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: 12. August 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 **1**

2 Augmented Reality **3**
2.1 Augmented Reality - Software Development Kits . . . 3
2.1.1 Marktübersicht - Software Development Kits . . . 4
2.2 Voraussetzungen für Augmented Reality . . . 4
2.3 Arten des Augmented Reality Trackings . . . 5
2.3.1 Referenzmarken-basiertes Tracking . . . 5
2.3.2 Hybrid-basiertes Tracking . . . 6
2.3.3 Modell-basiertes Tracking . . . 6
2.3.4 Natürliches Feature Tracking . . . 7
2.4 Photogrammetrie vs. SLAM für Augmented Reality . . . 8

3 Photogrammetrie **9**
3.1 Einführung in die Photogrammetrie . . . 9
3.2 Feature & Image Matching . . . 11
3.2.1 Scale Invariant Feature Transform . . . 13
3.2.2 Features from Accelerated Segment Test . . . 14
3.2.3 Binary Robust Independent Elementary Features . . . 14
3.2.4 Outlier Removal . . . 15
3.3 Der Bündelblockausgleich . . . 16
3.4 Nicht-lineare Methode der kleinsten Quadrate . . . 18
3.4.1 Gauß-Newton-Verfahren . . . 20
3.4.2 Levenberg-Marquardt Verfahren . . . 21
3.4.3 Die Kollinearitätsbedingung . . . 22
3.4.4 Die Koplanaritätsbedingung . . . 24
3.4.5 Das direkte lineare Transformationsverfahren . . . 26
3.5 Structure from Motion . . . 27
3.6 Depth Maps . . . 27

4 Simultaneous Localisation and Mapping **28**
4.1 Loop closure detection . . . 31
4.2 Visual SLAM . . . 32
4.3 Extended Kalman Filter - SLAM . . . 33
4.4 FAST-SLAM . . . 34
4.5 ORB-SLAM . . . 35
4.6 SLAM für mobiles Augumented Reality . . . 37

5 Vergleich von Photogrammetrie und SLAM **40**
5.1 Ähnlichkeiten und Unterschiede 40
5.2 Analyse der Echtzeitfähigkeit 40

6 Implementation einer AR Anwendung für Android **41**
6.1 Verwendete Hard und Software 41
6.1.1 Ar Core 41
6.1.2 Sceneform 41
6.1.3 Google Location Service 41
6.1.4 Dexter 41
6.1.5 Volley 41
6.2 Implentierung 41

7 Praxistest **42**
7.1 Anwendungsbeispiele 42
7.2 Benchmarks 42
7.3 Tests 42

8 Weitere Verfahren **43**
8.1 Depth Map mit Dual Camera 43
8.2 3-D Rekonstruktion 43

9 Zusammenfassung und Ausblick **44**

Literaturverzeichnis **45**

Abbildungsverzeichnis **50**

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 sind es 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 große Potenzial der Bereitstellung und Gewinnung von dreidimensionalen Daten, war in der Photogrammetrie seit Beginn eine der wichtigsten Stärken der Technologie. Photogrammetrie wird von Raumsonden im All bis hin zur Mikroskopie eingesetzt. Auch wenn die Kartenerstellung die ursprüngliche Kernanwendung war, sind nichttopographische Anwendungen, seit der Einführung digitaler Kameras stark expandiert. Dazu zählt zum Beispiel die hochpräzise industrielle Messtechnik, die architektonische Photogrammetrie, die medizinische und forensische Bildgebung und Analyse sowie eine Reihe weiterer Verfahren. (vgl. [26] S.1)

Das Gebiet der Computer Vision, das sich großteils parallel mit der Photogrammetrie entwickelt hat, verfolgt den gleichen Ansatz und ist inhaltlich sehr stark mit der digitalen Photogrammetrie verbunden. Das Forschungsziel der Computer Vision ist es, anhand von zeidimensionalen Bilddaten die geometrischen Informationen von dreidimensionalen Objekten, wie Form, Position, räumlicher Lage, Bewegung oder ähnliche Daten zu gewinnen. Unter diesem Gesichtspunkt gibt es sehr viele Gemeinsamkeiten zwischen Computer Vision und digitaler Photogrammetrie. (vgl. [25] S.1) 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).

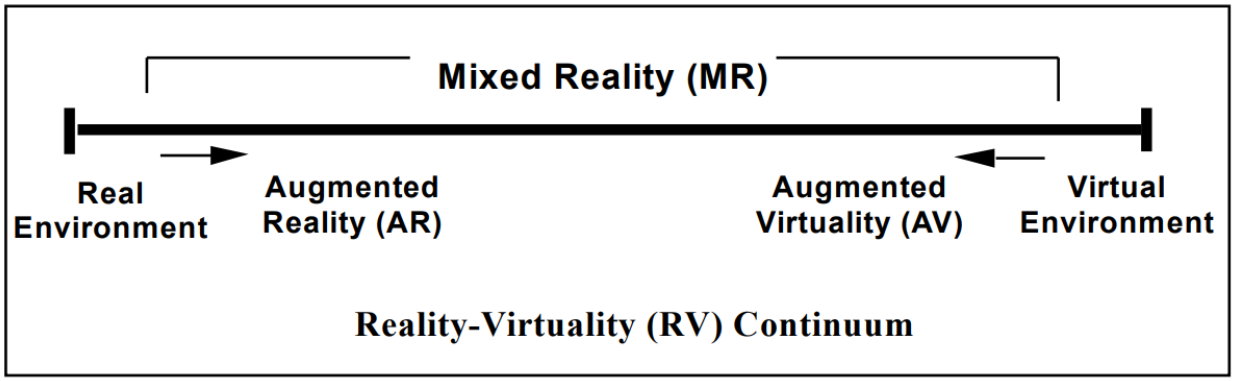

Abbildung 2.1: Das Realität - Virtualität Kontinuum, Bildquelle [11]

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 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 - Software Development Kits

Die folgende Tabelle gibt eine Übersicht an gängigen SDKs, und deren Plattformkompatibilität.

	Vuforia	Wikitude	Metaio	ARToolKit	Kudan	EasyAR	MaxST	ARCore	ARKit
Android	✓	✓	✓	✓	✓	✓	✓	✓	x
iOS	✓	✓	✓	✓	✓	✓	✓	✓	✓
Windows	✓	✓	✓	✓	x	✓	✓	x	x

Bis auf das von Apple entwickelte ARKit, sind alle hier genannten Frameworks für Android verwendbar. Weiterhin unterstützen alle das Betriebsystem iOS, sowie Windows, bis auf Kudan, ARCore und ARKit. Im praktischen Teil dieser Arbeit werden mehrere dieser Software Development Kits auf der Android Plattform verwendet, getestet und analysiert.

2.2 Voraussetzungen für Augmented Reality

Augmented Reality Anwendungen haben hohe Anforderungen an die Rechnenpower der Technik, die Verarbeitungsgeschwindigkeit der Algorithmen und Robustheit der verwendeten Verfahren. (vgl. [18] S.1)

- **Hohe räumliche Genauigkeit**: 6 „Degrees of Freedom“ (Freiheitsgrade) in Position und Ausrichtung.
- **Sehr geringer Jitter (Zittern)**: Das Rauschen im Tracking System muss minimal gehalten werden.

- **Hohe Aktualisierungsraten**: mindestens 30Hz, besser mehrere 100Hz.
- **Sehr geringer Lag**: Die Verzögerung von Messung bis zur Trackerausgabe muss minimal sein.
- **Volle Mobilität**: Bewegungsfreiheit für den Nutzer: Keine Kabel, kein eingeschränkte Umfang an Bedienmöglichkeiten.

Erst durch die Entwicklung der Technik in den letzten zwei Jahrzehnten und einer damit eingehenden Steigerung der Rechenpower von mobilen Geräten hat sich Augmented Reality auf Smartphones durchsetzen können.

2.3 Arten des Augmented Reality Trackings

Das Erkennen der Umgebung und die Lokalisierung der Kamera (Camera Pose Estimation), ist der ausschlaggebende Schritt zur Realisierung von Augmented Reality. Erst dies erlaubt die Projektion von digitalen Modellen in der richtigen Position auf den echten Bildern. Präzise und robuste Kamerapositionsdaten sind eine Grundvoraussetzung für eine Vielzahl an Anwendungen, wie dynamischer Szenenanalyse und Interpretation, 3D-Szenestrukturerkennung und Videodatenkompression. Augmented Reality Umgebungen sind ein Hauptanwendungsgebiet der Kameralokalisierung, da ein eingeschränkter Arbeitsbereich hohe Anforderungen an die Robustheit und Schnelligkeit stellt. Es existieren viele verschiedene Ansätze um die Kameralokalisierung im Raum zu lösen. Das Problem wird als nichtlineares Problem betrachtet und wird meistens durch die „Method of least squares“ (Methode der kleinsten Quadrate) oder nichtlineare Optimierungsalgorithmen gelöst, typischerweise durch das Gauß-Newton oder Levenberg-Marquardt Verfahren. (vgl. [17] S.1) Im Folgenden werden die vier gängigsten Ansätze zur Lösung dieses Tracking Problems erläutert.

2.3.1 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, zu registrieren und zu verwenden, was bei der Verwendung der Anwendung und damit der Nutzerfreundlicheit, immer mit einem Mehraufwand verbunden ist. (vgl. [13] S.13)

2.3.2 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)

(Evtl bessere quellen für ref und hybrid)

2.3.3 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

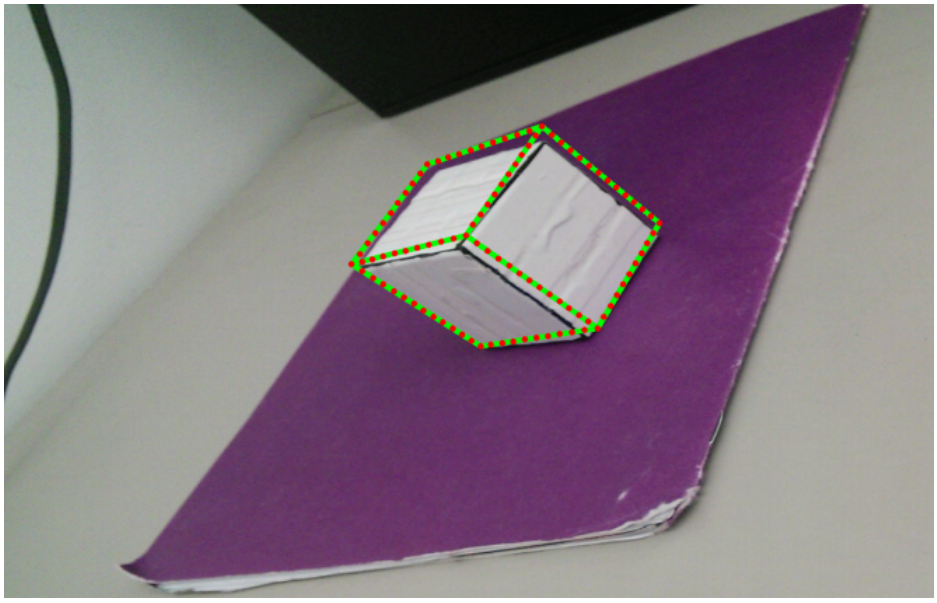

Abbildung 2.2: Kantenbasiertes rekursives Tracking, Bildquelle [14] S.3

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)

2.3.4 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 ihre natürlichen Eigenschaften. 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, Rauschen oder Verdeckung. (vgl. [13] S.13) Dies wird, meistens basierend auf dem Verfahren der kleinsten Quadrate, mit Gauß-Newton oder Levenberg-Marquardt durchgeführt, um eine Reduzierung des „re-projection error“ (Reprojektionsfehlers) zu erreichen, um alle internen und externen Kameraparameter zu bestimmen. Dies sind nicht-lineare Methoden zur Lösung des Problems der kleinsten Quadrate. Typischerweise sind zwei bis vier Wiederholungen

genug. (vgl. [15] S.28-29) Die genaue Funktionsweise des Natürlichen Feature Trackings wird in Kapitel 3.5 und 3.6 weiter ausgeführt.

