Erarbeitung geeigneter Optimierungskriterien zur Berechnung von Kameraparametern und Szenengeometrie aus Bildfolgen

Diplomarbeit im Fach Informatik

vorgelegt
von

Jochen Schmidt
geb. am 26. Mai 1972 in Kronach

Angefertigt am

Lehrstuhl für Mustererkennung (Informatik 5)
Institut für Mathematische Maschinen und Datenverarbeitung
Friedrich-Alexander-Universität Erlangen-Nürnberg

Betreuer: Benno Heigl, Joachim Hornegger

Beginn der Arbeit: 24. November 1999

Abgabe der Arbeit: 24. Mai 2000

Inhaltsverzeichnis

1 Einleitung **7**
- 1.1 Das Lichtfeld . . . 7
- 1.2 Struktur aus Bewegung . . . 10
- 1.3 Aufgabenstellung . . . 11
- 1.4 Literatur . . . 12
- 1.5 Aufbau der Arbeit . . . 13

2 Grundlagen **15**
- 2.1 Notation . . . 15
- 2.2 Kameraparameter . . . 16
- 2.3 Selbstkalibrierung . . . 17

3 Optimierungskriterien **21**
- 3.1 Überblick . . . 21
- 3.2 Optimalität des Bündelausgleichs . . . 23
  - 3.2.1 Optimalität der Maximum-Likelihood-Schätzung . . . 23
  - 3.2.2 Bündelausgleich als Maximum-Likelihood-Schätzung . . . 25

4 Bündelausgleich **29**
- 4.1 Einführung und Problemstellung . . . 29
- 4.2 Parametrisierung . . . 30
  - 4.2.1 Intrinsische Parameter, Translation und Weltpunkte . . . 31
  - 4.2.2 Rotation . . . 32
- 4.3 Lösung des Minimierungsproblems . . . 38
  - 4.3.1 Zwei Ansätze zur Minimierung . . . 39
  - 4.3.2 Das Gauß-Newton-Verfahren . . . 40
  - 4.3.3 Die Levenberg-Marquardt-Erweiterung . . . 41
  - 4.3.4 Die Jacobi-Matrix . . . 43
- 4.4 Berücksichtigung von nicht sichtbaren Punkten . . . 45

5 Erweiterungen des Verfahrens **47**
- 5.1 Gewichtung des Fehlers . . . 47
- 5.2 Korrektur von Linsenverzerrungen . . . 48
  - 5.2.1 Einführung . . . 48
  - 5.2.2 Modellierung von Linsenverzerrungen . . . 49
  - 5.2.3 Berücksichtigung beim Bündelausgleich . . . 52

5.2.4 Simulation von Linsenverzerrungen 52

6 Untersuchungen zur Leistung des Bündelausgleichs 55
6.1 Berechnung der Startwerte für den Bündelausgleich 55
6.2 Experimente mit synthetischen Daten 57
6.2.1 Anzahl der nötigen Iterationsschritte 61
6.2.2 Einfluss des Rauschens auf den Bildpunkten 67
6.2.3 Unterschiedliche Parametrisierung der Rotation 70
6.2.4 Korrektur von Linsenverzerrungen 72
6.2.5 Verhalten unabhängig von der vorangegangenen Rekonstruktion 76
6.3 Experimente mit realen Daten 81

7 Zusammenfassung und Ausblick 85

Literaturverzeichnis 89

A Quaternionen 93

B Singulärwertzerlegung 97

C Berechnung der Kameraparameter 101

D Tabellen 103
D.1 Anzahl der nötigen Iterationsschritte 103
D.2 Einfluss des Rauschens auf den Bildpunkten 109
D.3 Unterschiedliche Parametrisierung der Rotation 113
D.4 Korrektur von Linsenverzerrungen 116
D.5 Unabhängigkeit vom vorher laufenden Verfahren 118
D.5.1 Verrauschte Parameter 118
D.5.2 Selbstkalibrierung 121

Abbildungsverzeichnis

1.1 Bildqualität bei der Lichtfeldrekonstruktion 8
1.2 Lichtfeld: Approximation der Szenengeometrie 9

4.1 Rotation um Drehachse $\boldsymbol{a}$ mit dem Winkel Θ. 32
4.2 Parametrisierung der Rotation mit Quaternionen 35
4.3 Bestimmung der Koordinaten des gesuchten Quaternions 38
4.4 Ein Levenberg-Marquardt-Iterationsschritt 42

5.1 Linsenverzerrung 51

6.1 Veränderung des Fehlers bei Orthogonalisierung der Rotationsmatrix 57
6.2 Szenenaufbau in der Simulation 59
6.3 Rückprojektionsfehler/Anzahl der Iterationen, Szene 1, $\sigma = 0.3$ Pixel 62
6.4 Rückprojektionsfehler/Anzahl der Iterationen, Szenen 1 und 7, jeweils $\sigma = 1.0$ Pixel 63
6.5 Rückprojektionsfehler/Anzahl der Iterationen, Szene 5, $\sigma = 0.3$ Pixel 64
6.6 Fehler in der Rotation, Szene 1 64
6.7 Fehler in der Rotation, Szenen 5 und 7 65
6.8 Fehler in den 3-D-Punkten, Szene 1 65
6.9 Fehler in den 3-D-Punkten, Szenen 5 und 7 66
6.10 Fehler in den intrinsischen Parametern, Szene 1 67
6.11 Rückprojektionsfehler Szene 1 69
6.12 Rückprojektionsfehler Szene 5 69
6.13 Rückprojektionsfehler Szene 7 70
6.14 Rückprojektionsfehler Szenen 1 und 7 / Linsenverzerrungen 76
6.15 Echte Szene: Karten und Tuch 81
6.16 Rückprojektionsfehler echte Szenen 83
6.17 Echte Szenen: Karten und Tuch – Rekonstruktion 83
6.18 Kamerapositionen vor und nach der Optimierung 84

Tabellenverzeichnis

6.1 Szene 1: Anzahl Iterationen, $\sigma = 0.3$ Pixel 62
6.2 Szene 1: Rauschen verschiedener Stärke 68
6.3 Szene 5: Rauschen verschiedener Stärke 68
6.4 Szene 7: Rauschen verschiedener Stärke 68
6.5 Parametrisierung der Rotation – Auswahl 71
6.6 Prozentuale Änderung Rotation/3-D-Punkte 72
6.7 Szene 1: Korrektur von Linsenverzerrungen 74
6.8 Szene 7: Korrektur von Linsenverzerrungen 74
6.9 Szenen 1, 7 und 8: Korrektur von Linsenverzerrungen 75
6.10 Optimierung mit tatsächlichen Matrizen als Startwerte 78
6.11 Szene 1: Verrauschte Parameter 79
6.12 Szene 5: Verrauschte Parameter 79
6.13 Selbstkalibrierungseffekt beim Bündelausgleich 80
6.14 Echte Szenen: Rückprojektionsfehler 82

D.1 Szene 1: Anzahl Iterationen, $\sigma = 0.3$ 103
D.2 Szene 1: Anzahl Iterationen, $\sigma = 1.0$ 104
D.3 Szene 2: Anzahl Iterationen, $\sigma = 0.3$ 104
D.4 Szene 2: Anzahl Iterationen, $\sigma = 1.0$ 104
D.5 Szene 3: Anzahl Iterationen, $\sigma = 0.3$ 105
D.6 Szene 3: Anzahl Iterationen, $\sigma = 1.0$ 105
D.7 Szene 4: Anzahl Iterationen, $\sigma = 0.3$ 105
D.8 Szene 4: Anzahl Iterationen, $\sigma = 1.0$ 106
D.9 Szene 5: Anzahl Iterationen, $\sigma = 0.3$ 106
D.10 Szene 5: Anzahl Iterationen, $\sigma = 1.0$ 106
D.11 Szene 6: Anzahl Iterationen, $\sigma = 0.3$ 107
D.12 Szene 6: Anzahl Iterationen, $\sigma = 1.0$ 107
D.13 Szene 7: Anzahl Iterationen, $\sigma = 0.3$ 107
D.14 Szene 7: Anzahl Iterationen, $\sigma = 1.0$ 108
D.15 Szene 8: Anzahl Iterationen, $\sigma = 0.3$ 108
D.16 Szene 8: Anzahl Iterationen, $\sigma = 1.0$ 108
D.17 Szene 1: Rauschen verschiedener Stärke 109
D.18 Szene 2: Rauschen verschiedener Stärke 109
D.19 Szene 3: Rauschen verschiedener Stärke 110
D.20 Szene 4: Rauschen verschiedener Stärke 110
D.21 Szene 5: Rauschen verschiedener Stärke 111
D.22 Szene 6: Rauschen verschiedener Stärke 111

D.23 Szene 7: Rauschen verschiedener Stärke . . . 112
D.24 Szene 8: Rauschen verschiedener Stärke . . . 112
D.25 Szene 1: Quaternionen – Achse/Winkel . . . 113
D.26 Szene 2: Quaternionen – Achse/Winkel . . . 113
D.27 Szene 3: Quaternionen – Achse/Winkel . . . 113
D.28 Szene 4: Quaternionen – Achse/Winkel . . . 114
D.29 Szene 5: Quaternionen – Achse/Winkel . . . 114
D.30 Szene 6: Quaternionen – Achse/Winkel . . . 114
D.31 Szene 7: Quaternionen – Achse/Winkel . . . 114
D.32 Szene 8: Quaternionen – Achse/Winkel . . . 115
D.33 Szene 1: Korrektur von Linsenverzerrungen . . . 116
D.34 Szene 7: Korrektur von Linsenverzerrungen . . . 116
D.35 Szenen 1–8: Korrektur von Linsenverzerrungen . . . 117
D.36 Szene 1: Verrauschte Parameter . . . 118
D.37 Szene 2: Verrauschte Parameter . . . 118
D.38 Szene 3: Verrauschte Parameter . . . 119
D.39 Szene 4: Verrauschte Parameter . . . 119
D.40 Szene 5: Verrauschte Parameter . . . 119
D.41 Szene 6: Verrauschte Parameter . . . 120
D.42 Szene 7: Verrauschte Parameter . . . 120
D.43 Szene 8: Verrauschte Parameter . . . 120
D.44 Szene 1: Selbstkalibrierung . . . 121
D.45 Szene 2: Selbstkalibrierung . . . 121
D.46 Szene 3: Selbstkalibrierung . . . 121
D.47 Szene 4: Selbstkalibrierung . . . 122
D.48 Szene 5: Selbstkalibrierung . . . 122
D.49 Szene 6: Selbstkalibrierung . . . 122
D.50 Szene 7: Selbstkalibrierung . . . 123
D.51 Szene 8: Selbstkalibrierung . . . 123

Kapitel 1

Einleitung

Diese Arbeit beschäftigt sich mit einem Teilgebiet des Rechnersehens, dem Problem der Berechnung der 3-D-Geometrie einer starren Szene sowie der Bewegung der Kamera und ihrer Abbildungseigenschaften (z. B. Brennweite) aus einem Bildstrom. Dies wird üblicherweise unter dem Begriff *Struktur aus Bewegung* zusammengefasst. Eine Anwendung des Verfahrens liegt beispielsweise in der Echtzeitvisualisierung einer Szene mit unbekannter Geometrie. Die herkömmliche Methode besteht in der manuellen Konstruktion eines geometrischen Modells am Rechner, was gerade bei komplexen Szenen in vielen Fällen nur mit großem Aufwand möglich ist – man denke z. B. an die realistische Darstellung von Pflanzen.

Das Ziel ist es daher, alle benötigten Daten zur Visualisierung aus der Bildfolge automatisch zu ermitteln. Es existieren hierfür insbesondere zwei unterschiedliche Ansätze. Zum einen kann aus der berechneten 3-D-Information ein geometrisches Modell erstellt und mit Texturen überzogen werden. Die Darstellung von Spiegelungen und Reflexionen für eine photorealistische Darstellung der Szene ist damit allerdings schwierig. Mit dem zweiten Ansatz, der bildbasierten Modellierung und Visualisierung, ist dies dagegen auf einfache Weise möglich. Ein weiterer Vorteil ergibt sich daraus, dass es nicht nötig ist, ein explizites und vollständiges geometrisches Modell zu rekonstruieren. Soweit aber 3-D-Information über die Szenengeometrie vorhanden ist, kann diese auch genutzt werden und liefert eine bessere Bildqualität.

Auch das Teilprojekt C2 des SFB 603 am Lehrstuhl für Mustererkennung, im Rahmen dessen die vorliegende Arbeit entstand, befasst sich mit der bildbasierten Modellierung und Visualisierung von Szenen, welche mit einer handgeführten Kamera aufgenommen wurden. Verwendet wird dabei das sogenannte *Lichtfeld*, auf welches im folgenden Abschnitt eingegangen wird.

1.1 Das Lichtfeld

Das *Lichtfeld* [LH96], manchmal auch als *Lumigraph* [GGSC96] bezeichnet, ist ein Verfahren zur bildbasierten Modellierung und Visualisierung von Szenen. Weiterführende Informationen sind den beiden eben genannten Artikeln bzw. [HKP+99] zu entnehmen.

Das Lichtfeld ist eine 5-D-Funktion, welche der *plenoptischen* Funktion entspricht. Diese gibt für jeden 3-D-Punkt die Strahlungsdichte[1] in jede Richtung an. Für die übliche Definition des Lichtfelds kann jedoch stattdessen eine 4-D-Darstellung gewählt werden, da die Strahlungsdichte entlang eines von einem Punkt ausgesandten Lichtstrahls (im Vakuum) konstant bleibt,

[1] engl.: Radiance

(a) Eine globale Ebene

(b) Lokale Ebenen

(c) Verfeinerung der lokalen Ebenen um eine Detailstufe

Bild 1.1: Unterschiede in der Bildqualität bei Verwendung der verschiedenen Detailstufen bei der Modellierung der Szenengeometrie. Die Eckpunkte der Dreiecke in den Bildern (a) und (b) geben die Position einer Kamera bei der Aufnahme der Bildfolge an. In Bild (c) wurde jedes Dreieck in vier kleinere unterteilt.

solange dieser nicht auf ein anderes Objekt trifft. Die Verwendung einer 4-D-Funktion an Stelle der höherdimensionalen 5-D-Funktion bietet einige Vorteile [LH96]: Es müssen erheblich weniger Daten gespeichert werden und die Rekonstruktion des Lichtfelds aus den aufgenommenen Bildern wird einfacher.

Zur Speicherung des Lichtfelds wird üblicherweise eine Datenstruktur benutzt, welche aus zwei Ebenen besteht. Durch die Angabe von jeweils einem Punkt in jeder Ebene wird ein Sichtstrahl festgelegt, der durch diese beiden Punkte verläuft; gespeichert wird zu jedem Sichtstrahl ein Farbwert. Bei Verwendung einer handgeführten Kamera hat diese Struktur jedoch einige Nachteile, die sich in der Bildqualität niederschlagen [HKP+99]. Problematisch sind insbesondere die unregelmäßigen Abstände der einzelnen Kamerapositionen. In [GGSC96] wird daher eine Methode zur Wiederabtastung der Datenstruktur beschrieben[2], welche dieses Problem löst. Aufgrund der Wiederabtastung, die zweimal durchgeführt werden muss – einmal für den Zugriff auf die Datenstruktur und einmal für die Generierung der neuen Ansicht der Szene – entstehen allerdings Fehler, welche bei der Visualisierung sichtbar werden.

Um dies zu vermeiden, wird in [HKP+99] ein Verfahren vorgestellt, das neue Ansichten direkt aus der Bildfolge generieren kann, wenn die Kamerapositionen bekannt sind. Dadurch wird die obige Datenstruktur nicht verwendet und die damit einhergehenden Probleme treten nicht auf. Es werden dabei drei Ansätze unterschieden, bei denen unterschiedlich viel geometrische Information über den betrachteten Szenenausschnitt benötigt wird. Je besser die tatsächliche Szenengeometrie approximiert wird, desto besser ist auch die Qualität der entstehenden Bilder, wie man an den Beispielen in Bild 1.1 sieht.

Im einfachsten Fall wird die Szenengeometrie durch eine einzige globale Ebene approximiert, welche so gewählt wird, dass sie den mittleren quadratischen Abstand zu allen rekonstruierten 3-D-Punkten minimiert. Diese Ebene wird für alle neu generierten Ansichten verwendet. Je nachdem, wie weit die Ebene in einem bestimmten Punkt von der tatsächlichen Szenengeometrie abweicht, entstehen mehr oder weniger starke Artefakte bei der Darstellung der neuen

[2] das sog. *Rebinning*

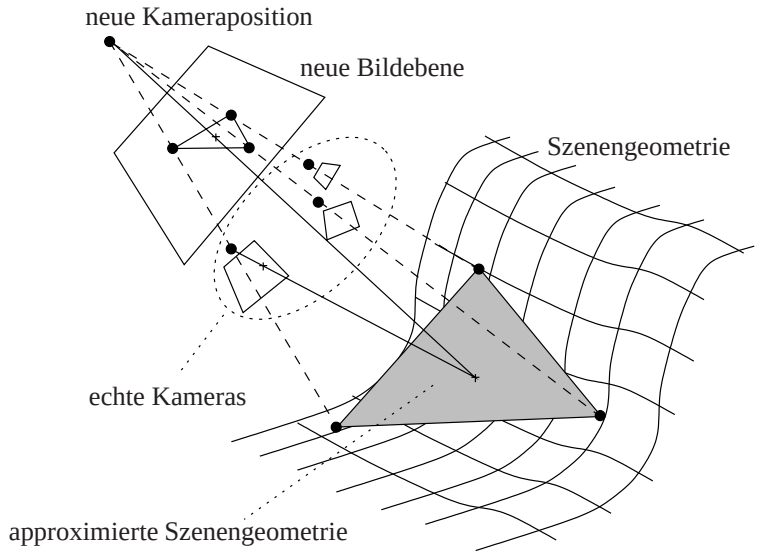


Bild 1.2: Approximation der Szenengeometrie durch drei Kamerapositionen.

Ansicht. Ein Beispiel für die damit erreichbare Bildqualität sieht man in Bild 1.1(a).

Eine bessere Darstellung ergibt sich, wenn man stattdessen für jede Ansicht lokale Ebenen benutzt. Dafür wird eine genauere Approximation der Geometrie der Szene benötigt, was wie folgt erreicht werden kann: Alle Kameras werden in die neue Aufnahme projiziert, anschließend wird eine Delaunay-Triangulierung durchgeführt (siehe auch Bild 1.2). Dadurch ergibt sich aus jeweils drei projizierten Kamerapositionen ein Dreieck, mit dem die Szenengeometrie approximiert wird. Die Verbesserung der Bildqualität bei Verwendung von lokalen Ebenen gegenüber einer einzigen globalen Ebene kann man in Bild 1.1(b) sehen.

Insbesondere bei relativ weit auseinander liegenden echten Kamerapositionen oder wenn sich die Kamera für die neu zu generierende Ansicht nahe an den echten Kamerapositionen befindet, können die lokalen Dreiecke noch so groß sein, dass man ähnliche Artefakte wie vorher bei Verwendung nur einer globalen Ebene sieht. Daher können die lokalen Ebenen weiter verfeinert werden, indem jedes Dreieck in vier kleinere unterteilt wird. Genügt die damit erreichte Bildqualität noch nicht, so kann dieses Verfahren wiederum auf die neuen Dreiecke angewandt werden. Damit erhält man eine immer genauere Approximation der Szenengeometrie. Ein Beispiel sieht man in Bild 1.1(c).

Der Vorteil dieser Vorgehensweise gegenüber der Verwendung eines einzigen 3-D-Modells der gesamten Szene liegt darin, dass man die Bildqualität und somit die für die Darstellung benötigte Rechenzeit auf einfache Weise steuern kann. Zudem wird immer nur der gerade sichtbare Teil der Szenengeometrie approximiert, nicht dagegen Teile, die in der aktuellen Ansicht sowieso verdeckt sind.

Wie man an den Beispielbildern sieht, bringt die Verwendung von zusätzlicher geometrischer Information einen erheblichen Zuwachs an Bildqualität bei der Generierung von neuen Ansichten. Daher ist es nötig, diese Information aus der aufgenommenen Bildfolge möglichst vollautomatisch zu generieren, was mit Methoden aus dem Bereich *Struktur aus Bewegung* erreicht werden kann. Das ist auch das Themengebiet dieser Arbeit. Einen Überblick bietet der folgende Abschnitt.

1.2 Struktur aus Bewegung

Zunächst soll hier eine kurze Einführung in das Gebiet *Struktur aus Bewegung* gegeben werden, wobei dieser Begriff normalerweise die Rekonstruktion der Szenenstruktur (3-D-Punkte) *und* der Kamerabewegung allein aus Bildinformation umfasst. Zusätzlich wird bei Verwendung einer unkalibrierten Kamera auch die Berechnung der intrinsischen Kameraparameter[3] einbezogen. Eine exzellente Einführung in dieses Gebiet ist [TV98]. Umfangreicher und bereits etwas älter, aber ein Standardwerk ist [Fau93].

Es sollen nun die Grundprinzipien sowie die Unterschiede und Gemeinsamkeiten der wichtigsten Verfahren im Bereich *Struktur aus Bewegung* erläutert werden. Dieser Abschnitt dient nur dem Überblick, Details können in den jeweiligen Artikeln nachgelesen werden.

Allen im Folgenden genannten Verfahren ist gemeinsam, dass sie die Lösung des Problems mit Methoden der linearen Algebra erlauben. Im Normalfall werden von den Verfahren Punktmerkmale verwendet, wobei davon ausgegangen wird, dass das Korrespondenzproblem bereits gelöst ist. Zum Finden von Korrespondenzen siehe z. B. [Fau93, Nie90]; das am Lehrstuhl verwendete Verfahren ist in [ST94] beschrieben.

Die damit gewonnene Rekonstruktion kann anschließend mit einem nichtlinearen Optimierungsverfahren verbessert werden, was auch das Thema dieser Arbeit ist.

Eine Methode zur Rekonstruktion von Struktur und Bewegung wurde im Jahr 1981 von Longuet-Higgins im Rechnersehen eingeführt [LH81], wobei das Verfahren an sich ursprünglich aus der Photogrammmetrie stammt. Es werden dabei nur *zwei* Aufnahmen einer Szene bei perspektivischer Projektion betrachtet. Die Beziehung zwischen beiden Bildern lässt sich durch eine 3×3 Matrix vom Rang zwei, die sogenannte *Kernmatrix*[4] $\boldsymbol{E}$, beschreiben. Diese enthält die extrinsischen Kameraparameter[5]. Zur Bestimmung der Parameter aus der Kernmatrix siehe [Fau93]. Die 3-D-Punkte erhält man durch anschließende Triangulation. Eine Erweiterung davon ist die *Fundamentalmatrix* $\boldsymbol{F}$, welche zusätzlich zu den extrinsischen auch die intrinsischen Parameter enthält. Diese ist ebenfalls eine 3×3 Matrix vom Rang zwei. Details dazu findet man z. B. in [TV98, Fau93, Zha98a, LF97]. Die Fundamentalmatrix erhält man allein aus vorhandenen Punktkorrespondenzen in den beiden Bildern durch die Lösung eines linearen Gleichungssystems, welches sich aus der *Epipolarbedingung* ergibt. Für eine numerisch stabile Lösung sind einige wichtige Punkte zu beachten; diese werden in [Har97a] ausführlich erläutert.

Eine Erweiterung dieser Methode auf drei Bilder wird in [Har97b] beschrieben. An die Stelle der Fundamentalmatrix tritt nun ein $3 \times 3 \times 3$ Tensor, der sogenannte *trifokale Tensor*, welcher eine Beziehung zwischen allen drei Bildern beschreibt. Eine Verallgemeinerung dieses Ansatzes, bei der eine ähnliche Beziehung zwischen *allen* Aufnahmen einer beliebig langen Bildfolge hergestellt wird, existiert nicht.

Prinzipiell kann man die Rekonstruktion aus beliebig langen Sequenzen mit Hilfe der Fundamentalmatrix durchführen, wenn man diese immer zwischen zwei Aufnahmen berechnet. Das ist jedoch problematisch: Man muss immer Paare von Bildern auswählen, wobei für die Berechnung der Fundamentalmatrix zwischen jeweils zweien die in den übrigen Aufnahmen vorhandenen Informationen für die Rekonstruktion nicht verwendet werden.

Wünschenswert ist somit ein Verfahren, bei dem *alle* Aufnahmen *zusammen* verrechnet

[3] z. B. der Brennweite . Näheres dazu folgt in Abschnitt 2.2

[4] engl.: essential matrix

[5] die Begriffe *intrinsische* und *extrinsische* Kameraparameter werden in Abschnitt 2.2 erläutert.

werden, ohne dass eine vorhergehende Auswahl von Bildpaaren nötig ist. Zudem sollte es bei Hinzunahme weiterer Aufnahmen insgesamt stabiler werden.

Diesen Ansatz verfolgen die *Faktorisierungsverfahren*: Hier wird zunächst eine sogenannte *Messmatrix* gebildet, welche im Prinzip die detektierten Bildpunkte aus allen Aufnahmen enthält. Je nach verwendetem Verfahren sind dafür noch weitere Vorarbeiten nötig, auf welche hier jedoch nicht weiter eingegangen werden soll. Details findet man in den weiter unten angegebenen Artikeln. Die Messmatrix wird durch den Algorithmus in das Produkt aus zwei Matrizen zerlegt, eine enthält sämtliche Projektionsmatrizen der Kameras, die andere alle 3-D-Punkte.

Die im Folgenden genannten Methoden unterscheiden sich im verwendeten Projektionsmodell: Ein Faktorisierungsverfahren, welches mit Orthogonalprojektion arbeitet, wurde 1992 von Tomasi und Kanade vorgestellt [TK92], eines für die paraperspektivische Projektion 1994 von Poelman und Kanade [PK94]. Das auch am Lehrstuhl verwendete [HN99] Verfahren von Sturm und Triggs [ST96] stammt aus dem Jahr 1996 und kann mit perspektivischer Projektion arbeiten. Allerdings ist für die Bildung der Messmatrix eine Schätzung der sogenannten *projektiven Tiefen* nötig, welche wiederum mit Hilfe der Fundamentalmatrix zwischen zwei benachbarten Bildern berechnet werden müssen. Eine Methode für die perspektivische Projektion ähnlich der von Tomasi und Kanade für die orthogonale, bei der dies nicht nötig ist, existiert bisher nicht. Ein weiteres Problem ist, dass alle verwendeten 3-D-Punkte auch in allen Bildern sichtbar sein müssen, was die Behandlung von Verdeckungen schwierig macht. Einen Lösungsansatz hierfür kann man in [HN99] nachlesen. Einen guten Überblick über die verschiedenen Faktorisierungsverfahren findet man in [KM98].
Zu beachten ist, dass sich unabhängig vom verwendeten Verfahren folgende Einschränkungen bezüglich der Rekonstruktion ergeben, je nachdem wie viel Information über die benutzte Kamera vorhanden ist [TV98]:

- Bei bekannten intrinsischen Kameraparametern ist die Rekonstruktion der Kamerabewegung sowie der Weltpunkte bis auf einen unbekannten Skalierungsfaktor möglich.
- Bei unbekannten intrinsischen Kameraparametern ist die Rekonstruktion nur bis auf eine unbekannte projektive Transformation möglich.
- Bei unbekannten intrinsischen Kameraparametern ist die Rekonstruktion bis auf eine unbekannte Ähnlichkeitstransformation[6] möglich, wenn man annimmt, dass die Achsen des Bildkoordinatensystems senkrecht aufeinander stehen [PKV98]. Dies ist der Fall, der hier betrachtet werden soll.

Weitere Details folgen in Abschnitt 2.3, wo das Thema *Selbstkalibrierung* behandelt wird.

1.3 Aufgabenstellung

Die in den Faktorisierungsverfahren auftretenden Matrizen müssen bestimmte Rangkriterien erfüllen [HN99], was aufgrund von verrauschten Bildpunkten nur näherungsweise möglich ist. Daher muss die Einhaltung der Rangkriterien erzwungen werden, wobei ein algebraisches Fehlermaß minimiert wird, kein physikalisch sinnvolles.

[6] also: Rotation, Translation und Skalierung

Ziel dieser Arbeit ist es, in einem nachfolgenden nichtlinearen Optimierungsschritt ein geeignetes Kriterium zu minimieren, um die aus dem linearen Verfahren stammende approximative Lösung zu verbessern. Zur optimalen Ermittlung der Kameraparameter und der 3-D-Punkte soll eine Maximum-Likelihood-Schätzung verwendet werden. Hierzu ist die Annahme einer geeigneten Wahrscheinlichkeitsverteilung für die Fehler in den Punktmerkmalen erforderlich.

Durch Experimente mit simulierten und realen Daten soll abschließend die Leistungsfähigkeit des gewählten Verfahrens belegt werden.

1.4 Literatur

Der folgende Literaturüberblick betrifft ausschließlich den Kernbereich der Arbeit. Weitere Literaturangaben, die sich allgemein auf das Thema *Struktur aus Bewegung* beziehen, befinden sich in den Abschnitten 1.2 und 2.3.

Optimierungskriterien. Unterschieden werden zwei Arten von Kriterien, algebraische und geometrische[7]. Einen Überblick über die Anwendung algebraischer Kriterien im Rechnersehen findet man z. B. in [Har98]. Ein Vergleich der drei besten bekannten geometrischen Kriterien für die nichtlineare Optimierung wird in [Zha98b] durchgeführt, wobei zu beachten ist, dass sich dieser Artikel auf die Verwendung von nur zwei Ansichten einer Szene bezieht.

