Hochschule für Technik und Wirtschaft Berlin

Internationaler Studiengang Medieninformatik

Masterarbeit

von

Daniel **Schneider**

Photogrammetrie zur Platzierung von standortbezogenen dynamischen Inhalten in AR

Photogrammetry for placement of location-based dynamic content in AR

Hochschule für Technik und Wirtschaft Berlin
Fachbereich Informatik, Kommunikation und Wirtschaft

Studiengang Internationaler Studiengang
Medieninformatik

Masterarbeit

von

Daniel **Schneider**

Photogrammetrie zur Platzierung von standortbezogenen dynamischen Inhalten in AR

Photogrammetry for placement of location-based dynamic content in AR

Bearbeitungszeitraum: von 13.05.2019
bis 16.09.2019

1. Prüfer: Prof. Dr. Tobias Lenz

2. Prüfer: Prof. Dr. Klaus Jung

Hochschule für Technik und Wirtschaft Berlin
Fachbereich Informatik, Kommunikation und Wirtschaft

Eigenständigkeitserklärung

Name und Vorname
der Studentin/des Studenten: **Schneider, Daniel**

Studiengang: **Internationaler Studiengang Medieninformatik**

Ich bestätige, dass ich die Masterarbeit mit dem Titel:

Photogrammetrie zur Platzierung von standortbezogenen dynamischen Inhalten in AR

selbständig verfasst, noch nicht anderweitig für Prüfungszwecke vorgelegt, keine anderen als die angegebenen Quellen oder Hilfsmittel benutzt sowie wörtliche und sinngemäße Zitate als solche gekennzeichnet habe.

Datum: 17. Juli 2019

Unterschrift:

Hochschule für Technik und Wirtschaft Berlin
Fachbereich Informatik, Kommunikation und Wirtschaft

Masterarbeit Zusammenfassung

Studentin/Student (Name, Vorname):	**Schneider, Daniel**
Studiengang:	Internationaler Studiengang Medieninformatik
Aufgabensteller, Professor:	Prof. Dr. Tobias Lenz
Durchgeführt in (Firma/Behörde/Hochschule):	HTW Berlin
Betreuer in Firma/Behörde:	
Ausgabedatum: 13.05.2019	Abgabedatum: 16.09.2019

Titel:

Photogrammetrie zur Platzierung von standortbezogenen dynamischen Inhalten in AR

Zusammenfassung:

"Zusammenfassung"

Schlüsselwörter: Photogrammetrie, AR, standortbezogene Daten, Android, Java, SfM, SLAM

Inhaltsverzeichnis

1 Motivation

Photogrammetrie ist "die Wissenschaft und Technologie der Gewinnung von Informationen über die physische Umwelt aus Bildern, mit einem Schwerpunkt auf Vermessung, Kartierung und hochgenauer Messtechnik". (Heipke, 2017, S.5 [1]) Die Photogrammetrie beschäftigt sich mit der Rekonstruktion von dreidimensionalen Daten aus zweidimensionalen Informationsträgern, wie Bildern oder Laserscan Daten. Dabei gehen diese Daten alle auf das Prinzip der Aufnahme der elektromagnetischen Strahlung zurück. Bei Bildern ist das die Helligkeits und Farbverteilung, bei Laserscans, Entfernungsbilder, beziehungsweise Punktwolken. Die Disziplin der Photogrammetrie ist dabei dem Bereich der Fernerkundung zuzuordnen, die sich mit der Auswertung von geometrischen oder semantischen Informationen beschäftigt. Beides sind Fachbereiche, die sich über die Jahrzehnte entwickelt haben und sich dem Gebiet der Geodäsie zuordnen lassen. Die Geodäsie erfasst Geoinformationen über die Erde, die dann beispielsweise mit Kartographie visualisiert werden können. Das Gebiet der Computer Vision, das sich großteils parallel mit der Photogrammetrie entwickelt hat, verfolgt den gleichen Ansatz. Auch hier ist die Auswertung digitaler Bilder das zentrale Element. Nachdem sich Photogrammetrie und Computer Vision lange unabhängig voneinander entwickelt haben, ist Photogrammetrie heute als Grundlage von Computer Vision anerkannt. (vgl. [1] S. 5-7)

Durch die Entwicklung der Technik und der damit eingehenden Steigerung der Rechenpower im mobilen Bereich, haben sich die Bereiche, in denen Photogrammetrie eingesetzt werden kann vergrößert. Im Rahmen dieser Arbeit soll evaluiert werden, ob photogrammetrische Verfahren für Augmented Reality Anwendungen im Bereich von Smarthphones eingesetzt werden können, um in Echtzeit aus Videodaten die dreidimensionalen Beschaffenheit der gefilmten Objekte zu rekonstruieren. Dazu wird die photogrammetrische Pipeline analysiert und mit aktuellen Verfahren verglichen.

Im Rahmen dieser Arbeit ist ebenfalls eine auf Android basierende Anwendung erstellt worden, welche eine der neuen Technologien im Bereich Augmented Reality implementiert.

2 Augmented Reality

Im Gegensatz zu „Virtual Reality“ (VR), welche eine interaktive, dreidimensionale, computergenerierte, immersive Umgebung schafft, in die eine Person versetzt wird, erlaubt „Augmented Reality“ (AR) die Überblendung von digitalen Medieninformationen über die Wahrnehmung der echten Welt. Dadurch fällt AR in die Definition von „Mixed Reality“ (MR).

