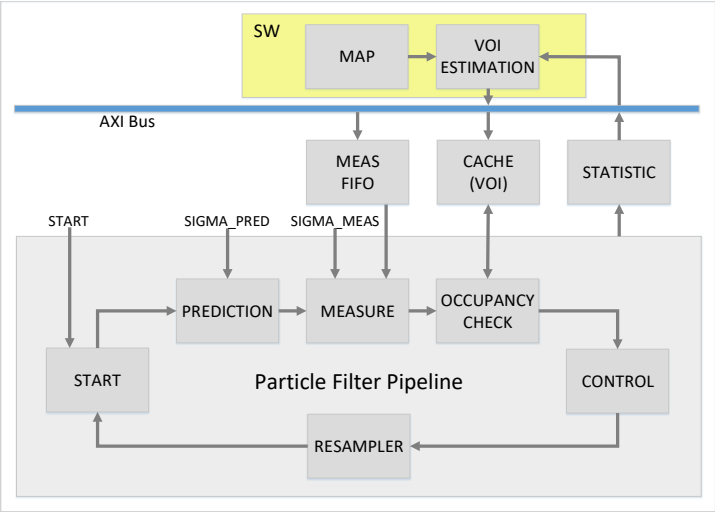


Abb. 1: Partikelfilter mit Anbindung an das Umgebungsmodell

Bei probabilistischen Ansätzen wird der Raum in Zellen unterteilt, denen die Wahrscheinlichkeit einer Belegung durch ein Objekt zugeordnet ist. Die sogenannten *Evidence* oder *Occupancy Grids* wurden in [ME85] für die 2-dimensionale Darstellung einer Umgebung vorgestellt und später in [RTJ89] und [MM94] für die dritte Dimension erweitert. Aufgrund der Rasterisierung des Raumes ist die Genauigkeit der Occupancy Grids im Vergleich zur Punktwolke geringer, dadurch wird jedoch eine verbesserte Speichereffizienz erreicht. Eine weitere Effizienzsteigerung wurde in [Me82] durch die Nutzung der *Octree* Datenstruktur vorgestellt. Dabei ist es möglich, Bereiche mit unterschiedlichen Auflösungen im gleichen Umgebungsmodell abzubilden. Die Nachteile dieses Ansatzes liegen in der höheren Komplexität des Zugriffsverfahrens, wodurch größere Zugriffszeiten, die zusätzlich nicht äquivalent bzw. vorhersagbar sind, zu erwarten sind.

Für die Integration der Belegtheitsprüfung in die hardware-orientierte Pipeline des Schätzprozesses ist ein schneller und konstanter Zugriff auf die Modelldaten entscheidend. Die zur Adressierung dieser Daten notwendigen Berechnungen bzw. Transformationen müssen daher für in Hardware günstig zu realisierende Vorschriften abgebildet werden. Der geometrische Ansatz nach dem Modell der Punktwolke erfüllt diese Anforderungen. Die damit verbundenen hohen Speicheranforderungen erlauben es allerdings kaum, ein vollständiges Umgebungsmodell im Block-RAM nahe der Pipeline mit direkten Zugriff vorzuhalten. Daher wird im Weiteren ein hierarchischer Ansatz auf Basis einer Hardware/Software-Systemarchitektur verfolgt und untersucht, bei dem die Modelldaten der kompletten Umgebung im Fest- oder Hauptspeicher einer Prozessorumgebung liegen und nur ein relevanter Teil davon, beispielsweise auch mittels Prozessierung in einem Rendering Prozess, durch Software extrahiert und einem Zwischenspeicher (Cache) der Partikelfilter-Pipeline

zugeführt wird. Dieser Ausschnitt entspricht der unmittelbaren Umgebung der geschätzten Position, in der sich die Partikel mit großer Wahrscheinlichkeit befinden. Dieser Ausschnitt wird im Weiteren *Volume of Interest (VOI)* genannt (vgl. Abbildung 1).