2.4 Photogrammetrie vs. SLAM für Augmented Reality

In diesem Kapitel wurde eine kurze Einführung in Augmented Reality gegeben und deren Voraussetzungen sowie aktuellen Umsetzungen dargelegt. Im folgenden Teil dieser Arbeit wird Photogrammetrie, sowie SLAM (Simultaneous Localisation and Mapping), welches von den meisten Augmented Reality Software Development Kits inzwischen verwendet wird, beschrieben. Anschließend wird ein Vergleich der beiden Verfahren durchgeführt, um Gemeinsamkeiten, Unterschiede, sowie Möglichkeiten und Schwächen der einzelnen Verfahren aufzuzeigen und in Kontext zu bringen. Anschließend wird evaluiert ob photogrammetrische Verfahren in Kontext der Augmented Reality eingesetzt werden können.

3 Photogrammetrie

3.1 Einführung in die Photogrammetrie

Das Grundprinzip der Messung mit Kameras ist trivial. Licht breitet sich mit einer bestimmten Wellenlänge, in annähernd geraden Strahlen aus. Diese Strahlen werden vom Sensor der Kamera aufgenommen, sodass diese die Richtungen im dreidimensionalen Raum misst. Der grundlegende geometrische Zusammenhang der Photogrammetrie ist somit die Zentralprojektion, die sich mathematisch durch die Kollinearitätsgleichung beschreiben lässt. Ein dreidimensionaler Punkt in der echten Welt, sein Bild in der Kamera und das Projektionszentrum müssen alle auf einer geraden Linie liegen. (vgl. [20] S.1) Das fundamentale photogrammetrische Problem besteht in der Bestimmung von internen und externen Ausrichtungsparametern der Kamera und der Messung von Objekt und Raumkoordinaten der aufgenommenen Fotografien.

- **Interne Orientierung**: Bei der internen Orientierung werden Kameraparameter gemessen und ausgewertet. Dazu wird die „principle distance“ (Brennweite) und der „principle point“ (Optisches Zentrum) betrachtet.

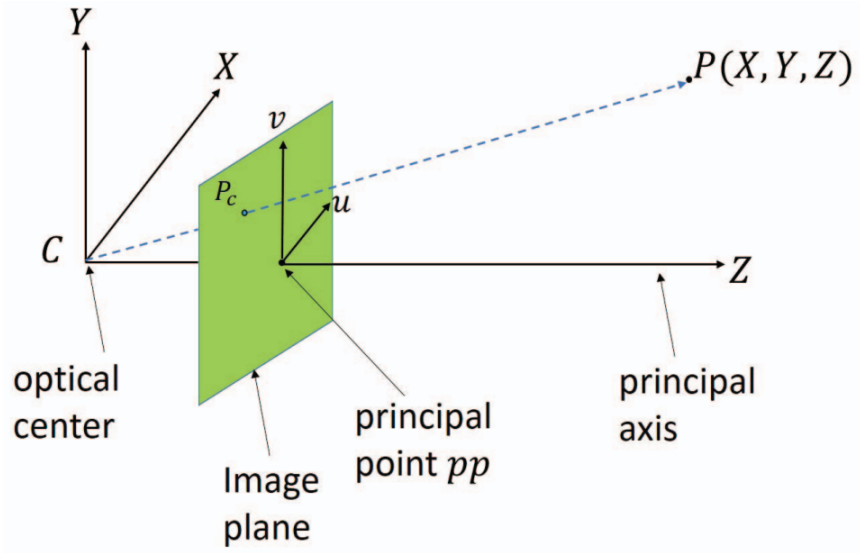

Abbildung 3.1: Kamera Kalibierungsmodell, Bildquelle [21]

Weiterhin müssen Parameter, welche die Verzeichnung, also die nicht maßstabsgetreue Abbildung von Objekten, betrachtet werden. Diese Parameter, die beispielsweise in der Objektivkorrektur verwendet werden, müssen, um die interne Orientierung der Kamera genau abzubilden, mit in die Berechnung einfließen.

- **Externe Orientierung**: Bei der externen Orientierung wird versucht die genaue räumliche dreidimensionale Lage der Kamera zum Zeitpunkt der Belichtung des Bildes zu rekonstruieren. Für die Bestimmung der Orientierung von ein oder mehreren Fotos, können verschiedene Methoden verwendet werden. Dies kann in Teilschritten (relative und absolute Orientierung) oder gleichzeitig (Bündelblockausgleich) durchgeführt werden.

In der Photogrammetrie werden häufig drei grundlegende Bedingungen verwendet, um die Parameter der externen Orientierung zu bestimmen. Dies sind die Bedingungen der Kollinearität, der Koplanarität und der Koangularität. Diese Verfahren verwenden alle Punktkoordinaten als Eingabedaten, aber in vielen Fällen sind auch räumliche oder kamerabedingte Einschränkungen möglich. In Bereich der Computer Vision wird die Bestimmung der externen Orientierung als „pose estimation problem“ (Posenschätzungsproblem) bezeichnet. Im Vergleich zur Photogrammetrie ist dieser Bereich darauf fokussiert die Lösung der Posenschätzung unter der Verwendung von minimalen Objektinformationen zu erreichen. Direkte lineare Lösungen basieren hier hauptsächlich auf Konzepten der projektiven algebraischen Geometrie. Homogene Koordinaten werden verwendet um Kameraparameter abzuleiten, entsprechend der direkten linearen Transformation, welche häufig in der Photogrammetrie und in der Fernerkundung eingesetzt wird. (vgl. [22] S.616)

Die Grundsätzliche Pipeline um anhand von zweidimensionalen Bildern die internen und externen Kameraparameter, sowie die Lage der Bilder zueinander im dreidimensionalen Raum zu rekonstruieren kann wie in Abbildung 2.3 dargestellt, beschrieben werden.

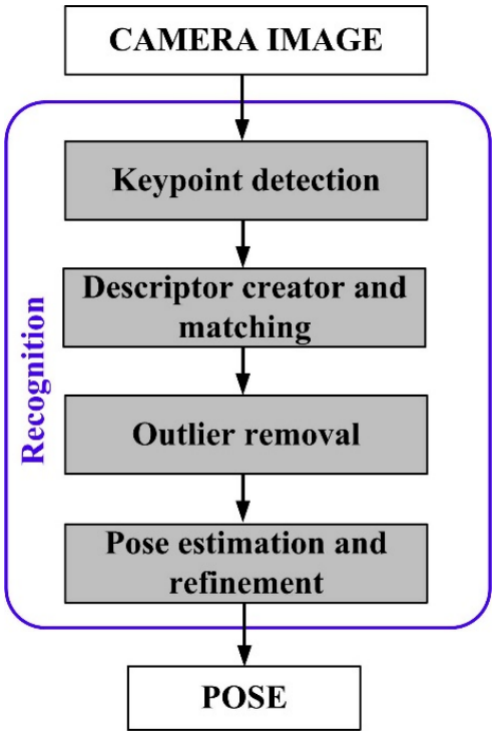

Abbildung 3.2: Tracking Pipeline Bildquelle [15]

Die Schritte „Keypoint detection“, „Descriptor creator and matching“ sowie „Outlier Removal“ werden in Kapitel 3.2, die „Pose estimation and refinement“ in Kapitel 3.3 beschrieben.

3.2 Feature & Image Matching

Um verschiedene Bilder miteinander in Beziehung setzen zu können, müssen die Features einer dreidimensionalen Szene in den verschiedenen Bildern abgeglichen werden. Seit einigen Jahren sind „Image Feature Detectors and Descriptors“ die am weitesten verbreiteten Techniken für diese Anwendungen. Zu dem Anwendungsbereich dieser Algorithmen zählen beispielsweise 3D-Rekonstruktion, Panoramaerstellung, Bildklassifizierung, Objekterkennung oder Roboterlokalisierung. Heute gibt es eine Vielzahl an verschiedenen Algorithmen, welche die Erkennung und Beschreibung von Features implementieren, um „Regions of Interest“, Ecken oder Kanten zu erhalten. (vgl. [35] S.3) Weiterhin sollte zwischen „Feature Detektors“ und „Feature Descriptors“ unterschieden werden. Detektoren sind Operationen, welche zweidimensionale Positionen im Bild suchen (ein Bildpunkt oder eine Region), die unter verschiedenen Transformatio-

nen stabil sind und einen hohen Informationsgehalt enthalten. Die Ergebnisse werden als „Interest Points“, affine Regionen, invariante Regionen, oder Ecken bezeichnet. Deskriptoren hingegen analysieren das Bild und liefern für einen bestimmten „Point of Interest“ im Bild einen 2D-Vektor mit Pixelinformationen. Diese Informationen sind dann Features, um diesen bestimmten Punkt im Bild zu klassifizieren und ihn mit anderen Features aus anderen Bildern zu vergleichen. (vgl. [36] S.1)

Die Eigenschaften eines guten Feature Matching Systems sind:

- **Invarianz**: Unabhängig von geometrischen oder radiometrischen Verzerrungen (Drehung, Skalierung, Rotation)
- **Stabilität**: Rrobust gegen Bildrauschen
- **Unterscheidbarkeit**: Klare Unterscheidbarkeit zum Hintergrund
- **Einzigartigkeit**: Jedes Feature ist zu jedem anderen einzigartig

Das Merkmalserkennung und -anpassung wird in drei Schritte unterteilt [35]:

(1) **Detektion**: Finden der Keypoints in jedem Bild.

(2) **Beschreibung**: Im Idealfall sollte die lokale Umgebung um ein Feature immer invariant gegenüber Skalierung, Drehung, Rauschen, Änderung der Beleuchtung oder affinen Transformationen sein. Die Merkmalsbeschreibungen für jeden Keypoint werden anhand seiner Nachbarschaft bestimmt. Zu jedem Keypoint wird ein „descriptor vector“ (Beschreibungsvektor) berechnet.

(3) **Matching**: Um ähnliche Merkmale zwischen verschiedenen Bildern zu identifizieren werden die Deskriptoren verglichen. Ist ein Feature in zwei Bildern enthalten, zeigen die Bilder das gleiche Objekt.

Die häufigsten Verwendeten Verfahren sind „Scale Invariant Feature Transform“ (SIFT) (Lowe, 2004), „Features from Accelerated Segment Test“ (FAST) (Rosten and Drummond, 2005), „Speeded Up Robust Features“ (SURF) (Bay et al., 2006),„Binary Robust Independent Elementary Features“ (BRIEF) (Calonder et al. 2010) oder „Oriented FAST and Rotated BRIEF “ (ORB) (Rublee et al. 2011). Diese unterscheiden sich hauptsächlich durch die erkannten Features im Bild, die getrackt werden sollen, sowie dem erzeugtem Modell der Umgebung. (vgl. [35] S.3, [15] S.28-29)

Die Erstellung und das Matching der Deskriptoren hängt von der Wahl des Deskriptorssystems ab.