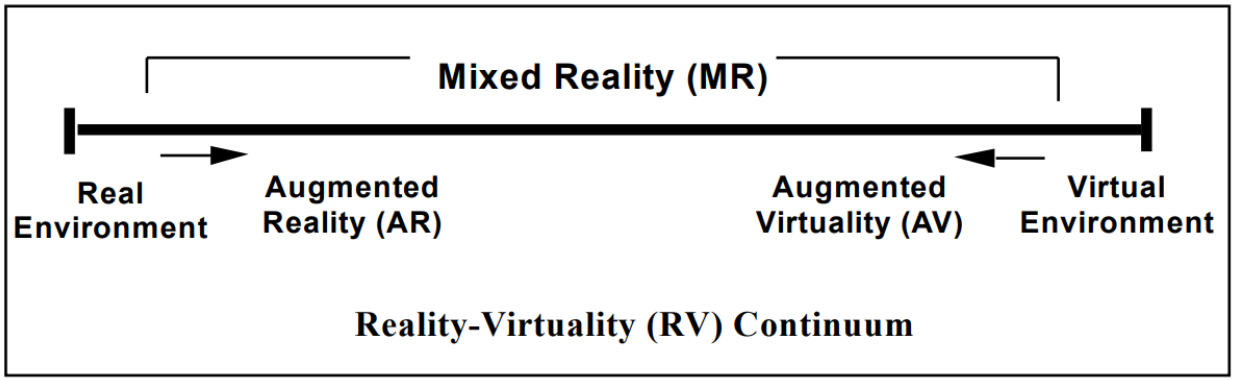

Abbildung 2.1: Das Realität - Virtualität Kontinuum, Bildquelle [11]

Im Rahmen dieser Arbeit wird sich, wenn der Begriff Augmented Reality (AR) verwendet wird, auf Monitor basierte, nicht immersive Geräte bezogen, da sich in der Analyse der Verfahren und der Implementation einer AR Anwendung, auf Smartphones bezogen wird. Diese Anzeigesysteme werden auch als „window-on-the-world“ bezeichnet, da computergenerierte Bilder oder Informationen digital über das Echtzeit Kamera Bild überlagert werden. (vgl. [11] S.284)

2.1 Augmented Reality - Software Development Kits

„Software Development Kits“(SDK) oder auch Frameworks, sind Werkzeuge und Bibliotheken, welche eine Programmierumgebung und Basistechnologien liefern, um

Programme zu entwickeln. Im Bereich von Augmented Reality SDKs umfassen Frameworks meistens die drei Hauptkomponenten: (vgl. [12] S.3)

- **Recognition**: Erkennung von Bildern, Objekten, Gesichtern oder Räumen, auf welche die virtuellen Objekte oder Informationen überlagert werden können.
- **Tracking**: Echtzeit-Lokalisierung der erkannten Objekte und Berechnung der lokalen Position des Gerätes zu diesen.
- **Rendering**: Überlagerung der virtuellen Medieninformationen über das Bild und Anzeige der generieren Mixed Reality.

2.1.1 Marktübersicht - SDKs

- Vuforia
- Wikitude
- Metaio
- ARToolKit
- Kudan
- EasyAR
- MaxST
- ARcore
- ARKit

2.2 Arten des Augmented Reality Trackings

(Evtl bessere quellen für ref und hybrid)

2.2.1 Referenzmarken-basiertes Tracking

Markerbasiertes Tracking war lange Zeit eine der häufigsten verwendeten Techniken um Augmented Realtiy zu realisieren. Dies liegt in der einfachen Erkennung der typischerweise schwarz-weißen Marker mit hohem Kontrast. Dadurch kann neben der

Relation des Geräts zum Marker auch relativ einfach die Entfernung und der Winkel berechnet werden. Der Nachteil liegt in der Limitierung der Anwendungsgebiete, in denen diese Technik verwendet werden kann, da Marker immer im Sichtfeld der Kamera lokalisiert sein müssen und nicht von anderen Objekten verdeckt werden dürfen. Weiterhin müssen immer externe Ressourcen verwendet werden um diese Marker zu erstellen und zu verwenden, was bei der Verbreitung der Anwendung immer mit einem Mehraufwand verbunden ist. (vgl. [13] S.13)

2.2.2 Hybrid-basiertes Tracking

Hybrid basiertes Tracking verwendet mehrere Datenquellen wie das Global Positioning System (GPS), Kompass oder Beschleunigungssensoren zur Bestimmung der Orientierung und Lokalisierung des Geräts. Dabei wird per GPS der Standort des Geräts bestimmt, um Objekte in der Nähe zu identifizieren, die augmentiert werden sollen. Mit Hilfe des Kompasses kann dann ein Pfad erstellt und überprüft werden, ob die Orientierung des Geräts auch in diese Richtung zeigt. Der Beschleunigungssensor bestimmt die Ausrichtung des Geräts mithilfe der Gravitation. Durch die Vereinigung all dieser Informationen kann berechnet werden, was im Sichtfeld ergänzt werden soll, ohne dass eine Auswertung und Verarbeitung des realen aufgenommen Bildes stattzufinden hat. Anschließend werden die Informationen über das Kamerabild gelegt. (vgl. [13] S.13)

2.2.3 Modell-basiertes Tracking

Beim Modell-basiertem Tracking wird ein rekursiver Algorithmus verwendet. Hierbei wird die vorherige Kameraposition als Grundlage für die Berechnung der aktuellen Kameraposition verwendet. Durch die Rekursivität ist dieses Verfahren nicht sehr rechenintensiv und benötigt eine relativ geringe Prozessorleistung. Weiterhin kann zwischen verschiedenen Merkmalen unterschieden werden, welche für das Tracking verwendet werden. Bei der kantenbasierten Methode wird versucht ein dreidimensionales Wireframe mit den Kanten des Objekts in der realen Welt zuzuordnen