Wie bereits in Abschnitt 2 gesehen, lässt sich der Partikelfilter in die Phasen Zustandsvorhersage (engl. Prediction), Gewichtsberechnung (oder Measurement Update) und Resampling einteilen. Es stellt sich die Frage, in welcher Stufe der Pipeline es günstig ist die Modellprüfung zu integrieren und ein Partikel entsprechend dem Ergebnis der Prüfung zu beeinflussen. Nach dem Prediction-Schritt ist ein Partikel ausschließlich durch dessen Zustand, der Position im Raum, charakterisiert, womit die Prüfung gegen das Umgebungsmodell damit erfolgen kann. Liegt das Partikel in einem Objekt, könnte das Partikel in einen freien, also nicht durch ein Objekt belegten Ort verschoben werden. Dazu muss ein freier Bereich identifiziert bzw. gesucht und anhand einer zu definierenden Regel ausgewählt werden.

Alternativ könnte die Prüfung nach der Measurement-Stufe erfolgen. Neben dem Zustand ist jedes Partikel mit einem Gewicht gekennzeichnet, dessen Einfluss die gesamte Positionsschätzung repräsentiert. Bei negativer Modellprüfung kann das Gewicht entsprechend auf nahe Null reduziert werden. Im nachfolgenden Resampling Schritt werden alle Partikel mit (sehr) kleinem Gewicht aussortiert bzw. verworfen und tragen so nicht mehr zur Positionsschätzung bei. Die Gewichte der Partikel mit positiver Modellprüfung bleiben unberührt. Letztendlich werden sich die Partikel nur in freien Gebieten ansammeln und somit zur Erhöhung der Genauigkeit der Positionsschätzung beitragen. Diese Variante wurde für den vorliegenden Beitrag umgesetzt und im Block Occupancy Check in die Pipeline integriert.

4 Implementierung

Die Umsetzung des beschriebenen Ansatzes erfolgte auf einem Xilinx-FPGA der ZYNQ-Klasse, da hier bereits eine geeignete Plattform-Architektur aus programmierbarer Schaltkreisstruktur für die Hardware-Implementierung und dedizierten ARM-Prozessoren zur Realisierung einer Software-Umgebung integriert ist. Die Partikelfilter-Pipeline verfügt über eine AXI-Schnittstelle, an die die Busstruktur der Plattform angebunden ist. Eine entsprechende Register-Schnittstelle dient zur Steuerung der Pipeline. Ein- und Ausgabe erfolgt über Speicherschnittstellen, die als FIFO umgesetzt sind. Abbildung 1 zeigt den Occupancy Check, der über einen weiteren AXI-Port zur Speicherung und Auswahl des VOI verfügt.

Der in diesem Beitrag umgesetzte Partikelfilter erzeugt insgesamt 8192 Partikel, wobei jedes davon durch 14 Bit pro Dimension und somit durch einen insgesamt 42 Bit breiten Zustands-Vektor repräsentiert ist. Die Partikel befinden sich in einem kubischen Berechnungsraum mit maximal 16384 Längeneinheiten pro Dimension. Sie durchlaufen einzeln nacheinander die Pipeline-Stufen, so auch den in Abbildung 2 dargestellten Block zur Modellprüfung.

Das Extrahieren des VOI durch die Software und das anschließende Füllen des Block-RAM dauert länger als ein Zyklus des Partikelfilters. Die vorliegende Implementierung