3.2.1 Scale Invariant Feature Transform

Bei SIFT schätzt der Algorithmus die dominante Orientierung des Keypoints mit Hilfe von Gradienten und beschreibt anschließend die Keypoints in Bezug zu den Gradienten der Umgebung. Die Features werden in einem „Differce of Gaussian“ (DoG) Skalenraum erfasst, der die Differenz der „Gaussian convolutions“ (Gaußschen Faltung) des Originalbilds repräsentiert (Abbildung 3.3-a).

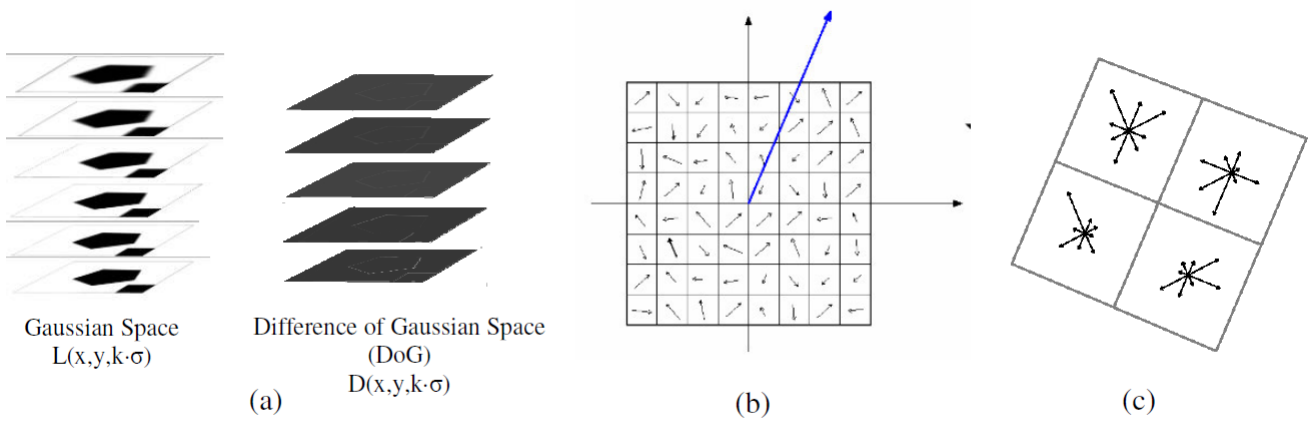

Abbildung 3.3: (a) Unterschied des Gaußschen (DoG) Skalenraums. (b) Überwiegende Ausrichtung des radiometrischen Gradienten. (c) SIFT Deskriptor. Bildquelle: [37]

Jedem lokalem Maximum der DoG Funktion ist eine dominante Orientierung des radiometrischen Gradienten zugewiesen, um die Invarianz gegenüber Rotation zu gewährleisten (Abbildung 3.3-b). Schließlich wird jedem Keypoint ein Deskriptor zugeordnet, welcher die extrahierten Features, welche die Helligkeitsinvarianz in der Umgebung zum Keypoint beschreiben, als 128-dimensionalen Vektor zusammenfasst (Abbildung 3.3-c). Den Deskriptor kann man als 3D-Histogramm der Lage und Orientierung des Gradienten verstehen. Die erkannten Deskriptoren werden in eine Datenbank gespeichert, in welcher dann gegen andere Bilder, auf Gemeinsamkeiten geprüft werden kann. Dies geschieht über die Auswertung des Mindestabstands zwischen zwei Deskriptoren. Dazu werden die Entferungen mithilfe des Euklidischen Abstands oder des Mahalanobis-Abstands berechnet. (vgl. [15] S.28-29, [36] S.3, [37] S.3)

3.2.2 Features from Accelerated Segment Test

FAST hat eine ähnliche Vorgehensweise, ist jedoch um ein vielfaches schneller als SIFT. Wie in Abbildung 2.4 zu sehen ist, arbeitet FAST mit einem Segementtest mit einem Kreis von 16 Pixeln um einen Eckenkandidaten p.

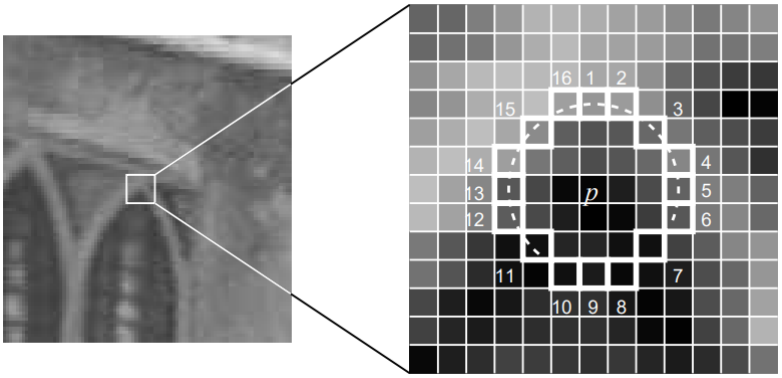

Abbildung 3.4: 12 Punkt Segment Test für die Eckenerkennung [19]

Der Detektor klassifiziert p als Ecke, wenn es ein Set von n zusammenhängenden Pixeln im Kreis gibt, die alle heller als der Kandidat I_p, addiert mit einem Grenzwert t, oder alle dunkler als $I_p - t$ sind. n ist zwölf, da dies einen High Speed Test ermöglicht, mit dem eine große Anzahl an Nicht-Ecken schnell ausgeschlossen werden kann. Der Test untersucht nur die vier Pixel 1, 5, 9 und 13. Wenn p eine Ecke ist, dann müssen mindestens drei der vier Pixel heller als $I_p + t$ oder dunkler als $I_p - t$ sein. Wenn dies nicht zutrifft, ist p keine Ecke. (vgl. [19] S.4-5)

3.2.3 Binary Robust Independent Elementary Features

Die Verwendung von BRIEF (Binary Robust Independent Elementary Features) Deskriptoren hat sich aus den steigenden Anforderungen an eine stärker einschränkende Arbeitsumgebung für rechenintensive Computer Vision basierte Systeme entwickelt. Ältere Methoden wie etwa SIFT oder SURF (Speeded Up Robust Features) neigen dazu bei einem großen Set an Features viel Speicher zu benötigen. Der 128-dimensionale Vektor mit dem beispielsweise SIFT die Features beschreibt ist für Echtzeitanwendungen zu rechenintensiv. SURF etwa speichert die Histogramme der lokalen Gradienten als Gleitkommazahlen in einem 64-dimensionalen Vektor. Für Echtzeitanwendungen ist SURF jedoch ebenfalls zu teuer, da die Anzahl an zu speichernden Deskriptoren

nicht gut skalierbar ist. BRIEF verwendet einen Ansatz, bei dem Bitvektoren aus Ausschnitten (Patches) des Bilds berechnet werden, nachdem diese geglättet wurden. Die Intensitätswerte der Pixel werden paarweise ausgewertet, um einzigartige oder gleiche Bits in jedem Bitvektor zu finden.

Für jeden geglätteten Bildpunkt wird ein Test τ für den Patch p durchgeführt (vgl. [42] S.3, [40] S.13-14):

$$\tau(p; x, y) = \begin{cases} 1, & p(x) < p(y) \\ 0, & otherwise \end{cases} \tag{3.1}$$

Wobei $p(x)$ die Intensität des Pixels bei $x = (u, v)^T$ im zu untersuchenden geglätteten Bild Patch ist. Die Auswahl eines Sets an $n_d(x, y)$-Bildpatches, definiert eindeutig ein Set an Binären Tests. Der BRIEF Deskriptor wird als n_d dimensionaler Bitstring definiert:

$$f_{n_d}(p) := \sum_{1 \geq i \geq n_d} 2^{i-1} \tau(p; x_i, y_i) \tag{3.2}$$

Die Metrik, um die Deskriptoren zu vergleichen, ist die Hamming-Distanz, welche die Unterschiedlichkeit zwischen zwei Zeichenketten, anhand der Auswertung der Anzahl verschiedener Zeichen für die gleichen Positionen in zwei Strings der selben Länge, beschreibt. (vgl. [40] S.13-14)

3.2.4 Outlier Removal

„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 dreidimensionale Lage zwischen zwei Bildebenen. 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.

$$s \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = H \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{3.3}$$

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

$$h_33 = 1 \tag{3.4}$$

oder

$$h^2{}_{11} + h^2{}_{12} + h^2{}_{13} + h^2{}_{21} + h^2{}_{22} + h^2{}_{23} + h^2{}_{31} + h^2{}_{32} + h^2{}_{33} = 1 \tag{3.5}$$

Homographie wird in vielen Anwendungsbereichen, wie Panoramaerstellung, Bildausrichtung, perspektivischer Entzerrung oder für die Schätzung der Kameraposition in Augmented Reality verwendet. (vgl. [16]) Die Resultate der Homographie werden als Ausgangspunkt für die „Pose Estimation“ (Positionsschätzung) der Kamera verwendet. Eric Marchand et al. [15] beschreiben die Positionsschätzung als Problem, welches ursprünglich in der Photogrammetrie ihren Ursprung fand und als „Space Resection“ bekannt ist. Sie definieren sie folgendermaßen: „given a set of correspondences between 3D features and their projections in the images plane, pose estimation consists in computing the position and orientation of the camera “. Bei photogrammetrischen Verfahren wird die Positionsschätzung heute hauptsächlich mit dem Bündelblockausgleich durchgeführt, der alle Parameter gleichzeitig in mehreren Iterationsschritten anpasst.

3.3 Der Bündelblockausgleich

Das Verfahren des Bündelblockausgeleichs, verwendet nun die Methoden der „collinearity condition “ (Kollinearitätsbedingung), der „coplanarity condition“ (Koplanaritätsbedingung) oder die Methode der direkten linearen Transformation. Die gewünschten Parameter aller Fotos werden gleichzeitig durch eine iterative Wiederholung der „least square“ Methode (Methode der kleinsten Quadrate) angepasst und korrigiert (Siehe Abbildung 3.5). Die Iterationen sind durch die nicht-Linearität der Konditionsgleichungen notwendig. Die Resulate des Bündelblockausgleichs aller Fotos sind dann die Ergebnisse der externen Orientierung der Kamera für jedes einzelne Foto. Weiterhin ergibt sich eine Auflistung der Objektraumkoordinaten der gemessenen Punkte aller Fotos, sowie deren gemessene statische Genauigkeit. (vgl. [23] S.66-67)

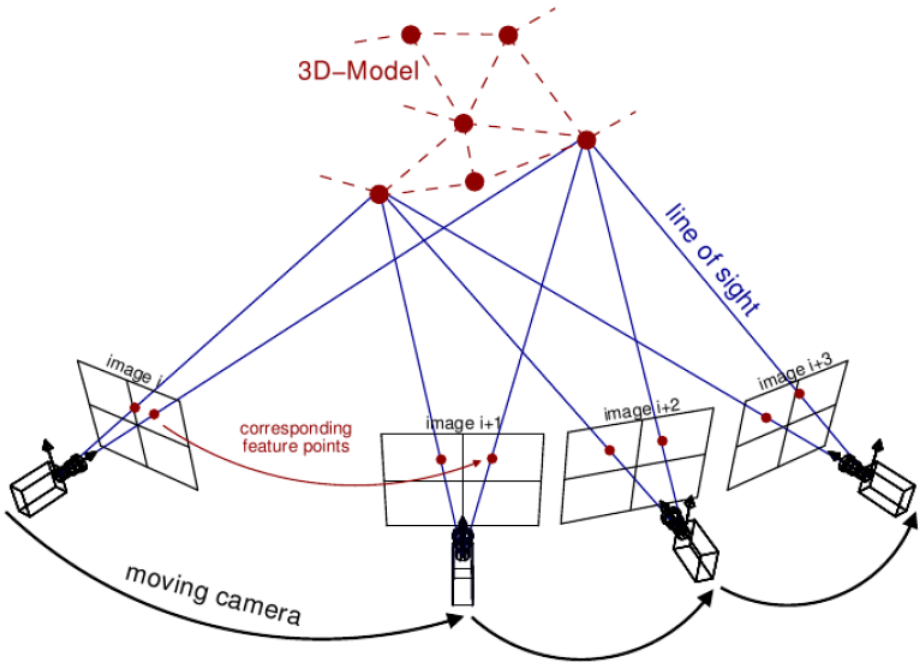

Abbildung 3.5: Visualisierung des Bündelblockausgleichs. Bildquelle [38]