Mit der optimalen Schätzung von Struktur und Bewegung befasst sich [WAH93]. Betrachtet wird insbesondere die nichtlineare Minimierung des Rückprojektionsfehlers unter Annahme einer Wahrscheinlichkeitsverteilung des Rauschens auf den Bildpunkten.

Grundlegende Arbeiten zu statistischen Ansätzen im Rechnersehen stammen von Kanatani. Dieser befasst sich insbesondere mit der Korrektur des statistischen Bias bei der Schätzung von Parametern. Eine Einführung findet man in [Kan93]; wesentlich ausführlicher ist das gesamte Gebiet jedoch in [Kan96] dargestellt.

Untersuchungen zur Wahl einer geeigneten Parametrisierung bei der Schätzung von Parametern findet man in [HT99].

Bündelausgleich. Auf dem Bündelausgleich liegt der Schwerpunkt dieser Arbeit. Der Begriff bezeichnet die Minimierung des Rückprojektionsfehlers und kommt aus der Photogrammmetrie, weshalb Beschreibungen das Verfahrens auch in der zugehörigen Literatur zu finden sind, so z. B. in [Sla80]. Die dort erläuterte Methode zur effizienten Berechnung der Jacobi-Matrix für das Gauß-Newton-Verfahren beim Bündelausgleich wird von Hartley für die euklidische Rekonstruktion aufgegriffen und in [Har94] beschrieben. Zur Parametrisierung des Verfahrens bei euklidischer Rekonstruktion siehe auch [HÅ97]. Informationen zum Bündelausgleich im projektiven Fall befinden sich in [McL99].

Der auch in der vorliegenden Arbeit benutzte eingebettete Bündelausgleich wird in [SKZ99] und [ST98] verwendet.

Weitere Anwendungsbeispiele findet man z. B. in [FZ98, Ris99, Fua99].

[7] weitere Informationen hierzu befinden sich in Abschnitt 3.1

Numerik. Aus dem Bereich der numerischen Mathematik werden in dieser Arbeit insbesondere zwei Verfahren benötigt. Eines ist die Singulärwertzerlegung, welche eine Methode aus der linearen Algebra zur Faktorisierung von Matrizen ist. Einen kurzen Überblick bietet Anhang B. Das Verfahren wird üblicherweise in jedem Buch beschrieben, welches sich auch mit numerischer linearer Algebra beschäftigt. Zu empfehlen sind insbesondere [TD97] (nur numerische lineare Algebra) und [PTVF92] (numerische Methoden allgemein).

Zur nichtlinearen Optimierung wird das Gauß-Newton-Verfahren mit Levenberg-Marquardt-Erweiterung verwendet. Eine ausführliche Darstellung findet man in [DS83], kürzere Erläuterungen in [PTVF92] und [Sch93]. Da dieses Verfahren oft beim Bündelausgleich eingesetzt wird, kann man kurze Beschreibungen auch in [HÅ97, Har94, HÅ99, McL99] nachlesen.

1.5 Aufbau der Arbeit

Im folgenden Kapitel werden die für das weitere Verständnis nötigen Grundlagen geschaffen. Es werden die Begriffe *intrinsische* und *extrinsische* Kameraparameter erläutert, daran anschließend wird kurz auf die *Selbstkalibrierung* eingegangen, welche aus einer projektiven Rekonstruktion eine euklidische macht.

In Kapitel 3 werden einige zur Auswahl stehende Optimierungskriterien vorgestellt. Aus diesen wurde der *Rückprojektionsfehler* ausgewählt. Die Minimierung dieses Kriteriums wird auch als *Bündelausgleich* bezeichnet. Auf die Optimalität des Kriteriums im Sinne einer Maximum-Likelihood-Schätzung wird in Abschnitt 3.2 eingegangen.

Für die Durchführung der Minimierung ist zunächst die Auswahl einer für das Problem geeigneten Parametrisierung nötig. Diese wird in Abschnitt 4.2 vorgestellt. Im darauf folgenden Abschnitt 4.3 wird beschrieben, wie die Optimierung auf effiziente Weise mit Hilfe des Gauß-Newton-Verfahrens möglich ist. Das geschieht unter Ausnutzung einer Blockstruktur der Jacobi-Matrix der ersten Ableitungen der Fehlerfunktion, welche sich aus dem Optimierungskriterium ergibt. In Abschnitt 4.4 wird erläutert, wie das Verfahren so angepasst werden kann, dass auch diejenigen 3-D-Punkte zur Optimierung herangezogen werden können, die z. B. aufgrund von Verdeckungen nicht in allen Aufnahmen der Bildfolge sichtbar sind.

In Kapitel 5 werden zwei Erweiterungen des Standardverfahrens betrachtet, zum einen die unterschiedliche Gewichtung des Rückprojektionsfehlers in den einzelnen Bildpunkten, zum anderen die Korrektur von Linsenverzerrungen.

Experimente mit simulierten (Abschnitt 6.2) und realen (Abschnitt 6.3) Bildfolgen findet man in Kapitel 6. Auf die Probleme bei der Berechung der Startwerte für die nichtlineare Optimierung wird vorher in Abschnitt 6.1 eingegangen.

Den Abschluss der Arbeit bildet eine Zusammenfassung der wichtigsten Ergebnisse.

Kapitel 2

Grundlagen

In diesem Kapitel sollen die zum weiteren Verständnis notwendigen Grundlagen geschaffen werden. Zunächst wird die verwendete Notation erläutert. Im darauf folgenden Abschnitt 2.2 wird auf die Abbildungseigenschaften einer Kamera eingegangen, die intrinsischen und extrinsischen Kameraparameter werden eingeführt. Der Übergang von einer projektiven zu einer euklidischen Rekonstruktion sowie die damit einhergehenden Restriktionen werden im Abschnitt *Selbstkalibrierung* beschrieben.

2.1 Notation

Im Folgenden soll die in der Arbeit verwendete Notation erläutert werden. Generell gilt: Vektoren sind fettgedruckte Kleinbuchstaben ($\boldsymbol{a}$), Matrizen fettgedruckte Großbuchstaben ($\boldsymbol{A}$). Skalare Größen erscheinen in normaler Schrift (a).
Homogene Weltpunkte werden mit $\boldsymbol{w}_j$ bezeichnet. Die Elemente dieses 4-D-Vektors lauten:

$$\boldsymbol{w}_j = \begin{pmatrix} w_{jx} \\ w_{jy} \\ w_{jz} \\ w_{jh} \end{pmatrix} \quad . \tag{2.1}$$

Im weiteren Verlauf wird angenommen, dass die 3-D-Punkte so normiert sind, dass $w_{jh} = 1$ gilt[1]. Die Projektion eines Weltpunktes $\boldsymbol{w}_j$ ins i-te Bild wird in der homogenen Form mit $\boldsymbol{q}_{ij}$ bezeichnet:

$$\boldsymbol{q}_{ij} = \begin{pmatrix} q_{ijx} \\ q_{ijy} \\ q_{ijh} \end{pmatrix} \quad . \tag{2.2}$$

Daraus entstehen die 2-D-Bildpunkte $\boldsymbol{u}_{ij}$ durch

$$\boldsymbol{u}_{ij} = \begin{pmatrix} u_{ij} \\ v_{ij} \end{pmatrix} = \begin{pmatrix} \frac{q_{ijx}}{q_{ijh}} \\ \frac{q_{ijy}}{q_{ijh}} \end{pmatrix} \quad . \tag{2.3}$$

Der Nenner q_{ijh} in Gleichung (2.3) wird immer verschieden von null sein, da nur solche Punkte im Bild detektiert werden können, die nicht im Unendlichen liegen.

[1] Punkte im Unendlichen werden also nicht betrachtet. Dies ist die für den Bündelausgleich bei einer euklidischen Rekonstruktion übliche Vorgehensweise, wie sie auch in [Har94, HÅ97, SKZ99] verwendet wird.

Weiterhin werden die tatsächlich detektierten Bildpunkte mit $\boldsymbol{u}_{ij}$ und die durch Rückprojektion eines rekonstruierten Weltpunktes entstandenen Bildpunkte mit $\tilde{\boldsymbol{u}}_{ij}$ bezeichnet.

Die auftretenden homogenen Punkte ($\boldsymbol{w}_j$ und $\boldsymbol{q}_{ij}$) sowie die Kameraparameter sind immer Schätzwerte, welche aus einer Rekonstruktion kommen. Daher werden sie nicht besonders gekennzeichnet. Alle übrigen Schätzwerte von Parametern, wie z. B. die Standardabweichung einer Normalverteilung, werden mit einem Dach versehen ($\hat{\sigma}$). Für den wahren Wert eines Parameters wird ein Querbalken benutzt ($\bar{\sigma}$).
Die Elemente einer 3×3 Rotationsmatrix $\boldsymbol{R}$ werden wie folgt verwendet:

$$\boldsymbol{R} = \begin{pmatrix} \boldsymbol{r}_1^T \\ \boldsymbol{r}_2^T \\ \boldsymbol{r}_3^T \end{pmatrix} = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} \quad . \tag{2.4}$$

Eine 3×4 Projektionsmatrix $\boldsymbol{P}$ sieht wie folgt aus:

$$\boldsymbol{P} = \begin{pmatrix} \boldsymbol{p}_1^T \\ \boldsymbol{p}_2^T \\ \boldsymbol{p}_3^T \end{pmatrix} = \begin{pmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{pmatrix} \quad . \tag{2.5}$$

Mit $[\boldsymbol{v}]_\times$ wird folgende, aus dem Vektor $\boldsymbol{v} \in \mathbb{R}^3$ gebildete, antisymmetrische Matrix bezeichnet:

$$[\boldsymbol{v}]_\times = \left[\begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix} \right]_\times = \begin{pmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{pmatrix} \quad . \tag{2.6}$$

Weiterhin wird die Bezeichnung $\boldsymbol{I}$ für die Einheitsmatrix benutzt; die Größe der Matrix ergibt sich in den meisten Fällen automatisch aus dem Zusammenhang; wo dies evtl. unklar sein könnte, ist sie mit einem zusätzlichen Index versehen, also z. B. $\boldsymbol{I}_3$ für die 3×3 Einheitsmatrix.

2.2 Kameraparameter

Es sollen nun die intrinsischen und extrinsischen Kameraparameter erläutert werden, wobei für die Linse der Kamera ein Lochkameramodell angenommen wird; das verwendete Projektionsmodell sei die perspektivische Projektion. Ein Überblick über die unterschiedlichen Projektionsmodelle findet man in [XZ96].

Durch Verwendung homogener Koordinaten erhält man folgende lineare Abbildung eines 3-D-Weltpunktes $\boldsymbol{w}$ (homogen: 4-D) auf einen homogenen Bildpunkt $\boldsymbol{q}$:

$$\boldsymbol{q} = \boldsymbol{P}\boldsymbol{w} = \boldsymbol{K}\boldsymbol{R}^T(\boldsymbol{I}_3 | -\boldsymbol{t})\boldsymbol{w} \quad . \tag{2.7}$$

$\boldsymbol{K}$ ist eine obere Dreiecksmatrix, die sogenannte *Kalibrierungsmatrix*. Diese sieht wie folgt aus:

$$\boldsymbol{K} = \begin{pmatrix} f/s_x & -1/(s_x \tan\Theta) & u_0 \\ 0 & f/(s_y \sin\Theta) & v_0 \\ 0 & 0 & 1 \end{pmatrix} \quad . \tag{2.8}$$

Sie enthält die intrinsischen Kameraparameter, welche die Abbildungseigenschaften der verwendeten Kamera definieren [TV98]:

- Die Brennweite f der Kamera (in mm).
- Der Hauptpunkt (u_0, v_0) (in Pixel) definiert eine Verschiebung des Ursprungs des Bildkoordinatensystems, da dieser im Allgemeinen nicht mit dem Schnittpunkt der optischen Achse und der Bildebene zusammenfällt.
- Die Pixel sind nicht quadratisch, daher geben s_x bzw. s_y die Längen in x- bzw. y-Richtung pro Pixel an (in mm/Pixel).
- Die Achsen des CCD-Chips der Kamera stehen nicht exakt senkrecht sondern im Winkel Θ aufeinander.

Zum letzten Punkt ist anzumerken, dass ein Winkel von 90° in der Praxis in guter Näherung gilt. Dies ist einer der Gründe, weshalb man zur Vereinfachung oft $\Theta = 90°$ annimmt. Ein weiterer Grund ergibt sich aus den Einschränkungen für die Rekonstruktion aus Bildfolgen selbst. Näheres dazu folgt weiter unten.

Wie man sieht, treten die Faktoren s_x bzw. s_y immer nur als Produkt zusammen mit f auf; man kann sie nie getrennt voneinander beobachten. D. h. eine Änderung der Brennweite f ist nicht von einer Änderung der Pixeleinheiten unterscheidbar. Daher fasst man diese Produkte zu einem einzigen Parameter zusammen:

$$f_x = \frac{f}{s_x}, \qquad f_y = \frac{f}{s_y} \quad . \tag{2.9}$$

Dies ist jeweils die Brennweite der Kamera, ausgedrückt in horizontalen bzw. vertikalen Pixeln, welche im Folgenden einfach als Brennweiten bezeichnet werden.

Die extrinsischen Kameraparameter werden durch eine Rotationsmatrix $\boldsymbol{R}$ und einen Translationsvektor $\boldsymbol{t}$ festgelegt. Sie geben eine Transformation vom Weltkoordinatensystem ins jeweilige Kamerakoordinatensystem an, also die Position der Kamera im Weltkoordinatensystem. Es gilt:

- $\boldsymbol{t}$ ist ein 3-D-Vektor,
- $\boldsymbol{R}$ ist eine orthogonale 3×3 Matrix mit Determinante eins.

2.3 Selbstkalibrierung

Wie bereits erwähnt, ist die Rekonstruktion von Struktur und Kameraparametern bei unbekannten intrinsischen Kameraparametern nur bis auf eine unbekannte projektive Transformation $\boldsymbol{T}$ möglich, da für bereits vorhandene Schätzwerte $\boldsymbol{P}_i$ und $\boldsymbol{w}_j$ gilt:

$$\boldsymbol{q}_{ij} = \boldsymbol{P}_i \boldsymbol{w}_j = (\boldsymbol{P}_i \boldsymbol{T}^{-1})(\boldsymbol{T} \boldsymbol{w}_j) = \boldsymbol{P}'_i \boldsymbol{w}'_j \quad . \tag{2.10}$$

Dabei ist $\boldsymbol{T}$ eine beliebige invertierbare 4×4 Matrix. Ohne weitere Informationen über die verwendete Kamera oder Annahmen über diese, ist es nicht möglich die unbekannte Transformation $\boldsymbol{T}$ weiter einzuschränken. Führt man zusätzliche Restriktionen ein, so kann man eine Rekonstruktion bis auf eine unbekannte Ähnlichkeitstransformation erreichen. Die dabei verwendete Technik wird als *Selbstkalibrierung* bezeichnet. Da es beim hier vorliegenden Anwendungsbereich darum geht, aus einer Bildfolge, die von einer handgeführten Kamera mit

unbekannten intrinsischen und extrinsischen Kameraparametern aufgenommen wurde, sowohl diese Parameter als auch die Position der 3-D-Weltpunkte zu ermitteln, wird im Folgenden kurz auf die Selbstkalibrierung eingegangen.

Weiterführende Informationen zu diesem Thema findet man in [PKV98, Pol99, HN99]. Für eine Einführung in die projektive Geometrie siehe [MT96]. Eine ausführliche Abhandlung über die Verwendung der projektiven Geometrie im Bereich des Rechnersehens befindet sich in [Kan93].

Die Rekonstruktion wird, je nach Aussehen der unbekannten Transformation $\boldsymbol{T}$, wie folgt bezeichnet [Pol99]:

projektiv. Die 4×4 Matrix $\boldsymbol{T}$ kann beliebig aussehen, allerdings muss sie invertierbar sein. $\boldsymbol{T}$ ist nur bis auf einen Skalierungsfaktor eindeutig.

affin. Eine affine Rekonstruktion kann aus einer projektiven berechnet werden, indem man die sogenannte *Ebene im Unendlichen* ermittelt. $\boldsymbol{T}$ darf dann noch wie folgt aussehen:

$$\boldsymbol{T} = \begin{pmatrix} t_{11} & t_{12} & t_{13} & t_{14} \\ t_{21} & t_{22} & t_{23} & t_{24} \\ t_{31} & t_{32} & t_{33} & t_{34} \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad . \tag{2.11}$$

euklidisch/metrisch. Die euklidische Rekonstruktion ist die hier interessante, und ohne Zusatzinformationen über die Größe eines Objekts im Bild oder ähnliche Maßangaben ist es nicht möglich, die Transformation $\boldsymbol{T}$ noch weiter einzuschränken. Diese hat im euklidischen die Form einer *Ähnlichkeitstransformation*

$$\boldsymbol{T} = \begin{pmatrix} \lambda \boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0}^T & 1 \end{pmatrix} \quad , \tag{2.12}$$

wobei $\boldsymbol{R}$ eine 3×3 Rotationsmatrix, $\boldsymbol{t}$ ein 3-D-Translationsvektor und λ ein Skalierungsfaktor ist. Erst in einer euklidischen Rekonstruktion haben die Begriffe *Winkel* und *relative Länge* eine Bedeutung. In der Literatur (und auch hier) werden die Begriffe *euklidisch* und *metrisch* häufig synonym verwendet. In [Pol99] bezeichnet der Begriff *euklidisch* dagegen eine Rekonstruktion, bei der der Skalierungsfaktor λ bekannt ist. Dieser Fall wird üblicherweise nicht weiter betrachtet, da eine solche Rekonstruktion ohne eine zusätzliche Maßangabe allein aus Bildinformationen nicht möglich ist.

Unter Selbstkalibrierung versteht man die Ermittlung einer Transformation $\boldsymbol{T}'$, welche eine anfängliche projektive Rekonstruktion in eine euklidische überführt. Dies geschieht mit Hilfe der *absoluten Quadrik* $\boldsymbol{\Omega}$, welche invariant bezüglich des Wechsels der euklidischen Basis ist und durch eine 4×4 Matrix dargestellt werden kann. Die einzige notwendige Annahme, um eine euklidische Rekonstruktion überhaupt durchführen zu können, ist, dass die Kalibrierungsmatrizen $\boldsymbol{K}$ folgende Form haben [PKV98]:

$$\boldsymbol{K} = \begin{pmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{pmatrix} \quad . \tag{2.13}$$

Die Achsen des Bildkoordinatensystems der Kamera müssen also senkrecht aufeinander stehen, was in der Praxis meist in guter Näherung der Fall ist [TV98].

Um überhaupt eine anfängliche Schätzung der gesuchten Transformation vom Projektiven ins Euklidische mit Hilfe eines linearen Verfahrens zu erhalten, sind weitere Einschränkungen nötig [PKV98]:

- Die Bildpunkte werden als quadratisch angenommen, d. h. $f = f_x = f_y$,
- die Koordinaten des Hauptpunkts (u_0, v_0) sind $(0, 0)$. In der Praxis kann der Punkt (u_0, v_0) in guter Näherung als in der Bildmitte liegend angenommen werden, d. h. man kann diese Voraussetzung durch eine einfache Verschiebung der Koordinaten des Bildes erreichen.

Die Kalibrierungsmatrizen müssen also folgende Form haben:

$$\boldsymbol{K} = \begin{pmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad . \tag{2.14}$$

Damit erhält man nach Anwendung der gesuchten Transformation eine näherungsweise euklidische Rekonstruktion, d. h. die entstehenden Kalibrierungsmatrizen haben *im Mittel* die obige Form. Betrachtet man jedoch eine einzelne Projektionsmatrix, so hat diese immer noch eine geringe projektive Verzerrung, was sich darin äußert, dass man nach der Zerlegung der Matrix in ihre Bestandteile entsprechend Gleichung (2.7) eine Kalibrierungsmatrix erhält, die *nicht* exakt diese Form hat, sondern

$$\boldsymbol{K} = \begin{pmatrix} \hat{f}_x & \hat{s} & \hat{u}_0 \\ 0 & \hat{f}_y & \hat{v}_0 \\ 0 & 0 & 1 \end{pmatrix} \quad . \tag{2.15}$$

Es ist also $\hat{f}_x \neq \hat{f}_y$, der Hauptpunkt liegt nicht genau bei $(0, 0)$ und vor allem befindet sich $\hat{s}$ nahe bei null, ist aber nicht exakt null. Auf die dadurch auftretenden Probleme wird in Abschnitt 6.1 eingegangen.

Diese initiale Schätzung kann anschließend mit Hilfe eines nichtlinearen Verfahrens verbessert und in eine exakt euklidische Rekonstruktion überführt werden, wobei auch nichtquadratische Pixel und (u_0, v_0) verschieden von $(0, 0)$ berücksichtigt werden können. Dies geschieht in der Implementierung am Lehrstuhl mit dem in [PKV98] vorgestellten Kriterium. Auch der in dieser Arbeit vorgestellte Bündelausgleich kann als Optimierung der Selbstkalibrierung aufgefasst werden; in [Pol99] entspricht das dem Maximum-Likelihood-Ansatz.

Kapitel 3

Optimierungskriterien

In Abschnitt 3.1 dieses Kapitels erhält der Leser einen Überblick über die zur Auswahl stehenden Optimierungskriterien. Aus diesen wurde der *Rückprojektionsfehler* ausgewählt. In Abschnitt 3.2 wird dargestellt, in wie weit dieses Kriterium optimal im Sinne einer Maximum-Likelihood-Schätzung ist.

3.1 Überblick

Da es in dieser Arbeit um die Verbesserung einer bereits vorhandenen Rekonstruktion von Struktur und Kameraparametern unter Verwendung eines nichtlinearen Optimierungsverfahrens geht, ist es nötig ein Kriterium zu finden, das optimiert werden soll. Es werden im Folgenden einige Alternativen aufgezeigt, aus denen ein Kriterium ausgewählt wurde. Wie damit die Minimierung des Fehlers möglichst geschickt erfolgt, wird im Verlauf der Arbeit erläutert.
Generell muss man zwischen zwei Arten von Optimierungskriterien unterscheiden:

Algebraische Kriterien. Diese werden üblicherweise von den in Abschnitt 1.2 vorgestellten linearen Algorithmen zur 3-D-Rekonstruktion, wie z. B. den Faktorisierungsverfahren verwendet. So minimiert beispielsweise die Singulärwertzerlegung beim Lösen eines überbestimmten Gleichungssystems $\boldsymbol{M}\boldsymbol{x} = \boldsymbol{0}$ die Norm des Fehlervektors $\|\boldsymbol{M}\boldsymbol{x}\|$. Oft enthält der Lösungsvektor $\boldsymbol{x}$ die Elemente einer Matrix, welche z. B. zusätzliche Rangkriterien erfüllen muss. Diese Kriterien werden bei der Verwendung eines algebraischen Fehlermaßes normalerweise nicht bereits bei der Minimierung berücksichtigt, sondern erst nachträglich erzwungen.
Diese Art von Kriterien hat ihre Berechtigung, da deren Minimierung meist zu schnellen linearen Algorithmen für die zu lösenden Probleme führt. Allerdings sind sie in gewisser Weise willkürlich, und sie optimieren kein physikalisch vorhandenes Maß wie die geometrischen Kriterien.

Geometrische Kriterien. Diese minimieren ein Fehlermaß, welches eine physikalische Bedeutung hat, und sie werden üblicherweise für eine nichtlineare Optimierung im Anschluss an ein lineares Verfahren verwendet. Da man es anfangs mit einer projektiven Rekonstruktion zu tun hat (die nur näherungsweise eine euklidische ist), sind die aufgenommenen Bilder die einzige Quelle für metrische Information, weil es in einem projektiven Raum keine Metrik gibt. Dies bedeutet, es werden hier generell Maße verwendet, die sich in der einen oder anderen Form auf den aufgenommenen Bildern berechnen lassen.

Weiterführende Informationen zu den algebraischen Kriterien findet man z. B. in [Har98]. Die drei besten bekannten geometrischen Kriterien für die nichtlineare Optimierung werden in [Zha98b] verglichen. Zu beachten ist dabei, dass sich der letztgenannte Artikel auf die Verwendung von nur zwei Ansichten einer Szene bezieht.
Folgende geometrische Optimierungskriterien werden in [Zha98b] vorgestellt:

Abstand Bildpunkte – Epipolarlinien. Hier wird, für jeweils zwei Bilder, vom Punkt $\boldsymbol{u}_{j1}$ im ersten Bild ausgehend die korrespondierende Epipolarlinie im zweiten Bild berechnet. Nun ermittelt man den Abstand des Punktes $\boldsymbol{u}_{j2}$ von dieser Linie. Ebenso wird ausgehend vom Punkt $\boldsymbol{u}_{j2}$ verfahren. Als Fehler wird die Summe der quadratischen Abstände der Bildpunkte von den korrespondierenden Epipolarlinien verwendet.
Optimiert wird die Fundamentalmatrix zwischen den beiden Bildern; die für die Berechnung dieses Fehlers benötigten Epipolarlinien erhält man auf einfache Weise aus dieser Matrix.

Gewichteter Fehler in der Epipolar-Bedingung. Bei diesem Maß wird der Fehler in der Epipolarbedingung verwendet. Diese besagt, dass für alle korrespondierenden Bildpunkte $\boldsymbol{u}_{j1}$ und $\boldsymbol{u}_{j2}$ in zwei Bildern gilt

$$\boldsymbol{u}_{j2}^T \boldsymbol{F} \boldsymbol{u}_{j1} = 0 \quad , \tag{3.1}$$

wobei $\boldsymbol{F}$ die Fundamentalmatrix ist. Die Bedingung (3.1) ist für zwei Punkte immer nur näherungsweise erfüllt, weshalb man als Fehler die Summe der quadratischen Abweichungen von null über alle Punkte verwendet. Dieser Fehler wird zusätzlich gewichtet. Optimiert wird auch hier die Fundamentalmatrix.

Rückprojektionsfehler. Der Rückprojektionsfehler wird ermittelt, indem man den quadratischen Abstand der detektierten Bildpunkte von den wieder in jedes Bild projizierten rekonstruierten 3-D-Punkten berechnet. Optimiert werden hier sowohl die Projektionsmatrizen für jedes Bild als auch die 3-D-Punkte.

In den ersten beiden Fällen benötigt man eine initiale Schätzung der Fundamentalmatrix zwischen *zwei* Bildern, welche dann während der Optimierung verbessert wird. Dies bedeutet also, man muss immer zwei Aufnahmen paarweise betrachten, was in einer Bildfolge mit vielen Aufnahmen nicht wünschenswert ist. Aus den gleichen Gründen wie bei den in Abschnitt 1.2 vorgestellten linearen Verfahren zur Rekonstruktion ist es besser, alle Bilder auf einmal zu verarbeiten. Zudem werden mit den ersten beiden Fehlermaßen nur die Fundamentalmatrizen optimiert, die 3-D-Punkte gehen nicht direkt in die Berechnung ein. Verwendet man dagegen den Rückprojektionsfehler, ist dies sehr wohl der Fall.
Die Minimierung des Rückprojektionsfehlers hat folgende Vorteile:

- Alle vorhandenen Bilder gehen gleichzeitig in die Optimierung ein. Eine Auswahl von Paaren ist nicht nötig.
- Es werden Kameraparameter und 3-D-Punkte zusammen optimiert.
- Die Minimierung des Rückprojektionsfehlers ist in einem statistischen Sinne optimal: Sie entspricht einer Maximum-Likelihood-Schätzung der zu optimierenden Parameter. Näheres hierzu folgt in Kapitel 3.2.

Aus den genannten Gründen wird dieses Fehlermaß für Bildfolgen mit mehr als zwei Aufnahmen in vielen Fällen verwendet – oft unter der Bezeichnung *Bündelausgleich*. Ein Nachteil soll hier jedoch nicht verschwiegen werden: Die Optimierung benötigt wesentlich mehr Zeit als mit denjenigen Kriterien, die auf der Schätzung der Fundamentalmatrix beruhen – auch dann, wenn man mit nur zwei Aufnahmen arbeitet [Zha98b].

Der Bündelausgleich wurde auch für die vorliegende Anwendung zur Optimierung verwendet. Bei der Lichtfeldrekonstruktion hat er den zusätzlichen Vorteil, dass der Rückprojektionsfehler *das* entscheidende Kriterium für die resultierende Bildqualität ist.

Im folgenden Abschnitt wird erläutert, unter welchen Voraussetzungen die Minimierung des Rückprojektionsfehlers statistisch gesehen optimal ist.

3.2 Optimalität des Bündelausgleichs

In diesem Abschnitt werden einige Ergebnisse vorgestellt, die die Wahl des Bündelausgleichs zur Optimierung der gesuchten Kameraparameter und Weltpunkte aus statistischer Sicht rechtfertigen. Dabei werden zunächst die Grundlagen aus der Statistik betrachtet, die besagen, dass Maximum-Likelihood-Schätzung von Parametern einer Wahrscheinlichkeitsverteilung optimal in dem Sinne ist, dass es kein Verfahren gibt, welches kleinere Kovarianzen der geschätzten Parameter liefert. Anschließend wird gezeigt, unter welchen Voraussetzungen der Bündelausgleich als Maximum-Likelihood-Schätzung aufgefasst werden kann.

3.2.1 Optimalität der Maximum-Likelihood-Schätzung

Zuerst soll kurz erläutert werden, in welchem Sinn eine Maximum-Likelihood-Schätzung optimal ist. Dazu werden zunächst einige grundlegende Ergebnisse aus der Statistik präsentiert. Nachzulesen sind diese beispielsweise in [Kan96, WAH93].