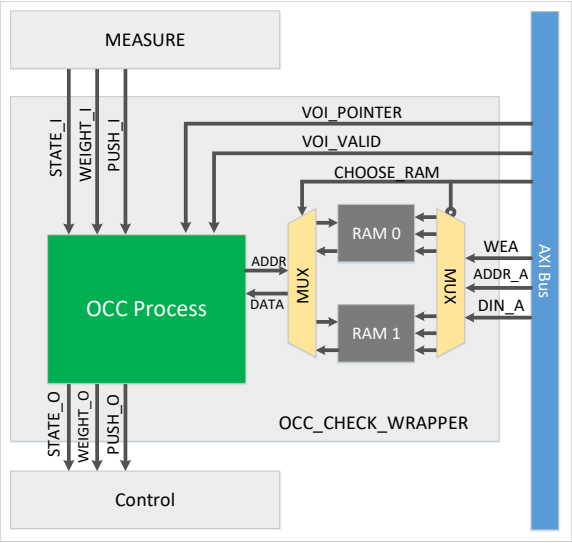


Abb. 2: Occupancy Check

sieht daher zwei VOI-Speicher vor, um das Laden des Speichers durch die Software und den Zugriff des OCC voneinander zu entkoppeln. So ist die Konsistenz der VOI-Informationen sicher gestellt und ein ununterbrochener Betrieb der Pipeline und damit die Echtzeitfähigkeit der Implementierung möglich. Über das Signal *CHOOSE_RAM* wird die entsprechende Instanz ausgewählt und das Adress- sowie Datensignal auf den gewünschten Block-RAM gemultiplext. Die Speicher sind als Dual Port RAM Instanzen innerhalb der programmierbaren Logik ausgeführt, wobei von Seiten des AXI Bus mit 128 Bit Datenbreite geschrieben und durch den OCC Daten mit 8 Bit gelesen werden. Die Größe der Speicher ergibt sich aus den Dimensionen des VOI und der Tiefe der Belegtheitsinformation im Modell. Wird die Belegung eines Punktes im Raum durch genau ein Bit (0 ... unbelegt, 1 ... belegt) repräsentiert, ergibt sich für ein Volumen von 128 x 128 x 128 Längeneinheiten^3 ein Speicherbedarf von 2048 kBit. Der Zugriff auf die Belegtheitsinformation für ein Partikel erfolgt über dessen Zustand, der eine Position im gesamten Berechnungsraum entspricht. Da das VOI nur einen Ausschnitt dieses Raumes darstellt und auf einem eindimensionalen Speicher abgelegt ist, bedarf es einer Berechnungsvorschrift für die Adressierung der Modelldaten.

Abbildung 3 zeigt die Abbildungsvorschrift beispielhaft. Zunächst muss für das Partikel der dazugehörige Punkt im VOI-Koordinatensystem bestimmt werden. Das Signal *VOI_POINTER* gibt die Lage des VOI im Berechnungsraum des Partikelfilter an und zeigt auf Punkt (0,0,0) des VOI-Kubus. Dieser Zeiger ist vom Zustand des Partikels in X, Y und Z Richtung abzuziehen. Für die Speicheradresse eines Byte teilen sich entsprechend 8 Punkte der X-Achse eine Adresse. Entsprechend werden die niederwertigen Bits der

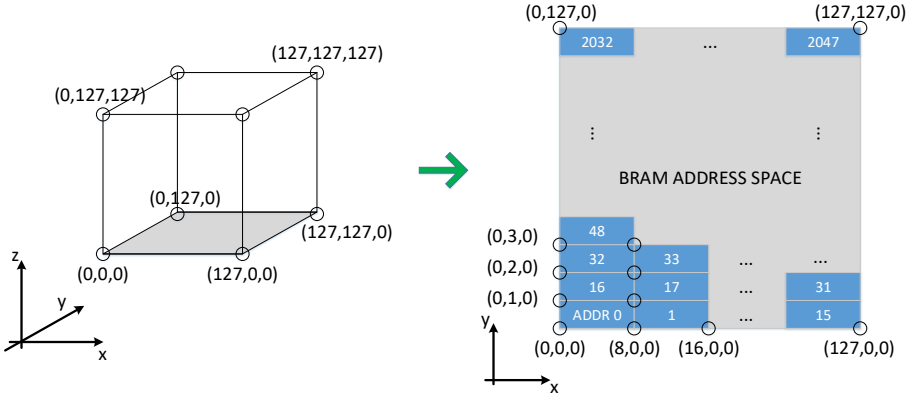


Abb. 3: Block-RAM Address Mapping

Adresse aus den um drei Bit nach rechts verschobenen X-Achsenwert bestimmt. Die Y und Z Achswerte im VOI-Koordinatensystem bilden als Offset den höherwertigen Teil der Adresse. Das Signal *VOI_POINTER* wird in Software nach dem Extrahieren des Kartenausschnitts gebildet und über den AXI Bus dem Schätzer zugeführt. Nach korrektem Schreiben der Belegtheitsinformationen in den Block-RAM, wird mittels *VOI_VALID* der Occupancy Check aktiviert.