Khalid El-AShmawy [23] beschreibt die Verwendung von Strahlenbündel, die durch Fotos generiert werden, als zweifelsfrei den flexibelsten Ansatz zur Blockbildung, Blockanpassung und für Photogrammetrie im Allgemeinen und mit den besten Ergebnissen. In der Nahbereichsphotogrammetrie, bei der mehrstufige und konvergente Konfigurationen möglich sind, ist der Bündelansatz in seiner stärksten Form vertreten. Der Ausgleich der Strahlenbündel in einem Set an Fotos beinhaltet die Rotation und Translation von jedem Bündel im Raum in eine Position, in der alle Strahlen sich an der korrekten Position im Objektraum schneiden. (vgl. [23] S.66)

Im Allgemeinen besteht ein Bündelblockausgleichsproblem aus den folgenden neun Elementen [34]:

(1) Ein Projektionsmodell, welches die Projektion von 3D-Objektpunkte in 2D-Bildpunkte beschreibt. Dazu gehören innere, sowie äußere Orientierung.

(2) Ein unbekannter Parametersatz, der bestimmt werden soll, dies können entweder innere oder äußere Parameter sowie Objektpunkte sein.

(3) Eine Reihe an Beobachtungen, typischerweise Messungen von 2D-Bildpunkten oder Kamerapositionen.

(4) Ein Reihe bekannter Parameter, wie innere Orientierung, oder Koordinaten von

Kontrollpunkten.

(5) Optional eine Reihe von Einschränkungen, zwischen Parametern oder Beobachtungen.

(6) Ein Verfahren, um die initialen Werte der unbekannten Parameter zu bestimmen.

(7) Ein Anpassungsalgorithmus, um die optimalen Parameterwerte anhand der initialen Werte zu finden.

(8) Ein Verfahren zur Erkennung von Ausreißern in den Beobachtungen.

(9) Eine Methode zur automatischen Erkennung von instabilen Parametern.

In den folgenden Abschnitten wird sich hauptsächlich auf Schritt (7) bezogen, dem Finden der optimalen Parameterwerte anhand der initialen Daten.

3.4 Nicht-lineare Methode der kleinsten Quadrate

Die „Method of Least Squares“ (Methode der kleinsten Quadrate) ist eine sehr leistungsfähige und flexible Technik, um alle Arten von Datenabgleichsproblemen zu lösen. Das Verfahren wird eingesetzt, um eine parametrisierbare Funktion an ein Datenset von Messwerten anzupassen, durch Minimierung der Summe des quadratischen Fehlers zwischen Funktion und Datenpunkten. Die verschiedenen Werkzeuge dieser Methode können auch eingesetzt werden, um beispielsweise die Korrelationsqualität der Messdaten zu bewerten. Gleichzeitig ermöglicht das System die Stabilisierung und Verbesserung der Korrelation, durch die Berücksichtigung geometrischer Randbedingungen. Wenn die anzupassende Funktion nicht linear ist, ist das Problem der kleinsten Quadrate ebenfalls nicht linear. Nichtlineare Lösungen der Methode der kleinsten Quadrate reduzieren iterativ die Summe der quadratischen Fehler zwischen Funktion und Messwerten, durch eine Abfolge von Aktualisierungen der Parameterwerte. (vgl. [29] S.1, [32] S.1)

In den folgenden Abschnitten werden verschiedene Methoden zur Bestimmung der externen Kameraparameter während des Bündelblockausgleichs vorgestellt. Die Methode der kleinsten Quadrate ist dabei essentiell für die Anwendbarkeit dieser Algorithmen, weswegen diese sowie das Gauss-Newton und das Levenberg-Marquardt Verfahren vorgestellt wird.

Die nicht lineare Methode der kleinsten Quadrate ist ein Standardoptimierungsproblem

und wird definiert als [**?**] :

$$minimiere : \sum_{i=1}^{m} f_i(x)^2 = ||f(x)||^2 \tag{3.6}$$

Wobei:

$f_1(x), ..., f_m(x)$ differenzierbare Funktionen einer Vektorvariablen x sind.

f eine Funktion von $\mathbf{R}^n$ zu $\mathbf{R}^m$ mit den Komponenten $f_i(x)$ ist:

$$f(x) = \begin{bmatrix} f_1(x) \\ f_2(x) \\ \cdots \\ f_m(x) \end{bmatrix} \tag{3.7}$$

Das **Kameramodell** wird beschrieben durch die Parameter: $A \in \mathbf{R}^{2\times3}, b \in \mathbf{R}^2, c \in \mathbf{R}^3, d \in \mathbf{R}$ welche die Position und Orientierung charakterisieren. Ein Objekt an Position $x \in \mathbf{R}^3$ erzeugt ein Bild an der Position $x' \in \mathbf{R}^2$ auf der Bildebene.

$$x\prime = \frac{1}{c^T x + d}(Ax + b) \tag{3.8}$$

$c^T x + d > 0$, wenn das Objekt vor der Kamera ist. Angenommen ein Objekt an Position x_{ex} wird von l Kameras betrachtet. (Beschrieben durch A_i, b_i, c_i, d_i) Das Bild des Objekts in der Bildebene von Kamera i ist an Position:

$$y_i = \frac{1}{c_i^T x_{ex} + d_i}(A_i x_{ex} + b_i) + v_i \tag{3.9}$$

Wobei v_i der Mess- oder Quantisierungsfehler ist. Das Ziel ist es nun die dreidimensionale Position e_{ex} von den l Beobachtungen $y_1, ..., y_l$ zu schätzen. Dies kann nun mit der nicht-linearen Methode der kleinsten Quadrate berechnet werden. $\hat{x}$ wird durch die Minimierung von (3.5) berechnet. (vgl. [**?**] S.5-6)

$$\sum_{i=1}^{l} ||y_i = \frac{1}{c_i^T x_{ex} + d_i}(A_i x_{ex} + b_i) + v_i||^2 \tag{3.10}$$

3.4.1 Gauß-Newton-Verfahren

Das Gauß-Newton Verfahren ist eine Technik zur Lösung der nicht linearen Methode der kleinsten Quadrate. Das Verfahren besteht aus einer Folge von linearen Annäherungen der kleinsten Quadrate an das nicht lineare Problem, bei dem jedes einzelne durch einen direkten oder iterativen Prozess gelöst wird. (vgl. [31] S.1) Gegeben ist die Definition der Problemstellung der kleinsten Quadrate, siehe (3.1). Beginnend mit einer anfänglichen Schätzung $x^{(1)}$, werden weitere Näherungslösungen $k = 1, 2, ...$ berechnet (vgl. [**?**] S.16-17).

Dazu wird f um $x^{(k)}$ linearisiert:

$$\overline{f}(x; x^{(k)}) = f(x^{(k)}) + Df(x^{(k)})(x - x^{(k)}) \tag{3.11}$$

Ersetze die affine Annäherung $\overline{f}(x; x^{(k)})$ für f im Problem der kleinsten Quadrate (3.1):

$$minimiere : ||\overline{f}(x; x^{(k)})||^2 \tag{3.12}$$

Die Lösung dieses linearen Problems wird nun als $x^{(k+1)}$ verwendet.

Das Problem der kleinsten Quadrate ist in Wiederholung k des Gauß-Newton Verfahrens gelöst:

$$minimiere : ||f(x^{(k)}) + Df(x^{(k)})(x - x^{(k)})||^2 \tag{3.13}$$

Wenn $Df(x^{(k)})$ linear unabhängige Spalten hat, wird die Lösung gegeben durch:

$$x^{k+1} = x^{(k)} - \Big(Df(x^{(k)})^T Df(x^{(k)})\Big)^{-1} Df(x^{(k)})^T f(x^{(k)}) \tag{3.14}$$

Der Gauß-Newton Schritt $\Delta x^{(k)} = x^{(k+1)} - x^{(k)}$ ist:

$$\begin{aligned} \Delta x^{(k)} &= -\Big(Df(x^{(k)})^T Df(x^{(k)})\Big)^{-1} Df(x^{(k)})^T f(x^{(k)}) \\ &= -\frac{1}{2}\Big(Df(x^{(k)})^T Df(x^{(k)})\Big)^{-1} \nabla g(x^{(k)}) \end{aligned} \tag{3.15}$$

Wobei $\nabla g(x)$ der Gradient der Kosten für nichtlineare kleinste Quadrate ist.

Das Gauß-Newton Verfahren eignet sich besonders gut für die Verarbeitung von großen Datenmengen mit hoher Varianz. Im Vergleich zum normalen Newton-Verfahren ist

der Algorithmus attraktiver, da er keine Auswertung der zweiten Ableitungen in der Hesse-Matrix der Zielfunktion benötigt. (vgl.[31] S.1, [**?**] S.16-17)

3.4.2 Levenberg-Marquardt Verfahren

Das Levenberg-Marquardt Verfahren wurde 1944 von Kenneth Levenberg [33] veröffentlicht, in den 1960er Jahren von Donald Marquardt wiederentdeckt und entwickelt um nicht lineare Probleme der kleinsten Quadrate zu lösen. Der Algorithmus kombiniert zwei Minimierungsmethoden: „Gradient descent" (Gradientenverfahren) und das Gauß-Newton Verfahren. Beim Gradientenverfahren wird die Summe der quadratischen Fehler durch die Aktualisierung der Parameter in Richtung der steilsten Richtung zum Minimum hin reduziert. Das Levenberg-Marquardt Verfahren verhält sich wie das Gradientenverfahren, wenn die Parameter weit von ihrem optimalen Wert entfernt sind und wie das Gauß-Newton Verfahren, wenn die Parameter nahe an ihrem optimalen Wert liegen. Es wechselt also adaptiv die Parameterupdates zwischen den beiden Verfahren. (vgl. [32] S.1)

Der Algorithmus befasst sich mit zwei Problembereichen des Gauß-Newton Verfahrens:

- Wie aktualisiert man $x^{(k)}$, wenn die Spalten von $Df(x^{(k)})$ linear Abhängig sind.
- Wie verfährt man, wenn das Gauß-Newton Update $||f(x)||^2$ nicht reduziert.

Das Verfahren wird mathematisch wie folgt beschrieben:

Berechnung von $x^{(k+1)}$ durch Lösung eines normalisierten Problems der kleinsten Quadrate:

$$minimiere : ||\overline{f}(x; x^{(k)})||^2 + \lambda^{(k)}||x - x^{(k)}||^2 \tag{3.16}$$

$\overline{f}(x; x^{(k)})$ ist dabei wie in Gleichung 3.6 gegeben. Mit $\lambda^{(k)} > 0$ gibt es immer eine eindeutige Lösung.