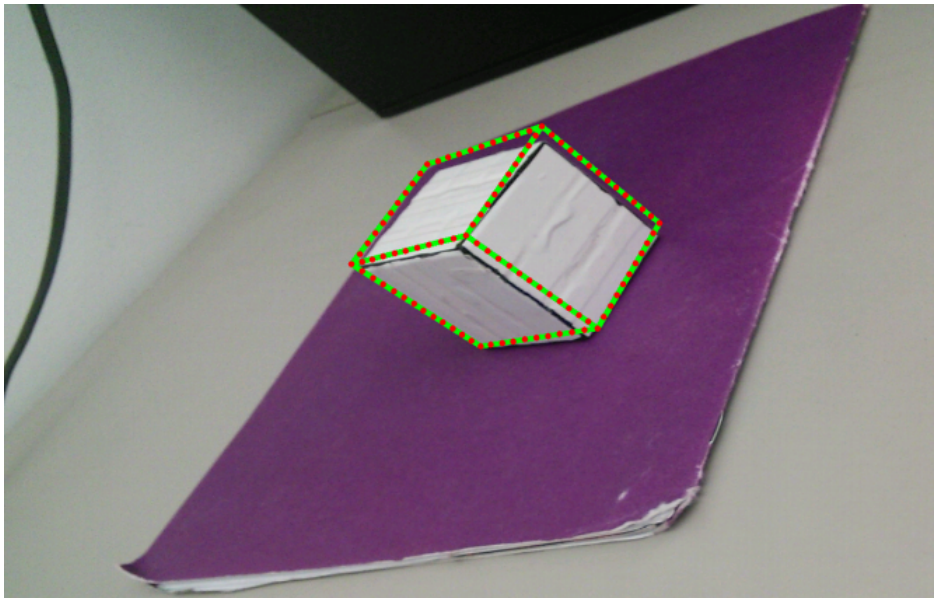

Abbildung 2.2: Kantenbasiertes rekursives Tracking, Bildquelle [14] S.3

Außerdem sind Ansätze wie „optical flow based tracking“, was zeitliche Informationen, entnommen aus der Bewegung der Projektion des Objekts relativ zur Bildebene verwendet, sowie texturbasierte Ansätze verbreitet. (vgl. [14] S.1-2)

2.2.4 Natürliches Feature Tracking

Natürliches Feature Tracking ist ein bildbasiertes Verfahren und kann die Position des Gerätes zur Umgebung, ohne das Wissen über einen vorherigen Zustand bestimmen. Diese Methode ist in der Regel sehr rechenintensiv und benötigt hohe Prozessorleistung. (vgl. [14] S.1-2) Diese Technik verwendet die Merkmale von Objekten in der echten Welt und erkennt die natürlichen Eigenschaften dieser. Diese Merkmale werden Features genannt und sind typischerweise, basierend auf einem mathematischem Algorithmus, sehr gut unterscheidbar und außern sich in der Form von Ecken, Kanten oder starke Kontrasten. Die Feature Deskriptoren eines Bildes werden zur späteren Erkennung gespeichert. Anhand des gespeicherten Datensets aus Merkmalen kann dann erkannt werden, ob ein Bild den gleichen Inhalt zeigt, unabhängig von Entfernung, Orientierung, Beleuchtungsintensität oder Verdeckung. (vgl. [13] S.13) Es gibt eine Vielzahl an natürlichen Feature Tracking Systemen, wie SIFT (Scale-Invariant Feature Transform), SURF (Speeded Up Robust Features) oder „Random Ferns“. Diese unterscheiden sich hauptsächlich durch die Bildmerkmale zwischen Videobild und dem Modell der Umgebung die verfolgt werden soll. Die Grundsätzliche Pipeline kann wie in Abbildung 2.3 dargestellt, beschrieben werden.

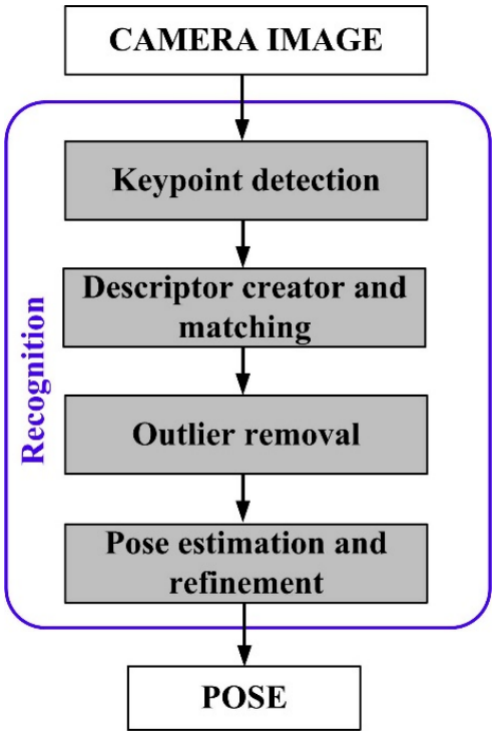

Abbildung 2.3: Tracking Pipeline Bildquelle [15]

Die Keypoint Erkennung wird normalerweise mit einer „FAST corner detector“ bewerkstelligt.