Der Block *OCC Process* ist für die eigentliche Belegtheitsprüfung der Partikel zuständig und ist als Zustandsautomat mit zwei Zuständen realisiert. In Zustand 1 wird die Position des anliegenden Partikels in die entsprechende Block-RAM Adresse umgewandelt und diese an den RAM angelegt. Da der RAM Zugriff einen Taktzyklus benötigt und mit jedem neuen Takt bereits das nächste Partikel am Eingang des OCC anliegen kann, wird das aktuelle Partikel sowie dessen Gewicht für den nächsten Zustand des Automaten in einem Register zwischengespeichert. Mit dem nächsten Partikel welches durch ein aktiv High von *PUSH_I* angezeigt wird, wechselt der Automat in Zustand 2. Erst hier wird geprüft, ob das zwischengespeicherte Partikel im VOI liegt, da nur für dieses die RAM Adresse und letztendlich die ausgelesene Belegtheitsinformation korrekt sind. Bei positiver Belegtheit wird das Gewicht des Partikels angepasst, wobei es ansonsten unverändert bleibt. Durch die Realisierung als Zustandsautomat wird ein durch den Occupancy Check entstehender konstanter Timing Overhead gewährleistet. Am Ende von Zustand 2 wird das Partikel und dessen ggf. neues Gewicht ausgegeben und an den *Control* Block zur Vorbereitung auf die Resampling Phase weitergeleitet.

5 High-Level Referenzmodell und Testumgebung

Zur Evaluation der unterschiedlichen Entwurfsmöglichkeiten und abschließenden Verifikation der Implementierung wurde ein Referenzmodell des Partikelfilters sowie eine synthetische Testumgebung in MATLAB erstellt. Deren Struktur ist in Abbildung 4 dargestellt. Sämtliche Berechnungen erfolgen hier in Fließkomma-Arithmetik, sodass die Auswirkungen der ungenaueren Festkomma-Arithmetik der Hardware-Implementierung festgestellt und bewertet werden können. Die Testumgebung erzeugt entsprechende Referenzpunkte als reale Position eines Objektes bzw. des Partikelfilter-Referenzmodells, sowie Messungen von Ankerpunkten zu diesen Referenzpunkten als Stimuli für die Hardware-Implementierung. Dabei werden die Messungen in einem zeitlichen Bezug zu einer synthetischen Simulationszeit erzeugt, sodass in diesem Zusammenhang auch der Zugriff auf das Funkmedium modelliert ist. Die Auswahl eines Ankerpunktes zu einem bestimmten Simulationszeitpunkt erfolgt iterativ und die Messungen werden mit einem entsprechenden Messfehler beaufschlagt. Die VHDL-Testbench liest die Stimuli auf Basis einer Textdatei ein und führt diese der RTL-Simulation der Hardware-Implementierung des Partikelfilters zu. Die Analyse der Simulationsergebnisse erfolgt post-simulativ in der MATLAB Testumgebung.