Grundlagen

Wir betrachten folgendes, im Allgemeinen nichtlineare, Modell der beobachteten Daten $\{\boldsymbol{y_i}\}$:

$$\boldsymbol{y_i} = \boldsymbol{f}(\boldsymbol{x_i}; \boldsymbol{\theta}) + \boldsymbol{\epsilon}_i \quad , \tag{3.2}$$

wobei $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_n)$ ein n-dimensionaler Parametervektor ist. Jedes beobachtete Datum $\boldsymbol{y_i}$ ist verrauscht, es weicht also vom tatsächlichen Wert $\bar{\boldsymbol{y}}_{\boldsymbol{i}}$ ab. Dieses Rauschen wird als Zufallsvariable $\boldsymbol{\epsilon}_i$ modelliert. Die Wahrscheinlichkeitsdichte der Daten $\{\boldsymbol{y_i}\}$ sei $p(\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_N; \boldsymbol{\theta})$; zunächst ist dies nicht notwendigerweise eine Normalverteilung. Das Ziel ist es, die Parameter $\boldsymbol{\theta}$ zu schätzen.
Ein Schätzer heisst *erwartungstreu*, wenn für die Schätzwerte $\bar{\boldsymbol{\theta}}$ der Parameter gilt:

$$E(\bar{\boldsymbol{\theta}}) = \boldsymbol{\theta} \quad . \tag{3.3}$$

Hierbei ist $E(\cdot)$ der Erwartungswert.
Die (a-posteriori) Kovarianzmatrix $\boldsymbol{V}(\bar{\boldsymbol{\theta}})$ der Parameter ist definiert als

$$\boldsymbol{V}(\bar{\boldsymbol{\theta}}) = E\left((\bar{\boldsymbol{\theta}} - \boldsymbol{\theta})(\bar{\boldsymbol{\theta}} - \boldsymbol{\theta})^T\right) \quad . \tag{3.4}$$

Die Kovarianzmatrix $\boldsymbol{V}(\boldsymbol{\theta})$ eines erwartungstreuen Schätzers wird als Maß der Güte des Schätzers verwendet: Je kleiner $\boldsymbol{V}(\boldsymbol{\theta})$ ist, desto besser ist der Schätzer; was genau „klein“ in diesem Zusammenhang bedeutet, wird weiter unten in Gleichung (3.9) erläutert.

Für die Größe der Kovarianzmatrix gibt es eine untere Schranke, die sogenannte *Cramér-Rao-Schranke*. Um diese angeben zu können wird zunächst der *Score* $\boldsymbol{l}$ eingeführt:

$$\boldsymbol{l} = \nabla_{\boldsymbol{\theta}} \ln p(\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_N; \boldsymbol{\theta}) \quad , \tag{3.5}$$

mit

$$\nabla_{\boldsymbol{\theta}} = \begin{pmatrix} \frac{\partial}{\partial \theta_1} \\ \vdots \\ \frac{\partial}{\partial \theta_n} \end{pmatrix} \quad . \tag{3.6}$$

Mit Hilfe von $\boldsymbol{l}$ lässt sich die *Fisher-Informationsmatrix* $\boldsymbol{F}$ wie folgt definieren:

$$\boldsymbol{F} = E(\boldsymbol{l}\boldsymbol{l}^T) \quad . \tag{3.7}$$

Die sogenannte *Cramér-Rao-Ungleichung* besagt nun, dass es keinen erwartungstreuen Schätzer gibt, der eine Kovarianzmatrix liefert, welche kleiner ist als die Inverse der Fisher-Informationsmatrix. Es gilt also immer

$$\boldsymbol{V}(\boldsymbol{\theta}) \succ \boldsymbol{F}^+ \quad . \tag{3.8}$$

Hierbei wird mit $\boldsymbol{F}^+$ die Pseudoinverse der Fisher-Informationsmatrix bezeichnet. Die Pseudoinverse ist für beliebige Matrizen berechenbar; falls $\boldsymbol{F}$ regulär ist, gilt $\boldsymbol{F}^+ = \boldsymbol{F}^{-1}$. In der Praxis kann man die Pseudoinverse z. B. mit Hilfe der Singulärwertzerlegung ermitteln. Nähere Informationen hierzu befinden sich in Anhang B.
$\boldsymbol{A} \succ \boldsymbol{B}$ ist eine „größer“-Relation für Matrizen, die wie folgt definiert ist:

$$\boldsymbol{A} \succ \boldsymbol{B} \;\Leftrightarrow\; \boldsymbol{A} - \boldsymbol{B} \text{ ist eine symmetrische, positiv semi-definite Matrix.} \tag{3.9}$$

$\boldsymbol{F}^+$ wird als *Cramér-Rao-Schranke* bezeichnet: Diese ist eine untere Schranke für die Kovarianzmatrix eines erwartungstreuen Schätzers, d. h. $\boldsymbol{V}(\boldsymbol{\theta})$ kann nicht kleiner sein als diese Schranke, wenn man $\succ$ wie beschrieben definiert. Ein erwartungstreuer Schätzer, der die Cramér-Rao-Schranke erreicht, heisst *effizient*.

Der Beweis, dass die Ungleichung (3.8) tatsächlich immer gilt, soll hier nicht geführt werden. Er ist nachzulesen in [Kan96, Rao73, WAH93].

Maximum-Likelihood-Schätzung und Cramér-Rao-Schranke

Der Maximum-Likelihood-Schätzer bestimmt die Parameter $\boldsymbol{\theta}$ einer Wahrscheinlichkeitsverteilung so, dass die Wahrscheinlichkeit $p(\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_N; \boldsymbol{\theta})$ einer Beobachtung $\{\boldsymbol{y}_i\}$ maximiert wird. Dies ist äquivalent zur Minimierung der *Log-Likelihood-Funktion*

$$g = -\ln p(\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_N; \boldsymbol{\theta}) \quad . \tag{3.10}$$

Maximum-Likelihood-Schätzung ist erwartungstreu, d. h. die Cramér-Rao-Ungleichung (3.8) gilt. Verwendet man als Wahrscheinlichkeitsverteilung des Rauschens eine Normalverteilung, so ist die Maximum-Likelihood-Methode sogar effizient, d. h. es gilt

$$\boldsymbol{V}(\boldsymbol{\theta}) = \boldsymbol{F}^+ \quad . \tag{3.11}$$

Verwendet man eine andere Verteilung, so wird die Schranke immerhin noch asymptotisch für $N \to \infty$ erreicht, wobei N die Anzahl der Daten ist [Kan96].

Die Maximum-Likelihood-Schätzung (und damit der Bündelausgleich) ist also optimal, wenn man als Maß für die Optimalität die Kovarianzmatrix verwendet. Wie die Gleichheit in (3.11) zur Schätzung der Kovarianzmatrix der geschätzten Parameter beim Bündelausgleich verwendet werden kann, wird im folgenden Abschnitt erläutert.

3.2.2 Bündelausgleich als Maximum-Likelihood-Schätzung

Das beim Bündelausgleich verwendete Modell für die Projektion des Weltpunktes $\boldsymbol{w}_j$ ins i-te Bild sieht folgendermaßen aus:

$$\boldsymbol{u}_{ij} = \boldsymbol{f}(\boldsymbol{\theta}) + \boldsymbol{\epsilon}_{ij} \quad , \tag{3.12}$$

mit

$$\boldsymbol{f}(\boldsymbol{\theta}) = \tilde{\boldsymbol{u}}_{ij} = \begin{pmatrix} \frac{\boldsymbol{p}_{i1}^T \boldsymbol{w}_j}{\boldsymbol{p}_{i3}^T \boldsymbol{w}_j} \\ \frac{\boldsymbol{p}_{i2}^T \boldsymbol{w}_j}{\boldsymbol{p}_{i3}^T \boldsymbol{w}_j} \end{pmatrix} \quad . \tag{3.13}$$

Die beobachteten Bildpunkte $\boldsymbol{u}_{ij}$ entstehen also durch ein additives Rauschen aus den $\tilde{\boldsymbol{u}}_{ij}$, welches durch die Zufallsvariable $\boldsymbol{\epsilon}_{ij}$ modelliert wird. Es gelten für den euklidischen Fall die in Kapitel 4 genannten Einschränkungen für die Elemente $\boldsymbol{p}_{ik}$ der Projektionsmatrizen. Zu beachten ist, dass sowohl die Parameter aller Projektionsmatrizen $\boldsymbol{P}_i$ als auch die Koordinaten der 3-D-Punkte $\boldsymbol{w}_j$ zu schätzen sind. Diese werden im Parametervektor $\boldsymbol{\theta}$ zusammengefasst.

Beim Bündelausgleich wird angenommen, dass die Zufallsvariable $\boldsymbol{\epsilon}_{ij}$ normalverteilt und mittelwertfrei ist. Unter der Voraussetzung statistischer Unabhängigkeit und gleicher Kovarianzmatrizen $\boldsymbol{\Sigma}$ ergibt sich damit folgende Wahrscheinlichkeitsdichte für n Punkte in m Bildern:

$$p(\boldsymbol{u}_{11}, \ldots, \boldsymbol{u}_{1n}, \boldsymbol{u}_{21}, \ldots, \boldsymbol{u}_{mn}; \boldsymbol{\theta}) = \frac{1}{\sqrt{|2\pi\boldsymbol{\Sigma}|}^{mn}} \, e^{-\frac{1}{2}\sum_{j=1}^{n}\sum_{i=1}^{m}(\boldsymbol{u}_{ij}-\tilde{\boldsymbol{u}}_{ij})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{u}_{ij}-\tilde{\boldsymbol{u}}_{ij})} \quad . \tag{3.14}$$

Die Wahl einer Normalverteilung kann durchaus gerechtfertigt werden, da das Rauschen auf den Bildpunkten durch Überlagerung mehrerer Fehlerquellen entsteht: Nach dem zentralen Grenzwertsatz ist die daraus entstehende Verteilung asymptotisch normalverteilt [Kre91, WAH93]. In der Praxis kommen die verwendeten Punktmerkmale aus einem Algorithmus zur Detektion und Verfolgung von Punktkorrespondenzen in mehreren Bildern einer Sequenz. Dessen Leistung wird von vielen möglichen Fehlerquellen beeinflusst, wie z. B. [WAH93]:

- der Struktur (z. B. Verdeckungen) und Beleuchtung der Szene,
- der Auflösung der Bilder und dem daraus resultierenden Einfluss auf die Genauigkeit der Lokalisation der Punkte sowie
- dem Sensorrauschen in der Kamera.

Bei anisotropem Rauschen auf den Bildpunkten muss die Kovarianzmatrix $\boldsymbol{\Sigma}$ zumindest bis auf einen Faktor bekannt sein [Kan96]. Da dies bei der vorliegenden Anwendung normalerweise

nicht der Fall sein wird, wird zur Vereinfachung von isotropem Rauschen ausgegangen.[1] Damit gilt allgemein für $\boldsymbol{\Sigma}$ [WAH93]:

$$\boldsymbol{\Sigma} = \sigma^2 \, \boldsymbol{I} \quad . \tag{3.15}$$

Hierbei ist $\boldsymbol{I}$ die $N \times N$ Einheitsmatrix, N ist die Dimension der Zufallsvariable $\boldsymbol{\epsilon}$. Im hier vorliegenden Fall ist also $N = 2$. Damit erhält man

$$p\left(\boldsymbol{u}_{11}, \ldots, \boldsymbol{u}_{1n}, \boldsymbol{u}_{21}, \ldots, \boldsymbol{u}_{mn}; \boldsymbol{\theta}\right) = \frac{1}{\sqrt{\left(2\pi\sigma^2\right)^2}^{mn}} \, e^{-\frac{1}{2\sigma^2} \sum_{j=1}^{n} \sum_{i=1}^{m} \left(\boldsymbol{u}_{ij} - \tilde{\boldsymbol{u}}_{ij}\right)^T \left(\boldsymbol{u}_{ij} - \tilde{\boldsymbol{u}}_{ij}\right)} \quad . \tag{3.16}$$

Die Maximum-Likelihood-Schätzung minimiert also

$$R = \sum_{j=1}^{n} \sum_{i=1}^{m} \left(\boldsymbol{u}_{ij} - \tilde{\boldsymbol{u}}_{ij}\right)^T \left(\boldsymbol{u}_{ij} - \tilde{\boldsymbol{u}}_{ij}\right) \quad . \tag{3.17}$$

Man sieht: Für die Maximum-Likelihood-Schätzung ist beim Vorliegen einer isotropen Verteilung *keine* Kenntnis der Varianz erforderlich. Diese kann vielmehr nach der Minimierung geschätzt werden, und zwar wie folgt [Kan96, HÅ97]:

Das Minimum von R, welches im Folgenden mit $\hat{R}$ bezeichnet werden soll, ist abhängig von den verwendeten Daten; es ist also selbst eine Zufallsvariable. Betrachtet man den Ausruck $\frac{\hat{R}}{\sigma^2}$, so kann man nachweisen, dass dieser eine χ^2-verteilte Zufallsvariable mit $2mn - r$ Freiheitsgraden ist, wobei r die Anzahl der zu schätzenden Parameter bezeichnet; im hier vorliegenden Fall[2] ist also $r = 10m + 3n$. Ein erwartungstreuer Schätzer $\hat{\sigma}^2$ für die Varianz ist damit

$$\hat{\sigma}^2 = \frac{\hat{R}}{2mn - (10m + 3n)} \quad . \tag{3.18}$$

Berücksichtigt man zusätzlich, dass die Rekonstruktion nur bis auf eine unbekannte Ähnlichkeitstransformation möglich ist, so hat man sieben zusätzliche Freiheitsgrade[3] und erhält den Schätzer

$$\hat{\sigma}^2 = \frac{\hat{R}}{2mn - (10m + 3n - 7)} \quad . \tag{3.19}$$

Bei einer realen Szene mit vielen Punkten und vielen Aufnahmen sind diese beiden Formeln gleichwertig.

Im vorhergehenden Abschnitt wurde erläutert, dass die Cramér-Rao-Schranke bei Maximum-Likelihood-Schätzung und Verwendung einer Normalverteilung erreicht wird (3.11). Für den Bündelausgleich bedeuten diese Ergebnisse insbesondere, dass die Jacobi-Matrix $\boldsymbol{J}$ der ersten Ableitungen der Residuumfunktion zur Schätzung der Kovarianzmatrix $\boldsymbol{V}$ der Parameter $\boldsymbol{\theta}$ verwendet werden kann.[4] Im hier vorliegenden Fall erhält man einen Schätzwert der Kovarianzmatrix $\boldsymbol{V}$ der geschätzten Parameter $\hat{\boldsymbol{\theta}}$, die aus dem Optimierungsverfahren stammen, durch [WAH93]

$$\boldsymbol{V}\left(\hat{\boldsymbol{\theta}}\right) = \boldsymbol{F}^{+} = \sigma^2 \left(\boldsymbol{J}^T \boldsymbol{J}\right)^{+} \quad , \tag{3.20}$$

[1] die im Abschnitt 5.1 vorgestellte Erweiterung des Verfahrens durch eine Gewichtung des Fehlers bietet die Möglichkeit, auch den allgemeineren Fall zu behandeln.

[2] zu schätzen: 4 intrinsische und 6 extrinsische Parameter pro Bild und 3 Koordinaten pro Weltpunkt

[3] 3 für die Rotation, 3 für die Translation und 1 für den Skalierungsfaktor

[4] zur Berechnung der Jacobi-Matrix siehe Abschnitt 4.3.2

wobei für σ^2 der erwartungstreue Schätzwert $\hat{\sigma}^2$ aus Gleichung (3.18) bzw. (3.19) verwendet werden kann.

Zusammengefasst hat man folgendes Ergebnis: Der klassische Bündelausgleich entspricht einer Maximum-Likelihood-Schätzung, wobei ein normalverteiltes, mittelwertfreies und isotropes Rauschen auf den Bildpunkten angenommen wird. Er ist damit optimal in dem Sinne, dass es kein Verfahren gibt, welches für diesen Fall eine kleinere Kovarianzmatrix liefert.

Kapitel 4

Bündelausgleich

Im vorhergehenden Kapitel wurde gezeigt, dass die Minimierung des Rückprojektionsfehlers als Maximum-Likelihood-Schätzung der Parameter aufgefasst werden kann und somit die gewonnene Lösung in einem statistischen Sinne optimal ist.

Nun soll nach einer kurzen Darstellung des Minimierungsproblems erläutert werden, wie das Problem für die nichtlineare Optimierung parametrisiert werden kann (Abschnitt 4.2), d. h. wie die zu optimierenden Parameter im Verlauf des Verfahrens verändert werden sollen. In Abschnitt 4.3 wird anschließend das zur Optimierung verwendete Gauß-Newton-Verfahren mit Levenberg-Marquardt-Erweiterung vorgestellt, wobei insbesondere Hinweise zur effizienten Minimierung gegeben werden. Eine Anpassung zur Behandlung von Fällen, in denen nicht jeder 3-D-Punkt in jeder Aufnahme sichtbar ist, wird in Abschnitt 4.4 dargestellt.

4.1 Einführung und Problemstellung

Der Begriff *Bündelausgleich* stammt ursprünglich aus der Photogrammmetrie, wo das Verfahren bereits seit langer Zeit verwendet wird [Sla80]. Man versteht darunter die Minimierung des Rückprojektionsfehlers, wozu ein *lokales* nichtlineares Optimierungsverfahren verwendet wird, was insbesondere bedeutet, dass gute Startwerte für die zu optimierenden Parameter unabdinglich sind. Diese kommen hier aus dem in [HN99] beschriebenen linearen Verfahren zur 3-D-Rekonstruktion. Es sind also folgende Daten vorhanden:

- Die in den Bildern detektierten 2-D-Punkte $\boldsymbol{u}_{ij}$. Dies ist der j-te Punkt im i-ten Bild,
- Schätzwerte für die intrinsischen und extrinsischen Kameraparameter aus Projektionsmatrizen $\boldsymbol{P}_i$,
- Schätzwerte $\boldsymbol{w}_j$ für die Koordinaten der Weltpunkte.

Daraus ergeben sich durch Rückprojektion der Weltpunkte $\boldsymbol{w}_j$ in die einzelnen Bilder die homogenen Bildpunkte $\boldsymbol{q}_{ij}$ mit Hilfe der 3×4 Projektionsmatrizen $\boldsymbol{P}_i$ wie folgt:

$$\boldsymbol{q}_{ij} = \boldsymbol{P}_i \boldsymbol{w}_j \quad . \tag{4.1}$$

Die aus den homogenen Punkten $\boldsymbol{q}_{ij}$ entstehenden rückprojizierten Bildpunkte $\tilde{\boldsymbol{u}}_{ij}$ ergeben sich insgesamt aus

$$\tilde{\boldsymbol{u}}_{ij} = \begin{pmatrix} \tilde{u}_{ij} \\ \tilde{v}_{ij} \end{pmatrix} = \begin{pmatrix} \frac{\boldsymbol{p}_{i1}^T \boldsymbol{w}_j}{\boldsymbol{p}_{i3}^T \boldsymbol{w}_j} \\ \frac{\boldsymbol{p}_{i2}^T \boldsymbol{w}_j}{\boldsymbol{p}_{i3}^T \boldsymbol{w}_j} \end{pmatrix} \quad . \tag{4.2}$$

Unter Bündelausgleich versteht man nun die Minimierung des Rückprojektionsfehlers:

$$\min_{\boldsymbol{P}_i, \boldsymbol{w}_j} \sum_{j=1}^{n} \sum_{i=1}^{m} \left((u_{ij} - \tilde{u}_{ij})^2 + (v_{ij} - \tilde{v}_{ij})^2 \right) \quad . \tag{4.3}$$

Dabei ist m die Anzahl der Aufnahmen und n die Anzahl der Weltpunkte.

Da hier eine euklidische Rekonstruktion der Kameramatrix sowie der Weltpunkte erfolgen soll[1], darf die Minimierung nicht mit den Elementen der Projektionsmatrizen direkt erfolgen, denn dann würde man eine projektive Rekonstruktion erhalten. Bei einer euklidischen Rekonstruktion muss für eine Projektionsmatrix $\boldsymbol{P}_i$ gelten

$$\boldsymbol{P}_i = \boldsymbol{K}_i \boldsymbol{R}_i^T (\boldsymbol{I}_3 | - \boldsymbol{t}_i) = \begin{pmatrix} f_{xi} & 0 & u_{0i} \\ 0 & f_{yi} & v_{0i} \\ 0 & 0 & 1 \end{pmatrix} \boldsymbol{R}_i^T \begin{pmatrix} 1 & 0 & 0 & -t_{xi} \\ 0 & 1 & 0 & -t_{yi} \\ 0 & 0 & 1 & -t_{zi} \end{pmatrix} \quad . \tag{4.4}$$

Wie man sieht, wird angenommen, dass die Koordinatenachsen des Bildkoordinatensystems senkrecht aufeinander stehen; weitere Einschränkungen bzgl. der intrinsischen Kameraparameter gibt es nicht. Die 3×3 Rotationsmatrix $\boldsymbol{R}_i$ und der 3-D-Vektor $\boldsymbol{t}_i$ enthalten die extrinsischen Kameraparameter; sie definieren die Bewegung der Kamera. Auch bei der Rotationsmatrix werden nicht alle Elemente direkt als zu optimierende Parameter verwendet, da diese zwar neun Elemente, aber wegen der geforderten Orthogonalität sowie der Einschränkung auf Determinante gleich eins nur drei Freiheitsgrade hat. Zwei mögliche Parametrisierungen werden in Abschnitt 4.2.2 angegeben.

4.2 Parametrisierung

Zur Lösung des in Gleichung (4.3) angegebenen nichtlinearen Optimierungsproblems soll eine lokale Parametrisierung verwendet werden, d. h. man geht von einer bereits vorhandenen Lösung aus, die z. B. mit dem in [HN99] vorgestellten linearen Verfahren gewonnen werden kann. Die zu ermittelnden Parameter sollen also die – nicht notwendigerweise additiven – Änderungen der intrinsischen und extrinsischen Kameraparameter sowie der Weltpunkte angeben.

Dieser Abschnitt basiert im wesentlichen auf [HÅ97]. Abgesehen von der Rotationsmatrix wird die Parametrisierung in der Literatur allerdings in den meisten Fällen ähnlich gehandhabt. Oft variiert auch die Anzahl der gesuchten Parameter, so auch in [HÅ97], wo f_{xi} und f_{yi} gleich sind. Es werden dort also im Gegensatz zu dieser Arbeit quadratische Pixel angenommen.

Gesucht sind die Parameter der Projektionsmatrizen $\boldsymbol{P}_i$ sowie die Koordinaten der Weltpunkte $\boldsymbol{w}_j$:

$$f_{xi}, f_{yi}, u_{0i}, v_{0i}, t_{ix}, t_{iy}, t_{iz}, \boldsymbol{R}_i, w_{jx}, w_{jy}, w_{jz} \quad , \tag{4.5}$$

[1] also bis auf eine Ähnlichkeitstransformation

wobei jede Rotationsmatrix $\boldsymbol{R}_i$ nur drei Freiheitsgrade hat. Bei m Aufnahmen hat man also insgesamt $10m$ Freiheitsgrade für die Projektionsmatrizen. Hinzu kommen noch $3n$ für die Weltpunkte, insgesamt ist somit ein $10m + 3n$ dimensionales Minimierungsproblem zu lösen. Bedenkt man, dass bereits kleine reale Bildfolgen z. B. aus 50 Bildern und 1000 Punkten bestehen[2], so hat man einen 3500-dimensionalen Suchraum. Dies kann möglicherweise zu Problemen mit dem *Fluch der Dimension* führen [HPN99]: In hochdimensionalen Räumen

- sind die Abstände zwischen allen Punkten groß und alle in etwa gleich,
- liegen alle Punkte am Rand des Raums,
- sind Nachbarschaften nicht lokal.

Dies hat zur Konsequenz, dass eine möglichst kleine Zahl zu optimierender Parameter verwendet werden sollte, d. h. eine Überparametrisierung, die insbesondere bei der Rotation möglich ist, sollte unter allen Umständen vermieden werden. Der in Abschnitt 4.3.1 vorgestellte *eingebettete Bündelausgleich*[3] trägt dem Fluch der Dimension durch eine Reduktion der Dimension des Suchraums Rechnung.

Für das Optimierungsverfahren werden die gesuchten Parameteränderungen in einem $(10m + 3n)$ Vektor $\boldsymbol{x}$ wie folgt zusammengefasst:

$$\boldsymbol{x} = \left(\Delta \boldsymbol{a}_1^T, \ldots, \Delta \boldsymbol{a}_m^T, \Delta \boldsymbol{b}_1^T, \ldots, \Delta \boldsymbol{b}_n^T\right)^T \quad . \tag{4.6}$$

Die Vektoren $\Delta \boldsymbol{a}_i$ parametrisieren jeweils eine Projektionsmatrix, jeder Vektor hat damit 10 Elemente

$$\Delta \boldsymbol{a}_i = \left(\Delta a_{i,1}, \ldots, \Delta a_{i,10}\right)^T \quad . \tag{4.7}$$

Die Vektoren $\Delta \boldsymbol{b}_j$ parametrisieren jeweils einen Weltpunkt:

$$\Delta \boldsymbol{b}_j = \left(\Delta b_{j1}, \Delta b_{j2}, \Delta b_{j3}\right)^T \quad . \tag{4.8}$$

Wie diese Parameteränderungen mit den aktuellen Werten verrechnet werden, wird in den folgenden Abschnitten beschrieben.

4.2.1 Intrinsische Parameter, Translation und Weltpunkte

Zunächst soll der einfache Teil behandelt werden, nämlich die Veränderung der intrinsischen Parameter, des Translationsvektors sowie der Koordinaten der Weltpunkte. All diese Änderungen sind schlicht additiv.

Die ersten vier Elemente von $\Delta \boldsymbol{a}_i$ geben die Änderung in der Kalibrierungsmatrix an:

$$\boldsymbol{K}_i := \begin{pmatrix} f_{xi} + \Delta a_{i,1} & 0 & u_{0i} + \Delta a_{i,3} \\ 0 & f_{yi} + \Delta a_{i,2} & v_{0i} + \Delta a_{i,4} \\ 0 & 0 & 1 \end{pmatrix} \quad . \tag{4.9}$$

[2] siehe z. B. Abschnitt 6.3

[3] engl.: interleaved bundle-adjustment

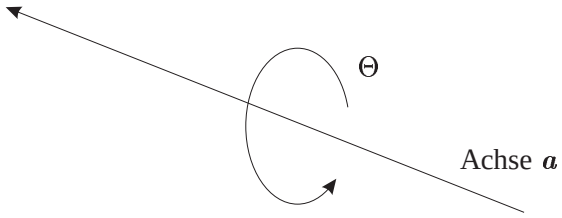


Bild 4.1: Rotation um Drehachse $\boldsymbol{a}$ mit dem Winkel Θ.

Die Elemente 5, 6 und 7 parametrisieren die Rotation; näheres dazu folgt weiter unten. Zunächst zur Translation:

$$\boldsymbol{t}_i := \begin{pmatrix} t_{ix} + \Delta a_{i,8} \\ t_{iy} + \Delta a_{i,9} \\ t_{iz} + \Delta a_{i,10} \end{pmatrix} \quad . \tag{4.10}$$

Die Parametrisierung der Weltpunkte sieht wie folgt aus[4]:

$$\boldsymbol{w}_j := \begin{pmatrix} w_{jx} + \Delta b_{j1} \\ w_{jy} + \Delta b_{j2} \\ w_{jz} + \Delta b_{j3} \\ 1 \end{pmatrix} \quad . \tag{4.11}$$

4.2.2 Rotation

Für die Parametrisierung der Rotation ist zunächst festzustellen, dass eine Rotationsmatrix zwar neun Elemente, jedoch nur drei Freiheitsgrade hat. Würde man eine ähnliche Parametrisierung wie oben verwenden, so wäre diese nicht minimal, was insbesondere die Dimension des Minimierungsproblems erhöhen würde. Zudem muss bei jeder Änderung der Rotationsmatrix darauf geachtet werden, dass auch wieder eine Rotationsmatrix entsteht, was bei einer beliebigen (additiven) Änderung nicht der Fall sein wird.

Die wohl bekannteste Form der Parametrisierung erfolgt über Eulerwinkel [TV98]: Jede Rotation kann dargestellt werden als Hintereinanderschaltung von drei Rotationen um die Koordinatenachsen. Diese wird zwar von Hartley verwendet [Har94], sie soll hier jedoch nicht weiter behandelt werden, da Eulerwinkel keine gute[5] Parametrisierung von Rotationen im Sinne von [HT99] sind. Die Verwendung einer guten Parametrisierung führt insbesondere zu einer höheren numerischen Stabilität bei der Parameterschätzung.

Achse/Winkel Darstellung

Am weitesten verbreitet ist die folgende Parametrisierung, sie wird z. B. in [HÅ97, HÅ99, SKZ99, ST98] verwendet: Eine beliebige Rotation $\boldsymbol{R}$ lässt sich darstellen als Rotation um *eine einzige* Drehachse $\boldsymbol{a} \in \mathbb{R}^3$ mit dem Winkel Θ (siehe auch Bild 4.1). Da die Länge der Drehachse $\boldsymbol{a}$ nicht von Bedeutung ist, hat $\boldsymbol{a}$ nur zwei Freiheitsgrade, weshalb man Achse und Winkel in einem einzigen Vektor $\boldsymbol{\omega}$ kombinieren kann, dessen Richtung die Drehachse und dessen Länge

[4] wie in Abschnitt 2.1 beschrieben, sind alle homogenen Weltpunkte so normiert, dass die vierte Koordinate immer eins ist.

[5] engl.: fair

den Drehwinkel darstellt:

$$\theta = |\boldsymbol{\omega}| \qquad \boldsymbol{a} = \frac{\boldsymbol{\omega}}{|\boldsymbol{\omega}|} \quad . \tag{4.12}$$

Dies ist eine gute Parametrisierung für Rotationen [HT99].