`http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/AV1011/AV1FeaturefromAccelerat pdf`

Die Erstellung und das Matching der Deskriptoren hängt von der Wahl des Deskriptors ab. Bei SIFT beispielsweise schätzt der Algorithmus die dominante Orientierung des Keypoints mit Hilfe von Gradienten, kompensiert die erkannte Orientierung und beschreibt abschließend die Keypoints in Bezug zu den Gradienten der Umgebung. Die erkannten Deskriptoren werden in eine Datenbank gespeichert, in welcher dann, während des Echtzeit Trackings, auf Gemeinsamkeiten geprüft werden kann. „Outlier removal“ oder auch die Ausreißerbeseitigung besteht aus einer Reihe von Techniken zur Entfernung von unerwünschten, falsch erkannten Keypoints, beginnend mit günstigen Methoden (einfache geometrische Tests) und abschließend mit teuren, homographie basierten Tests. (vgl. [15] S.28-29)

Die planare Homographie bezieht sich auf die Transformation zwischen zwei Ebenen. Betrachtet man das erst Set and korrespondierender Punkte, (x, y) im ersten Bild und (x', y') im zweiten. Dann bildet die Homographie H diese wie folgt ab.

$$s \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = H \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} h_1 1 & h_1 2 & h_1 3 \\ h_2 1 & h_2 2 & h_2 3 \\ h_3 1 & h_3 2 & h_3 3 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{2.1}$$

Die Homographiematrix ist eine 3x3 Matrix mit 8 DoF (Degrees of Freedom). Sie wird standardmäßig normalisiert mit:

$$h_3 3 = 1 \tag{2.2}$$

oder

$$h^2{}_{11} + h^2{}_{12} + h^2{}_{13} + h^2{}_{21} + h^2{}_{22} + h^2{}_{23} + h^2{}_{31} + h^2{}_{32} + h^2{}_{33} = 1 \tag{2.3}$$

Homographie wird in vielen Anwendungsbereichen, wie Panoramaerstellung, Bildausrichtung, perspektivischer Entzerrung oder für die Schätzung der Kameraposition in Augmented Reality verwendet. (vgl. [16])

Die verschiedenen Resultate der Homographie werden als Ausgangspunkt für die „Pose Estimation“(Positionsschätzung) verwendet.

`http://sci-hub.tw/10.1109/TVCG.2015.2513408`

Abschließend wird, basierend auf dem Gauß-Newton-Verfahren eine Reduzierung des „re-projection error“ erreicht.

urlhttps://mathepedia.de/Gausz-Newton-Verfahren.html

Typischerweise sind zwei bis vier Wiederholungen genug. (vgl. [15] S.28-29)

SIFT SURF

2.3 Warum beleuchte ich Photogrammetrie im Rahmen von AR

3 Photogrammetrie

3.1 Einführung in die Photogrammetrie

3.2 Mathe in der Photogrammetrie

3.3 Pipeline

3.3.1 Feature Matching

3.3.2 Image Matching

3.3.3 Structure from Motion

6.2 Structure from motion

http://sci-hub.tw/https://doi.org/10.1007/s10462-012-9365-8

3.3.4 Depth Maps

3.4 Photogrammetrie für Smartphones

3.5 Evaluierung der photogrammetrischen Technologie für Echtzeit AR Anwendungen

4 Simultaneous Localisation and Mapping

Simultaneous Localisation and Mapping, kurz SLAM, ist das Problem der Auswertung einer unbekannten Umgebung und Erstellung einer Map, während gleichzeitig die lokale Position innerhalb dieser Map bestimmt wird. Die Lösung dieses SLAM Problems war vorallem in der Robotik eine fundamentale Aufgabe der letzten zwei Jahrzehnte. Dabei ist SLAM ein Alltagsproblem: Das Problem der räumlichen Erkundung. Jeder Mensch und jedes Tier hat dieses Verfahren gemeistert und benutzt es unterbewusst zur Navigation in unserer Realität. Die Lösung dieses Problems, wenn es für einen Roboter automatisiert ausgeführt werden soll, ist dagegen sehr komplex. Durch das Meistern dieser Technik kann man Roboter wirklich autonom steuern. Bei SLAM wird die Bewegung des Objekts an sich durch den Raum und die Position aller zur positionsbestimmung notwendigen Merkmale berechnet, ohne auf vorheriges Wissen, über Position oder Lage im Raum, Kenntniss zu haben. (vgl. [2] S. 1-2)

Dabei benötigt der Roboter mindestens einen exterozeptiven Sensor um äußere Informationen zu sammeln. SLAM besteht aus drei grundlegenden Operationen, die iterativ pro Zeitintervall ausgeführt werden.

Der Roboter bewegt sich und erreicht eine neue Position in der Umwelt. Diese Bewegung erzeugt, durch unvermeidbares Rauschen und Fehler, Ungewissheit über die wirkliche Position des Roboters. Eine automatisierte Lösung benötigt ein mathematisches Modell für diese Bewegung. Dies ist das „*Motion Model*“

Der Roboter entdeckt neue Features in seiner Umgebung, welche in die Umgebungskarte aufgenommen werden müssen. Diese Features heißen „Landmarks“. Da die Position der Landmarks, durch Fehler in den exterozeptiven Sensoren und die Position des Roboters ungewiss ist, müssen diese beiden Faktoren passend arrangiert werden. Eine automatisierte Lösung benötigt ein mathematisches Modell, das die Position der Landmarks anhand er Sensordaten bestimmt. Dies ist das „*Inverse Oberservation Model* “.