Das normalisierte Problem der kleinsten Quadrate wird in Iteration k gelöst:

$$minimiere : ||f(x^{(k)}) + Df(x^{(k)})(x - x^{(k)})||^2 + \lambda^{(k)}||x - x^{(k)}|^2 \tag{3.17}$$

Die Lösung ist gegeben durch:

$$x^{(k+1)} = x^{(k)} - \left(Df(x^{(k)})^T Df(x^{(k)}) + \lambda^{(k)} I\right)^{-1} Df(x^{(k)})^T f(x^{(k)}) \tag{3.18}$$

Der Levenberg-Marquardt Schritt $\Delta x^{(k)} = x^{(k+1)} - x^{(k)}$ ist:

$$\begin{aligned}\Delta x^{(k)} &= -\Big(Df(x^{(k)})^T Df(x^{(k)}) + \lambda^{(k)} I\Big)^{-1} Df(x^{(k)})^T f(x^{(k)}) \\ &= -\frac{1}{2}\Big(Df(x^{(k)})^T Df(x^{(k)}) + \lambda^{(k)} I\Big)^{-1} \nabla g(x^{(k)})\end{aligned} \tag{3.19}$$

Für $\lambda^{(k)} = 0$ ist das der Gauß-Newton Schritt; für große $\lambda(k)$:

$$\Delta x^{(k)} \approx -\frac{1}{2}\lambda^{(k)} \nabla g(x^{(k)}) \tag{3.20}$$

Es gibt verschiedene Strategien um $\lambda^{(k)}$ anzupassen:

- Nach Iteration k, berechne Lösung $\hat{x}$ von Gleichung (3.11)
- Wenn $||f(\hat{x})||^2 < ||f(x^{(k)})||^2$, verwende $x^{(k+1)} = \hat{x}$ und verringere λ
- Wenn $||f(\hat{x})||^2 > ||f(x^{(k)})||^2$, lasse x gleich (verwende $x^{(k+1)} = x^{(k)}$), und erhöhe λ

(vgl. [**?**] S.20-22)

3.4.3 Die Kollinearitätsbedingung

Die Bestimmung der externen Kameraparameter mit Hilfe der Kollinearitätsbedingung, um die gleichzeitige Anpassung der Parameter im Bündelblockausgleich zu berechnen, erfolgt durch die Aufstellung von zwei Gleichungen für jeden gemessenen Bildpunkt. Die Lösung all dieser Gleichungen erfolgt dann nach der Methode der kleinesten Quadrate. Die Bedingung der Kollinearität sagt aus, dass ein Objektpunkt P, sein Bildpunkt p und das perspektivische Zentrum O, alle auf der gleichen Geraden liegen müssen. Mathematisch wird die Bedingung wie folgt ausgedrückt [24]:

$$\begin{aligned}x_p &= -f\frac{(X_p - X_O)m_{11} + (Y_p - Y_O)m_{12} + (Z_p - Z_O)m_{13}}{(X_p - X_O)m_{31} + (Y_p - Y_O)m_{32} + (Z_p - Z_O)m_{33}} \\ y_p &= -f\frac{(X_p - X_O)m_{21} + (Y_p - Y_O)m_{22} + (Z_p - Z_O)m_{23}}{(X_p - X_O)m_{31} + (Y_p - Y_O)m_{32} + (Z_p - Z_O)m_{33}}\end{aligned} \tag{3.21}$$

Dabei sind x_p und y_p die korrigierten Foto Koordinaten, X_p, Y_p, Z_p die Objektpunktkoordinaten von P, X_O, Y_O, Z_O die Koordinaten des perspektivischen Zentrums O, f die

kalibrierte Brennweite der Kamera und m_{ij} die Elemente der Orientierungsmatrix M des Fotos.

Die linearisierte Form der Gleichung, für die Lösung der Methode der kleinsten Quadrate, wird gegeben als ([23] S.67):

$$V + B \cdot \triangle = \varepsilon \tag{3.22}$$

Wobei:

- $\triangle$ der Korrekturvektor zu dem aktuellen Werteset, für die unbekannten Werte (innere und äußere Orientierung, Objektkoordinaten der Punkte) der iterativen Lösung ist.
- B die Matrix der partiellen Ableitungen von Gleichung (3.1), in Bezug auf die Unbekannten ist.
- V der Korrekturvektor zu den Beobachtungen ist.
- ε der Abweichungsvektor ist.

El-Ashmawy [23] schlägt weitere Beschränkungen vor, indem ergänzende Beobachtungsgleichungen berücksichtigt werden, die sich aus den a priori Kenntnissen der Raumkoordinaten der Objekte der Kontrollpunkte in Gleichung (3.2) ergeben. Solche zusätzlichen Gleichungen können wie folgt beschrieben werden:

$$V^c - \triangle^c = \varepsilon^c \tag{3.23}$$

Wobei:

- $\triangle^c$ der Vektor der beobachtbaren Korrekturen zu den Objektkoordinaten der Kontrollpunkte ist.
- ε^c der Abweichungsvektor zwischen Beobachtungswerten und aktuellen (in iterativer Lösung) Werten der Objektkoordinaten der Kontrollpunkte ist.

Beobachtungsgleichungen können dann durch das Zusammenführen von Gleichung (3.2) und (3.3) erhalten werden. Die grundlegenden Voraussetzungen an den Bündelblockausgleich sind die Schätzungen der Parameter für innere und äußere Orientierung der Kamera. Weiterhin können die - je nach spezifischem Ansatz - Schätzungen für die Objekt und Raumkoordinaten aller Kontrollpunkte nützlich sein. Deshalb sollte ein Bündelverfahren immer eine praktikable Methode enthalten, mit der die ungefähren

geschätzten initialen Werte ermittelt werden können. Dieses Vorwissen sorgt nicht nur für eine reduzierte Anzahl an Iterationen, sondern resultiert auch in schnelleren und genaueren Ergebnissen. (vgl. [23] S.67)

3.4.4 Die Koplanaritätsbedingung

Die Koplanaritätsbedingung ist ein weiteres Verfahren zur Lösung des Bündelblockausgleichs und impliziert, dass die beiden perspektivischen Zentren von zwei Aufnahmen, ein beliebiger Objektpunkt und die entsprechenden Bildpunkte auf den beiden Fotos, alle in einer gemeinsamen Ebene liegen müssen (vgl. Abbildung 3.2). Die Koplanaritätsbedingung kann wie folgt dargestellt werden ([25] S.1204):

$$F_i = \begin{vmatrix} b_X & b_Y & b_Z \\ X_1 & Y_1 & Z_1 \\ X_2 & Y_2 & Z_2 \end{vmatrix} = 0 \tag{3.24}$$

Dabei sind b_X, b_Y, b_Z die Komponenten des Basisvektors b und X_1, Y_1, Z_1 sowie X_2, Y_2, Z_2 sind die Komponenten des Vektors $\vec{R}_1$ (von O_1 zu P) respektive $\vec{R}_2$ (von O_2 zu P).

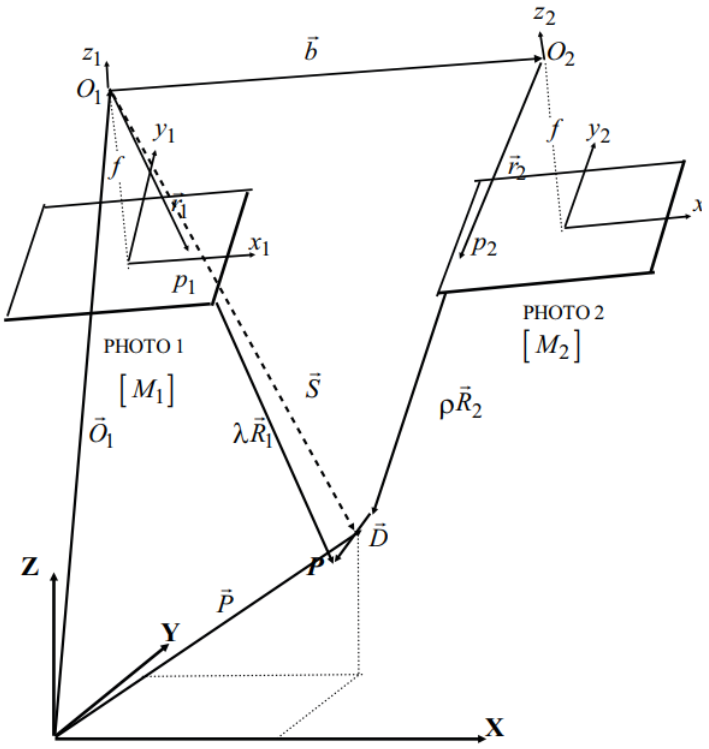

Abbildung 3.6: Koplanaritätsbedinung, Bildquelle [23]

Das mathematische Modell besteht aus vier skalaren Gleichungen:

$$\begin{aligned} X_p - (X_{O_1} + 0.5(b_X + \lambda \cdot X_1 + p \cdot X_2)) &= 0.0 \\ Y_p - (Y_{O_1} + 0.5(b_Y + \lambda \cdot Y_1 + p \cdot Y_2)) &= 0.0 \\ Z_p - (Z_{O_1} + 0.5(b_Z + \lambda \cdot Z_1 + p \cdot Z_2)) &= 0.0 \\ D_Y = \lambda \cdot X_1 - p \cdot X_2 - b_Y &= 0.0 \end{aligned} \tag{3.25}$$

Wobei $X_{O_1}, Y_{O_1}, Z_{O_1}$ die Objektkoordinaten der ersten Kameraposition während der Belichtung sind und λ und p die Skalierungsfaktoren der entsprechenden Positionsvektoren $\vec{r_1}$ und $\vec{r_2}$ im Kameraraum sind.

Die linearisierte Form von Gleichung (3.5), mit ebenfalls von El-Ashmawy [23] vorgeschlagenen zusätzlichen Beschränkungen, für das Verfahren der kleinsten Quadrate, wird gegeben als:

$$\begin{aligned} A \cdot V + B \cdot \triangle &= \varepsilon \\ V^c - \triangle^c &= \varepsilon^c \end{aligned} \tag{3.26}$$

Wobei:

- $\triangle$ der Korrekturvektor zu dem aktuellen Werteset, für die unbekannten Werte (innere und äußere Orientierung, Objektkoordinaten der Punkte) der iterativen Lösung ist.
- A die Matrix der partiellen Ableitungen von Gleichung (3.5), in Bezug auf die Beobachtungen (korrigierte Foto-Koordinaten auf den linken und rechten Fotos, des gleichen Objektpunkts) ist.
- B die Matrix der partiellen Ableitungen von Gleichung (3.5), in Bezug auf die Unbekannten ist.
- V der Korrekturvektor zu den Beobachtungen ist.
- ε der Abweichungsvektor ist.