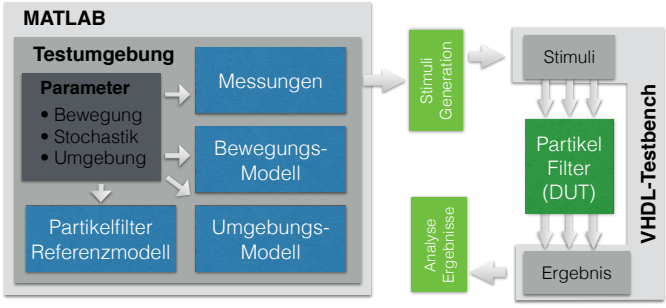
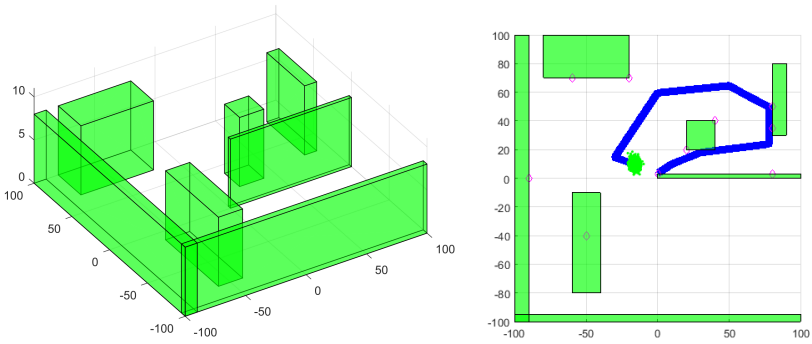


Abb. 4: Synthetische Test- und Verifikationsumgebung

Das Referenzmodell bildet auf Basis algorithmischer Verhaltensbeschreibung neben der in Abschnitt 2 beschriebenen Implementierung auch die VOI-basierte Umgebungsmodellprüfung sowie das Verhalten der Partikel an den Grenzen des Berechnungsraumes nach. Parametrierbar sind neben der Partikelanzahl auch die stochastischen Aspekte wie Wahrscheinlichkeitsverteilungen und Sigma-Werte. In der Testumgebung wurde ein Umgebungsmodell zur Nachbildung eines Untersuchungsraumes realisiert. Parametrierbar sind die Größe des Raumes, die Größe und Lage von quader- und kugelförmigen Objekten sowie die Anzahl und Lage von Anker- bzw. Referenzpunkten. Weiterhin erzeugt ein Bewegungsmodell eine synthetische Bewegung eines zu ortenden Objektes auf der Basis zu passierender Wegpunkte mit entsprechenden Geschwindigkeits- und Beschleunigungswerten. Abbildung 5 zeigt beispielhaft einen Umgebungsraum und eine Bewegung innerhalb der Objekte sowie vorhandene Anker.



(a) Raum mit Objekten

(b) Draufsicht mit Bewegung, Referenzpunkten und Partikelwolke

Abb. 5: Modellierung einer synthetischen Testumgebung

6 Ergebnisse

Zur Evaluierung der Integration des Umgebungsmodells in den Schätzprozess des Partikelfilters wurden verschiedene Testfälle entwickelt und zunächst im High-Level Referenzmodell geprüft. Ein Testfall wird dabei durch den bestimmten Bewegungsablauf eines Objektes charakterisiert. Das Referenzmodell erzeugt je nach Art der Bewegung Punkte, welche das simulierte Objekt durchschreitet. Anhand eines Messmodells werden fehlerhafte Messungen zur simulierten Position erzeugt und dem Partikelfilter zugeführt. In der Evaluierung wird dann die Wurzel des mittleren quadratischen Fehlers (engl. Root-Mean-Square-Error, RMSE) aus der tatsächlichen Position des Objektes mit der Vorhersage des Schätzers gebildet und entsprechend der Art der Bewegung für den Fall ohne Umgebungsinformationen mit dem Partikelfilter mit Belegtheitsprüfung verglichen. Die Auswertung des High-Level Modells zeigte bereits den positiven Einfluss des Umgebungsmodells durch eine Verringerung des Positionsfehlers. Da sich die Softwareanbindung noch in Entwicklung befindet, wurde zunächst der Testfall eines statischen Objektes in der Testbench der Hardware-Implementierung untersucht und dessen Resultate mit denen aus dem High-Level Modell verglichen. Hierbei ist die VOI zur Hälfte mit einem Objekt belegt. Tabelle 1 fasst diese zusammen, wobei hier exemplarisch Meter als Längeneinheit genommen wurde. Mit Belegtheitsinformationen