Die Berechnung einer Rotationsmatrix aus dem Vektor $\boldsymbol{\omega}$ erfolgt mit Hilfe der Formel von Rodrigues [Fau93, TV98]:

$$\boldsymbol{R} = e^{[\boldsymbol{\omega}]_\times} = \boldsymbol{I}_3 + \frac{\sin\theta}{\theta}[\boldsymbol{\omega}]_\times + \frac{1-\cos\theta}{\theta^2}[\boldsymbol{\omega}]_\times^2 \quad , \tag{4.13}$$

wobei $[\boldsymbol{\omega}]_\times$ eine antisymmetrische Matrix entsprechend Gleichung (2.6) ist.

Als Parameter für die Optimierung der Rotationsmatrix des i-ten Bildes werden die Elemente des Vektors $\boldsymbol{\omega}_i$ verwendet:

$$\boldsymbol{\omega}_i = \begin{pmatrix} \Delta a_{i,5} \\ \Delta a_{i,6} \\ \Delta a_{i,7} \end{pmatrix} \quad . \tag{4.14}$$

Die Änderung der Rotationsmatrix $\boldsymbol{R}_i$ erfolgt dann durch:

$$\boldsymbol{R}_i := \boldsymbol{R}_i \boldsymbol{R}_i' \quad , \tag{4.15}$$

wobei die Matrix $\boldsymbol{R}_i'$ aus $\boldsymbol{\omega}_i$ mit Gleichung (4.13) berechnet wird. Der umgekehrte Weg, die Berechnung der Achse und des Winkels aus einer Rotationsmatrix, wird für das hier beschriebene Verfahren nicht benötigt, da nur *Änderungen* in der Rotation parametrisiert werden. Als Startwert für $\boldsymbol{\omega}_i$ wird also immer der Nullvektor verwendet. Dennoch ist dies im Anhang A beschrieben, da man Rotationsachse und -winkel für die im nächsten Abschnitt dargestellte Quaternionenparametrisierung benötigt.

In der Literatur wird oft folgende Änderung der Rotationsmatrix angegeben:

$$\boldsymbol{R}_i := \boldsymbol{R}_i \left(\boldsymbol{I}_3 + [\boldsymbol{\omega}_i]_\times\right) = \boldsymbol{R}_i \begin{pmatrix} 1 & -\Delta a_{i,7} & \Delta a_{i,6} \\ \Delta a_{i,7} & 1 & -\Delta a_{i,5} \\ -\Delta a_{i,6} & \Delta a_{i,5} & 1 \end{pmatrix} \quad . \tag{4.16}$$

Gleichung (4.16) ergibt sich aus (4.13), wenn die Reihenentwicklung der Exponentialfunktion nach dem zweiten Glied abgebrochen wird. Sie ist also nur eine Näherungsformel, welche für kleine Änderungen zulässig ist. Für die Implementierung sollte also aus numerischen Gründen immer der Weg über die Rodrigues-Formel gewählt werden.

Quaternionen

Dieser Abschnitt beschreibt die Parametrisierung der Rotation mit Quaternionen, wobei zunächst auf die dabei auftretenden Probleme eingegangen werden soll. Eine Einführung in die Verwendung von Quaternionen zur Darstellung von Rotationen befindet sich in Anhang A.

Quaternionen werden z. B. in [SK93] verwendet[6]; sie sind eine gute Parametrisierung für Rotationen [HT99]. Da Quaternionen in vielen Anwendungen Vorteile gegenüber den anderen Parametrisierungen besitzen, sollen sie auch hier näher betrachtet werden. Sie werden im Rechnersehen z. B. zur numerisch stabilen Berechnung der Rotation aus der Kernmatrix $\boldsymbol{E}$ benutzt

[6] wobei allerdings nur erwähnt wird, *dass* diese verwendet werden, jedoch nicht *wie* genau parametrisiert wurde

[Fau93] oder auch in der Computergrafik zur Interpolation zwischen zwei gegebenen Rotationen, da dies zu glatten Bewegungen führt [WW92].

Das Ziel ist es, ausgehend von einem Quaternion $\boldsymbol{h}_0$ als initialem Schätzwert für die Rotation deren Änderung zu parametrisieren. Das Hauptproblem bei der Verwendung von Quaternionen rührt daher, dass ein Quaternion, welches eine Rotation darstellen soll, zwar vier Parameter, aber nur drei Freiheitsgrade[7] hat. Will man eine minimale Parametrisierung, welche beim Bündelausgleich schon aus Geschwindigkeitsgründen essenziell ist, muss man darüber nachdenken, wie diese erreicht werden kann. Zunächst sollen einige Vorschläge näher betrachtet werden:

Änderung aller vier Parameter des Quaternions. Hier wird die Normbedingung bei der Parametrisierung gar nicht berücksichtigt, man normiert vielmehr nach einer Veränderung das Quaternion wieder auf eins. Dieser Vorschlag hat zwei Nachteile: Zum einen werden vier Parameter verändert, wo nur drei nötig wären, und zum anderen wird die Norm erst nach der Veränderung erzwungen. Besser wäre eine Änderung, bei der die Norm des Quaternions erhalten bleibt.

Verwendung des Verfahrens von Lagrange. Das Verfahren von Lagrange zur nichtlinearen Optimierung ist ein Standardansatz, wenn es um die Einhaltung von Nebenbedingungen bei der Lösung von Optimierungsproblemen geht. Dazu müsste man die Zielfunktion (4.3) so erweitern, dass für jedes Bild ein Ausdruck der Form $\lambda_i(\boldsymbol{h}_i^T\boldsymbol{h}_i - 1)$ hinzukommt. Dies ist kein prinzipielles Problem, hat aber den Nachteil, dass sich die Anzahl der zu schätzenden Parameter sogar von vier auf fünf *erhöht* (und zwar pro Bild), da jetzt noch die Lagrange-Faktoren λ_i in der Optimierung berücksichtigt werden müssen.
Eine Möglichkeit wäre, zur Regularisierung des Problems die Faktoren λ_i vor der Optimierung festzulegen und nicht mit zu schätzen. Dann tritt allerdings die Frage auf, *wie groß* die Faktoren zu wählen sind. Dieser Ansatz bestraft Abweichungen des Quaternions von der Norm eins mit einer Erhöhung des Fehlers. Eine (innerhalb der numerischen Genauigkeit) exakte Einhaltung der Normbedingung ist jedoch nicht gewährleistet.

Änderung der Imaginärteile des Quaternions, der Realteil wird hieraus berechnet. Diese Parametrisierung wird in [AP95] verwendet, zwar auch im Zusammenhang mit dem Problem *Struktur aus Bewegung*, allerdings nicht zum Bündelausgleich. Verändert werden dort die drei Parameter ω_x, ω_y und ω_z. Daraus ergibt sich dann das Quaternion $\boldsymbol{h}$ wie folgt:

$$\boldsymbol{h} = \left(\sqrt{1 - \frac{(\omega_x^2 + \omega_y^2 + \omega_z^2)}{4}}, \frac{\omega_x}{2}, \frac{\omega_y}{2}, \frac{\omega_z}{2} \right) \quad . \tag{4.17}$$

Der Vorteil ist offensichtlich: Es werden nur drei Parameter geändert, also die minimal nötige Zahl. Allerdings muss sichergestellt sein, dass die Änderung nur so erfolgt, dass der Radikant zur Berechnung des Realteils des Quaternions immer positiv ist. Leider kann dies bei dem zum Bündelausgleich verwendeten Verfahren nicht garantiert werden, weshalb auch dieser Ansatz nicht verwendet werden soll.

Fazit: Es muss eine Parametrisierung gefunden werden,

- die nur die minimal benötigte Zahl von drei Parametern verwendet,

[7] es muss nämlich gelten: $|\boldsymbol{h}| = 1$

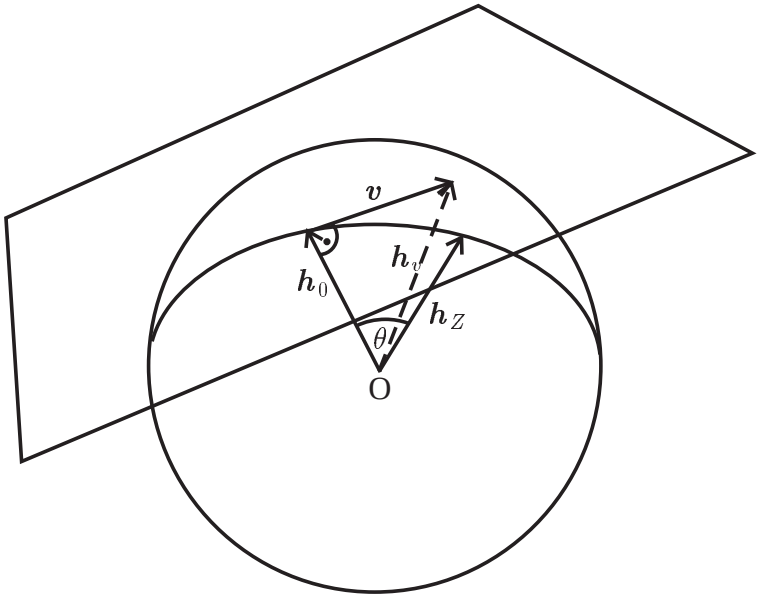


Bild 4.2: Parametrisierung der Rotation mit Quaternionen: Die Einheitskugel im 4-D, hier um eine Dimension erniedrigt dargestellt. Die Tangentialebene berührt die Kugel im Punkt $\boldsymbol{h}_0$.

- bei der diese drei Parameter beliebig verändert werden können und
- wobei das resultierende Quaternion trotzdem immer Norm eins hat.

Ein Verfahren, mit dem dies erreicht werden kann, wird im Folgenden vorgeschlagen.

Verwendet werden soll eine Parametrisierung, die ausgehend von einem Startquaternion angibt, in welcher Richtung und in welcher Entfernung auf der Einheitskugel um den Ursprung im $\mathbb{R}^4$ das neue Quaternion liegt. Statt beliebige Bewegungen im Raum der Quaternionen zuzulassen, beschränkt man sich auf Bewegungen auf der Oberfläche der Einheitskugel, welche eine 3-D-Mannigfaltigkeit des $\mathbb{R}^4$ darstellt. Es genügen hierfür folglich drei Parameter. Die Bewegung soll entlang eines Großkreises erfolgen, also auf der kürzesten Verbindung zwischen zwei Punkten auf der Kugeloberfläche.

Dies ist die grundlegende Idee. Nun soll erläutert werden, wie diese genau umgesetzt werden kann; eine graphische Darstellung dazu zeigt Bild 4.2. Zunächst muss die Frage beantwortet werden, wie die Richtung der Bewegung ausgehend von einem Startquaternion[8] $\boldsymbol{h}_0$ angegeben werden kann. Hierfür soll ein Vektor verwendet werden, der in der Tangential(hyper)ebene liegt, die im Startquaternion die Kugel berührt. Diese ist selbst ein 3-D-Unterraum des $\mathbb{R}^4$, man kann folglich Vektoren, die in der Tangentialebene liegen, als 3-D-Vektoren bezogen auf ein lokales Koordinatensystem für diese Ebene angeben. Der Ursprung des lokalen Systems soll mit demjenigen Punkt zusammenfallen, in dem die Ebene die Kugel berührt.

Um auf einer 3-D-Kugeloberfläche eine Richtung zu kodieren, genügt ein Vektor $\boldsymbol{v} \in \mathbb{R}^3$ mit nur zwei Freiheitsgraden. Der dritte Freiheitsgrad ist die Länge des Vektors, also die zurückzulegende Entfernung auf der Kugeloberfläche. Da wir es hier mit der *Einheitskugel* zu tun haben, entspricht diese Länge aber genau dem Winkel zwischen Startquaternion $\boldsymbol{h}_0$ und Zielquaternion $\boldsymbol{h}_Z$ im Bogenmaß. Der Großkreis, entlang dessen die Bewegung erfolgen soll, wird

[8] dieses ergibt sich aus der anfänglichen Schätzung der Rotation

definiert durch den Schnitt der Kugel mit derjenigen (2-D-)Ebene, welche durch das Startquaternion und den Richtungsvektor aufgespannt wird und die durch den Ursprung verläuft.
Die weitere Beschreibung ist in drei Teile gegliedert:

1. Berechnung einer (orthogonalen) Basis der Tangentialebene, welche das lokale Koordinatensystem der Ebene ist, auf den sich der 3-D-Richtungsvektor bezieht.

2. Umrechnung des 3-D-Richtungsvektors auf 4-D-Koordinaten, also einen 4-D-Vektor, der in der Hyperebene liegt.

3. Bestimmung des Zielquaternions.

Es sei im Folgenden $\boldsymbol{h}_0$ das Startquaternion, $\boldsymbol{h}_Z$ das gesuchte Zielquaternion, $\boldsymbol{v} \in \mathbb{R}^3$ der Richtungsvektor mit Koordinaten bezogen auf die Basis der Tangentialebene.

Berechnung der Basis der Tangentialebene. Gesucht ist zunächst die Tangentialebene an die Einheitskugel im Punkt $\boldsymbol{h}_0$. Diese ist definiert durch alle Punkte $\boldsymbol{x} \in \mathbb{R}^4$ mit

$$\boldsymbol{h}_0^T \boldsymbol{x} = 1 \quad , \tag{4.18}$$

da man $\boldsymbol{h}_0$ als Normalenvektor der Tangentialebene auffassen kann. Um Verwirrungen zu vermeiden, wird der Normalenvektor im Folgenden mit $\boldsymbol{n}$ bezeichnet; man sollte jedoch in Erinnerung behalten, dass immer gilt

$$\boldsymbol{n} = \begin{pmatrix} n_1 \\ n_2 \\ n_3 \\ n_4 \end{pmatrix} = \boldsymbol{h}_0 \quad . \tag{4.19}$$

Damit ergibt sich die Gleichung für die Hyperebene in Normalform:

$$n_1 x_1 + n_2 x_2 + n_3 x_3 + n_4 x_4 - 1 = 0 \quad . \tag{4.20}$$

Benötigt wird hier aber nicht diese Gleichung, sondern vielmehr eine Basis des durch die Ebene definierten Unterraums. Diese erhält man durch Umrechnung von (4.20) in Parameterform.

Hierzu wird zunächst ein Element des Normalenvektors $\boldsymbol{n}$ ausgewählt, das verschieden von null ist.[9] Dies sei o. B. d. A. n_1. Nun wählt man die drei Parameter für die Richtungsvektoren, hier $\lambda = x_2$, $\mu = x_3$ und $\nu = x_4$. Einsetzen in (4.20) und Auflösen nach x_1 ergibt

$$\begin{aligned} x_1 &= \frac{1}{n_1}(1 - n_2 x_2 - n_3 x_3 - n_4 x_4) \quad , \\ x_2 &= \lambda, \qquad x_3 = \mu, \qquad x_4 = \nu \quad . \end{aligned} \tag{4.21}$$

Es ergibt sich die Parameterdarstellung der Ebenengleichung:

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 1/n_1 \\ 0 \\ 0 \\ 0 \end{pmatrix} + \lambda \begin{pmatrix} -n_2/n_1 \\ 1 \\ 0 \\ 0 \end{pmatrix} + \mu \begin{pmatrix} -n_3/n_1 \\ 0 \\ 1 \\ 0 \end{pmatrix} + \nu \begin{pmatrix} -n_4/n_1 \\ 0 \\ 0 \\ 1 \end{pmatrix} \quad . \tag{4.22}$$

[9] eines der n_i wird immer verschieden von null sein, da $|\boldsymbol{n}| = 1$.

Als Basis werden nun die drei Richtungsvektoren verwendet, wobei diese im Folgenden mit $\boldsymbol{b}_i$ ($i = 1, 2, 3$) bezeichnet werden sollen:

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 1/n_1 \\ 0 \\ 0 \\ 0 \end{pmatrix} + \lambda \boldsymbol{b}_1 + \mu \boldsymbol{b}_2 + \nu \boldsymbol{b}_3 \quad . \tag{4.23}$$

Anmerkung: Diese Basis ist natürlich im Allgemeinen nicht orthogonal. Möchte man eine orthogonale Basis verwenden, so kann man mit Hilfe der Singulärwertzerlegung eine solche berechnen (siehe Anhang B). Dies wurde in der Implementierung auch getan, da so evtl. auftretende numerische Probleme durch beinahe linear abhängige Vektoren vermieden werden.

Umrechnung des 3-D-Richtungsvektor auf 4-D-Koordinaten. Das zur Darstellung des Richtungsvektors $\boldsymbol{v}$ verwendete Koordinatensystem besteht nun aus der Basis, gebildet durch die Vektoren $\boldsymbol{b}_i$. Allerdings soll als Aufhängepunkt nicht derjenige aus Gleichung (4.23) verwendet werden, sondern das Startquaternion $\boldsymbol{h}_0$, welches ja ein Punkt der Ebene ist, da diese in $\boldsymbol{h}_0$ die Kugel berührt. Hat man den Vektor $\boldsymbol{v} = (v_1, v_2, v_3)^T$ gegeben, so erhält man die Koordinaten des Quaternions $\boldsymbol{h}_v$, auf welches $\boldsymbol{v}$ zeigt, durch:

$$\boldsymbol{h}_v = \boldsymbol{h}_0 + v_1 \boldsymbol{b}_1 + v_2 \boldsymbol{b}_2 + v_3 \boldsymbol{b}_3 \quad . \tag{4.24}$$

Benötigt wird jedoch eigentlich nicht $\boldsymbol{h}_v$, sondern die Darstellung von $\boldsymbol{v}$ durch einen 4-D-Vektor, welcher mit $\boldsymbol{v}_4$ bezeichnet werden soll. Es gilt:

$$\boldsymbol{v}_4 = \boldsymbol{h}_v - \boldsymbol{h}_0 = v_1 \boldsymbol{b}_1 + v_2 \boldsymbol{b}_2 + v_3 \boldsymbol{b}_3 \quad . \tag{4.25}$$

Bestimmung des Zielquaternions. Das gesuchte Quaternion $\boldsymbol{h}_Z$ liegt auf dem Großkreis, der durch den Schnitt der Kugel mit derjenigen (2-D-)Ebene definiert wird, welche die Vektoren $\boldsymbol{h}_0$ und $\boldsymbol{v}_4$ aufspannen:

$$\boldsymbol{x} = \lambda \boldsymbol{h}_0 + \mu \boldsymbol{v}_{4N} \quad , \tag{4.26}$$

wobei

$$\boldsymbol{v}_{4N} = \frac{\boldsymbol{v}_4}{|\boldsymbol{v}_4|} \quad . \tag{4.27}$$

Diese Normierung ist für die Darstellung der Ebene natürlich überflüssig, die nachfolgende Berechnung des Zielquaternions wird jedoch dadurch etwas übersichtlicher. Man kann feststellen: Die Vektoren $\boldsymbol{h}_0$ und $\boldsymbol{v}_{4N}$ bilden ein rechtwinkliges Koordinatensystem, dessen Ursprung mit dem Ursprung des 4-D-Raums zusammenfällt. Das Zielquaternion $\boldsymbol{h}_Z$ ist nun ein Vektor, der in dieser Ebene liegt und mit dem Startquaternion den Winkel θ bildet. Man erhält $\boldsymbol{h}_Z$ dann als Linearkombination der beiden Richtungsvektoren der Ebene, wie aus Bild 4.3 leicht ersichtlich ist:

$$\boldsymbol{h}_Z = \sin(\theta)\, \boldsymbol{v}_{4N} + \cos(\theta)\, \boldsymbol{h}_0 \quad , \tag{4.28}$$

wobei

$$\theta = |\boldsymbol{v}_4| \quad . \tag{4.29}$$

Wie leicht nachzurechnen ist, hat das resultierende Quaternion $\boldsymbol{h}_Z$ automatisch wieder Norm eins.

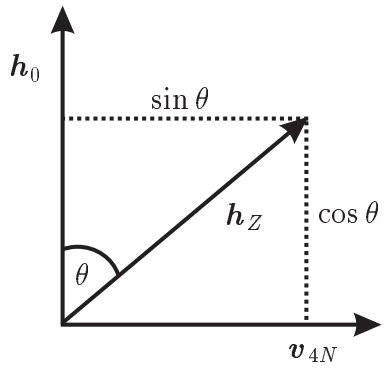


Bild 4.3: Die Vektoren $\boldsymbol{h}_0$ und $\boldsymbol{v}_{4N}$ bilden eine orthogonale Basis der Ebene, in der das gesuchte Quaternion $\boldsymbol{h}_Z$ liegt.

Als Parameter in der Optimierung wird also der 3-D-Vektor $\boldsymbol{v}$ verwendet, aus dem wie oben beschrieben ein Quaternion $\boldsymbol{h}_Z$ berechnet werden kann, welches die neue Rotation der Kamera angibt:

$$\boldsymbol{v} = \begin{pmatrix} \Delta a_{i,5} \\ \Delta a_{i,6} \\ \Delta a_{i,7} \end{pmatrix} \quad . \tag{4.30}$$

Zusammengefasst muss man vor der Optimierung für jede Aufnahme einmalig folgende Berechnungen durchführen:

- Berechnung des Startquaternions $\boldsymbol{h}_0$ aus der Rotationsmatrix (siehe Anhang A),
- Bestimmung der Basis der zugehörigen Tangentialebene, Zusammenfassung der Basisvektoren in einer 4×3 Matrix $\boldsymbol{B}$,
- Bestimmung einer orthogonalen Basis $\boldsymbol{B}'$ aus $\boldsymbol{B}$ mit Hilfe der Singulärwertzerlegung (siehe Anhang B).

Während der Optimierung sind nur noch folgende Berechnungen nötig:

- Umrechnung des 3-D-Vektors $\boldsymbol{v}$ in einen 4-D-Vektor $\boldsymbol{v}_4$ durch

$$\boldsymbol{v}_4 = \boldsymbol{B}'\boldsymbol{v} \quad , \tag{4.31}$$

- Normierung von $\boldsymbol{v}_4$ und Bestimmung des resultierenden Quaternions $\boldsymbol{h}_Z$ mit Formel (4.28),
- Umrechnung von $\boldsymbol{h}_Z$ in eine Rotationsmatrix (siehe Anhang A).

4.3 Lösung des Minimierungsproblems

Nun soll beschrieben werden, wie das vorgestellte Minimierungsproblem mit Hilfe der Gauß-Newton-Methode – einem lokalen Optimierungsverfahren für nichtlineare Gleichungssysteme – gelöst werden kann. Wegen der größeren numerischen Stabilität wird daran anschließend die Levenberg-Marquardt-Erweiterung des Gauß-Newton-Verfahrens erläutert. Näheres

zu diesen Optimierungsverfahren findet man in [DS83, Sch93, PTVF92]. Kurze Beschreibungen der beiden Methoden sowie ihre Anwendung für den Bündelausgleich findet man in [HÅ97, Har94, HÅ99, McL99].

Prinzipiell ist auch die Anwendung eines beliebigen anderen nichtlinearen Optimierungsverfahrens denkbar. Der Grund dafür, dass hier gerade Gauß-Newton das Mittel der Wahl ist, liegt im außerordentlich hohen Rechenaufwand, den eine nichtlineare Optimierung erfordert. Es zeigt sich nämlich, dass die beim Gauß-Newton-Verfahren auftretende Jacobi-Matrix der ersten Ableitungen der Fehlerfunktion im Fall des Bündelausgleichs eine besondere Blockstruktur besitzt. Berücksichtigt man diese Struktur im Algorithmus, so ergibt sich eine Reduktion des Berechnungsaufwands *von kubischer auf lineare Komplexität* in der Anzahl der Aufnahmen.

In Abschnitt 4.3.1 folgt nun zunächst eine Beschreibung zweier Ansätze zur Minimierung. Bei Verwendung des dort vorgestellten *eingebetteten Bündelausgleichs* erreicht man eine weitere Reduktion des Rechenaufwands durch geschickte Umformulierung des Optimierungsproblems. Anschließend wird das Gauß-Newton-Verfahren (Abschnitt 4.3.2) sowie die Levenberg-Marquardt-Erweiterung (Abschnitt 4.3.3) erläutert, noch unabhängig von der oben erwähnten Blockstruktur der Jacobi-Matrix. Wie diese genutzt werden kann, um die Optimierung effizienter zu machen, steht in Abschnitt 4.3.4.

4.3.1 Zwei Ansätze zur Minimierung

Bevor das Gauß-Newton-Verfahren erläutert wird, sollen hier zwei etwas unterschiedliche Ansätze zur Minimierung vorgestellt werden, wobei der erstere offensichtlich ist – es wird einfach der Rückprojektionsfehler minimiert (Formel (4.3)) – während der zweite sich durch eine Umformulierung der Fehlerfunktion ergibt. Der zu minimierende Ausdruck lautet

$$\min_{\boldsymbol{P}_i,\boldsymbol{w}_j} \sum_{j=1}^{n} \sum_{i=1}^{m} \left(\left(u_{ij} - \frac{\boldsymbol{p}_{i1}^T \boldsymbol{w}_j}{\boldsymbol{p}_{i3}^T \boldsymbol{w}_j} \right)^2 + \left(v_{ij} - \frac{\boldsymbol{p}_{i2}^T \boldsymbol{w}_j}{\boldsymbol{p}_{i3}^T \boldsymbol{w}_j} \right)^2 \right) \quad . \tag{4.32}$$

Dabei ist m die Anzahl der Aufnahmen und n die Anzahl der Weltpunkte. Außerdem sind die Einschränkungen der Parameter der Projektionsmatrizen bei euklidischer Rekonstruktion zu beachten (siehe Abschnitt 4.2).

Die zwei möglichen Ansätze zur Lösung dieses Problems sind:

Gleichzeitige Optimierung von Struktur und Kameraparametern. Dieser Ansatz wird in [Har94, HÅ97] verfolgt, er benötigt jedoch erheblich mehr Rechenzeit als der weiter unten stehende. Außerdem betrachtet man hier ein Optimierungsproblem in einem hochdimensionalen Raum ($10m + 3n$ Dimensionen), weshalb es möglicherweise früher oder später zu Problemen mit dem Fluch der Dimension kommt.

Eingebetteter Bündelausgleich. Dieser Ansatz wird in [SKZ99, ST98] verwendet, sowie in [Zha98a] zur Schätzung der Fundamentalmatrix. Man nutzt dabei aus, dass die Schätzung der 3-D-Struktur bei festen Kameraparametern für jeden Weltpunkt unabhängig durchgeführt werden kann. Dieses Vorgehen soll hier ebenfalls verwendet werden, da es das Optimierungsproblem in Räumen löst, die eine wesentlich niedrigere Dimension haben, als bei gleichzeitiger Optimierung von Struktur und Kameraparametern. Zudem ist der Berechnungsaufwand geringer.

Betrachten wir den eingebetteten Bündelausgleich näher. Hierbei wird Gleichung (4.32) in folgendes Optimierungsproblem überführt[10] [SKZ99]:

$$\min_{\boldsymbol{P}_i} \sum_{j=1}^{n} \min_{\boldsymbol{w}_j} \sum_{i=1}^{m} \left(\left(u_{ij} - \frac{\boldsymbol{p}_{i1}^T \boldsymbol{w}_j}{\boldsymbol{p}_{i3}^T \boldsymbol{w}_j} \right)^2 + \left(v_{ij} - \frac{\boldsymbol{p}_{i2}^T \boldsymbol{w}_j}{\boldsymbol{p}_{i3}^T \boldsymbol{w}_j} \right)^2 \right) \quad . \tag{4.33}$$

Innerhalb eines Optimierungsschrittes für die Kameraparameter $\boldsymbol{P}_i$ wird also jedesmal die 3-D-Struktur auf die aktuellen Parameter hin optimiert. Dabei ist die Optimierung eines Weltpunktes unabhängig von allen anderen Punkten. Man hat somit erreicht, dass an Stelle eines $10m + 3n$-dimensionalen Problems nur noch ein $10m$-dimensionales zu lösen ist, wobei in jedem Schritt außerdem n 3-dimensionale Optimierungsprobleme auftreten. Um dies an dem bereits in 4.2 verwendeten Beispiel einer Bildfolge mit 50 Aufnahmen und 1000 Weltpunkten deutlich zu machen: Es ergibt sich eine Reduktion der Dimension des Suchraums von 3500 auf 500 mit 1000 3-dimensionalen Einzeloptimierungen pro Schritt.

Zusätzlich ergibt sich bei Verwendung des eingebetteten Bündelausgleichs eine geringere Zeitkomplexität. Mehr hierzu befindet sich in Abschnitt 4.3.2.

4.3.2 Das Gauß-Newton-Verfahren

Es wird jetzt zunächst die Gauß-Newton-Methode zur lokalen nichtlinearen Optimierung beschrieben [DS83, Sch93, HÅ97, Har94]. Wegen der größeren numerischen Stabilität sollte dieses Verfahren jedoch nur zusammen mit der Levenberg-Marquardt-Erweiterung[11] verwendet werden.