Zur Verwendung der iterativen Lösung der kleinsten Quadrate, ist die Berechnung der Ausgangswerte von Unbekannten, wie beim Verfahren mit der Kollinearitätsbedinung, notwendig. (vgl. [23] S.68)

3.4.5 Das direkte lineare Transformationsverfahren

Die Methode der „direct linear transformation“ modelliert die Transformation zwischen Bildkoordinatensystem und Objektkoordinatensystem als lineare Funktion und wurde von Abdel-Aziz und Karara [28] eingeführt. Die direkte lineare Transformation kann aus den Kollinearitätsgleichungen abgeleitet werden und lassen sich mathematisch wie folgt ausdrücken ([27] S.72)

$$\begin{aligned} x &= \frac{L_1X + L_2Y + L_3Z + L4}{L_9X + L_{10}Y + L_{11}Z + 1} \\ y &= \frac{L_5X + L_6Y + L_7Z + L_8}{L_9X + L_{10}Y + L_{11}Z + 1} \end{aligned} \tag{3.27}$$

Wobei x, y die Bildkoordinaten, $L_1, ..., L_{11}$ die Transformationskoeffizienten und X, Y, Z die Objektkoordinaten des Punktes sind. Die Werte der inneren und externen Kameraparameter werden dann berechnet durch:

$$\begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \end{bmatrix} = - \begin{bmatrix} L_1 & L_2 & L_3 \\ L_5 & L_6 & L_7 \\ L_9 & L_{10} & L_{11} \end{bmatrix}^{-1} \begin{bmatrix} L_4 \\ L_8 \\ 1.0 \end{bmatrix} \tag{3.28}$$

$$\begin{aligned} x_0 &= (L_1L_9 + L_2L_{10} + L_3L_{11})/(L^2{}_9 + L^2{}_{10} + L^2{}_{11}); \\ y_0 &= (L_5L_9 + L_6L_{10} + L_7L_{11})/(L^2{}_9 + L^2{}_{10} + L^2{}_{11}); \\ \omega &= tan^{-1}(-L_{10}/L_{11}); \\ \phi &= sin^{-1}(-L_9\sqrt{(L^2{}_9 + L^2{}_{10} + L^2{}_{11})} \\ \kappa &= cos^{-1}((x_0L_9 - L_1)/(cos\phi\sqrt{(x_0L_9 - L_1)^2 + (x_0L_{10} - L_2)^2 + (x_0L_{11} - L_3)^2})); \\ f &= (x_0L_9)/(cos\kappa \cdot \phi\sqrt{L^2{}_9 + L^2{}_{10} + L^2{}_{11}}) \end{aligned} \tag{3.29}$$

Wobei $X_0, Y_0, Z_0, \omega, \phi, \kappa$ die externen Kameraparameter, x_0, y_0 die Bildkoordinaten des optischen Zentrums und f die Brennweite ist.

Das direkte lineare Transformationsverfahren hat lange Zeit in den Bereichen Photogrammetrie, Computer Vision, Robotik und Biomechanik Verwendung gefunden. Dies liegt an der linearen Formulierung der Beziehung zwischen Objekt- und Bildkoordinaten. Weiterhin können Bildkoordinaten in einem nicht-orthogonalem System, mit unterschiedlichen Skalen ausgedrückt werden und die Position des Koordinaten-

systems kann unbekannt, sowie die Brennweite beliebig sein und von Bild zu Bild variieren. (vgl. [27] S.72)

3.5 Structure from Motion

Structure from Motion (SfM) ist ein Oberbegriff aus der Photogrammetrie unter dem verschiedene Verfahren zusammengefasst werden können. Immer wenn es eine relative Bewegung zwischen Kamera und den realen 3D-Objekten gibt, werden die 3D-Objekte aus unterschiedlichen Blickrichtungen auf das 2D-Bild projiziert. Die Rekonstruktion der 3D-Struktur der Originalobjekte aus diesen 2D-Bildsequenzen wird als SfM bezeichnet. Structure from Motion ist nicht nur in der Photogrammetrie vertreten, sondern hat auch in der Computervision Anwendungsbereiche gefunden, wie etwa bei der 3D-Geometrie-Rekonstruktion oder bei SLAM. In den letzten Jahrzehnten sind verschiedene Ansätze zur Lösung dieses SfM Problems entstanden, wie die „Maximum likehood estimation“, „Kalman und Extended Kalman Filter“, „Particle Filter“ oder das „Hidden Markov Model“. (vgl. [38] S.5)

`https://www.sciencedirect.com/science/article/pii/S0169555X12004217`

gehört das noch dazu??

3.6 Depth Maps

gehört das noch dazu?

4 Simultaneous Localisation and Mapping

Simultaneous Localisation and Mapping, kurz SLAM, ist das Problem der Auswertung einer unbekannten Umgebung und Erstellung einer Karte, während gleichzeitig die lokale Position innerhalb dieser Karte 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. Wichtige Anwendungsbereiche von SLAM sind die automatische Fahrzeugsteuerung auf unbekannten Terrain, Rettungseinsätze in Umgebungen mit hohem Risiko, planetarische, terrestrische oder ozeanische Exploration, Augmented Reality Anwendungen, visuelle Überwachungssysteme oder in der Medizin. 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, [9] S.56)

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

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

Neue Features werden in der Umgebung entdeckt, welche in die Umgebungskarte aufgenommen werden müssen. Diese Features heißen „Landmarks“. Da die Position der Landmarks, durch Fehler in den exterozeptiven Sensoren genauso wie die aktuelle

Position der Kamera ungewiss ist, müssen diese beiden Faktoren passend arrangiert und werden. Eine automatisierte Lösung benötigt ein mathematisches Modell, das die Position der Landmarks anhand er Sensordaten bestimmt. Dies ist das „*Inverse Oberservation Model* “.

Schon gemappte Landmarks werden entdeckt welche dann verwendet werden um die eigene Position, sowie die aller anderen Landmarks zu korrigieren. Diese Operation reduziert die Unsicherheit über den aktuellen Standort, sowie die der Landmarks. Die automatisierte Lösung erfordert ein mathemathisches Modell, um die Werte der Messungen aus den prognostizierten Positionen der Landmarks und der aktuellen Position der Kamera berechnet. 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, organisiert und aktuell halten, sowie die korrekten Entscheidungen bei jedem Iterationsschritt 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 und Landmwarks in der beliebig großen Karte erkannt werden. Wegen der möglichen Komplexität von großen Maps ist dies auch eins der größten Hindernisse, wenn es um die Skalierbarkeit der Lösung des SLAM Problems geht. Wichtig ist, dass die Loop Closure Detection keine Falsch-Positiven Ergebnisse liefert, da dies die Integrität und Korrektheit der kompletten Karte beeinflussen würde. (vgl. [10] S.4)

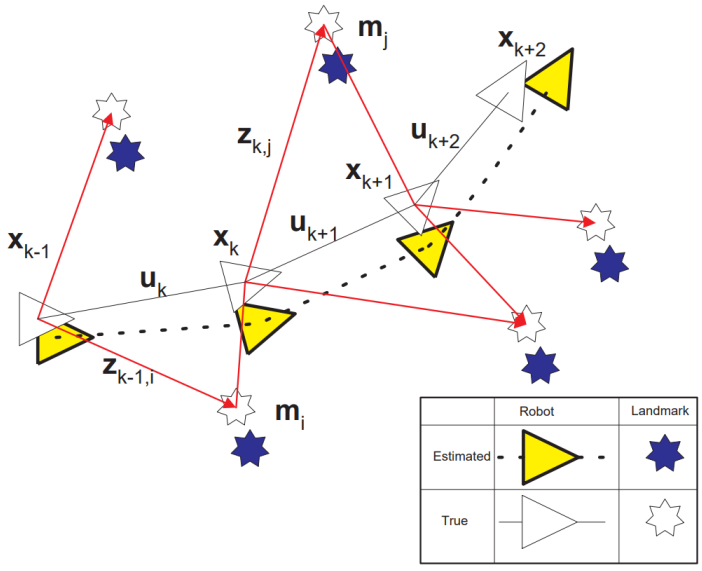

Abbildung 4.1: Das SLAM Problem: Die wahren absoluten Positionen der extrahierten Features sind nie wirklich bekannt. Bildquelle [2]

Wie in Abbildung 4.1. erkennbar ist, bewegt sich in diesem Beispiel ein Roboter durch eine unbekannte Umgebung und nimmt mit seinem Sensor Features der näheren Objekte (Landmarks) auf. Wobei x_k der Vektor des Roboters, u_k der Bewegungsvektor, m_i der Vektor des Landmarks und $z_i k$ 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, $m_i - m_j$ zueinander sehr genau sein kann, auch wenn die absolute Position sehr ungenau ist.

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

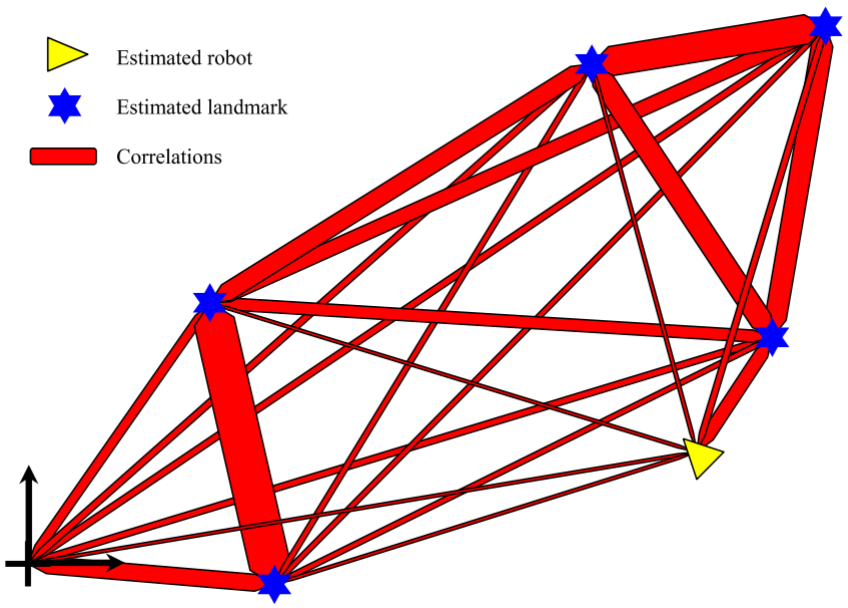

Abbildung 4.2: Die Landmarks sind durch „Federn“ verbunden, welche die Korrelation zwischen ihnen darstellen. Bildquelle [2]

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.

4.1 Loop closure detection

Die „Loop closure detection“ (Schleifenerkennung) besteht in der Erkennung von bereits besuchten Orten, nach der zyklischen Erkundung der Umwelt nach beliebiger Länge des Weges. Dies ist eins der größten Hindernisse um SLAM im großen Maßstab zu verwenden und der Vermeidung von kritischen Fehlern. Daraus ergibt sich ein weiteres Problem, das „perceptual aliasing“ genannt wird und bei dem zwei verschiedene Orte aus der Umgebung als ein gleicher Ort erkannt werden. Dies stellt vor allem bei sich wiederholenden Features in der Umgebung, wie Fluren, ähnlichen architektonischen Elementen oder Zonen mit vielen Büschen oder Bäumen ein großes Problem dar. Eine gute Methode zur Schleifenerkennung darf keine falsch-positiven Ergebnisse und nur ein Minimum an falsch-negativen Ergebnissen liefern. Deshalb müssen alle Loop Closure Systeme auf eine Präzision von 100 % abzielen. Dies wird benötigt, da ein einzelnes falsch-positives Ergebnis zu irreparablen Fehlern in der Erstellung der Karte führt. Falsch-negative hingegen reduzieren nur den Anteil der Rückrufquote, haben aber keinen Einfluss auf die Präzision der Genauigkeit der Karte. (vgl. [9] S.64-65)