Der Roboter entdeckt Landmarks, die schon gemappt wurden und verwendet diese um seine eigene Position, sowie die aller Landmarks zu korrigieren. Diese Operation reduziert die Unsicherheit über den Standort des Roboters, sowie der Landmarks. Die automatisierte Lösung erfordert ein mathemathisches Modell, um die Werte der Messungen aus den prognostizierten Positionen der Landmarks und der Position des Roboters zu berechnen. Dies ist das „*Direct Observation Model* “

Mit diesen drei Modellen ist es möglich eine automatisierte Lösung für SLAM zu entwerfen. Diese Lösung muss diese drei Modelle verbinden und alle Daten korrekt und oganisiert halten, sowie die korrekten Entscheidungen bei jedem Schritt machen. (vgl. [4] S.2-3)

Eine erfolgreiche Lösung des SLAM Problems setzt weiterhin die Lösung des „Loop Closure Detection“Problems vorraus. Dabei müssen bereits besuchte Orte in der beliebig großen Map erkannt werden. Wegen der möglichen Komplexität von großen Maps ist es auch eins der größten Hindernisse, wenn es um die Skalierbarkeit der Lösung geht. Wichtig ist, dass die Loop Closure Detection keine Falsch-Positiven Ergebnisse liefert, da dies die Integrität und Korrektheit der kompletten Map beeinflusst. (vgl. [10] S.4)

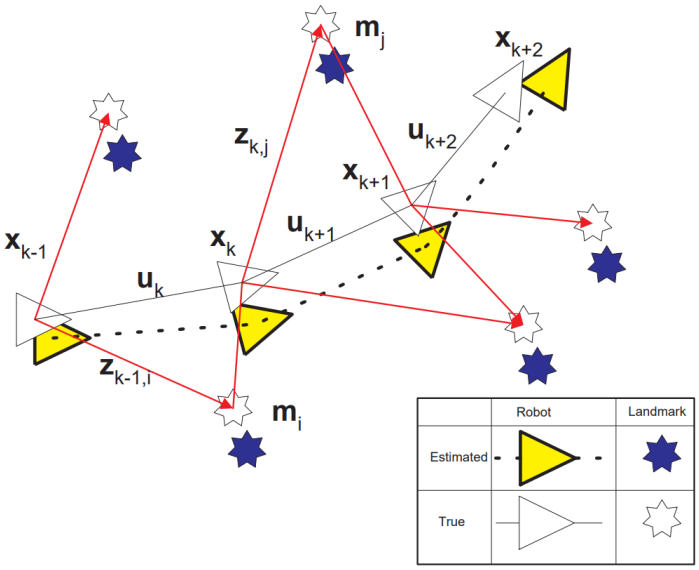

Abbildung 4.1: Das SLAM Problem: Die wahren absoluten Positionen der extrahierten Features sind nie wirklich bekannt. Bildquelle [2]

Wie in Abbildung 3.1. erkennbar ist, bewegt sich ein Roboter durch eine unbekannte Umgebung und nimmt mit seinem Sensor Features der näheren Objekte (Landmarks) auf. Wobei $\mathbf{x}k$ der Vektor des Roboters, $\mathbf{u}k$ der Bewegungsvektor, $\mathbf{m}i$ der Vektor des

Landmarks und $\mathbf{z}ik$ die Oberservation eines Landmarks durch den Roboter zur Zeit k sind. Wie man sehen kann, ist der Fehler zwischen echten und geschätzten Landmarks, bei allen geschätzten Landmarks ähnlich, was an der initialen Betrachtung der Umgebung liegt. Zu diesem Zeitpunk wird nur das erste Feature erkannt. Daraus kann man schließen, dass die Fehler in der Schätzung der Landmarkpositionen korrelieren. Praktisch bedeutet dies, dass die relative Position zweier Landmarks, $\mathbf{m}i$ - $\mathbf{m}j$ zueinander sehr genau sein kann, auch wenn die absolute Position sehr ungenau ist.

Je mehr Landmarks in das Modell aufgenommen werden, desto gleichbleibend besser wird das Modell der relativen Positionen, egal wie sich der Roboter bewegt. Dieser Prozess wird in Abbildung 3.2. veranschaulicht.

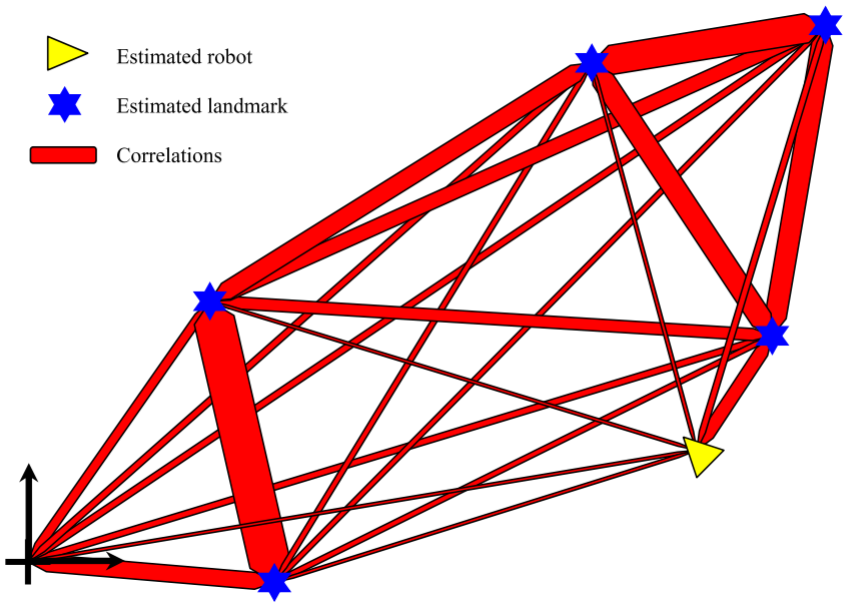

Abbildung 4.2: Die Landmarks sind durch Federn verbunden, welche die Korrelation zwischen ihnen darstellen. Bildquelle [2]

Während sich der Roboter durch die Umgebung bewegt, werden die Korrelationen stetig aktualisiert. Je mehr Beobachtungen über die Umwelt gemacht werden, desto steifer werden die Federn in diesem Modell. Im Nachhinein werden neue Beobachtungen von Landmarks durch das ganze Netzwerk propagiert und je nach Input, kleinere oder größere Anpassungen vorgenommen.