Es wird von einer bereits vorhandenen Näherungslösung $\boldsymbol{x}_k$ für die gesuchten Parameter im k-ten Iterationsschritt ausgegangen.[12] Es sei weiterhin $\boldsymbol{\epsilon}(\boldsymbol{x}) \in \mathbb{R}^{2nm}$ eine Residuumfunktion, die den (nicht quadrierten) Fehler auf den Bildpunkten bei der Rückprojektion mit den Parametern $\boldsymbol{x}$ liefert. [13] Dabei kommen zuerst alle Fehler in den Koordinaten der Bildpunkte der ersten Aufnahme, dann die der zweiten usw. Der Fehler im j-ten Bildpunkt in der i-ten Aufnahme hat also die Indizes $2n(i-1)+2(j-1)+1$ für die u-Koordinate und $2n(i-1)+2(j-1)+2$ für die v-Koordinate:[14]

$$\boldsymbol{\epsilon}(\boldsymbol{x}) = \begin{pmatrix} \epsilon_1 \\ \vdots \\ \epsilon_{2n(i-1)+2(j-1)+1} \\ \epsilon_{2n(i-1)+2(j-1)+2} \\ \vdots \\ \epsilon_{2nm} \end{pmatrix} = \begin{pmatrix} u_{1,1} - \tilde{u}_{1,1} \\ v_{1,1} - \tilde{v}_{1,1} \\ \vdots \\ u_{ij} - \tilde{u}_{ij} \\ v_{ij} - \tilde{v}_{ij} \\ \vdots \\ u_{mn} - \tilde{u}_{mn} \\ v_{mn} - \tilde{v}_{mn} \end{pmatrix} \quad . \tag{4.34}$$

[10] die Zielfunktion ist dabei immer noch dieselbe, das Problem wird hier nur geschickter formuliert

[11] siehe Abschnitt 4.3.3

[12] bei einer lokalen nichtlinearen Optimierung werden immer gute Startwerte benötigt, da das Verfahren zum nächsten (lokalen) Minimum hin konvergiert, welches nicht das gesuchte sein muss. Die Startwerte für den Bündelausgleich kommen aus einem vorhergehenden linearen Verfahren zur 3-D-Rekonstruktion [HN99].

[13] Zur Erinnerung: m ist die Anzahl der Aufnahmen und n die Anzahl der Weltpunkte.

[14] wobei gilt: $i = 1, \ldots, m$ und $j = 1, \ldots, n$

Wie man sieht, wird hierbei davon ausgegangen, dass jeder Weltpunkt auch in jedem Bild sichtbar ist. Dies wird in realen Bildfolgen natürlich nicht der Fall sein, weshalb in Abschnitt 4.4 erläutert wird, auf welche Weise dieses Problem gelöst werden kann.

Minimiert werden soll der quadratische Rückprojektionsfehler $\boldsymbol{\epsilon}^T\boldsymbol{\epsilon}$ entsprechend (4.3), was mit der Gauß-Newton-Methode wie folgt funktioniert:
Die Funktion $\boldsymbol{\epsilon}(\boldsymbol{x})$ wird um den Wert $\boldsymbol{x}_k$ herum linearisiert. Dies liefert

$$\boldsymbol{\epsilon}(\boldsymbol{x}) = \boldsymbol{\epsilon}(\boldsymbol{x}_k + \Delta\boldsymbol{x}) \approx \boldsymbol{\epsilon}(\boldsymbol{x}_k) + \frac{\partial\boldsymbol{\epsilon}}{\partial\boldsymbol{x}}(\boldsymbol{x}_k)\Delta\boldsymbol{x} \quad . \tag{4.35}$$

Es ist

$$\boldsymbol{J} = \frac{\partial\boldsymbol{\epsilon}}{\partial\boldsymbol{x}}(\boldsymbol{x}_k) \tag{4.36}$$

die Jacobi-Matrix der ersten Ableitungen der Fehlerfunktion, ausgewertet an der Stelle $\boldsymbol{x}_k$. Gesucht ist die Lösung der Gleichung

$$\boldsymbol{\epsilon}(\boldsymbol{x}) = \boldsymbol{0} \quad , \tag{4.37}$$

und mit Formel (4.35) ergibt sich

$$\Delta\boldsymbol{x} = -\boldsymbol{J}^+\boldsymbol{\epsilon}(\boldsymbol{x}_k) \quad . \tag{4.38}$$

Einen besseren Näherungswert für die gesuchten Parameter liefert die Iterationsvorschrift

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \Delta\boldsymbol{x} \quad . \tag{4.39}$$

Bei Anwendung des Standard-Gauß-Newton-Verfahrens kann die Pseudoinverse $\boldsymbol{J}^+$ direkt berechnet werden. Verwendet man dazu die Singulärwertzerlegung (siehe Anhang B), so bedeutet das einen Aufwand von $O(MN^2)$ für eine $(M \times N)$-Matrix $(M \geq N)$ [TD97]. Im hier vorliegenden Fall ist $\boldsymbol{J}$ eine $(2nm \times 10m)$-Matrix, womit man einen Berechnungsaufwand von $O(2nm\,(10m)^2) = O(nm^3)$ erhält. Hierbei wurde angenommen, dass das im vorhergehenden Abschnitt vorgestellte eingebettete Verfahren verwendet wird und dass 10 Parameter der Projektionsmatrix zu schätzen sind. Hinzu kommt der Aufwand zur Optimierung der 3-D-Punkte. Diese erfordert die Bildung von n Pseudoinversen einer $2m \times 3$ Matrix, was einen Aufwand von $O(nm)$ zur Folge hat. Die Gesamtkomplexität beträgt somit $O(nm^3 + nm) = O(nm^3)$.

Würde man das Minimierungsproblem durch gleichzeitige Optimierung von Struktur und Kameraparametern lösen, so hätte die Jacobi-Matrix eine Größe von $2nm \times (10m + 3n)$ und man erhielte einen Aufwand von $O(2nm\,(10m + 3n)^2) = O(nm^3 + n^2m^2 + n^3m)$.

4.3.3 Die Levenberg-Marquardt-Erweiterung

Das Verfahren von Levenberg-Marquardt erweitert das oben vorgestellte Gauß-Newton-Verfahren, um eine höhere numerische Stabilität zu erreichen. Dazu wird zunächst Gleichung (4.38) betrachtet, wobei die Pseudoinverse ausgeschrieben wird:

$$\Delta\boldsymbol{x} = -\boldsymbol{J}^+\boldsymbol{\epsilon}(\boldsymbol{x}_k) = -\left(\boldsymbol{J}^T\boldsymbol{J}\right)^{-1}\boldsymbol{J}^T\,\boldsymbol{\epsilon}(\boldsymbol{x}_k) \quad . \tag{4.40}$$

Die Berechnung von $(\boldsymbol{J}^T\boldsymbol{J})^{-1}$ kann numerisch schlecht konditioniert sein, wenn die Singulärwerte von $\boldsymbol{J}^T\boldsymbol{J}$ klein sind [HÅ97]. Die Parameter werden dann zwar in die richtige Richtung verändert, allerdings ist die Schrittweite möglicherweise zu groß. Daher wird beim

	Berechne $\Delta \boldsymbol{x}$ mit Gleichung (4.41)	
	Aktualisiere die Parameter mit Gleichung (4.39)	
	Berechne den Fehler mit diesen neuen Parametern	
	WENN	Fehler kleiner als im letzten Schritt?
	DANN	Reduziere λ um einen Faktor von 10, d. h. $\lambda := \lambda/10$
	SONST	Erhöhe λ um einen Faktor von 10, d. h. $\lambda := 10 \cdot \lambda$
		Verwirf die aktualisierten Parameter $\boldsymbol{x}_{k+1}$, d. h. verwende weiterhin $\boldsymbol{x}_k$
BIS	neuer Fehler $<$ alter Fehler	

Bild 4.4: Ein Levenberg-Marquardt-Iterationsschritt.

Levenberg-Marquardt-Verfahren an Stelle dieser Gleichung folgende Erweiterung verwendet [DS83, HÅ97]:

$$\Delta \boldsymbol{x} = -\left(\lambda \boldsymbol{I} + \boldsymbol{J}^T \boldsymbol{J}\right)^{-1} \boldsymbol{J}^T \boldsymbol{\epsilon}(\boldsymbol{x}_k) \quad , \tag{4.41}$$

wobei λ eine kleine positive Zahl ist, die zu Beginn des Verfahrens typischerweise auf $\lambda = 10^{-3}$ gesetzt wird [PTVF92, Har94].

Zu Gleichung (4.41) ist anzumerken, dass es für die Erweiterung auch andere Möglichkeiten gibt. In [PTVF92, Har94] wird beispielsweise folgende verwendet:

$$\Delta \boldsymbol{x} = -\boldsymbol{N}^{-1} \boldsymbol{J}^T \boldsymbol{\epsilon}(\boldsymbol{x}_k) \quad , \tag{4.42}$$

wobei für die Elemente der Matrix $\boldsymbol{N}$ gilt:

$$N_{ii} = (1+\lambda)\,(\boldsymbol{J}^T \boldsymbol{J})_{ii} \qquad \text{und} \qquad N_{ij} = (\boldsymbol{J}^T \boldsymbol{J})_{ij}, \quad i \neq j \quad . \tag{4.43}$$

In beiden Fällen werden also nur die Diagonalelemente von $\boldsymbol{J}^T \boldsymbol{J}$ verändert, die restlichen bleiben gleich. Zur Implementierung wurde die Version aus Gleichung (4.41) verwendet.

In *einem* Levenberg-Marquardt-Iterationsschritt (siehe auch Bild 4.4) wird zunächst $\Delta \boldsymbol{x}$ mit Gleichung (4.41) berechnet. Daraus erhält man mit (4.39) die neuen Parameterwerte $\boldsymbol{x}_{k+1}$. Anschließend wird der Rückprojektionsfehler mit den neuen Werten ermittelt. Ist dieser kleiner als vorher, so wird λ um einen Faktor von 10 reduziert, die neuen Parameter $\boldsymbol{x}_{k+1}$ werden als besser akzeptiert und als Startwerte für den nächsten Iterationsschritt verwendet. Ist der Fehler jedoch größer als mit den Parametern des vorherigen Iterationsschrittes, so wird λ um einen Faktor von 10 erhöht. Die berechneten Parameter werden verworfen und es wird mit dem erhöhten λ und den Werten $\boldsymbol{x}_k$ aus dem letzten Iterationsschritt von vorne begonnen.

Dies bedeutet folglich: Ein Levenberg-Marquardt-Iterationsschritt besteht aus der Suche nach einem Wert für λ, welches einen kleineren Fehler liefert; erst wenn dieser gefunden wurde erhält man bessere Parameter $\boldsymbol{x}_{k+1}$.

Betrachtet man Gleichung (4.41) genauer, so erkennt man, dass das Verfahren von Levenberg-Marquardt dynamisch, je nach Größe von λ, zwischen zwei Optimierungsverfahren hin- und herschaltet:

- Für $\lambda = 0$ ergibt sich das vorher beschriebene Gauß-Newton-Verfahren.

- Für große Werte von λ geht die Änderung der Parameter in Richtung des steilsten Abstiegs (mit kleiner Schrittweite), das Verhalten entspricht dann einem Gradientenabstiegsverfahren.

In der Praxis ist die Verwendung der Levenberg-Marquardt-Erweiterung unbedingt erforderlich, da durch die numerischen Probleme bei der Matrixinversion die Änderungen in einem Schritt bei Gauß-Newton viel zu groß werden. Dies lässt sich auch durch die Verwendung der Singulärwertzerlegung zur Bildung der Pseudoinversen nicht verhindern, da das Ergebnis der Inversion bei Singulärwerten nahe null sehr davon abhängt, wo genau der Schwellwert liegt, ab dem ein Singulärwert auf exakt null gesetzt (und damit nicht invertiert) wird. Bereits Änderungen des Schwellwerts um eine Zehnerpotenz liefern dann völlig andere Ergebnisse.[15]

Weiterhin sollte auch die Inversion des Ausdrucks $\lambda \boldsymbol{I} + \boldsymbol{J}^T \boldsymbol{J}$ in Gleichung (4.41) mit Hilfe der Singulärwertzerlegung durchgeführt werden, da je nach aktuellem λ die zu invertierende Matrix durchaus singulär sein kann. Dies wird dann aber im nächsten Schritt durch Erhöhung von λ berücksichtigt.

4.3.4 Die Jacobi-Matrix

Durch direkte Verwendung des im vorhergehenden Abschnitt beschriebenen Levenberg-Marquardt-Verfahrens kann man nun die Optimierung durchführen. Jetzt soll erläutert werden, wie die Berechnung wesentlich effizienter gemacht werden kann, indem man die durch den Bündelausgleich entstehende Blockstruktur der Jacobi-Matrix $\boldsymbol{J}$ ausnutzt. Es wird hier der Fall des eingebetteten Bündelausgleichs betrachtet, bei dem offensichtlich ist, wie diese Struktur für die Beschleunigung des Verfahrens genutzt werden kann.

Auf das Aussehen der Jacobi-Matrix bei der gleichzeitigen Optimierung von Struktur und Kameraparametern wird hier nicht eingegangen. Auch in diesem Fall hat die Jacobi-Matrix eine Blockstruktur, welche aber wesentlich komplizierter ist als diejenige beim eingebetteten Bündelausgleich. Dies ist im Artikel von Hartley [Har94] beschrieben, der das Verfahren aus der Photogrammmetrie übernommen hat, wo es z. B. in [Sla80] erläutert wird.

Es wird von Gleichung (4.38) ausgegangen, die in der folgenden Form betrachtet wird:

$$\boldsymbol{J} \Delta \boldsymbol{x} = -\boldsymbol{\epsilon}(\boldsymbol{x}_k) \quad . \tag{4.44}$$

Aus dieser Gleichung soll $\Delta \boldsymbol{x}$ berechnet werden, was im allgemeinen Gauß-Newton-Verfahren durch die Berechnung der Pseudoinversen $\boldsymbol{J}^+$ geschieht, wie es oben beschrieben wurde.

Im hier betrachteten Fall des eingebetteten Bündelausgleichs ist die Jacobi-Matrix eine $(2nm \times 10m)$-Matrix, die folgende Elemente enthält:

$$(J_{ij}) = \left(\frac{\partial \epsilon_i}{\partial x_j} \right) \quad . \tag{4.45}$$

Eine Spalte von $\boldsymbol{J}$ enthält die Änderungen der Rückprojektionsfehler bei Änderung eines einzigen Kameraparameters[16]. Jeweils 10 Spalten gehören zu einer Projektionsmatrix.

[15] Details zur Bildung der Pseudoinversen befinden sich in Anhang B. Zur Wahl des Schwellwerts siehe [PTVF92].

[16] in der vorher gewählten Parametrisierung (siehe Abschnitt 4.2)

Dies hat folgende Konsequenz: Die Ableitungen nach den Kameraparametern einer Aufnahme wirken sich immer nur auf die Fehler der Bildpunkte in *dieser* Aufnahme aus; die anderen Bilder bleiben davon völlig unberührt. Damit ergibt sich für die Jacobi-Matrix eine Blockstruktur, wie sie in Gleichung (4.46) dargestellt ist. Jeder der m Blöcke – es entsteht pro Aufnahme einer – hat folglich eine Größe von $2n \times 10$. Die Blöcke sollen hier mit $\boldsymbol{A}_i$ bezeichnet werden. Man erhält an Stelle des Gleichungssystems (4.44) nun

$$\begin{pmatrix} \boldsymbol{A}_1 & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{A}_2 & \boldsymbol{0} & \boldsymbol{0} \\ \vdots & \vdots & \vdots & \vdots \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{A}_i & \boldsymbol{0} \\ \vdots & \vdots & \vdots & \vdots \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{A}_m \end{pmatrix} \begin{pmatrix} \Delta\boldsymbol{x}_1 \\ \Delta\boldsymbol{x}_2 \\ \vdots \\ \Delta\boldsymbol{x}_i \\ \vdots \\ \Delta\boldsymbol{x}_m \end{pmatrix} = - \begin{pmatrix} \boldsymbol{\epsilon}_1 \\ \boldsymbol{\epsilon}_2 \\ \vdots \\ \boldsymbol{\epsilon}_i \\ \vdots \\ \boldsymbol{\epsilon}_m \end{pmatrix} \quad , \tag{4.46}$$

wobei der Vektor $\Delta\boldsymbol{x}_i \in \mathbb{R}^{10}$ die gesuchte Änderung der Kameraparameter der Aufnahme i enthält und $\boldsymbol{\epsilon}_i \in \mathbb{R}^{2n}$ die zugehörigen Rückprojektionsfehler, wie in (4.34) definiert. An Stelle eines einzigen großen Gleichungssystems sind nun m Systeme der Form

$$\boldsymbol{A}_i \Delta\boldsymbol{x}_i = -\boldsymbol{\epsilon}_i \tag{4.47}$$

zu lösen. Dies erfordert einen Berechnungsaufwand für die Singulärwertzerlegung von $O(nm)$, im Gegensatz zu $O(nm^3)$, wenn das System auf einmal gelöst wird.[17] Es ergibt sich eine Reduktion von kubischem Aufwand auf linearen in der Anzahl der Aufnahmen und somit eine enorme Ersparnis an Rechenzeit. Die Parameteränderungen für einen Block erhält man mit

$$\Delta\boldsymbol{x}_i = -\boldsymbol{A}_i^{+} \boldsymbol{\epsilon}_i \quad . \tag{4.48}$$

Weitere Einsparungen ergeben sich bezüglich des Speicherplatzbedarfs der Jacobi-Matrix: Prinzipiell würde es genügen, an Stelle der gesamten Matrix immer nur den gerade benötigten Block $\boldsymbol{A}_i$ zu speichern. Dadurch würde sich die Speicherkomplexität von $O(nm^2)$ auf $O(n)$ reduzieren. Allerdings ist dies in der Praxis nicht empfehlenswert, da bei Verwendung der Levenberg-Marquardt-Erweiterung die einzelnen Gleichungssysteme (4.48) immer wieder mit verschiedenen λ-Werten gelöst werden müssen. Daher muss man zwischen Speicherplatzbedarf und Rechenzeit einen Kompromiss eingehen: Wird tatsächlich nur der aktuell benötigte Block gespeichert, so muss dieser für jedes λ wieder neu berechnet werden, und zwar so oft, bis ein λ gefunden wurde, das einen kleineren Fehler liefert. Und gerade die Ermittlung der Jacobi-Matrix ist es, was in der Praxis sehr lange dauert; man sollte also vor jedem Levenberg-Marquardt-Schritt einmal *alle* Blöcke berechnen und diese speichern. Daraus resultiert immer noch eine Reduktion des Speicherplatzbedarfs auf $O(nm)$, da m Blöcke der Größe $2n \times 10$ gespeichert werden müssen.

Bisher wurde nur das Gauß-Newton-Verfahren betrachtet. Bei Verwendung der Levenberg-Marquardt-Erweiterung muss an Stelle von (4.48) die erweiterte Formel (4.49) benutzt und das Verfahren blockweise angewendet werden:

$$\Delta\boldsymbol{x}_i = -\left(\lambda \boldsymbol{I} + \boldsymbol{A}_i^T \boldsymbol{A}_i\right)^{-1} \boldsymbol{A}_i^T \boldsymbol{\epsilon}_i \quad . \tag{4.49}$$

Zu beachten ist, dass λ für jeden Block gleich groß ist, es werden also nicht verschiedene λ-Werte verwendet.

[17] siehe hierzu auch Abschnitt 4.3.2

4.4 Berücksichtigung von nicht sichtbaren Punkten

Bei der Beschreibung des im Verlauf dieses Kapitels vorgestellten Verfahrens zur nichtlinearen Optimierung der Kameraparameter sowie der 3-D-Punkte wurde bisher davon ausgegangen, dass in jedem Bild alle 3-D-Punkte auch sichtbar sind. Dies wird in einer realen Anwendung natürlich nicht der Fall sein. Dort werden von Bild zu Bild Punkte verloren gehen, sei es durch Verdeckungen oder weil sie von einer Aufnahme zur nächsten nicht verfolgt werden konnten. Außerdem werden immer wieder neue, vorher nicht sichtbare Punkte hinzukommen.

Wie dieser Fall im linearen Verfahren zur 3-D-Rekonstruktion, welches die Startwerte für den Bündelausgleich liefert, berücksichtigt werden kann, wird in [HN99] erläutert. Wie beim eingebetteten Bündelausgleich *alle* in einem Bild sichtbaren 3-D-Punkte zur Optimierung herangezogen werden können, soll im Folgenden dargestellt werden.

Die zu minimierende Funktion lautet in diesem Fall[18]

$$\min_{\boldsymbol{P}_i} \sum_{j=1}^{n} \min_{\boldsymbol{w}_j} \sum_{i=1}^{m} \left(\left(u_{ij} - \frac{\boldsymbol{p}_{i1}^T \boldsymbol{w}_j}{\boldsymbol{p}_{i3}^T \boldsymbol{w}_j} \right)^2 + \left(v_{ij} - \frac{\boldsymbol{p}_{i2}^T \boldsymbol{w}_j}{\boldsymbol{p}_{i3}^T \boldsymbol{w}_j} \right)^2 \right) \quad . \tag{4.50}$$

Bei jedem Optimierungsschritt für die Kameraparameter werden die 3-D-Punkte optimiert, und zwar jeder Punkt getrennt von allen anderen. Wie man an Gleichung (4.50) sieht, können bei der Optimierung Punkte (u_{ij}, v_{ij}), die es in einem Bild nicht gibt, beim Aufsummieren einfach weggelassen werden. Man muss nur erkennen können, dass ein Punkt nicht berücksichtigt werden soll, denn die rekonstruierten 3-D-Punkte lassen sich natürlich in jedes Bild projizieren. Dies wird in der vorliegenden Implementierung dadurch realisiert, dass die 3-D-Punkte in einer Datenstruktur gehalten werden, in der sich auch Informationen darüber befinden, in welchem Bild der Punkt sichtbar ist und mit welchem 2-D-Punkt in diesem Bild er korrespondiert.

Wenden wir uns nun der Frage zu, wie sich das Weglassen eines Punktes auf die im Gauß-Newton-Verfahren benötigten Gleichungen auswirkt. Wegen des eingebetteten Bündelausgleichs muss man zwischen der Optimierung der Kameraparameter und der Optimierung eines 3-D-Punktes unterscheiden.

Im Fall der Kameraparameter muss das in Formel (4.46) angegebene Gleichungssystem gelöst werden, was sich wegen der Blockstruktur der Jacobi-Matrix auf die Lösung von m Systemen der Form

$$\boldsymbol{A}_i \Delta \boldsymbol{x}_i = -\boldsymbol{\epsilon}_i \tag{4.51}$$

reduzieren lässt[19]. Die Matrix $\boldsymbol{A}_i$ hat dabei eine Größe von $2n \times 10$. Sie enthält die Werte der partiellen Ableitungen der Fehlerfunktion nach den Kameraparametern; jeweils zwei aufeinanderfolgende Zeilen der Matrix gehören zusammen zu einem Punkt und geben die Änderung des Rückprojektionsfehlers bei einer Veränderung der Kameraparameter an. Für den Fall, dass ein Punkt im entsprechenden Bild gar nicht vorhanden ist, fehlen zwei Gleichungen des Gleichungssystems. Für die Implementierung hat dies folgende Konsequenz: Entweder man passt den Levenberg-Marquardt-Algorithmus so an, dass er mit variablen Blockgrößen in der Jacobi-Matrix arbeiten kann, oder man lässt alle Blöcke gleich groß und füllt die Matrizen $\boldsymbol{A}_i$ sowie die Vektoren $\boldsymbol{\epsilon}_i$ an den entsprechenden Stellen mit Nullen auf.

[18] siehe auch Gleichung (4.33)
[19] siehe auch Gleichung (4.47)

Bei der Optimierung eines 3-D-Punktes ist die Situation ähnlich, es gehen dort ebenfalls pro nicht vorhandenem Punkt zwei Gleichungen verloren. Der Unterschied besteht lediglich darin, dass die Jacobi-Matrix keine Blockstruktur hat, weshalb sie als Ganzes verwendet wird.

Besser ist diejenige Variante, bei der der Levenberg-Marquardt-Algorithmus mit variablen Blockgrößen arbeiten kann: Bei Anwendung des Verfahrens auf echte Bildfolgen wird der Anteil der in einem bestimmten Bild sichtbaren 3-D-Punkte, gemessen an der Gesamtzahl der Punkte überhaupt, oft relativ gering sein, d. h. die meisten 3-D-Punkte der Szene sind in einem Bild gar nicht sichtbar. Dadurch ist der größte der Teil der Gleichungen in Formel (4.51) überflüssig, es wird also unnötigerweise der Aufwand bei der Bildung der Pseudoinversen mit der Singulärwertzerlegung erhöht.

Daher wurde ein Gauß-Newton-Verfahren mit Levenberg-Marquardt-Erweiterung implementiert, bei dem es möglich ist, die Anzahl und Größe der Blöcke, aus denen die Jacobi-Matrix besteht, beliebig festzulegen. Natürlich kann dieses auch für eine „normale“ nichtlineare Optimierung verwendet werden, bei der die Jacobi-Matrix keine Blockstruktur besitzt. Sie besteht dann aus einem einzigen Block.

Kapitel 5

Erweiterungen des Verfahrens

In diesem Kapitel werden einige Erweiterungen des vorher beschriebenen Verfahrens zum Bündelausgleich vorgestellt. Im ersten Abschnitt wird erläutert, wie die Rückprojektionsfehler in verschiedenen Bildern unterschiedlich stark gewichtet werden können. Der zweite Abschnitt befasst sich mit der Korrektur von Linsenverzerrungen.

5.1 Gewichtung des Fehlers

Bei der Rekonstruktion von *Struktur aus Bewegung* benötigt man vor Anwendung des eigentlichen Verfahrens die Korrespondenzen von 2-D-Punkten in verschiedenen Aufnahmen. Das hierfür am Lehrstuhl verwendete Verfahren basiert auf der Berechnung des optischen Flusses zwischen jeweils zwei aufeinanderfolgenden Bildern [ST94]. Dadurch bedingt, akkumulieren sich kleine Fehler bei der Lokalisation der Punkte, so dass die Position eines Punktes im letzten Bild der Folge wesentlich unsicherer ist als bei solchen am Anfang. Diese Unsicherheit soll nun beim Bündelausgleich berücksichtigt werden, indem die Rückprojektionsfehler für jedes Bild unterschiedliche Gewichte erhalten. Eine Beschreibung der hier vorgestellten Vorgehensweise findet man z. B. in [Sla80, Har94, WAH93, SK97, McL99]. An Stelle von Formel (4.3) wird zur Minimierung folgende Funktion verwendet:

$$\min_{\boldsymbol{P}_i,\boldsymbol{w}_j} \sum_{j=1}^{n} \sum_{i=1}^{m} \left(c_{iju}^2 \left(u_{ij} - \frac{\boldsymbol{p}_{i1}^T \boldsymbol{w}_j}{\boldsymbol{p}_{i3}^T \boldsymbol{w}_j} \right)^2 + c_{ijv}^2 \left(v_{ij} - \frac{\boldsymbol{p}_{i2}^T \boldsymbol{w}_j}{\boldsymbol{p}_{i3}^T \boldsymbol{w}_j} \right)^2 \right) \quad . \tag{5.1}$$

Dabei sind $c_{iju}, c_{ijv} \in \mathbb{R}$ die Gewichtungsfaktoren der u- bzw. v-Koordinate des j-ten Punktes im i-ten Bild, welche hier zur Vereinfachung der nachfolgenden Formulierungen quadratisch eingehen.

Damit ändern sich die bei der nichtlinearen Optimierung mit dem Gauß-Newton-Verfahren zu verwendenden Gleichungen[1]. Zunächst werden alle Gewichtsfaktoren – diesmal nicht quadriert – in einer Matrix $\boldsymbol{W}$ zusammengefasst:

$$\boldsymbol{W} = \operatorname{diag}(c_{1,1,u}, c_{1,1,v}, \ldots, c_{iju}, c_{ijv}, \ldots, c_{mnu}, c_{mnv}) \quad . \tag{5.2}$$

Der Residuenvektor $\boldsymbol{\epsilon}$ wird nun mit dieser Matrix multipliziert, wobei der entstehende Vektor mit $\boldsymbol{\xi}$ bezeichnet werden soll. Beim Bündelausgleich wird dann der Ausdruck

$$\boldsymbol{\xi}^T \boldsymbol{\xi} = \boldsymbol{\epsilon}^T \boldsymbol{W}^T \boldsymbol{W} \boldsymbol{\epsilon} \tag{5.3}$$

[1] siehe Abschnitt 4.3.2 für das Gauß-Newton-Verfahren bzw. 4.3.3 für die Levenberg-Marquardt-Erweiterung

minimiert, was Formel (5.1) entspricht. Für die Linearisierung der Residuenfunktion um den Startwert $\boldsymbol{x}_k$ erhält man nun an Stelle von (4.35)

$$\boldsymbol{\xi}(\boldsymbol{x}) = \boldsymbol{W}\,\boldsymbol{\epsilon}(\boldsymbol{x}) = \boldsymbol{W}\,\boldsymbol{\epsilon}(\boldsymbol{x}_k + \Delta\boldsymbol{x}) \approx \boldsymbol{W}\,\boldsymbol{\epsilon}(\boldsymbol{x}_k) + \boldsymbol{W}\,\frac{\partial \boldsymbol{\epsilon}}{\partial \boldsymbol{x}}(\boldsymbol{x}_k)\Delta\boldsymbol{x} \quad , \tag{5.4}$$

wobei, mit der Abkürzung $\boldsymbol{J} = \frac{\partial \boldsymbol{\epsilon}}{\partial \boldsymbol{x}}(\boldsymbol{x}_k)$ für die Jacobi-Matrix, wieder gelten muss

$$\boldsymbol{W}\,\boldsymbol{\epsilon}(\boldsymbol{x}_k) + \boldsymbol{W}\,\boldsymbol{J}\Delta\boldsymbol{x} = \boldsymbol{0} \quad . \tag{5.5}$$

Auflösen nach $\Delta\boldsymbol{x}$ ergibt dann entsprechend Gleichung (4.38)

$$\Delta\boldsymbol{x} = -\left(\boldsymbol{J}^T\,\boldsymbol{W}^T\,\boldsymbol{W}\,\boldsymbol{J}\right)^{-1}\boldsymbol{J}^T\,\boldsymbol{W}^T\,\boldsymbol{W}\,\boldsymbol{\epsilon}(\boldsymbol{x}_k) \tag{5.6}$$

bzw. bei Verwendung der Levenberg-Marquardt-Erweiterung entsprechend Gleichung (4.41)

$$\Delta\boldsymbol{x} = -\left(\lambda\boldsymbol{I} + \boldsymbol{J}^T\,\boldsymbol{W}^T\,\boldsymbol{W}\,\boldsymbol{J}\right)^{-1}\boldsymbol{J}^T\,\boldsymbol{W}^T\,\boldsymbol{W}\,\boldsymbol{\epsilon}(\boldsymbol{x}_k) \quad . \tag{5.7}$$

Sofern bekannt, sollte als $\boldsymbol{W}^T\,\boldsymbol{W}$ die inverse Kovarianzmatrix $\boldsymbol{\Sigma}^{-1}$ der Messungen verwendet werden [SK97, Har94].[2] Damit werden Punkte, die relativ sicher lokalisiert sind, mit einem großen Faktor gewichtet, während Punkte mit hoher Varianz weniger stark gewichtet werden. Man verwendet dann

$$\Delta\boldsymbol{x} = -\left(\lambda\boldsymbol{I} + \boldsymbol{J}^T\,\boldsymbol{\Sigma}^{-1}\boldsymbol{J}\right)^{-1}\boldsymbol{J}^T\,\boldsymbol{\Sigma}^{-1}\,\boldsymbol{\epsilon}(\boldsymbol{x}_k) \quad . \tag{5.8}$$

5.2 Korrektur von Linsenverzerrungen

5.2.1 Einführung

Nun soll der Bündelausgleich in so fern erweitert werden, dass Verzerrungen der Kameraoptik während der Optimierung mit geschätzt und so ausgeglichen werden. In der Literatur wird dies im Zusammenhang mit dem Bündelausgleich üblicherweise nicht gemacht. Die einzigen Stellen, wo dies erwähnt wird, sind [KDRS98] und [Pol99], wobei dort allerdings keine Aussagen darüber gemacht werden, wie gut die damit erzielbaren Ergebnisse sind. Darüber sollen die in Kapitel 6.2.4 durchgeführten Auswertungen Klarheit bringen. Grundlagen über die Korrektur von radialen Verzerrungen findet man z. B. in [TV98, Nie90].