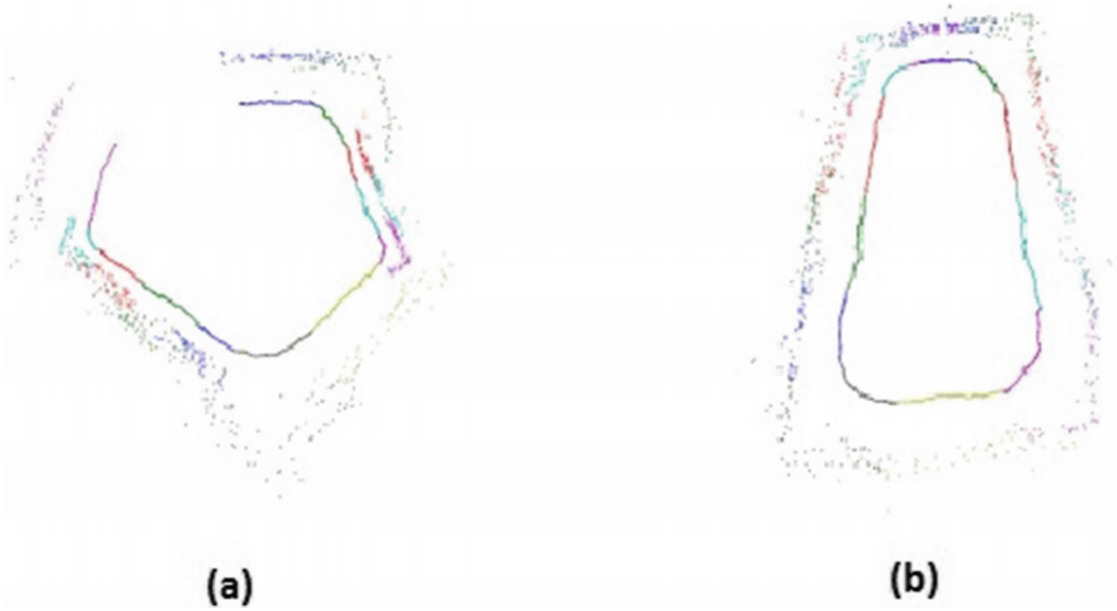

Abbildung 4.3: Lösung des SLAM Problems ohne (a) und mit (b) einem Loop closure Algorithmus. Die innere Kurve ist der Pfad des mobilen Systems, die äußeren Punkte stellen die Features in der Karte dar. Bildquelle [39]

Algorithmen für die Schleifenerkennung lassen sich in drei Gruppen unterteilen ([39]

S.212):

- **map-to-map**: Die Schleifenerkennung erfolgt durch die Aufteilung der globalen Karte in Teilkarten und dem finden von Übereinstimmungen zwischen diesen Teilkarten.
- **image-to-map**: Die Suche wird anhand von Übereinstimmungen zwischen Bild und Karte ermöglicht und die Position des Systems wird dann in Bezug zur Karte wieder hergestellt.
- **image-to-image**: Die Schleifenerkennung wird anhand von Bildfeatures zwischen verschiedenen Bildern durchgeführt.

Nach der Erkennung einer Schleife wird das globale Kartenmodell angepasst, um die gewonnenen Erkenntnisse in die Karte zu übernehmen. (vgl. [39] S.213)

4.2 Visual SLAM

Wenn Kameras als einziger exterozeptiver Sensor verwendet werden, spricht man von „Visual SLAM " oder „Vision-Based SLAM". Viele visuelle SLAM Ansätze haben jedoch Probleme wenn sie unter folgenden Bedingungen eingesetzt werden:

- In externen, dynamischen Umgebungen
- In Umgebungen mit sehr wenigen oder sehr vielen Markanten Merkmalen
- In sehr großflächigen Umgebungen
- Bei sehr unregelmäßigen Bewegungen der Kamera
- Bei partieller oder großflächiger Verdeckung des Sensors

Ein Schlüsselelement von erfolgreichen SLAM Systemen ist die Fähigkeit trotz dieser Schwierigkeiten korrekt zu arbeiten. (vgl. [9] S.56)

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), „FAST-SLAM“ und „ORB-SLAM“(vgl. [7] S. 2). Im folgenden wird der Extended Kalman Filter vorgestellt, der in vielen SLAM Systemen zum Einsatz gekommen ist, sowie seine Weiterentwicklungen FAST-SLAM und ORB-SLAM.

4.3 Extended Kalman Filter - SLAM

Eine der ersten Methoden zur Lösung des SLAM Problems war die Verwendung von EKF-SLAM, welches den Extendet Kalman Filter verwendet. Diese Methode wurde 1990 entwickelt ([40] S.1). 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 Extended Kalman Filter verwedet für die Vorhersage der Messungen und der Zustände hingegen nichtlineare Funktionen. (vgl. [8] S.16-17) Beim 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`

4.4 FAST-SLAM

Eine SLAM Methode, die sich aus dem EKF-SLAM entwicklelt hat ist FAST-SLAM ([40] S.1). FAST-SLAM (Features from Accelerated Segment Test 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 ver-

wendet, 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 Fast-SLAM 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)

4.5 ORB-SLAM

ORB-SLAM (Oriented FAST and Rotated BRIEF - SLAM) ist eine Weiterentwicklung und Kombination von den Feature Erkennugsalgorithmen FAST und BRIEF. ORB Deskriptoren sind sehr schnell berechnet und können sehr schnell verglichen werden, während sie gleichzeitig einen gute perspektivische Invarianz besitzen. ([41] S.3) Der erste Schritt von ORB-SLAM ist es ORB-Features aus dem Bild zu extrahieren. Dies geschieht durch das Finden von Ecken mit FAST und der anschließenden Erstellung eines Deskriptors mit BRIEF.

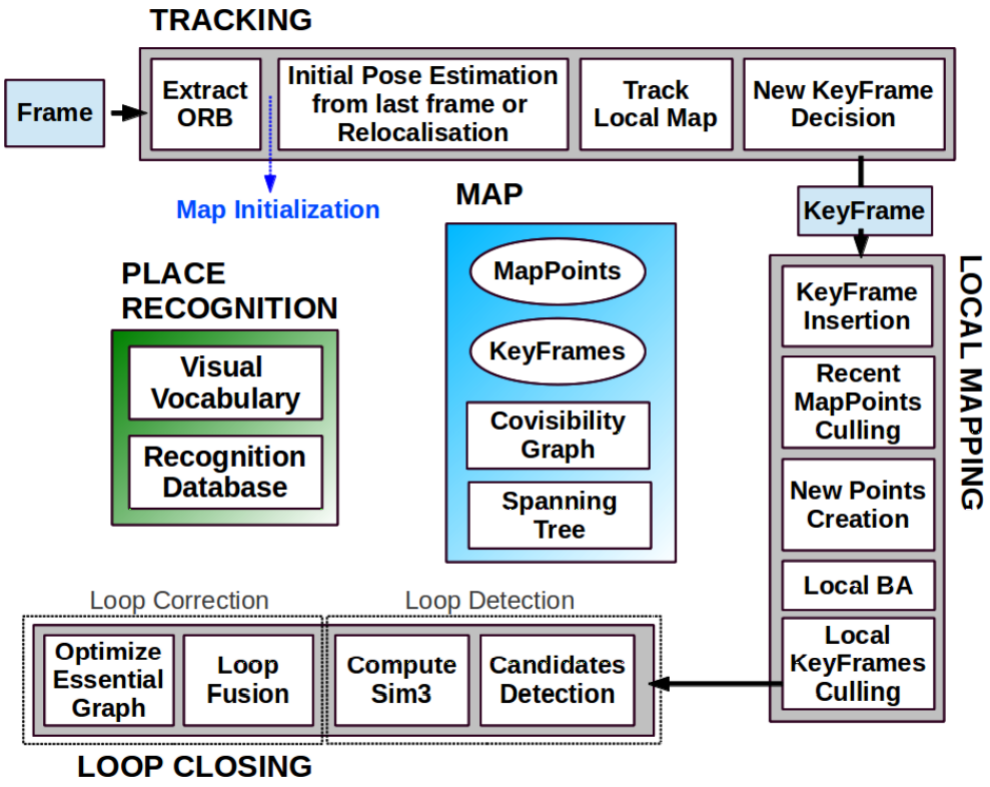

Abbildung 4.4: ORB-SLAM System Übersicht: Tracking, Mapping und Loop Closing. Bildquelle [41]

ORB-SLAM arbeitet mit verschiedenen Threads, um das Tracking, Mapping und Loop Closing parallel auszuführen.

Der **Tracking** Thread ist für die Lokalisierung der Kamera zuständig und entscheidet auch, welche Keyframes verwendet werden. Keyframes sind die Menge der ausgewählten Frames, die für die Erstellung der Map verwendet werden. Dabei wird mit einem „Survival of the Fittest“ Ansatz gearbeitet, der die nützlichsten Keyframes findet. Die initiale Positionsschätzung wird durch das Feature Matching zwischen ausgewähltem und vorherigem Keyframe erreicht. Danach wird die aktuelle Position mithilfe eines Bewegungsmodells optimiert, um die korrespondierenden Map Points auf dem Keyframe vorherzusagen und die 3D-Rekonstruktion zu optimieren. Wenn das Tracking eines vorherigen Frames verloren geht, wird ein globaler Lokalisierungsalgorithmus verwendet, um ein neues Keyframe zu finden, dass dann wieder mit dem aktuellen Frame verglichen werden kann. Ein erfolgreicher Tracking Schritt resultiert in dem Finden einer Kamerapose und einem initalen Set an Features, aus denen dann eine lokale Karte berechnet werden kann.

Die **Map** wird aus den ermittelten Keyframes und den darin enthaltenen Map Points im zweiten Thread erstellt. Wenn ein Keyframe erfasst wird, wird es in einem Graphen abgelegt, der Keyframes als Nodes und die gemeinsam beobachteten Map Points in den verschiedenen Keyframes als Verbindungen zwischen den Nodes modelliert. Die Gewichtung der Kanten ist die Anzahl der Map Points, die auf beiden Keyframes observiert wurden. Um eine unbeschränktes Wachstum des Graphen zu vermeiden werden Keyframes aussortiert, die im Vergleich zu anderen Keyframes zu ähnlich sind. Map Points werden ebenfalls aussortiert, wenn sie gleichzeitig von zu wenigen Keyframes erfasst werden. Auf diese Weise können Keyframes sehr großzügig in den Graphen eingefügt werden und bei Redundanz wieder entfernt werden.

Das **Loop Closing** wird vom dritten Thread ausgeführt. Wenn erkannt wird, dass ein bestimmter Ort schon einmal besucht wurde, kann dann der Drift, der sich beim Durchlaufen der Schleife angesammelt hat korrigiert werden. Bei ORB-SLAM wird die Schleifenerkennung beim Einfügen jedes Keyframes in den Graphen vorgenommen. Dazu wird die Ähnlichkeit zwischen den Keyframes betrachtet, indem ein Ähnlichkeitswert zwischen den Keyframes, mithilfe der Repräsentation des „Bag-of-words“ Modells, berechnet wird. Wenn ein Keyframe mehr Ähnlichkeiten mit einem neu eingefügten Keyframe aufweist, als seine Nachbarn im Graphen, ist es ein Kandidat für die Loop Closure. Wenn drei miteinander verbundene Kandidaten gefunden wurden, gilt der Loop als akzeptiert und die Schleife wird geschlossen. Abschließend wird der Graph aktualisiert. (vgl. [41] S.6-8, [40] S.4-5)

http://publications.lib.chalmers.se/records/fulltext/256291/256291.pdf

ORB SLAM: https://arxiv.org/pdf/1502.00956.pdf

ORB SLAM 2: https://arxiv.org/pdf/1610.06475.pdf

4.6 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, an-

hand 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)

Folgende Tabelle gibt eine Übersicht an Software Development Kits, die SLAM implementiert haben.