Lösungen für das SLAM Problem benötigen eine angemessene Repräsentation für die Observierungen der Landmarks, welche eine konsistente und schnelle Berechnung ermöglichen. Die geläufigste Repräsentation besteht in der Form einer Zustandsraumdarstellung mit Gaußschen Rauschen, was zur Verwendung des „Extended Kalman

Filter“(EKF) führt. (vgl. [2] S. 2-4)

Weitere gängige Lösungen für das SLAM Problem sind „Maximum Likelihood Techniques “, „Sparse Extended Information Filters “(SEIFs) und „Rao Blackwellized Particle Filters“(RBPFs). (vgl. [7] S. 2)

4.1 Extended Kalman Filter - SLAM

Der Kalman Filter ist eine Schätzfunktion für das „linear-quadratic-problem“, welches das Problem der Schätzung des augenblicklichen Zustands eines linearen dynamischen Systems, gestört durch weißes Rauschen, darstellt. Der Kalman Filter wird auch dazu benutzt um die mögliche Zukunft von dynamischen Systemen vorherzusagen, die von Menschen nicht kontrolliert werden können, wie zum Beispiel die Flugbahn von Himmelskörpern, oder der Kurs von gehandelten Rohstoffen. (vgl. [5] S.1)

Der Kalman Filter besteht aus drei Schritten. Zuerst wird eine Messung vorhergesagt, welche dann mit der realen Messung verglichen wird. Die resultierende Differenz wird mit der Varianz der Messung gewichtet, um daraus eine neue Schätzung des Zustands zu erhalten. (vgl. [8] S.13) Der Kalman Filter lässt sich jedoch nur auf lineare Systeme anwenden. Der EKF verwedet für die Vorhersage der Messungen und der Zustände hingegen nichtlineare Funktionen. (vgl. [8] S.16-17)

Bei Extended Kalman Filter - SLAM ist die Map ein großer Stapel an Vektor und Sensordaten, sowie Zuständen von Landmarks.

$$x = \begin{bmatrix} R \\ M \end{bmatrix} = \begin{bmatrix} R \\ L_1 \\ ... \\ L_n \end{bmatrix} \tag{4.1}$$

R ist der Zustand des Roboters und $M = (L_1, ..., L_n)$ ist das Set an Zuständen der Landmarks. Bei EKF wird die Map durch eine gaußsche Variable modelliert, die den Mittelwert und die Kovarianzmatrix des Zustandsvektors verwendet, die jeweils durch $\overline{x}$ und P beschrieben werden. Das Ziel ist es die Map $\{\overline{x}, P\}$ zu allen Zeiten auf dem aktuellsten Stand zu halten.

$$\overline{x} = \begin{bmatrix} \overline{R} \\ \overline{M} \end{bmatrix} = \begin{bmatrix} \overline{R} \\ \overline{L_1} \\ ... \\ \overline{L_n} \end{bmatrix} \qquad P = \begin{bmatrix} P_{RR} & P_{RM} \\ P_{MR} & P_{MM} \end{bmatrix} = \begin{bmatrix} P_{RR} & P_{RL1} & ... & P_{RLn} \\ P_{L1R} & P_{L1L1} & ... & P_{L1Ln} \\ ... & ... & ... & ... \\ P_{LnR} & P_{LnL1} & ... & P_{LnLn} \end{bmatrix} \tag{4.2}$$

Diese Map, die als stochastische Map bezeichnet wird, wird durch die Vorhersage- und Korrekturprozesse des EKF in Stand gehalten. Um eine echte Erkundung der Umgebung zu erreichen, wird der EKF Algorithmus mit einem extra Schritt der Landmark Erkennung und Initialisierung gestartet, bei dem neue Landmarks der Map hinzugefügt werden. Die Landmark Initialisierung erfolgt durch eine Umkehrung der Bewertungsfunktion und der Verwendung dieser und der Ableitungsmatrix, um die beobachteten Landmarks und die benötigten Co- und Crossvarianzen für den Rest der Map zu berechnen. Diese Beziehungen werden dann an den Zustandsvektor und die Kovarianzmatrix angehängt. (vgl. [4] S.6-7)

Eine zentrale Einschränkung des EKF basierten SLAM Ansatzes ist die Komplexität der Berechnung. Sensor-Updates benötigen Zeit, quadratisch zur Anzahl der Landmarks K, die zu berechnen sind. Diese Komplexität ergibt sich aus der Tatsache, dass die vom Kalman-Filter verwaltete Kovarianzmatrix $O(K^2)$ Elemente enthält, die alle aktualisiert werden müssen, auch wenn nur ein einzelnes Landmark beobachtet wurde. Diese Komplexität limitiert die Anzahl an Landmarks, die durch diesen Ansatz verarbeitet werden konnen, auf ein paar Hunderte, während natürliche Umgebungsmodelle häufig Millionen von Features enthalten. (vgl. [6] S.1)

`http://www.iri.upc.edu/people/jsola/JoanSola/objectes/curs_SLAM/SLAM2D/SLAM%20course.pdf`