	High-Level Modell	Hardware Implementierung
ohne Umgebungsmodell	0,3892 m	0,3823 m
mit Umgebungsmodell	0,3045 m	0,2885 m
Differenz	-0,0847 m	-0,0938 m

Tab. 1: Vergleich des RMSE zwischen High-Level Modell und Hardware-Implementierung

konnte der Fehler der Positionsschätzung entsprechend verringert werden. Die Resultate sind für beide Modelle vergleichbar. In der Hardware-Implementierung ist die Genauigkeitsverbesserung sogar leicht größer als im High-Level Modell. Dies lässt sich mit der

unterschiedlichen Berechnung der Zufallszahlen in beiden Modellen und somit der vorhergesagten Position der Partikel im Prediction Schritt erklären. Die im High-Level Modell gewählten Parameter zur Zufallszahlen-Generierung lassen sich nicht eins-zu-eins für den Hardware Partikelfilter übertragen. Eine weitere Untersuchung dieses Verhalten, sowie ein Nachbilden der Hardware Zufallszahlen-Generierung im High-Level Modell ist Aufgabe zukünftiger Betrachtungen, ebenso wie das Testen unterschiedlicher Bewegungsmodelle sobald die Softwareanbindung fertig gestellt wurde.

Das Design wurde zunächst für den ZYNQ 7020 SoC, welcher auf dem Zedboard zu finden ist, als Ziel-Plattform synthetisiert. Da dessen Ressourcen für die Hardware-Implementierung mit Kartenanbindung nicht ausreichen, wurde letztendlich der ZYNQ 7045, der er u.a. im ZC706 Evaluation Kit von Xilinx Verwendung findet, zur Implementierung genutzt. In

ZYNQ 7045	LUT	LUTRAM	FlipFlops	BRAM	DSP
ohne Occ. Check	5255	114	8350	47,5	38
mit Occ. Check	5955	114	8695	175,5	38
verfügbar	218600	70400	437200	545	900

Tab. 2: Vergleich der Hardware Ressourcen für den ZYNQ 7045 SoC

Tabelle 2 wird der Ressourcenbedarf des ursprünglichen Partikelfilters mit dem der Erweiterung verglichen. Die Integration der Belegtheitsprüfung im Schätzprozess führten zu einem Anstieg der genutzten Look-Up-Tables (LUT) um rund 13,3 % und der FlipFlops um 4,13 %. Der größte Zuwachs ist jedoch durch das Vorhalten eines Teils des Umgebungsmodells im Block RAM des FPGA begründet und beträgt rund 269,5 %. Der Bedarf an zusätzlichen FlipFlop Ressourcen übersteigt die Kapazitäten des Zedboard-ZYNQ um 25,4 %, wodurch ein größerer SoC, hier der Xilinx ZYNQ 7045, gewählt wurde. Generell hängt die Größe des VOI vom Anwendungsfall und dessen entsprechende Anforderungen an eine bestimmte Auflösung des Umgebungsmodells ab. Die in diesem Beitrag gewählte Größe von 128 x 128 x 128 Längeneinheiten^3 wurde für eine Genauigkeit von 1 Zentimeter konzipiert, wodurch ein Kubus mit 1,28 m Kantenlänge abgebildet wird. Wenn auf das Punktwolkenmodell verzichtet werden kann und stattdessen ein Voxel-Grid genutzt wird, lässt sich bei gleichzeitiger Verringerung der Auflösung des Modells der Bedarf an Block RAM Ressourcen signifikant verringern. So ist es zum Beispiel möglich, ein VOI mit 6,40 m Kantenlänge und einer Auflösung von 10 Zentimeter in nur 256 kBit zu speichern. Im Vergleich zu vorher ist das nur ein Achtel an Speicherplatz. Diese Variante ließe sich problemlos mit dem ZYNQ des Zedboards realisieren. An diesem Vergleich ist die Ineffizienz des Punktwolkenmodells erkennbar, weshalb in weitergehenden Untersuchungen die Realisierung mittels anderer Umgebungsmodelle betrachtet werden soll.