	Vuforia	Wikitude	Metaio	ARToolKit	Kudan	EasyAR	MaxST	ARCore	ARKit
SLAM	✓	✓	✓	x	✓	✓	✓	✓	✓
Open Source	x	x	x	✓	x	x	x	✓	x

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

5 Vergleich von Photogrammetrie und SLAM

5.1 Ähnlichkeiten und Unterschiede

Es ist inzwischen allgemein bekannt, dass Photogrammetrie und geometrische Computer Vision zwei eng zusammenhängende Disziplinen sind. Sie haben viele ähnliche Aufgabenstellungen und Ziele, wie Kalibrierung, Orientierung und Rekonstruktion. Viele Arbeiten und Forschungen beziehen sich auf beide Gebiete, wie relative Orientierung (Philip, 1996; Nistèr, 2004), die räumliche Analyse von Einzelbildern (Masry, 1981; Lepetit et al., 2009), Feature Erkennung (Förstner & Gülch, 1986; Lowe, 2004) oder etwa der Bündelblockausgeleich (Triggs et al., 2000). Dabei sollte beachtet werden, dass viele dieser Probleme erst in der Photogrammetrie untersucht und beschrieben worden sind und erst später in der Computer Vision signifikant weiter entwickelt wurden. Dies hat die Kommunikation zwischen beiden Fachbereichen gefördert. (vgl. [43] S.93)

`https://elib.uni-stuttgart.de/bitstream/11682/3934/1/Tang_Uni.pdf`

Seite 93, Unterschiede und gemeinsamkeiten

5.2 Analyse der Echtzeitfähigkeit

`https://phowo.ifp.uni-stuttgart.de/publications/phowo05/280foerstner.pdf`

6 Implementation einer AR Anwendung für Android

Im Rahmen dieser Arbeit ist eine Applikation für Android entstanden.

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 3-D Rekonstruktion

https://www.researchgate.net/publication/284013055_Recent_developments_in_large-scale_tie-point_matching

9 Zusammenfassung und Ausblick

Fortschritte in der Photogrammetrie sind viel zu sehr mit den Fortschritten der Computer Vision verflochten, als dass die Konvergenz der beiden Disziplinen sich umkehren wird. Photogrammetrie und Computer Vision haben die Extraktion und Rekonstruktion von Daten aus Bildmaterial zu unterschiedlichen Zeiten und mit unterschiedlichen Zielen begonnen. Als sich dann 3D-Modelle als Referenzziel herausstellten, wurde der Austausch von Ansätzen und Techniken zwischen den Disziplinen vorangetrieben. (vgl. [26] S.9) Dies hat dazu geführt, dass Photogrammetrie und Computer Vision verfahrenstechnisch kaum mehr unterscheidbar sind.

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)

[17] M. Maidi, J.Y. Didier, F. Ababsa, M. Mallem (2008), A performance study for camera pose estimation using visual marker based tracking. URL:

https://www.academia.edu/13152554/A_performance_study_for_camera_pose_estimation_using_visual_marker_based_tracking (Zuletzt aufgerufen am 18.07.2019)

[18] A. Pinz, M. Brandner, H. Ganster, A. Kusej, P. Lang, M. Ribo (2002) Hybrid Tracking for Augmented Reality. URL: https://www.researchgate.net/publication/229025765_Hybrid_tracking_for_augmented_reality (Zuletzt aufgerufen am 18.07.2019)

[19] E. Rosten, T. Drummond (2006) Machine learning for high-speed corner detection. URL: https://www.edwardrosten.com/work/rosten_2006_machine.pdf (Zuletzt aufgerufen am 18.07.2019)

[20] K. Schindler (2014) Mathematical Foundations of Photogrammetry. URL: https://ethz.ch/content/dam/ethz/special-interest/baug/igp/photogrammetry-remote-sensing-dam/documents/pdf/math-of-photogrammetry.pdf (Zuletzt aufgerufen am 19.07.2019)

[21] A.S. Alturki, J.S. Loomias (2016) Camera Principal Point Estimation from Vanishing Points. DOI: 10.1109/NAECON.2016.7856820 (Zuletzt aufgerufen am 19.07.2019)

[22] P. Grussenmeyer, O. Al Khalil (2002) Solutions for Exterior Orientation in Photogrammetry: A Review. Photogrammetric Record, 17(100):615-634. URL: https://hal.archives-ouvertes.fr/hal-00276983/document (Zuletzt aufgerufen am 19.07.2019)

[23] K.L.A. El-Ashmawy (2015). A comparison study between collinearity condition, coplanarity condition, and direct linear transformation (DLT) method for camera exterior orientation parameters determination. Geodesy and Cartography, 41(2), 66–73. URL:https://journals.vgtu.lt/index.php/GAC/article/view/2837/2334 (Zuletzt aufgerufen am 19.07.2019)

[24] E.E. Elnima (2015) A solution for exterior and relative orientation in photogrammetry, a genetic evolution approach. Journal of King Saud University – Engineering Sciences. URL: https://core.ac.uk/download/pdf/82822280.pdf (Zuletzt aufgerufen am 22.07.2019)

[25] C. Li, Y. Zhao (2012) Approach of Camera Relative Pose Estimation Based on Epipolar Geometry. Information Technology Journal, 11: 1202-1210. URL: https://

scialert.net/fulltext/?doi=itj.2012.1202.1210&org=11 (Zuletzt aufgerufen am 22.07.2019)

[26] G. Forlani, R. Roncella, C. Nardinocchi (2015) Where is photogrammetry heading to? State of the art and trends. Rendiconti Lincei, 26(S1), 85–96. DOI:10.1007/s12210-015-0381-x (Zuletzt aufgerufen am 24.07.2019)

[27] K. L. El-Ashmawy (2018) Using direct linear Transformation (DLT) Method for aerial Photogrammetry Applications. Geodesy and Cartography 2018 Volume 44 Issue 3: 71–79. URL: https://journals.vgtu.lt/index.php/GAC/article/download/1629/5048 (Zuletzt aufgerufen am 24.07.2019)

[28] Y.I. Abdel-Aziz, H. M. Karara (1971) Direct linear transformation into object space coordinates in close-range photogrammetry. In Proceedings Symposium on Close Range Photogrammetry (pp. 1-18). Urbana, Illinois

[29] A. W. Gruen (1985) Adaptive least Squares Correlations: A powerful Image Matching Technique. URL: https://www.researchgate.net/publication/265292615_Adaptive_Least_Squares_Correlation_A_powerful_image_matching_technique (Zuletzt aufgerufen am 24.07.2019)

[30] S. Boyd (2016) Nonlinear Least Squares. URL: https://stanford.edu/class/ee103/lectures/nlls_slides.pdf (Zuletzt aufgerufen am 24.07.2019)

[31] S. Gratton, A. S. Lawless, N. K. Nichols (2007) Approximate Gauss–Newton Methods for Nonlinear Least Squares Problems. URL: https://www.researchgate.net/publication/220133629_Approximate_Gauss-Newton_Methods_for_Nonlinear_Least_Squares_Problems (Zuletzt aufgerufen am 30.07.2019)

[32] H. P. Gavin (2019) The Levenberg-Marquardt algorithm for nonlinear least squares curve-fitting problems. URL: http://people.duke.edu/~hpgavin/ce281/lm.pdf (Zuletzt aufgerufen am 30.07.2019)

[33] K. Levenberg (1944) A method for the solution of certain non-linear problems in least squares. URL: https://www.ams.org/journals/qam/1944-02-02/S0033-569X-1944-10666-0/ (Zuletzt aufgerufen am 30.07.2019)

[34] N. Börlin, P. Grussenmeyer (2013) Bundle Adjustment With and Without Damping. The Photogrammetric Record, 28(144), 396–415. DOI: 10.1111/phor.12037 (Zuletzt aufgerufen am 07.08.2019)

[35] A. Abbas, S. Ghuffar (2018) Robust Feature Matching in terrestrial

Image Sequences. The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume XLII-3, URL: https://www.researchgate.net/publication/324855190_ROBUST_FEATURE_MATCHING_IN_TERRESTRIAL_IMAGE_SEQUENCES (Zuletzt aufgerufen am 07.08.2019)

[36] F. Remondino (2006) Detectors and descriptors for photogrammetric applications. URL: http://3dom.fbk.eu/sites/3dom.fbk.eu/files/pdf/remondino_ISPRS_III_06.pdf (Zuletzt aufgerufen am 07.08.2019)

[37] A. Lingua, D. Marenchio, F. Nex (2009) A comparison between "old and new" feature extraction and matching techniques in Photogrammetry. URL: https://pdfs.semanticscholar.org/b9c3/067b6fc111e1cc02f42357d5fb459a642152.pdf (Zuletzt aufgerufen am 07.08.2019)

[38] Z. Qu (2018) Efficient Optimization for Robust Bundle Adjustment. URL: https://vision.in.tum.de/_media/members/demmeln/qu2018msc.pdf (Zuletzt aufgerufen am 08.08.2019)

[39] A. Bokovoy, K. Yakovlev (2017) Original Loop-Closure Detection Algorithm for Monocular vSLAM. Analysis of Images, Social Networks and Texts, 210–220. URL: https://arxiv.org/pdf/1707.04771.pdf (Zuletzt aufgerufen am 08.08.2019)

[40] M. Andersson, M. Baerveldt (2018) Simultaneous localization and mapping for cars using ORB2-SLAM. URL: http://publications.lib.chalmers.se/records/fulltext/256291/256291.pdf (Zuletzt aufgerufen am 08.08.2019)

[41] R. Mur-Artal, J. M. M, Montiel (2015) ORB-SLAM: a Versatile and Accurate Monocular SLAM System. URL: https://arxiv.org/pdf/1502.00956.pdf(Zuletzt aufgerufen am 08.08.2019)

[42] M. Calonder, V. Lepetit, C. Strecha, P. Fua (2010) BRIEF: Binary Robust Independent Elementary Features. URL: https://www.researchgate.net/publication/221304115_BRIEF_Binary_Robust_Independent_Elementary_Features (Zuletzt aufgerufen am 08.08.2019)

[43] R. Tang (2013) Mathematical Methods for Camera Self-Calibration in Photogrammetry and Computer Vision. URL: https://elib.uni-stuttgart.de/bitstream/11682/3934/1/Tang_Uni.pdf (Zuletzt aufgerufen am 12.08.2019)

Abbildungsverzeichnis

2.1 Das Realität - Virtualität Kontinuum, Bildquelle [11] 3
2.2 Kantenbasiertes rekursives Tracking, Bildquelle [14] S.3 7

3.1 Kamera Kalibierungsmodell, Bildquelle [21] 9
3.2 Tracking Pipeline Bildquelle [15] 11
3.3 (a) Unterschied des Gaußschen (DoG) Skalenraums. (b) Überwiegende Aurichtung des radiometrischen Gradienten. (c) SIFT Deskriptor. Bildquelle: [37] 13
3.4 12 Punkt Segment Test für die Eckenerkennung [19] 14
3.5 Visualisierung des Bündelblockausgleichs. Bildquelle [38] 17
3.6 Koplanaritätsbedinung, Bildquelle [23] 24

4.1 Das SLAM Problem: Die wahren absoluten Positionen der extrahierten Features sind nie wirklich bekannt. Bildquelle [2] 29
4.2 Die Landmarks sind durch „Federn“ verbunden, welche die Korrelation zwischen ihnen darstellen. Bildquelle [2] 30
4.3 Lösung des SLAM Problems ohne (a) und mit (b) einem Loop closure Algorithmus. Die innere Kurve ist der Pfad des mobilen Systems, die äußeren Punkte stellen die Features in der Karte dar. Bildquelle [39] . . 31
4.4 ORB-SLAM System Übersicht: Tracking, Mapping und Loop Closing. Bildquelle [41] 36