4.2 ORB-SLAM

TODO

https://arxiv.org/pdf/1502.00956.pdf https://arxiv.org/pdf/1610.06475.pdf

4.3 FAST-SLAM

Fast-SLAM (Fast SLAM 1.0) zerlegt das SLAM-Problem in ein Lokalisierungsproblem des Roboters und eine Reihe von Landmark Schatzungsproblemen, die auf der Schätzung der Roboterposition beruhen. Fast-SLAM verwendet einen modifizierten Partikelfilter zur Schätzung der „A-posteriori-Wahrscheinlichkeit"für die Roboterposition. Partikelfilter sind vom Prinzip her ähnlich wie Kalman Filter. Diese werden auch zur Schätzung von Zuständen verwendet, können aber viele verschiedene mögliche Zustände betrachten. Diese Anzahl an Zuständen wird Partikel genannt. Jedes Partikel besitzt wiederum Kalman-Filter, welche die Positionen der Landmarks schätzen, abhängig von der Pfadschätzung. Eine naive Implementation dieser Idee führt zu einem Algorithmus, der $O(MK)$ Zeit benötigt, wobei M die Anzahl an Partikeln im Partikel Filter und K die Anzahl an Landmarks ist. Mit der Verwendung einer Baumstruktur kann die Laufzeit von FastSLAM auf $O(MlogK)$ reduziert werden, was diesen Algorithmus deutlich schneller als EKF basierte SLAM ALgortihmen macht. (vgl. [6] S.1-2, [8] S.18-19)

Bei Fast-SLAM werden nicht nur die unterschiedlichen geschätzten Positionen verwendet, sondern auch verschiedene Maps der Umgebung betrachtet. Da verschiedenste Maps mit verschieden Warscheinlichkeiten der geschätzten Positionen betrachtet werden, können Fehler in der Map, die sich durch falsche gemessene oder geschätzte Positionen ergeben, durch die Anzahl an verschieden Maps relativieren. Es können sich im Laufe der Zeit Maps, die vorher weniger wahrscheinliche Positionen hatten, als richtig herausstellen. (vgl. [8] S.23)

Fast-SLAM 2.0 ist eine weiterentwickelte Variante von Fast-SLAM. Hierbei wird eine Vorauswahl getroffen, berechnet durch einen weiteren Kalman Filtern, in wie weit und mit welcher Varianz die Partikel um den Mittelpunkt verteilt werden. Es ergibt sich eine bessere Auswahl des Mittelpunktes und des Streuradius für den Partikelfilter als in Fast-SLAM 1.0. Diese Methodik reduziert die Anzahl an Partikel und generierten Maps, ist demnach auch schneller berechnet. (vgl. [8] S.29-30)

4.4 SLAM für mobiles Augumented Reality

Das Ziel von Augumented Reality ist es virtuelle Objekte oder Informationen in die echte Welt zu integrieren, um den Benutzer zusätzliche Informationen in die betrachtete Szene zu liefern. Dazu ist es notwendig, die echte und die virtuelle Welt präzise

aneinander auszurichten. Dann kann für jeden Frame aus der Sequenz des Videobildes die genaue Position des mobilen Gerätes bestimmt werden. Um dieses Ziel des exakten Matchings von Realität und generierter virtueller Realität zu erreichen, ist "Camera Localization", also die Lokalisierung der Kamera im dreidimensionalen Raum, anhand von aufgenommenen zweidimensionalen Daten, die Schlüsseltechnologie für alle Augumented Reality Anwendungen. (vgl. [3] S.1) Weiterhin ist es wichtig die Umgebung zu kartographieren, um Projektionsflächen für virtuelle dreidimensionale Objekte zu finden. Eine mögliche Lösung für dieses Problem ist die Verwendung von SLAM. In den letzten 20 Jahren hat sich bei SLAM ein Trend gezeigt, bei dem Kameras als einzige exterozeptive Sensoren verwendet werden. Der Hauptgrund dafür ist die Fähigkeit eines kamerabasierten Systems, Tiefeninformationen, Farben, Texturen und Helligkeiten zu erkennen, was Robotern beispielsweise Objekt- oder Gesichtserkennung ermöglicht. Darüber hinaus sind Kameras preiswert, leicht und haben einen geringen Stromverbrauch. (vgl. [9] S.57) Erst in den letzten Jahren hat sich SLAM in Alltagsanwendungen profilieren können, hauptsächlich wegen dem Aufkommen von Smartphones. Smartphones sind mobil und haben die nötige Rechenleistung um SLAM Aufgaben in Echtzeit auszuführen. Dies hat zu einer Erweiterung der Forschungsmöglichkeiten von SLAM geführt und hat es zu einer robusteren Technologie gemacht. Weiterhin verfügen Smartphones über eine ganze Reihe an Sensoren, wie Beschleunigungssensoren, Gyroskop, Magentometer oder GPS, mit denen die visuellen Daten ergänzt werden können, um die Map genauer und weniger anfällig für innere Drifteffekte zu machen, was jedoch wieder eigene komplexe Probleme bei der Verbindung von verschiedenen Sensordaten mit sich bringt. (vgl. [10] S.4)

Die Verwendung von SLAM im mobilen Bereich hat jedoch einige Probleme zu meistern. Durch die Ungenauigkeiten der absoluten Lokalisierung, ist es schwierig kontextsensitive Augmentation zu erzeugen. Auch wenn es möglich ist bekannte Objekte im Raum während des Trackings zu bestimmen, und die Information um diese herum zu zeigen, ist es fast unmöglich den gesamten Raum einer Szenerie zu bestimmen. Das zweite Problem liegt in der Ergonomie und Benutzbarkeit der Anwendung durch den Endbenutzer. Typischerweise soll ein Endverbraucher keinem komplexem Protokoll folgen müssen, um sich selbst in der Szene zu lokalisieren. Das System benötigt also eine Verfahren zur schnellen und robusten Lokalisierung im Raum. (vgl. [3] S.1)

`https://hal.inria.fr/hal-00994756/document` `https://pdfs.semanticscholar.org/00f4/41387f04f40aad6491ce23bdeb0ece17d12e.pdf`

file:///C:/Users/Daniel/Downloads/AIREVSLAMSurvey.pdf

4.5 SLAM als Core für viele AR APIs

file:///C:/Users/Daniel/Downloads/COMPARATIVESTUDYOFAUGMENTEDREALITY.pdf

5 Vergleich der Verfahren

5.1 Ähnlichkeiten und Unterschiede

5.2 Analyse der Echtzeitfähigkeit

6 Implementation einer AR Anwendung für Android

6.1 Verwendete Hard und Software

6.1.1 Ar Core

6.1.2 Sceneform

6.1.3 Google Location Service

6.1.4 Dexter

6.1.5 Volley

6.2 Implentierung

7 Praxistest

7.1 Anwendungsbeispiele

7.2 Benchmarks

7.3 Tests

8 Weitere Verfahren

8.1 Depth Map mit Dual Camera

8.2 ???