Die Frequenz der Hardware-Implementierung wurde im Beispiel auf 50 MHz gesetzt. Ein Zyklus des Partikelfilter benötigt 16485 Takte, wobei der Overhead zur vorherigen Realisierung nur 2 Takte beträgt. Bei dieser Frequenz führt dies zu einer Dauer von rund 0,33 ms pro Zyklus und einem theoretischen Maximum von ca. 3000 Partikelfilter Zyklen pro Sekunde. Ausgehend von der Bewegung eines Menschen mit einer angenommenen Geschwindigkeit

von 1 ms^{-1} genügt jedoch bereits eine Frequenz von rund 2 MHz um dessen Position auf den Zentimeter genau zu verfolgen. Ein Update des partiellen Umgebungsmodells im Block-RAM ist bei dieser Geschwindigkeit und im Hinblick auf die VOI-Größe in jedem Prozessierungsschritt möglich. Da bereits eine relativ geringe Frequenz zur Verfolgung eines sich bewegenden Objektes ausreicht, ist der Vorteil dieser Implementierung im Vergleich zu einer reinen Softwarelösung vor allem in deren Energieeffizienz zu sehen.

7 Zusammenfassung

In diesem Beitrag wurde ein hardware-basierter Partikelfilter zur Positionsbestimmung um ein 3-dimensionales Umgebungsmodell erweitert. Durch Hinzufügen von bereits bekannten Informationen in den Schätzprozess, lässt sich der Suchraum einschränken und so die Qualität der Positionsbestimmung erhöhen. In dieser Arbeit wurde ein Punktwolken-Modell zur Beschreibung einer 3-dimensionalen Umgebung gewählt, wobei Teilbereiche als Volume of Interests aus der kompletten Karte mittels Software extrahiert und der Hardware-Implementierung zugeführt werden. Die Ergebnisse zeigen eine verbesserte Genauigkeit der Positionsbestimmung durch Integration der Umgebungsinformationen in den Schätzprozess des Partikelfilters. Der Hardware Overhead ist dabei vor allem auf das Vorhalten eines Teils der Umgebung im Block-RAM begründet. Die Größe des Teilbereichs in Zusammenhang mit der geforderten Auflösung des Umgebungsmodells definiert den FPGA zur Implementierung des erweiterten Partikelfilter. Die vorgestellte Lösung ermöglicht es die Bewegung einer Person in Echtzeit zu verfolgen und ist durch Realisierung in einem Hardware/Software-System energieeffizienter als eine pure Softwarelösung.

Literaturverzeichnis

[Fr10] Froß, D.; Langer, J.; Froß, A.; Rößler, M.; Heinkel, U.: Hardware implementation of a Particle Filter for location estimation. In: 2010 International Conference on Indoor Positioning and Indoor Navigation. S. 1–6, Sept 2010.

[Ma12] Mautz, Rainer: Indoor Positioning Technologies. Habilitation, ETH Zürich, 2012.

[Me82] Meagher, Donald: Geometric modeling using octree encoding. Computer graphics and image processing, 19(2):129–147, 1982.

[ME85] Moravec, H.; Elfes, A.: High resolution maps from wide angle sonar. In: Proceedings. 1985 IEEE International Conference on Robotics and Automation. Jgg. 2, S. 116–121, Mar 1985.

[MM94] Moravec, Hans P; Martin, Martin C: Robot navigation by 3D spatial evidence grids. Mobile Robot Laboratory. Robotics Institute, Carnegie Mellon University, 1994.

[RTJ89] Roth-Tabak, Y.; Jain, R.: Building an environment model using depth information. Computer, 22(6):85–90, June 1989.

[TBF05] Thrun, Sebastian; Burgard, Wolfram; Fox, Dieter: The Particle Filter. In: Probabilistic Robotics. MIT Press, Kapitel 4.3, S. 96–113, 2005.