Dass die Korrektur der Linsenverzerrungen so wenig Beachtung findet, hat insbesondere zwei Gründe: Zum einen sind die Verzerrungen oft gering, zum anderen wird oft keine so große Genauigkeit benötigt und deshalb kann man die Verzerrungen ohne Probleme vernachlässigen. Ist dies nicht der Fall, so erfolgt die Berechnung der zur Korrektur nötigen Parameter meist *vor* der Bildaufnahme in einem Kalibrierungsschritt. Da in der vorliegenden Anwendung aber gerade eine unkalibrierte Kamera verwendet werden soll, ist dies hier nicht möglich.

Untersuchungen über den Einfluss von Linsenverzerrungen auf die 3-D-Rekonstruktion im Allgemeinen gibt es natürlich: Zhang beispielsweise betrachtet die Epipolargeometrie mit Linsenverzerrungen [Zha96] und schätzt die Fundamentalmatrix. Er fasst sein Ergebnisse wie folgt zusammen:

[2] dies ist der vorher unberücksichtigte Fall von anisotropem Rauschen auf den Bildpunkten.

- Wann immer es möglich ist, die Korrektur in einem vorhergehenden Kalibrierungsschritt durchzuführen, sollte man dies tun.
- Eine Korrektur bringt dann bessere Ergebnisse, wenn die Verzerrungen relativ groß sind und die verwendeten Punktmerkmale mit großer Genauigkeit (Fehler $< 0,3$ Pixel) extrahiert werden können.
- Problematisch ist die Korrektur auch deshalb, weil die Verzerrungen am deutlichsten an den Rändern des Bildes auftreten: Verwendet man also die dort liegenden Punkte zur Schätzung, so könnte man stabilere Werte erwarten als bei Punkten in der Mitte des Bildes. Allerdings liegen an den Rändern normalerweise praktisch keine zur Rekonstruktion verwendbaren Punkte, da diese oft von einem Bild zum nächsten nicht mehr sichtbar sind und somit keine Korrespondenzen ermittelt werden können.

Allerdings ist nicht gesichert, dass dies auch auf den hier vorliegenden Anwendungsfall in dieser Form zutrifft, da Zhang eine Fundamentalmatrix zwischen zwei Bildern mit Hilfe einer erweiterten Epipolarbedingung berechnet, und seine Ergebnisse darauf bezogen sind. Zudem wird beim Bündelausgleich eine initiale Schätzung sämtlicher Parameter verwendet, die ohne die Berücksichtigung von Verzerrungen entstanden ist. Somit kommen diese erst im letzten Schritt der Rekonstruktion zum Tragen.

5.2.2 Modellierung von Linsenverzerrungen

In diesem Abschnitt wird erläutert, wie Linsenverzerrungen mit Hilfe von Polynomfunktionen modelliert werden können. Die Integration des Verfahrens in den Bündelausgleich wird im nachfolgenden Abschnitt 5.2.3 betrachtet.

Zunächst soll dargestellt werden, an welcher Stelle in der Abbildung eines 3-D-Weltpunktes $\boldsymbol{w}$ auf einen tatsächlich beobachteten Bildpunkt $\boldsymbol{u}$ die Linsenverzerrungen auftreten.[3] Die Reihenfolge, in der die Abbildung vonstatten geht, ist wie folgt [Zha96]:

1. Transformation des Punktes $\boldsymbol{w}$ vom 3-D-Welt- ins 3-D-Kamerakoordinatensystem (Rotation und Translation).
2. Perspektivische Projektion vom 3-D-Kamerakoordinatensystem auf ideale 2-D-Bildkoordinaten. Es entsteht der unverzerrte Punkt $\boldsymbol{x} = (x, y)$.
3. Berechnung der Linsenverzerrung. Man erhält den verzerrten Punkt $\boldsymbol{x}_d = (x_d, y_d)$. In der folgenden Gleichung wird der umgekehrte Weg angegeben, nämlich wie man von einem verzerrten Punkt zu einem unverzerrten kommt, da das Ziel ja die Korrektur der Verzerrung ist:
$$x = x_d + \delta_x, \qquad y = y_d + \delta_y \quad . \tag{5.9}$$
Dabei sind δ_x bzw. δ_y Korrekturterme für die Entzerrung, welche im Vektor $\boldsymbol{\delta} = (\delta_x, \delta_y)^T$ zusammengefasst werden.

[3] es soll hier exemplarisch ein 3-D-Punkt in einer Aufnahme betrachtet werden, die Indizes werden daher wegen der übersichtlicheren Darstellung weggelassen.

4. Transformation auf Pixelkoordinaten. Berücksichtigt wird hier nun das Seitenverhältnis der Bildpunkte sowie die Verschiebung um den Hauptpunkt (u_0, v_0). Wie bereits in den vorherigen Kapiteln wird auch hier angenommen, dass man ein rechtwinkliges Bildkoordinatensystem vorliegen hat. Die Pixelkoordinaten $\boldsymbol{u} = (u, v)$ ergeben sich mit

$$u = \frac{1}{s_x} x_d + u_0, \qquad v = \frac{1}{s_y} y_d + v_0 \quad . \tag{5.10}$$

Die in $\boldsymbol{\delta}$ enthaltenen Korrekturterme für die Verzerrung setzen sich aus zwei Komponenten zusammen: Dem radialen Anteil $\boldsymbol{\delta}_r$ und dem tangentialen $\boldsymbol{\delta}_t$ [Zha96, Sla80], die sich additiv überlagern:

$$\boldsymbol{\delta} = \boldsymbol{\delta}_r + \boldsymbol{\delta}_t = \begin{pmatrix} \delta_{rx} + \delta_{tx} \\ \delta_{ry} + \delta_{ty} \end{pmatrix} \quad . \tag{5.11}$$

Radiale Verzerrungen sind symmetrisch um den Punkt (u_0, v_0). Sie entstehen in erster Linie durch nicht exakt geschliffene Linsen. Tangentiale Verzerrungen entstehen beim Zusammenbau des Objektivs dadurch, dass die Zentren der einzelnen Linsen etwas von der Geraden, auf der sie liegen sollten, abweichen. So ergeben sich Verzerrungen, die symmetrisch entlang einer Geraden durch (u_0, v_0) sind.

Radiale Verzerrungen der Linse können mit Hilfe von Polynomen mit geradzahligen Exponenten (wegen der Symmetrie) modelliert werden. Die Elemente des Terms $\boldsymbol{\delta}_r$ ergeben sich dann wie folgt [Zha96, Sla80, TV98]:

$$\begin{aligned} \delta_{rx} &= x_d(k_1 r^2 + k_2 r^4 + k_3 r^6 + \dots) \quad , \\ \delta_{ry} &= y_d(k_1 r^2 + k_2 r^4 + k_3 r^6 + \dots) \quad , \end{aligned} \tag{5.12}$$

wobei

$$r^2 = x_d^2 + y_d^2 \quad . \tag{5.13}$$

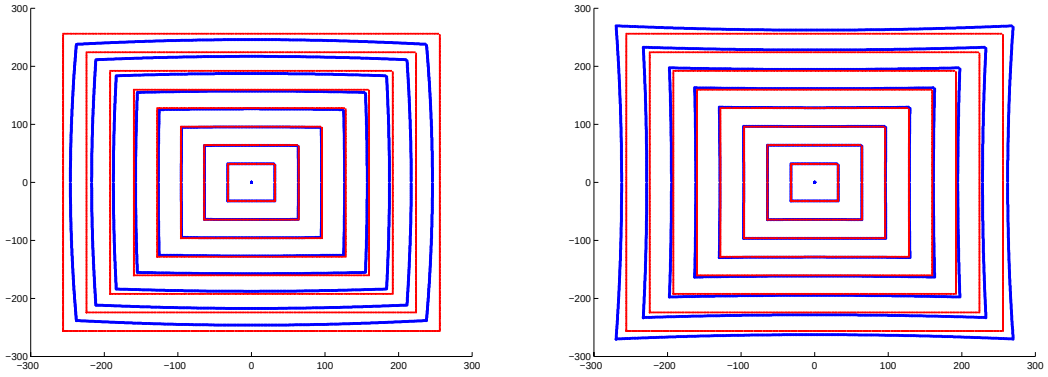
Die Terme für den tangentialen Anteil werden modelliert durch [Zha96]:

$$\begin{aligned} \delta_{tx} &= \left(p_1 \left(r^2 + 2x_d^2\right) + 2p_2 x_d y_d\right)\left(1 + p_3 r^2 + \dots\right) \quad , \\ \delta_{ty} &= \left(2p_1 x_d y_d + p_2 \left(r^2 + 2y_d^2\right)\right)\left(1 + p_3 r^2 + \dots\right) \quad . \end{aligned} \tag{5.14}$$

Zu bestimmen sind dann zusätzlich zu den anderen Kameraparametern die neuen Parameter k_i und p_j. Um das obige Modell verwenden zu können, muss natürlich ab einem bestimmten Exponenten von r der Term abgebrochen werden; dies ist normalerweise bereits bei r^4 oder sogar schon bei r^2 der Fall.
Zu den obigen Gleichungen ist anzumerken:

- Die Verzerrungen werden im Allgemeinen vom radialen Anteil bestimmt, der tangentiale ist demgegenüber vernachlässigbar [Zha96]. Insbesondere genügt außerdem oft die Verwendung nur eines Terms im radialen Anteil, also nur von k_1. Nach [TV98] ist das Modell selbst dann noch gut, wenn man an den Rändern des Bildes Verzerrungen von bis zu 5 Pixeln hat.

- Es gibt experimentelle Ergebnisse von Tsai [Tsa87, Zha96], die besagen, dass eine zu genaue Modellierung der Linsenverzerrungen die Parameterschätzung nicht nur nicht verbessert, sondern im Gegenteil zu einer größeren numerischen Instabilität führen kann.



(a) $\kappa = 0.5$ mm^{-2} (b) $\kappa = -0.5$ mm^{-2}

Bild 5.1: Beispielbilder zum Einfluss des Parameters κ bei Linsenverzerrungen. Die dünnen Linien sind die unverzerrten, die dickeren Linien die verzerrten Punkte.

Aus diesen Gründen werden zur Modellierung der Linsenverzerrungen hier folgende Gleichungen verwendet, welche sich aus der Kombination von (5.9) und (5.12) ergeben:

$$\begin{aligned} x &= x_d(1 + \kappa r^2) \quad , \\ y &= y_d(1 + \kappa r^2) \quad , \end{aligned} \tag{5.15}$$

wobei k_1 durch κ ersetzt wurde. Mit κ bleibt nur ein zusätzlich zu schätzender Parameter übrig. Wie sich die so berechneten Verzerrungen äußern, sieht man in Bild 5.1, wo Beispielbilder mit $\kappa = \pm 0.5$ mm^{-2} dargestellt sind.

Da im Bild nicht der Punkt $\boldsymbol{x}_d$ sondern $\boldsymbol{u}$ gemessen wird, bei dem nach (5.10) die intrinsischen Kameraparameter mit berücksichtigt werden, erhält man nach dem Auflösen von (5.10) nach x_d bzw. y_d für Formel (5.13) folgenden Ausdruck:

$$r^2 = s_x^2(u - u_0)^2 + s_y^2(v - v_0)^2 \quad . \tag{5.16}$$

Dadurch ergibt sich schließlich für (5.15) die Abbildung von verzerrten Pixelkoordinaten $\boldsymbol{u}$ auf unverzerrte Punkte $\boldsymbol{x}$:

$$\begin{aligned} x &= s_x(u - u_0)\left(1 + \kappa\left(s_x^2\left(u - u_0\right)^2 + s_y^2\left(v - v_0\right)^2\right)\right) \quad , \\ y &= s_y(v - v_0)\left(1 + \kappa\left(s_x^2\left(u - u_0\right)^2 + s_y^2\left(v - v_0\right)^2\right)\right) \quad . \end{aligned} \tag{5.17}$$

Weiterhin gelten für die Komponenten des Punktes $\boldsymbol{x}$ auch folgende Projektionsgleichungen:

$$x = f\,\frac{\tilde{\boldsymbol{p}}_1^T\boldsymbol{w}}{\tilde{\boldsymbol{p}}_3^T\boldsymbol{w}}, \qquad y = f\,\frac{\tilde{\boldsymbol{p}}_2^T\boldsymbol{w}}{\tilde{\boldsymbol{p}}_3^T\boldsymbol{w}} \quad , \tag{5.18}$$

wobei $\tilde{\boldsymbol{P}}$ eine 3×4 Matrix ist, die nur die extrinsischen Kameraparameter enthält:

$$\tilde{\boldsymbol{P}} = \begin{pmatrix} \tilde{\boldsymbol{p}}_1^T \\ \tilde{\boldsymbol{p}}_2^T \\ \tilde{\boldsymbol{p}}_3^T \end{pmatrix} = \boldsymbol{R}^T(\boldsymbol{I}_3| - \boldsymbol{t}) \quad . \tag{5.19}$$

Sowohl (5.17) als auch (5.18) werden beim Bündelausgleich verwendet. Näheres dazu befindet sich im nachfolgenden Abschnitt.

5.2.3 Berücksichtigung beim Bündelausgleich

Nun wird erläutert, wie die im vorherigen Abschnitt beschriebenen Gleichungen in den Bündelausgleich integriert werden können. Es wurde dort der neue Parameter κ eingeführt, welcher zusätzlich bei der Optimierung geschätzt werden muss. Wie bei allen anderen zu optimierenden Werten, wird auch für κ ein Startwert benötigt; im Gegensatz zu den übrigen Parametern jedoch erhält man κ nicht aus dem vorhergehenden linearen Rekonstruktionsverfahren, da dort die Linsenverzerrungen nicht berücksichtigt werden. Daher wird der Startwert auf $\kappa = 0$ festgesetzt [TV98], man geht also von einem unverzerrten Bild aus.

Weiterhin muss man sich überlegen, ob der neue Parameter in jeder Aufnahme verschieden sein kann oder ob dieser für alle Aufnahmen gleich ist. Hier werden für jede Aufnahme verschiedene κ_i gewählt, da von der Verwendung einer Kamera mit Zoom-Objektiv ausgegangen wird, bei der sich die optischen Eigenschaften von Bild zu Bild verändern können. Folglich erhöht sich die Anzahl der zu schätzenden Parameter von 10 auf 11 *pro Aufnahme*.

Eine weitere Veränderung muss bei der Berechnung des Rückprojektionsfehlers vorgenommen werden. Bisher wurde dieser für einen Bildpunkt wie folgt berechnet:

$$\left(\boldsymbol{u}_{ij} - \tilde{\boldsymbol{u}}_{ij}\right)^T \left(\boldsymbol{u}_{ij} - \tilde{\boldsymbol{u}}_{ij}\right) \quad . \tag{5.20}$$

Man musste also nur in jedem Iterationsschritt aus den neu geschätzten Parametern eine Projektionsmatrix $\boldsymbol{P}_i$ für jedes Bild zusammenbauen und damit die (ebenfalls veränderten) Weltpunkte ins jeweilige Bild projizieren. Anschließend wurde die Differenz der Rückprojektionen $\tilde{\boldsymbol{u}}_{ij}$ zu den tatsächlichen Positionen der Bildpunkte $\boldsymbol{u}_{ij}$ gemessen.

Nun liegt der Fall etwas anders: Der Parameter κ_i ist nicht in der Projektionsmatrix $\boldsymbol{P}_i$ enthalten, sondern er muss entsprechend (5.17) getrennt davon behandelt werden. Außerdem kommt hinzu, dass die Linsenverzerrung nicht erst nach der Projektion berücksichtigt wird, sondern wie in Abschnitt 5.2.2 in den Punkten 1–4 beschrieben, schon vor der Umrechnung ins Bildkoordinatensystem.

Daher wird die Berechnung des Fehlers nun von zwei Seiten her angegangen: Die vorhandenen (verzerrten) Bildpunkte $\boldsymbol{u}_{ij}$ werden durch Verwendung der Gleichungen (5.17) in unverzerrte Punkte $\boldsymbol{x}_{ij}$ umgerechnet, wobei die neu geschätzten Parameter verwendet werden. Die Weltpunkte werden wie in Gleichung (5.18) dargestellt mit Hilfe der neuen Projektionsmatrizen $\tilde{\boldsymbol{P}}_i$, die nur die extrinsischen Parameter enthalten, ins Bild projiziert. Man erhält die Rückprojektionen $\tilde{\boldsymbol{x}}_{ij}$. Der Fehler für einen Punkt ergibt sich dann aus

$$\left(\boldsymbol{x}_{ij} - \tilde{\boldsymbol{x}}_{ij}\right)^T \left(\boldsymbol{x}_{ij} - \tilde{\boldsymbol{x}}_{ij}\right) \quad . \tag{5.21}$$

5.2.4 Simulation von Linsenverzerrungen

Für die in Kapitel 6 vorgestellten Experimente ist es notwendig, Linsenverzerrungen zu simulieren und zwar so, dass der Parameter κ, der zur Korrektur nötig ist, bekannt ist. Wie dies geschieht [Zha96], wird im Folgenden erläutert. Die Gleichungen in Abschnitt 5.2.2 erlauben die Berechnung von unverzerrten Punkten aus verzerrten, da dies die Richtung ist, die man üblicherweise benötigt. Nun sollen aus diesen Gleichungen Formeln entstehen, die es erlauben, den umgekehrten Weg zu gehen, der für die Simulation benötigt wird: Ausgehend von idealen 2-D-Punkten $\boldsymbol{x}$ sollen verzerrte Punkte $\boldsymbol{u}$ berechnet werden.

Hierzu wird zunächst folgende Notation für Bildpunkte $\breve{\boldsymbol{u}}$ eingeführt, die aus den idealen Bildpunkten $\boldsymbol{x}$ entstehen, bei denen aber die Größe der Pixel berücksichtigt wird:

$$\breve{u} = \frac{x}{s_x}, \qquad \breve{v} = \frac{y}{s_y} \quad . \tag{5.22}$$

Mit dieser Notation wird Gleichung (5.17) zu

$$\breve{u} = (u - u_0)\left(1 + \kappa\left(s_x^2\,(u - u_0)^2 + s_y^2\,(v - v_0)^2\right)\right) \quad , \tag{5.23}$$

$$\breve{v} = (v - v_0)\left(1 + \kappa\left(s_x^2\,(u - u_0)^2 + s_y^2\,(v - v_0)^2\right)\right) \quad . \tag{5.24}$$

Nun wird Gleichung (5.23) durch Gleichung (5.24) dividiert, und man erhält

$$\frac{\breve{u}}{\breve{v}} = \frac{u - u_0}{v - v_0} \quad . \tag{5.25}$$

Auflösen von (5.25) nach dem Term $(u - u_0)$ und Einsetzen in (5.24) ergibt

$$\breve{v} = (v - v_0)\left(1 + \kappa\left(s_x^2\,(v - v_0)^2\,\frac{\breve{u}^2}{\breve{v}^2} + s_y^2\,(v - v_0)^2\right)\right) \quad . \tag{5.26}$$

Zusammengefasst erhält man ein Polynom 3. Grades im Term $(v - v_0)$, von dem die Nullstellen bestimmt werden müssen, um schließlich die verzerrten Bildpunkte $\boldsymbol{u} = (u, v)$ zu erhalten:

$$(v - v_0)^3 + p(v - v_0) + q = 0 \tag{5.27}$$

mit

$$p = \frac{\breve{v}^2}{\kappa\left(s_x^2\breve{u}^2 + s_y^2\breve{v}^2\right)}, \qquad q = -\breve{v}p = -\frac{\breve{v}^3}{\kappa\left(s_x^2\breve{u}^2 + s_y^2\breve{v}^2\right)} \quad . \tag{5.28}$$

Für das Polynom (5.27) gibt es bis zu drei reelle Lösungen, welche durch die Diskriminante der Lösungsformel für kubische Gleichungen charakterisiert werden können. Die Diskriminante lautet:

$$D = \left(\frac{p}{3}\right)^3 + \left(\frac{q}{2}\right)^2 \quad . \tag{5.29}$$

Zu unterscheiden sind dann die Fälle [Zha96]:

- $D > 0$: Es gibt nur eine reelle Lösung.
- $D = 0$: Tritt nur auf, wenn $\breve{v} = 0$ ist; es ergibt sich eine dreifache reelle Nullstelle, die Lösung lautet dann $v = v_0$.
- $D < 0$: Es gibt drei reelle Lösungen, wovon die mittlere die gesuchte ist.
- Sonderfall $\kappa = 0$: Es sind keine Linsenverzerrungen vorhanden; die Lösung lautet dann $v = \breve{v} + v_0$.

Hat man die Lösung für v, so ergibt sich das zugehörige u aus (5.25)

$$u = \frac{\breve{u}}{\breve{v}}(v - v_0) + u_0 \quad . \tag{5.30}$$

Kapitel 6

Untersuchungen zur Leistungsfähigkeit des Bündelausgleichs

In diesem Kapitel werden die Ergebnisse der Untersuchungen zur Leistungsfähigkeit des Bündelausgleichs vorgestellt. Es wurden dazu Experimente mit synthetischen und realen Bildfolgen durchgeführt, welche in den Abschnitten 6.2 bzw. 6.3 beschrieben werden. Für die Durchführung dieser Untersuchungen wurde der in Kapitel 4 beschriebene eingebettete Bündelausgleich mit den Erweiterungen aus Kapitel 5 in C++ implementiert.

Hierzu wurde eine Version des Gauß-Newton-Algorithmus mit Levenberg-Marquardt-Erweiterung entwickelt, welche mit variablen Blockgrößen der Jacobi-Matrix arbeiten kann – also speziell auch mit nur einem einzigen Block, nämlich der gesamten Matrix. Somit ist dieser universell für die nichtlineare Optimierung einsetzbar. Ebenfalls möglich ist die Gewichtung des Fehlers bei der Optimierung, wie sie in Abschnitt 5.1 erläutert wurde. Zur Fehlergewichtung wurden hier keine Experimente durchgeführt, da noch keine Daten darüber vorliegen, wie stark sich der Fehler bei der Verfolgen von Punkten von einer Aufnahme zur nächsten ändert.

Für die Berechnung des Rückprojektionsfehlers wurde die in Kapitel 4 beschriebene Parametrisierung verwendet, wobei die Rotation wahlweise mit Quaternionen oder mit der Achse/Winkel-Darstellung parametrisiert werden kann. Auch die Korrektur von Linsenverzerrungen, wie sie Abschnitt 5.2 dargestellt wurde, ist möglich. Zudem kann der Algorithmus mit Verdeckungen umgehen, wobei alle in einem Bild sichtbaren 3-D-Punkte auch zur Optimierung herangezogen werden. Dazu war die Implementierung einer Klasse nötig, die einen effizienten Zugriff auf diese Punkte ermöglicht.

Vor der Betrachtung der Ergebnisse der Experimente soll in Abschnitt 6.1 kurz darauf eingegangen werden, wie die für die Optimierung notwendigen Startwerte für die Parameter gewonnen werden und welche Probleme dabei auftreten.

6.1 Berechnung der Startwerte für den Bündelausgleich

Zu Beginn des Optimierungsverfahrens benötigt man möglichst gute Startwerte für die zu optimierenden Parameter, also die intrinsischen und extrinsischen Kameraparameter sowie die Koordinaten der Weltpunkte. Da die Startwerte von einem vor dem Bündelausgleich laufenden

linearen Verfahren zur projektiven Rekonstruktion mit anschließender linearer[1] Selbstkalibrierung [HN99] erzeugt werden, erhält man allerdings nicht direkt die Kameraparameter, sondern nur die 3×4 Projektionsmatrizen $\boldsymbol{P}_i$ mit

$$\boldsymbol{P}_i = \boldsymbol{K}_i \boldsymbol{R}_i^T (\boldsymbol{I}_3 | - \boldsymbol{t}_i) \quad , \tag{6.1}$$

für jede Aufnahme i eine. Wie bereits in Abschnitt 2.3 erwähnt, liefert die Selbstkalibrierung Kalibrierungsmatrizen der Form

$$\boldsymbol{K}_i = \begin{pmatrix} f_{xi} & s_i & u_{0i} \\ 0 & f_{yi} & v_{0i} \\ 0 & 0 & 1 \end{pmatrix} \tag{6.2}$$

mit $s_i \neq 0$ (aber klein[2]) und $\boldsymbol{R}_i$ exakt orthogonal. Man hat folglich eigentlich eine projektive Rekonstruktion, die nur näherungsweise euklidisch ist. Für eine euklidische Rekonstruktion bis auf eine unbekannte Ähnlichkeitstransformation ist es aber nötig, dass die s_i tatsächlich null und die $\boldsymbol{R}_i$ tatsächlich Rotationsmatrizen sind, also orthogonal und mit Determinante gleich eins. D. h. die Parameter s_i dürfen *nicht* einfach als zusätzliche Parameter in den Bündelausgleich eingehen, da man dann nur eine projektive[3] Rekonstruktion erhält.

Die Frage, die sich nun stellt, ist: Wie erhält man aus dieser näherungsweisen euklidischen Rekonstruktion eine exakt euklidische, da man eine solche für die Durchführung des Bündelausgleichs benötigt. Es gibt hierfür zwei mögliche Ansätze:

- Zerlege die $\boldsymbol{P}_i$ wie in (6.1) angegeben. Dabei entstehen Werte von $s_i \neq 0$, aber exakt orthogonale Rotationsmatrizen. Setze nun in jeder Projektionsmatrix $s_i = 0$.

- Berechne aus einer Projektionsmatrix alle Parameter unter der Annahme, dass $s_i = 0$ ist. Hierfür wurde eine leicht abgewandelte Version des in [TV98] beschriebenen Verfahrens zur Kamerakalibrierung verwendet (siehe Anhang C). Allerdings entstehen damit für die $\boldsymbol{R}_i$ *keine* exakt orthogonalen Matrizen; es ist also eine anschließende Orthogonalisierung notwendig, wobei hierfür die Singulärwertzerlegung verwendet wird (siehe Anhang B). Diese liefert die im Sinne der Frobeniusnorm nächstliegende echt orthogonale Matrix.

Egal welchen der beiden Ansätze man auch wählt, die ursprünglichen Projektionsmatrizen werden dadurch – wenn auch nur leicht – verändert, was zur Folge hat, dass der Rückprojektionsfehler ansteigt. Dieser Anstieg kann durchaus gravierend sein, abhängig davon, wie gut die Selbstkalibrierung funktioniert hat. Wie stark sich der Fehler tatsächlich ändert, wird man in den Auswertungen in den folgenden Abschnitten noch sehen.

Es wurden nun einige Simulationen durchgeführt, um herauszufinden, ob evtl. eines der beiden Verfahren immer besser ist, d. h. ob es immer einen geringeren Anstieg des Fehlers zur Folge hat. Hierzu wurden Projektionsmatrizen erzeugt, bei denen $s_i = 0$ gilt. Anschließend wurde der Parameter s_i variiert und die Weltpunkte mit der neuen Projektionsmatrix ins Bild projiziert. Nach dem Nullsetzen von s_i bzw. der Berechnung der Parameter unter Annahme $s_i =$

[1] Diese wird zusätzlich in einem nichtlinearen Optimierungsschritt unter Verwendung des in [PKV98] vorgestellten Kriteriums verbessert. An den folgenden Ausführungen ändert dies jedoch nichts.