9 Zusammenfassung und Ausblick

Literaturverzeichnis

[1] Heipke, C. (2017), „Photogrammetrie und Fernerkundung “, 1.Auflage, Berlin: Springer Verlag, S. 5-7.

[2] Hugh Durrant-Whyte, Tim Bailey (2006), Simultaneous Localisation and Mapping (SLAM): Part I The Essential Algorithms. URL: https://people.eecs.berkeley.edu/~pabbeel/cs287-fa09/readings/Durrant-Whyte_Bailey_SLAM-tutorial-I.pdf (Zuletzt abgerufen am 09.07.2019)

[3] P. Martin, E. Marchand, P. Houlier, I. Marchal. Mapping and re-localization for mobile augmented reality. IEEE Int. Conf. on Image Processing, Oct 2014, Paris, France. URL: https://hal.inria.fr/hal-00994756/document (Zuletzt abgerufen am 09.07.2019)

[4] Joan Sol'a, (2014), Simulataneous localization and mapping with the extended Kalman filter. URL: http://www.iri.upc.edu/people/jsola/JoanSola/objectes/curs_SLAM/SLAM2D/SLAM%20course.pdf (Zuletzt aufgerufen am 10.07.2019)

[5] Mohinder S. Grewal, Angus P. Andrews (2001), Kalman Filtering: Theory and Practice with MATLAB. URL: http://staff.ulsu.ru/semoushin/_index/_pilocus/_gist/docs/mycourseware/13-stochmod/2-reading/grewal.pdf (Zuletzt aufgerufen am 10.07.2019)

[6] M. Montemerlo, S. Thrun, D. Koller, B. Wegbreit (2002). FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem. URL: http://robots.stanford.edu/papers/montemerlo.fastslam-tr.pdf (Zuletzt aufgerufen am 11.07.2019)

[7] G. Grisetti, G.D. Tipaldi, C. Stachniss, W. Burgard, D. Nardi (2007). Fast and accurate SLAM with Rao-Blackwellized particle filters. URL: http://srl.informatik.uni-freiburg.de/publicationsdir/grisettiRAS07.pdf (Zuletzt aufgerufen am 11.07.2019)

[8] M. Mengelkoch (2007). Implementieren des FastSLAM Algorithmus zur Kar-

tenerstellung in Echtzeit. URL: https://kola.opus.hbz-nrw.de/opus45-kola/frontdoor/deliver/index/docId/183/file/sa-00.pdf (Zuletzt aufgerufen am 11.07.2019)

[9] J. Fuentes-Pacheco, J. Ruiz-Ascencio, J.M. Rendón-Mancha (2012). Visual simultaneous localization and mapping: a survey. In: J.M. Artif Intell Rev (2015) 43: 55. Springer Netherlands. DOI: https://doi.org/10.1007/s10462-012-9365-8

[10] J. Halvarsson, (2018), Using SLAM-based technology to improve directional navigation in an Augmented Reality game. URL: http://umu.diva-portal.org/smash/get/diva2:1245293/FULLTEXT01.pdf (Zuletzt aufgerufen am 11.07.2019)

[11] P. Milgram, H. Takemura, A. Utsumi, F. Kishino (1994), Augmented Reality: A class of displays on the reality-virtuality continuum. URL: http://etclab.mie.utoronto.ca/publication/1994/Milgram_Takemura_SPIE1994.pdf (Zuletzt aufgerufen am 16.07.2019)

[12] A. Hanafi, L. Elaachak, M. Bouhorma (2019), A comparative Study of Augmented Reality SDKs to Develop an Educational Application in Chemical Field. DOI: 10.1145/3320326.3320386

[13] D. Amin, S. Govilkar (2015), Comparative Study of Augmented Reality SDK's. International Journal on Computational Sciences & Applications (IJCSA) Vol.5, No.1, February 2015 URL: https://pdfs.semanticscholar.org/e752/17e8897cb46b466d6ba83e909cca4ecff8f2.pdf (Zuletzt aufgerufen am 16.07.2019)

[14] M. Lowney, A. S. Raj (2016), Model Based Tracking for Augmented Reality on Mobile Devices. URL: https://web.stanford.edu/class/ee368/Project_Autumn_1617/Reports/report_lowney_raj.pdf (Zuletzt aufgerufen am 16.07.2019)

[15] S. Ćuković, M. Gattullo, F. Pankratz, G. Devedzic, E. Carrabba, K. Baizid (2015), Marker Based vs. Natural Feature Tracking Augmented Reality Visualization of the 3D Foot Phantom. URL: https://www.researchgate.net/publication/278668320_Marker_Based_vs_Natural_Feature_Tracking_Augmented_Reality_Visualization_of_the_3D_Foot_Phantom (Zuletzt aufgerufen am 17.07.2019)

[16] Basic concepts of the homography explained with code. URL: https://docs.opencv.org/3.4.1/d9/dab/tutorial_homography.html (Zuletzt aufgerufen am 17.07.2019)

Abbildungsverzeichnis