[2] typische Werte liegen im Bereich von $\pm 10^{-2}$, extreme Werte bei ± 0.3. Aber bereits mit sehr kleinen s_i gibt es Probleme.

[3] siehe Abschnitt 2.3

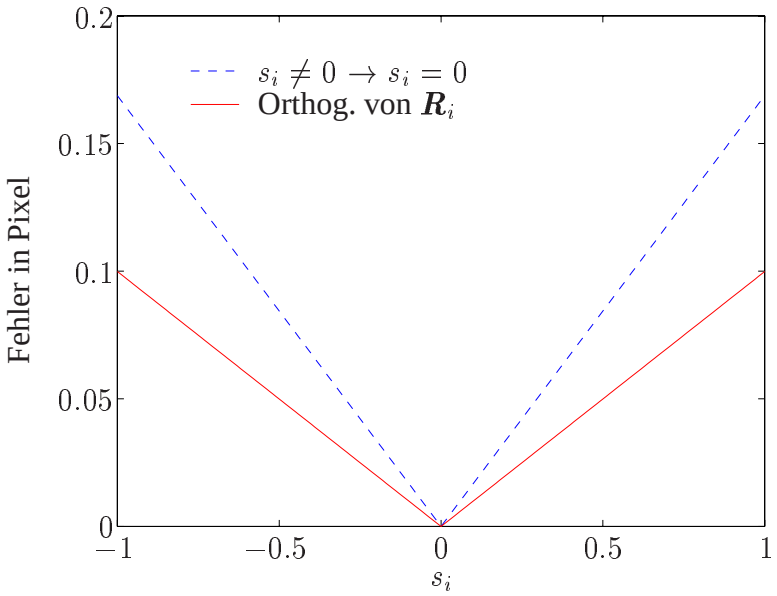


Bild 6.1: Veränderung des Fehlers bei Orthogonalisierung der Rotationsmatrix $\boldsymbol{R}_i$ bzw. beim Nullsetzen von s_i. Angetragen ist an der horizontalen Achse der s_i-Wert in der Projektionsmatrix, an der vertikalen der Rückprojektionsfehler pro Punkt. Die Zunahme des Fehlers ist bei der Orthogonalisierung immer kleiner als beim einfachen Nullsetzen von s_i.

0 und anschließender Orthogonalisierung der Rotationsmatrix wurde der Rückprojektionsfehler pro Bildpunkt berechnet.

In Bild 6.1 ist exemplarisch die Zunahme des Fehlers pro Bildpunkt für beide Verfahren graphisch dargestellt. In allen durchgeführten Versuchen verzeichnete das zweite Verfahren mit anschließender Orthogonalisierung einen geringeren Anstieg des Fehlers, weshalb es in allen folgenden Versuchen zur Berechnung der Kameraparameter aus den Projektionsmatrizen zur Bestimmung der Startwerte verwendet wurde.

Die Zunahme des Rückprojektionsfehlers ist im Übrigen für den Bündelausgleich kein Problem: In allen Fällen sank der Fehler nach der Optimierung nach genügend Iterationsschritten wieder unter den Fehler, welchen das lineare Verfahren liefert. Zusätzlich sind nach der Optimierung in allen Projektionsmatrizen die s_i gleich null und die $\boldsymbol{R}_i$ Rotationsmatrizen.

6.2 Experimente mit synthetischen Daten

In diesem Abschnitt werden die Ergebnisse der Experimente mit synthetischen Daten vorgestellt. Hierzu wurden insgesamt acht verschiedene Szenen verwendet, deren Gemeinsamkeiten und Unterschiede nun kurz erläutert werden sollen. Gemeinsam ist allen Szenen:

- Es wurden 100 3-D-Punkte in 20 Bildern zur Simulation verwendet, wobei die Punkte der Szene in einem Würfel der Kantenlänge 200 mm zufällig (gleich-)verteilt sind. Jeder Punkt ist in jeder Aufnahme sichtbar, da die derzeitige C++ Implementierung des vorher laufenden linearen Verfahrens, das die Startwerte der Parameter liefert, noch nicht mit Verdeckungen arbeiten kann.

- Sowohl der Abstand der Kamera vom Mittelpunkt des Würfels als auch die exakte Blickrichtung wurden zusätzlich zur vorgegebenen Bewegungsrichtung (siehe einzelne Szenen) zufällig variiert. Dies entspricht dem Verhalten bei Verwendung einer handgeführten Kamera.

- Die normierten Brennweiten[4] der Kameras wurden zufällig aus dem Intervall 800–1200 Pixel gewählt, wobei bei jeder Kamera $f_x = f_y$ gilt, d. h. es wurden quadratische Pixel angenommen. Diese Annahme ist für den Bündelausgleich irrelevant, nicht hingegen für die vorhergehende Selbstkalibrierung. [5] Diese kann bei zu großem Unterschied der beiden Brennweiten fehlschlagen. Das Resultat ist dann eine projektive Rekonstruktion, keine euklidische. Dadurch sind die Startwerte für den Bündelausgleich sehr schlecht, was für die Untersuchungen hier möglichst vermieden werden sollte, da die Leistungsfähigkeit bei einer guten vorhergehenden Rekonstruktion beurteilt werden soll. Bei Verwendung von echten Bildfolgen, wo meist keine quadratischen Pixel vorliegen, ist dies kein Problem, wie man in Abschnitt 6.3 sieht.

  Die Werte für f_x und f_y sind so gewählt, dass sie einer Kamera mit 1/2”-CCD-Chip und 8 mm Brennweite entsprechen. Die Koordinaten der Bildpunkte in der Simulation liegen damit in der gleichen Größenordnung, wie bei einem Bild in PAL-Auflösung.

- Der Hauptpunkt (u_0, v_0) wurde als $(0, 0)$ gewählt. Auch dies ist für den Bündelausgleich selbst nicht wichtig, aber genau wie bei den Brennweiten geht die Selbstkalibrierung von dieser Annahme aus. Bei den echten Aufnahmen wird hierfür der Bildmittelpunkt gewählt und das Bild für die Selbstkalibrierung entsprechend verschoben.

- Die Weltpunkte wurden in jedes Bild projiziert und mit einem normalverteilten Rauschen verschiedener Standardabweichungen σ (in Pixel) versehen.

- Die Rechenzeit für eine Levenberg-Marquardt-Iteration beträgt ca. 45 Sekunden. Diese Angabe bezieht sich auf einen PC mit Intel Celeron-Prozessor mit 366 MHz.

Insgesamt wurden acht Szenen verwendet, wobei immer zwei vom Aufbau her gleich sind und sich nur in den zufällig gewählten 3-D-Punkten und Kameraparametern unterscheiden. Die jeweils erste Szene eines Paares ist in Bild 6.2 graphisch dargestellt.

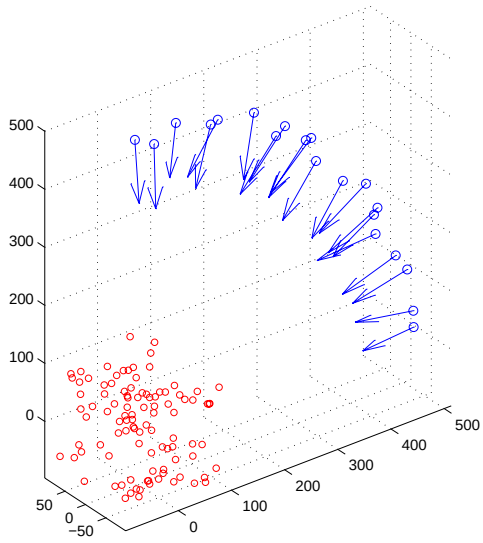
Szenen 1/2. Die Kamera bewegt sich in einem Abstand von 500 mm zum Ursprung radial um den Würfel, wobei ein Winkel von 90° abgedeckt wird (Bild 6.2(a)).

Szenen 3/4. Die Kamera bewegt sich in einem Abstand von 500 mm zum Ursprung radial um den Würfel, wobei ein Winkel von 45° abgedeckt wird. Die einzelnen Kameras liegen also enger zusammen als in denen Szenen 1/2 (Bild 6.2(b)).

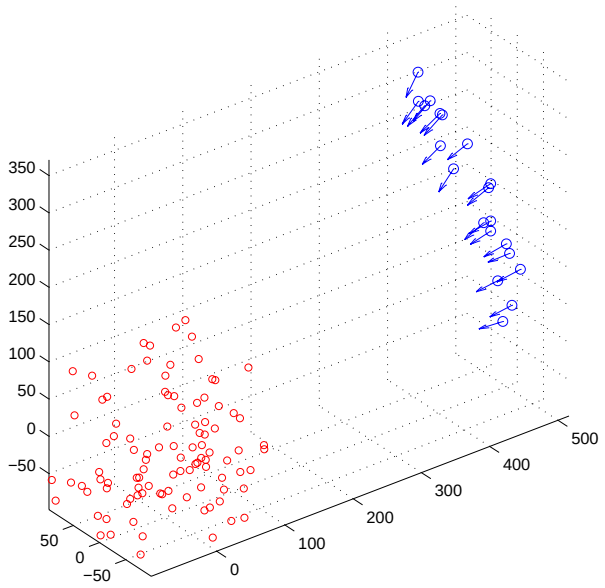
Szenen 5/6. Die Kamera bewegt sich entlang einer Linie der Länge 300 mm, die einen Abstand von 500 mm zum Ursprung des Würfels hat (Bild 6.2(c)).

Szenen 7/8. Die Kamera bewegt sich entlang einer Linie der Länge 1000 mm, die einen Abstand von 500 mm zum Ursprung des Würfels hat. Hier liegen die einzelnen Kamerapositionen also weiter auseinander als in den Szenen 5/6 (Bild 6.2(d)).

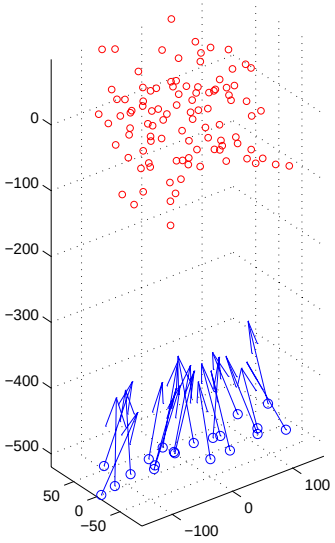
[4] vergleiche Formel (2.9) in Abschnitt 2.2
[5] siehe Abschnitt 2.3



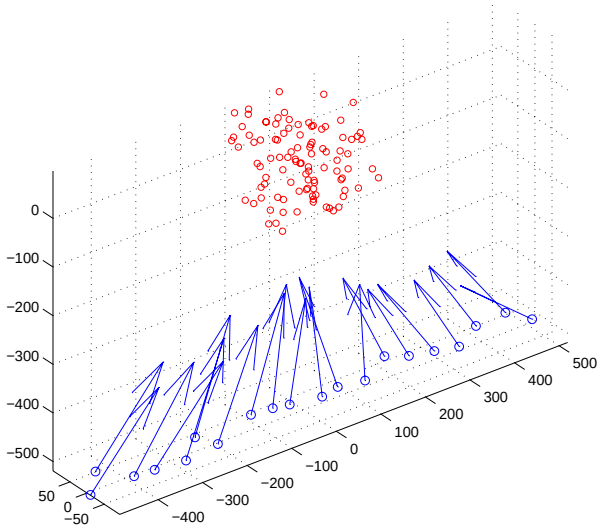
(a) Szenen 1/2



(b) Szenen 3/4



(c) Szenen 5/6



(d) Szenen 7/8

Bild 6.2: Beispiel des Aufbaus der in den Experimenten verwendeten Szenen: Die Weltpunkte (Kreise) befinden sich innerhalb eines Würfels um den Ursprung, die Kamera wurde in den Szenen 1–4 radial um die Szene herum bewegt, in den Szenen 5–8 translatorisch an der Szene vorbei. Die Pfeile geben die Blickrichtung der Kamera an der jeweiligen Position an. (Alle Einheiten in mm)

Die Startwerte für den Bündelausgleich kommen aus dem in [HN99] beschriebenen Verfahren. Um die Leistungsfähigkeit des Bündelausgleichs auch unabhängig vom vorhergehenden Algorithmus zur Rekonstruktion beurteilen zu können, wurden in Abschnitt 6.2.5 zusätzlich Versuche durchgeführt, bei denen an Stelle einer Rekonstruktion die bekannten wahren Werte verrauscht bzw. projektiv verzerrt und als Startwerte für die Optimierung verwendet wurden. Beurteilt wird in der überwiegenden Anzahl der Versuche also die Leistung des Gesamtsystems bestehend aus der initialen Rekonstruktion mit einem linearen Verfahren und der anschließenden nichtlinearen Optimierung in Form des eingebetteten Bündelausgleichs. Allerdings ist es gerade auch diese Gesamtleistung, die bei der Verarbeitung realer Bildfolgen wichtig ist.

Im Vergleich zu den realen Bildfolgen wurden sehr wenige Aufnahmen und 3-D-Punkte verwendet, um die Rechenzeiten in einem vernünftigen Rahmen zu halten. Allerdings sind die 100 Punkte in 20 Bildern mehr, als üblicherweise in der Literatur verwendet wird: In [HÅ97] beispielsweise sind es einmal 10 Punkte in 15 Bildern und einmal 50 Punkte in 20 Bildern, in [McL99] werden 20 Punkte in 10 Bildern verwendet.

Dabei ist anzumerken, dass der Algorithmus mit mehr Daten stabiler wird; es ist damit also kein schlechteres Verhalten als in den hier durchgeführten Versuchen zu erwarten.

In den kommenden Abschnitten wird versucht, folgende Fragen zu beantworten:

- Wieviele Iterationsschritte werden benötigt, bis der Rückprojektionsfehler stabil ist? Wie stark verändern sich die Fehler in den Parametern? (Abschnitt 6.2.1)
- Wie wirkt sich Rauschen auf den Bildpunkten mit verschiedener Varianz auf das Ergebnis aus? (Abschnitt 6.2.2)
- Macht es einen Unterschied, ob zur Parametrisierung der Rotation[6] Quaternionen oder die Achse/Winkel-Darstellung verwendet wird? (Abschnitt 6.2.3)
- Gelingt der Versuch, Linsenverzerrungen beim Bündelausgleich zu korrigieren? (Abschnitt 6.2.4)

In jedem Abschnitt werden nur einige aus den gesamten durchgeführten Simulationen ausgewählte Daten präsentiert, wobei versucht wurde diese Auswahl so repräsentativ wie möglich durchzuführen und auch Sonderfälle zu betrachten. Tabellen mit den gesamten Daten befinden sich im Anhang D.

In den Tabellen ist bei den jeweiligen Parametern immer der mittlere quadratische Fehler gegenüber den wahren Werten gemittelt über alle Aufnahmen bzw. 3-D-Punkte der Szene vor der Optimierung angegeben. Die dabei zugrunde gelegten Werte sind diejenigen, die sich *nach* der Orthogonalisierung der Rotationsmatrix ergeben, wie in Abschnitt 6.1 beschrieben. Da die Rekonstruktion nur bis auf eine unbekannte Ähnlichkeitstransformation eindeutig ist, wurde diese mit Hilfe der bekannten wahren Parameterwerte ermittelt. Alle Angaben zu Fehlern in den Parametern ergeben sich nach Anwendung dieser Transformation.

Weiterhin sind beim Rückprojektionsfehler vor der Optimierung zwei Werte angegeben: Der erste ist der Fehler *vor* der Orthogonalisierung, also der, den das lineare Verfahren liefert und der sich auf eine nur näherungsweise euklidische Rekonstruktion bezieht. Der zweite Wert ist der, der sich nach der Orthogonalisierung ergibt; dies ist der anfängliche Fehler in der Optimierung.

[6] siehe Abschnitt 4.2.2

Der Rückprojektionsfehler ist der mittlere quadratische Fehler pro 2-D-Punkt in Pixel, welcher sich entsprechend Formel (4.3) wie folgt ergibt:

$$\sqrt{\frac{1}{mn}\sum_{j=1}^{n}\sum_{i=1}^{m}\left(\left(u_{ij}-\tilde{u}_{ij}\right)^2+\left(v_{ij}-\tilde{v}_{ij}\right)^2\right)} \quad . \tag{6.3}$$

Abgesehen vom Rückprojektionsfehler geben die Werte *nach* der Optimierung die *Änderung* des Fehlers für den jeweiligen Parameter an, bezogen auf die Werte vor der Optimierung; dadurch kann man auf einen Blick abschätzen, wie groß die Veränderung überhaupt war, und ob der Fehler im jeweiligen Parameter durch die Optimierung im Mittel kleiner (negatives Vorzeichen) oder größer (positives Vorzeichen) wurde.

Die Werte für die nicht-skalaren Größen, also die 3-D-Punkte, den Translationsvektor und die Rotationsmatrix sind wie folgt angegeben:

3-D-Punkte. Ähnlich wie beim Rückprojektionsfehler bei den Bildpunkten ist auch hier der mittlere quadratische Fehler bzw. dessen Änderung pro 3-D-Punkt in mm angegeben.

Translationsvektor. Hier ist der mittlere quadratische Fehler bzw. dessen Änderung pro Komponente des Vektors in mm angegeben.

Rotationsmatrix. Für die Beurteilung des Fehlers in der Rotation wurden nicht die Rotationsmatrizen, sondern Quaternionen verwendet. Diese bieten sich dafür an, da eine glatte Änderung des Quaternions eine ebensolche Änderung der Rotation zur Folge hat. Insbesondere: Wenn zwei Quaternionen weit auseinander liegen, so sind auch die dazugehörenden Rotationen sehr verschieden.[7] Wie bei der Translation ist auch hier der mittlere quadratische Fehler bzw. dessen Änderung pro Komponente des Quaternions angegeben.

6.2.1 Anzahl der nötigen Iterationsschritte

Als Erstes wurde untersucht, wieviele Levenberg-Marquardt-Iterationsschritte nötig sind, um ein zufrieden stellendes Ergebnis zu erhalten. Natürlich ist es prinzipiell so, dass man mit mehr Iterationen auch bessere Werte erhält, wobei diese sich ab einem bestimmten Punkt nicht mehr stark ändern. Für die Anwendung des Verfahrens auf reale Bildfolgen ist es jedoch vorteilhaft, so wenig Iterationen wie möglich durchzuführen, da insbesondere die Berechnung der Jacobi-Matrix viel Rechenzeit kostet. Eine Iteration auf einem Bild mit 50 Aufnahmen und ca. 900 Punkten dauert beispielsweise etwa 30 Minuten[8].

Es wurden die Ergebnisse der Optimierung für alle acht Szenen betrachtet, wobei die Anzahl der Iterationen zwischen 1 und 30 variiert wurde. Da die Veränderung der Fehler für die hier näher betrachteten Szenen graphisch dargestellt ist, wurde auf den Abdruck der zugehörigen Tabellen in diesem Abschnitt verzichtet. Exemplarisch sind die Werte für Szene 1 mit $\sigma = 0.3$ Pixel in Tabelle 6.1 enthalten. In Anhang D.1 befinden sich die vollständigen Tabellen mit den Ergebnissen für alle acht Szenen. Daraus wurden folgende ausgewählt, die hier näher betrachtet werden sollen: Szene 1 mit $\sigma = 0.3$ Pixel (Tabelle 6.1) und $\sigma = 1.0$ Pixel (Tabelle D.2); dies ist eine rotatorische Szene. Szene 5 (translatorisch) mit $\sigma = 0.3$ Pixel (Tabelle D.9); diese

[7] Zu beachten ist die nicht eindeutige Darstellung einer Rotation als Quaternion; siehe hierzu Anhang A

[8] Versuche zur Anwendung des Bündelausgleichs auf reale Szenen dieser Größe befinden sich in Abschnitt 6.3

#Iter	f_x	f_y	u_0	v_0	$\boldsymbol{t}$	$\boldsymbol{R}$	3-D	Rückproj.
0	8.60	8.13	9.96	4.37	2.21369	0.00269973	0.243482	0.45/0.49
1	-0.189	0.168	-2.50	-0.0402	0.0364844	-0.000585481	-0.086026	0.407386
2	-0.466	-0.0420	-3.31	-0.471	-0.0154891	-0.000802894	-0.095357	0.405092
3	-0.803	-0.321	-3.95	-0.755	-0.0925322	-0.000967943	-0.102567	0.404078
4	-1.14	-0.608	-4.47	-1.02	-0.169569	-0.00110851	-0.108743	0.403375
5	-1.44	-0.882	-4.91	-1.26	-0.243064	-0.00122831	-0.114066	0.40284
7	-2.00	-1.37	-5.63	-1.68	-0.376945	-0.00143381	-0.122566	0.40207
10	-2.67	-1.97	-6.43	-2.13	-0.536825	-0.00166201	-0.131515	0.401348
15	-3.50	-2.73	-7.24	-2.57	-0.738703	-0.00190034	-0.140003	0.4007
20	-4.09	-3.28	-7.69	-2.74	-0.882988	-0.00201266	-0.14373	0.400397
30	-4.85	-4.00	-7.91	-2.83	-1.07228	-0.00207101	-0.145717	0.40017

Tabelle 6.1: Szene 1: Unterschiedliche Anzahl an Iterationen, $\sigma = 0.3$ Pixel.

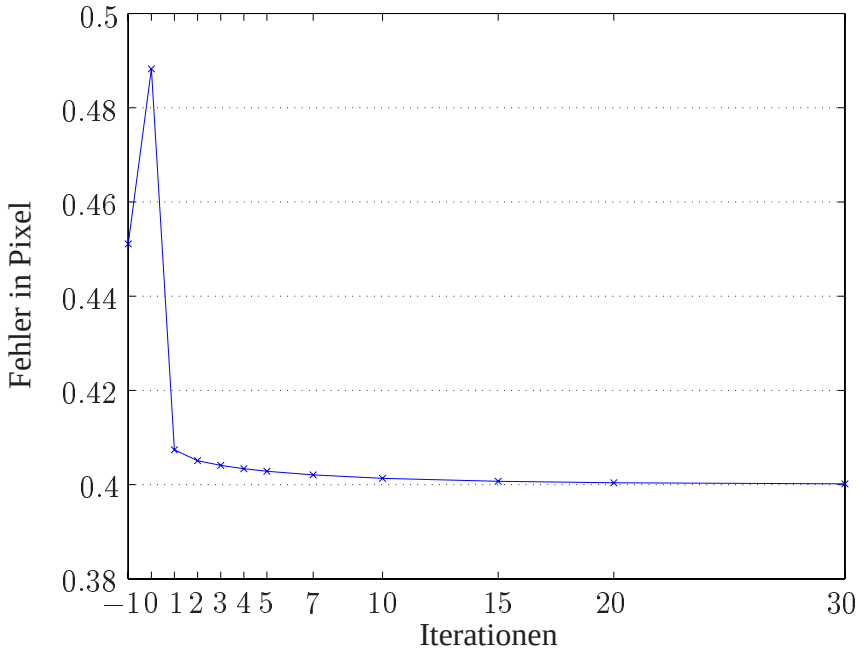


Bild 6.3: Rückprojektionsfehler in Abhängigkeit von der Anzahl der Iterationen beim Bündelausgleich. Szene 1, $\sigma = 0.3$ Pixel.

wurde gewählt, da der Rückprojektionsfehler durch die Orthogonalisierung der Rotationsmatrix hier sehr stark ansteigt. Das bedeutet, die Rekonstruktion, welche das vorangegangene lineare Verfahren geliefert hat, war noch stark projektiv verzerrt, und man hat folglich eher schlechte Startwerte für die Optimierung. Weiterhin wurde Szene 7 (translatorisch) mit $\sigma = 1.0$ Pixel (Tabelle D.14) ausgewählt.

Zunächst soll die Veränderung des Rückprojektionsfehlers im Verlauf der Optimierung betrachtet werden. Diese ist in den Bildern 6.3 (Szene 1, $\sigma = 0.3$ Pixel), 6.4 (Szenen 1 und 7, $\sigma = 1.0$ Pixel) sowie 6.5 (Szene 5, $\sigma = 0.3$ Pixel) graphisch dargestellt. Der Wert bei -1 gibt den Fehler an, den das vorhergehende Verfahren liefert, der Wert bei 0 entspricht dem Rückprojektionsfehler nach der Orthogonalisierung der Rotationsmatrix. In allen Fällen steigt der Fehler wie erwähnt durch die Orthogonalisierung vor dem Bündelausgleich erst einmal mehr

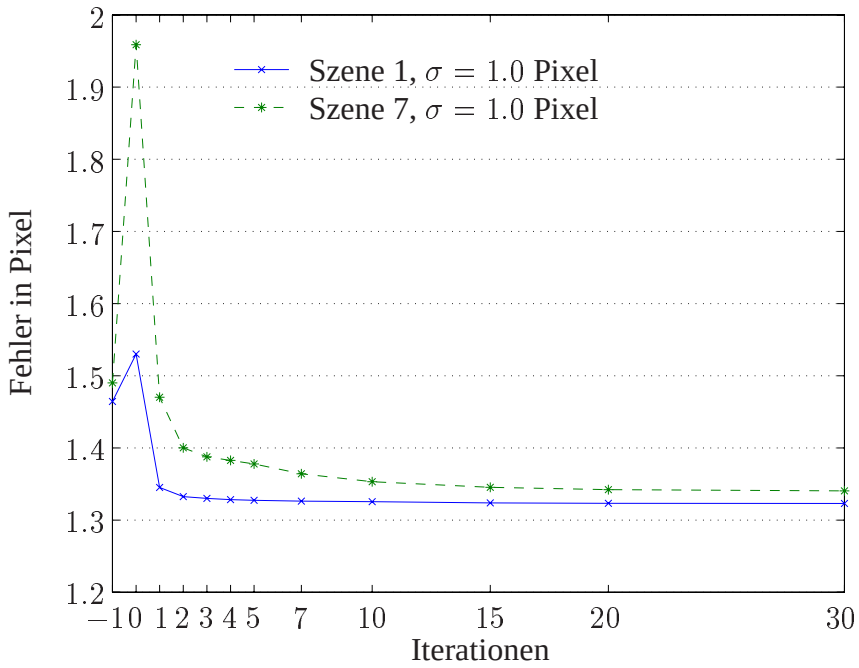


Bild 6.4: Rückprojektionsfehler in Abhängigkeit von der Anzahl der Iterationen beim Bündelausgleich. Szenen 1 und 7, jeweils $\sigma = 1.0$ Pixel.

oder weniger stark an. Dies ist abhängig davon, wie gut die vorhergehende Selbstkalibrierung funktioniert hat. Außerdem sieht man, dass der Rückprojektionsfehler bereits nach dem ersten Iterationsschritt steil abfällt, in den meisten Fällen wieder unter den Wert vor der Orthogonalisierung.

Im Allgemeinen kann man sagen, dass sich der Rückprojektionsfehler nach ca. 10 Iterationen nicht mehr gravierend ändert, wenn man ein Rauschen mit $\sigma = 0.3$ Pixel zugrunde legt. Je höher das Rauschen auf den Bildpunkten ist, desto mehr Iterationen werden benötigt. Dies ist in der Praxis natürlich vorher möglicherweise schwer abschätzbar, man kann als groben Richtwert allerdings den Rückprojektionsfehler vor der Orthogonalisierung verwenden, wenn man Formel (3.18) zur Berechnung eines Schätzwertes für σ zu Hilfe nimmt.[9]

Weiterhin werden mehr Iterationen benötigt, wenn das lineare Verfahren keine gute euklidische Rekonstruktion liefert. Dies kann man anhand des Anstiegs des Rückprojektionsfehlers bei der Orthogonalisierung vor der Optimierung feststellen. Es wird also empfohlen, die Anzahl der durchzuführenden Iterationen dadurch zu steuern.

Nun sollen die Veränderungen in den einzelnen zu optimierenden Parametern näher betrachtet werden. Zunächst die Rotation: Eine graphische Darstellung des Verlaufs der Änderung des Fehlers in der Rotation findet man in den Bildern 6.6 (Szene 1) und 6.7 (Szenen 5 und 7). Zunächst fällt auf, dass der Fehler in der Rotation bereits von Anfang an sehr gering ist, er liegt in allen durchgerechneten Fällen unter 0.1, in vielen sogar in der Größenordnung von 10^{-3}. Dadurch wirken gerade die Veränderungen nach vielen Iterationen evtl. größer als sie tatsächlich sind. Insgesamt gesehen nimmt der Fehler in der Rotation bezogen auf den Startwert immer sehr stark ab, meist deutlich über 50%. Auch hier sind die Veränderungen zu Beginn am stärksten, die Kurven flachen dann immer weiter ab. Betrachtet man Szene 5, so erkennt man, dass

[9] Diese gilt eigentlich erst *nach* der Optimierung; als Anhaltspunkt sollte der Wert aber genügen.

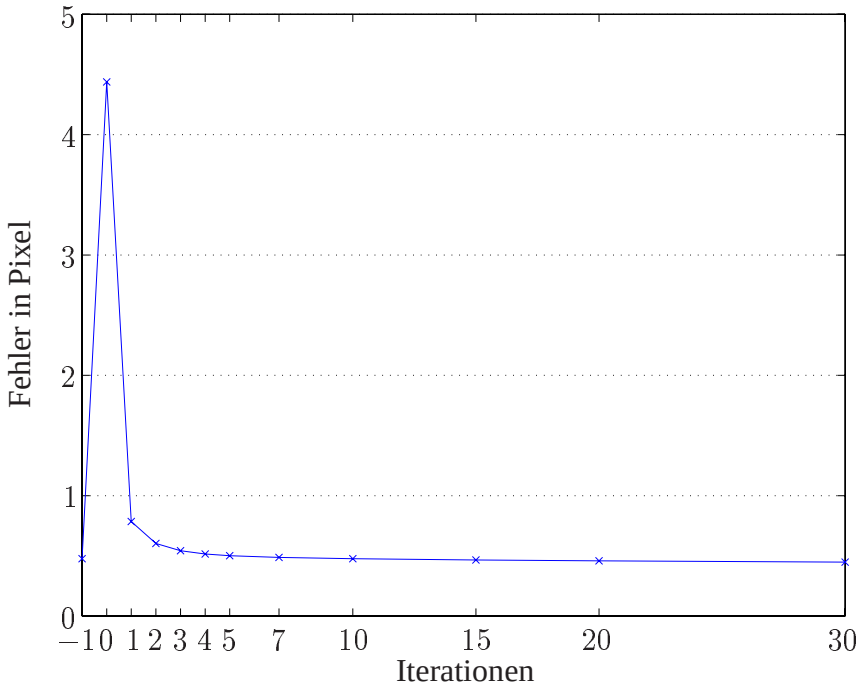


Bild 6.5: Rückprojektionsfehler in Abhängigkeit von der Anzahl der Iterationen beim Bündelausgleich. Szene 5, $\sigma = 0.3$ Pixel.

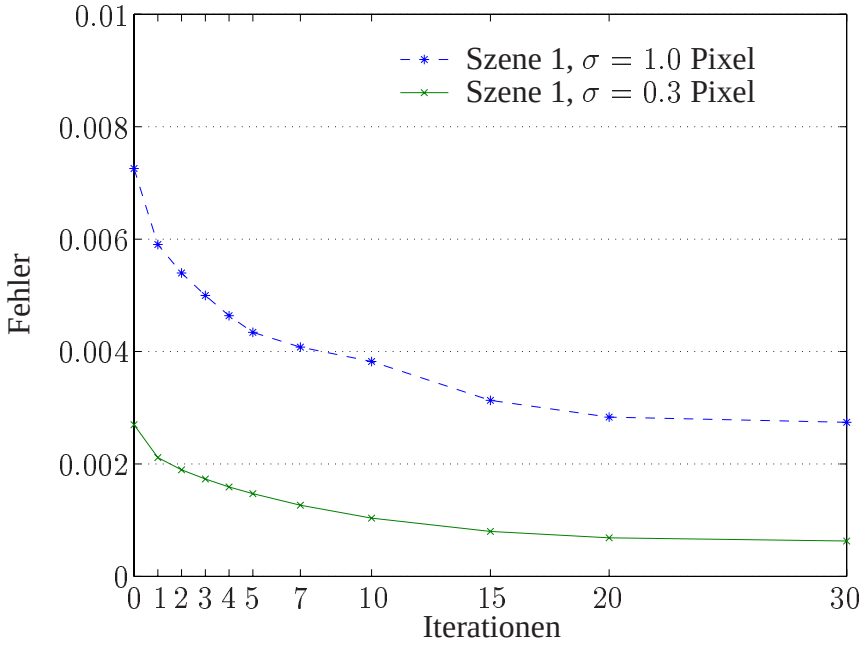


Bild 6.6: Fehler in der Rotation in Abhängigkeit von der Anzahl der Iterationen beim Bündelausgleich. Szene 1.

hier die Änderung des Fehlers von 20 auf 30 Iterationen im Vergleich zu den anderen Szenen noch relativ stark ist; es sind also mit weiteren Iterationen noch Verbesserungen zu erwarten.

Der Fehler in den 3-D-Punkten ist für Szene 1 in Bild 6.8 und für die Szenen 5 und 7 in

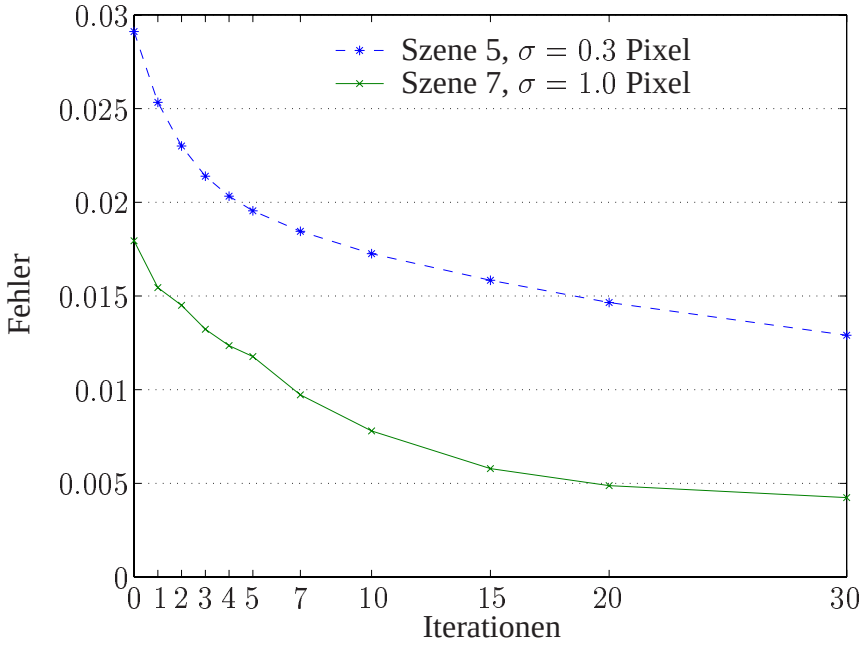


Bild 6.7: Fehler in der Rotation in Abhängigkeit von der Anzahl der Iterationen beim Bündelausgleich. Szenen 5 und 7.

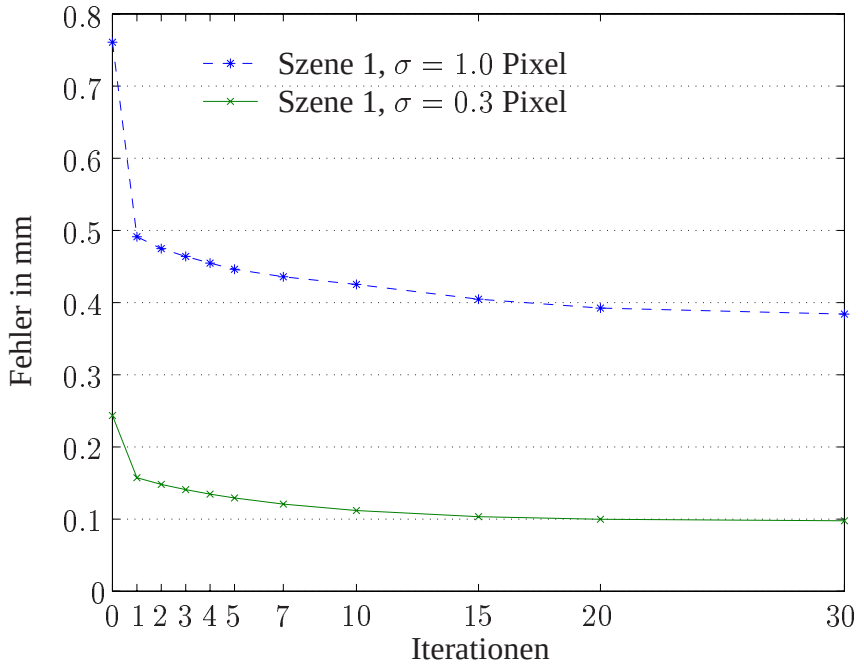


Bild 6.8: Fehler in den 3-D-Punkten in Abhängigkeit von der Anzahl der Iterationen beim Bündelausgleich. Szene 1.

Bild 6.9 dargestellt. Zunächst kann man erkennen, dass hier – ebenso wie bei der Rotation – mehr Iterationen als beim Rückprojektionsfehler nötig sind, bis sich der 3-D-Fehler nur noch gering ändert. Auch hier gilt: Bei höherem Rauschen sollte man mehr Iterationen durchführen.

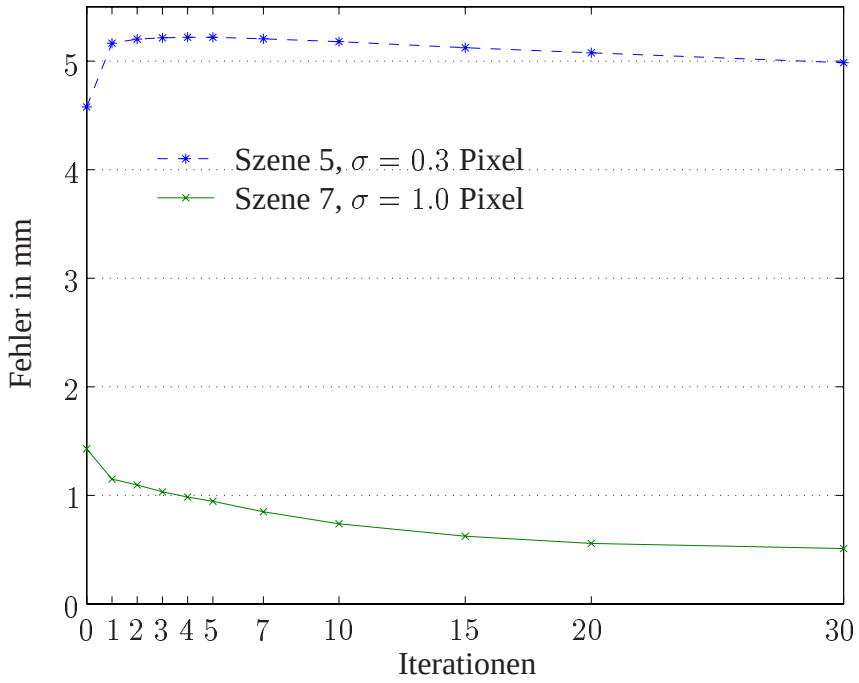


Bild 6.9: Fehler in den 3-D-Punkten in Abhängigkeit von der Anzahl der Iterationen beim Bündelausgleich. Szenen 5 und 7. Deutlich zu sehen ist der *Anstieg* des Fehlers bei Szene 5 aufgrund der fehlerhaften Selbstkalibrierung.

Szene 5 fällt, wie bereits vorher, etwas aus dem Rahmen: Hier steigt der 3-D-Fehler zu Beginn der Optimierung an statt zu sinken. Nach einigen Iterationen wird er zwar wieder geringer, allerdings bewegt er sich weiterhin auf höherem Niveau als vor dem Bündelausgleich. Auch dieses Verhalten hängt wieder mit der schlechten Selbstkalibrierung zusammen.

Ein Anstieg des Fehlers ist in einzelnen Parametern bei verschiedenen Szenen immer wieder zu beobachten. Im weitaus überwiegenden Teil der Fälle jedoch nimmt der Fehler in allen Parametern während der Optimierung ab. Eine Garantie dafür gibt es allerdings nicht, da ein lokales Optimierungsverfahren verwendet wird, welches immer zum nächstgelegenen Minimum hin konvergiert. Ausgenommen von einer Verschlechterung ist natürlich der Rückprojektionsfehler: Dieser wird immer kleiner, schließlich ist er das Optimierungskriterium.

Die Veränderung des Fehlers in den intrinsischen Kameraparametern ist exemplarisch für Szene 1 in Bild 6.10 dargestellt. Zu beobachten ist hier, dass die Fehler zu Beginn relativ groß sind. Nach ca. 15 Iterationen sind bei den meisten Parametern nur noch geringe Änderungen zu sehen; nur bei u_0 ($\sigma = 1.0$ Pixel) und f_x bzw. f_y ($\sigma = 0.3$ Pixel) ergeben sich auch danach noch größere Veränderungen. Betrachtet man die Werte zu Szene 5 in Tabelle D.9, so erkennt man, dass dies wiederum nicht für diese Szene gilt; die Veränderungen sind dort auch zwischen der 20. und 30. Iteration noch recht groß. Auch bei Szene 7 (Tabelle D.14) kann man nach dieser Anzahl Iterationen noch stärkere Änderungen sehen: Im Vergleich zu Szene 1 waren die Fehler in den intrinsischen Parametern bei gleichem Rauschen von Anfang an höher.

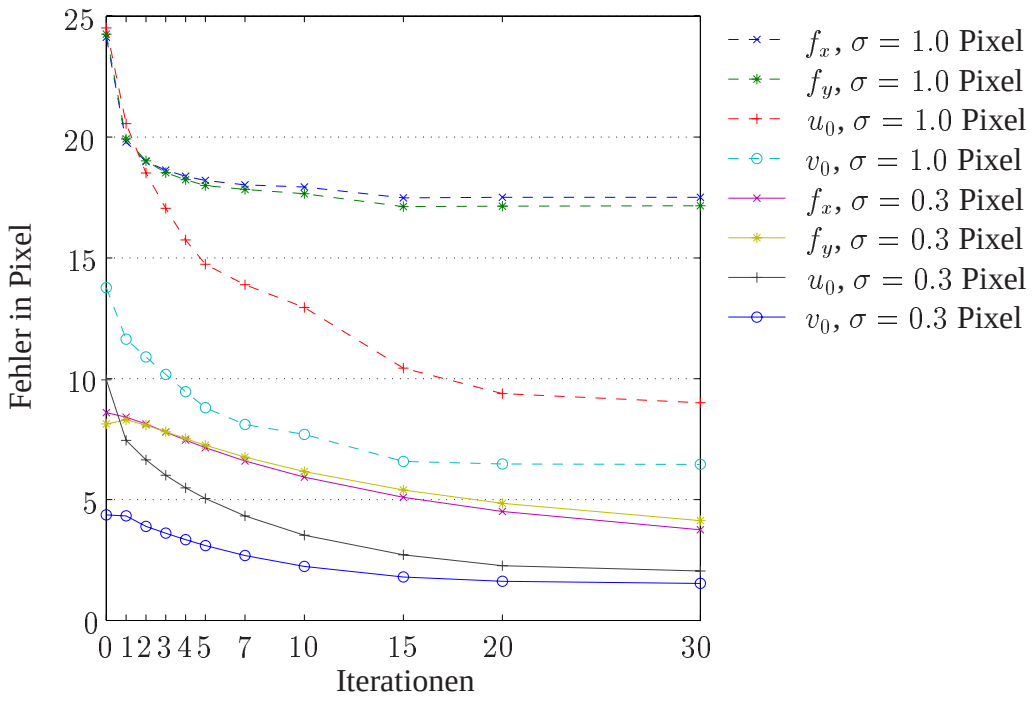


Bild 6.10: Fehler in den intrinsischen Kameraparametern in Abhängigkeit von der Anzahl der Iterationen beim Bündelausgleich. Szene 1.

6.2.2 Einfluss des Rauschens auf den Bildpunkten

In diesem Abschnitt wird untersucht, wie sich der Bündelausgleich in Abhängigkeit der Stärke des Rauschens auf den Bildpunkten verhält. Dazu wurden die Bildpunkte aller acht Szenen mit einem mittelwertfreien und normalverteilten Rauschen versehen, wobei die beiden Koordinaten der Bildpunkte unabhängig voneinander verrauscht wurden. Die verwendeten Standardabweichungen betrugen $\sigma = 0.1, 0.2, 0.3, 0.5, 0.7, 1.0$ und 2.0 Pixel. Es wurden jeweils 20 Iterationen durchgeführt, die Parametrisierung der Rotation erfolgte mit Quaternionen.

Aus den acht Szenen wurden wiederum die Szenen 1, 5 und 7 für eine nähere Betrachtung ausgewählt. Die zugehörigen Tabellen 6.2, 6.3 und 6.4 zeigen einen Ausschnitt aus den kompletten Daten der jeweiligen Szene. Die vollständigen Tabellen zu allen Szenen befinden sich in Anhang D.2.

Szene 1 besteht aus einer rotatorischen, Szene 7 aus einer translatorischen Kamerabewegung. Szene 5 ist diejenige, bei der die vorangegangene Selbstkalibrierung nur sehr schlecht war, was sich in einem starken Anstieg des Rückprojektionsfehlers durch die Orthogonalisierung der Rotationsmatrix äußert.

Die Bilder 6.11 (Szene 1), 6.12 (Szene 5) und 6.13 (Szene 7) zeigen graphische Darstellungen des Rückprojektionsfehlers in Abhängigkeit von der Stärke des Rauschens. In jedem Bild sind drei Kurven enthalten: Der Fehler direkt nach dem linearen Verfahren, welches eine nur näherungsweise euklidische Rekonstruktion liefert (mittlere Kurve), der Fehler vor der Optimierung, aber nach der Orthogonalisierung der Rotationsmatrix (obere Kurve) sowie der Fehler nach der Optimierung (untere Kurve).

Bei Szene 1 (Bild 6.11) ist zu beobachten, wie die Kurven aussehen sollten, wenn die vorhergehende Selbstkalibrierung funktioniert hat: Der Fehler vor der Orthogonalisierung ist immer

σ	f_x	f_y	u_0	v_0	$\boldsymbol{t}$	$\boldsymbol{R}$	3-D	Rückproj.	$\hat{\sigma}$
0.1	3.09	2.94	3.12	2.08	0.860836	0.00096633	0.0881747	0.14/0.16	
	-1.12	-0.931	-1.88	-1.35	-0.29028	-0.000610078	-0.0506682	0.13	0.093
0.5	18.9	18.5	12.5	5.25	5.36	0.00350493	0.410793	0.75/0.78	
	-2.69	-2.06	-5.13	0.0843	-0.632547	-0.00126083	-0.14391	0.66	0.48
2.0	42.6	44.8	45.6	41.3	13.0532	0.0159312	1.58283	3.00/3.22	
	-26.7	-26.8	-28.9	-24.2	-7.90669	-0.00991456	-0.85271	2.64	1.91

Tabelle 6.2: Szene 1: Normalverteiltes Rauschen verschiedener Stärke

σ	f_x	f_y	u_0	v_0	$\boldsymbol{t}$	$\boldsymbol{R}$	3-D	Rückproj.	$\hat{\sigma}$
0.1	6.76	6.88	2.63	1.89	2.11889	0.000700558	0.235186	0.16/0.16	
	-1.04	-1.22	-0.498	-0.0717	-0.278572	-5.58657e-06	-0.0885836	0.13	0.094
0.5	377	387	210	172	127.644	0.0481796	5.1844	0.83/12.8	
	-99.3	-113	-111	-87.1	-32.6377	-0.0237607	1.51837	0.75	0.54
2.0	482	483	174	122	169.656	0.0836537	32.3138	3.33/4.07	
	1.86	-2.19	-5.95	-2.31	-1.83063	0.000628537	-1.5482	2.83	2.05

Tabelle 6.3: Szene 5: Normalverteiltes Rauschen verschiedener Stärke

etwas kleiner als nachher, der Anstieg des Rückprojektionsfehlers jedoch nur gering. Mit zunehmender Stärke des Rauschens wird die Selbstkalibrierung immer schlechter, wodurch auch der Anstieg des Fehlers größer wird. Durch die Optimierung fällt er jedoch wieder unter den Wert vor der Orthogonalisierung.

Bei Szene 5 (Bild 6.12) erkennt man bei einer Standardabweichung von 0.3 Pixel und besonders 0.5 Pixel einen starken Anstieg des Fehlers durch die Orthogonalisierung, die Selbstkalibrierung hat hier versagt. Man erkennt auch: Ein direkter Zusammenhang zwischen der schlechten Selbstkalibrierung und der Szene oder der Stärke des Rauschens besteht nicht, da sich der Fehler bei höherem σ wieder ebenso wie bei Szene 1 verhält. Würde man die gleiche Szene mit gleicher Standardabweichung nochmals verrauschen, so sähe das Verhalten wieder anders aus, insbesondere kann dann auch die Selbstkalibrierung in allen Fällen gut funktionieren.

In Szene 7 (Bild 6.13) kann man bei $\sigma = 1.0$ Pixel ebenfalls eine fehlerhafte Selbstkalibrierung erkennen, wenn auch der Fehler nicht so stark ansteigt wie in Szene 5.

In allen Fällen – auch bei denjenigen, bei denen der Fehler vor der Optimierung stark ansteigt – liegt der Rückprojektionsfehler nach dem Bündelausgleich niedriger als vor der Orthogonalisierung.

Bezüglich der einzelnen zu optimierenden Parameter ergibt sich folgendes Bild: Mit steigender Stärke des Rauschens verschlechtern sich auch die Startwerte für den Bündelausgleich

σ	f_x	f_y	u_0	v_0	$\boldsymbol{t}$	$\boldsymbol{R}$	3-D	Rückproj.	$\hat{\sigma}$
0.1	3.75	3.97	2.02	2.05	1.43082	0.000712568	0.0902733	0.15/0.15	
	-0.681	-0.995	-0.537	-0.472	-0.397488	-0.000141937	-0.0392904	0.13	0.096
0.5	32.1	33.9	15.2	28.5	11.4042	0.00790888	0.667514	0.73/0.83	
	-12.2	-13.4	-4.54	-11.7	-4.48034	-0.00318831	-0.336339	0.66	0.48
2.0	76.3	76.5	30.1	20.7	29.8411	0.0102848	1.71714	2.83/2.87	
	-26.7	-25.5	-8.62	1.62	-10.9875	-0.00229837	-0.626682	2.61	1.89

Tabelle 6.4: Szene 7: Normalverteiltes Rauschen verschiedener Stärke

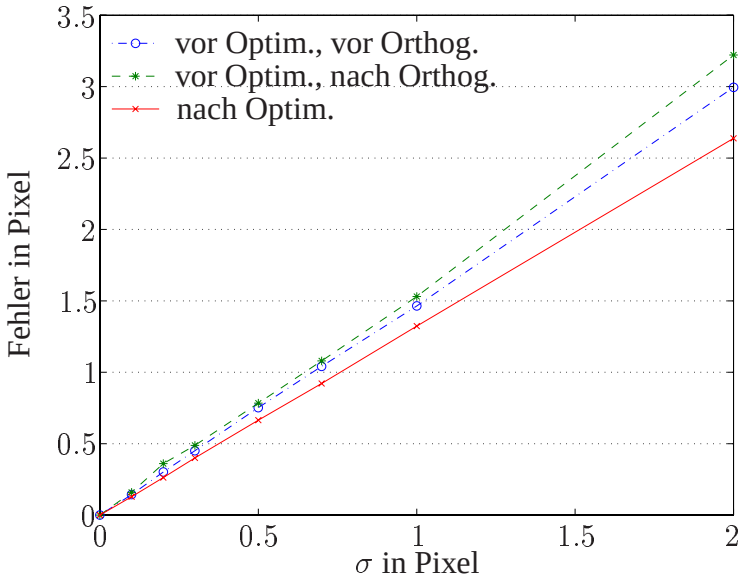


Bild 6.11: Einfluss der Stärke des Rauschens auf den Bildpunkten auf den Rückprojektionsfehler bei Szene 1.

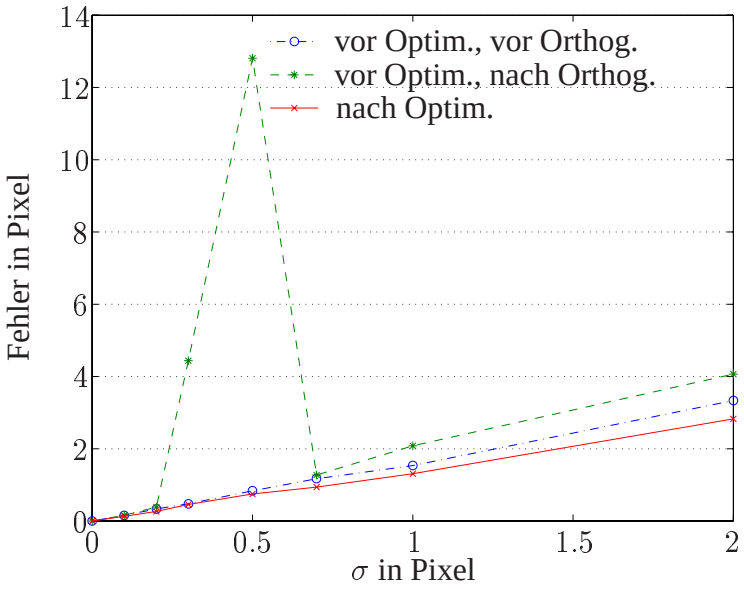


Bild 6.12: Einfluss der Stärke des Rauschens auf den Bildpunkten auf den Rückprojektionsfehler bei Szene 5. Deutlich erkennbar ist der starke Anstieg des Fehlers durch die Orthogonalisierung der Rotationsmatrix bei $\sigma = 0.3$ Pixel und $\sigma = 0.5$ Pixel. Doch auch dieser große Fehler wird durch den Bündelausgleich wieder unter das Niveau vor der Orthogonalisierung gedrückt.

immer mehr, dennoch wird eine Verbesserung der Werte durch die Optimierung erreicht.

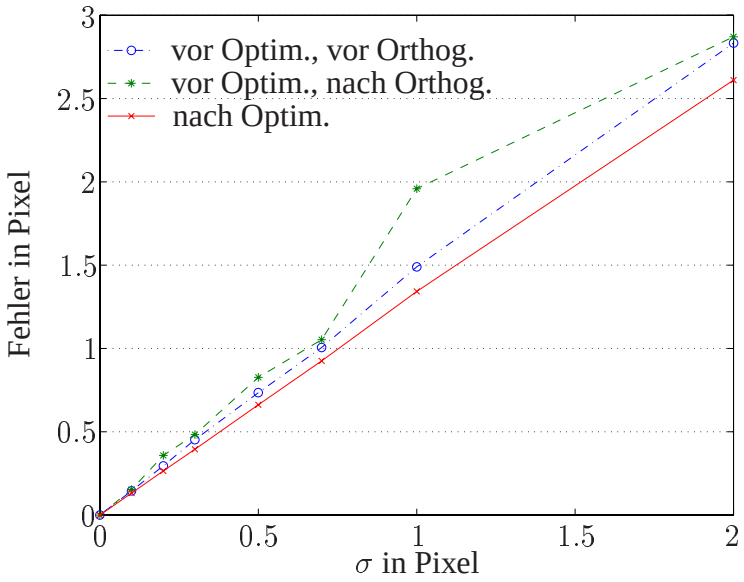


Bild 6.13: Einfluss der Stärke des Rauschens auf den Bildpunkten auf den Rückprojektionsfehler bei Szene 7. Auch hier sind die Fehler in der Selbstkalibrierung bei $\sigma = 0.5$ Pixel und $\sigma = 1.0$ Pixel zu erkennen, wenn sie auch nicht so groß sind wie in Szene 5.

6.2.3 Unterschiedliche Parametrisierung der Rotation

In Abschnitt 4.2.2 wurden zwei Parametrisierungen für Rotationen vorgestellt: Die Darstellung als Rotationsachse und -winkel sowie die Quaternionen. In der Literatur wird üblicherweise die Achse/Winkel-Darstellung gewählt; da in dieser Arbeit jedoch eine einfache Möglichkeit zur Verwendung von Quaternionen vorgestellt wurde, soll hier ein Vergleich der beiden Parametrisierungen durchgeführt werden. Es stellt sich die Frage, ob eine der beiden besser für den Bündelausgleich geeignet ist, d. h. insbesondere:

- Liefert eine Parametrisierung generell bessere Ergebnisse auf allen zu optimierenden Parametern?
- Führt sie schneller zu einer größeren Verbesserung des Rückprojektionsfehlers?
- Ergeben sich zumindest für einige Parameter (insbesondere die Rotation) mit einem Verfahren immer bessere Werte?

Hierzu wurden die anfangs genannten acht Szenen mit einem normalverteilten Rauschen mit den Standardabweichungen $\sigma = 0.3$ Pixel, $\sigma = 0.7$ Pixel sowie $\sigma = 1.0$ Pixel durchgerechnet, wobei die Optimierung nach jeweils 20 Iterationen abgebrochen wurde. Die Tabellen mit den Ergebnissen befinden sich in Anhang D.3, eine Auswahl daraus ist in diesem Abschnitt in Tabelle 6.5 abgedruckt.
Zu beobachten ist:

- Bei Szene 1 waren die beiden Parametrisierungen in etwa gleich gut.
- Bei Szene 2 waren die Quaternionen besser als die Darstellung der Rotation mit Achse und Winkel.

Szene	σ		f_x	f_y	u_0	v_0	$\boldsymbol{t}$	$\boldsymbol{R}$	3-D	Rückproj.
1	0.3	vor	8.60	8.13	9.96	4.37	2.21369	0.00269973	0.243482	0.45/0.49
		Q	-4.09	-3.28	-7.69	-2.74	-0.882988	-0.00201266	-0.14373	0.40
		A/W	-4.10	-3.28	-7.65	-2.75	-0.885005	-0.00200837	-0.14363	0.40
2	1.0	vor	36.0	36.5	38.9	26.1	9.57001	0.0110419	0.984122	1.50/1.86
		Q	-7.89	-8.12	-21.8	-15.2	-1.97096	-0.00637703	-0.504492	1.31
		A/W	-7.86	-7.83	-19.2	-15.0	-1.97941	-0.00573987	-0.490924	1.32
4	0.7	vor	88.7	91.4	47.5	41.1	26.6822	0.0138212	1.78132	1.20/1.73
		Q	-12.4	-15.4	-12.6	-13.5	-3.73035	-0.00545732	-0.81368	0.98
		A/W	-14.0	-17.3	-19.4	-16.6	-4.64984	-0.00657011	-0.967142	0.96
6	1.0	vor	340	346	171	128	91.9583	0.0623885	8.46183	1.86/3.95
		Q	-1.68	-8.46	-0.940	-1.50	0.173968	0.00652012	-0.397551	1.70
		A/W	-2.52	-9.48	0.281	-1.50	0.0102833	0.00716027	-0.254497	1.70
8	0.7	vor	27.7	27.0	28.6	19.7	9.38349	0.00846833	0.667536	1.01/1.11
		Q	-16.7	-16.1	-20.2	-11.3	-6.00133	-0.00561137	-0.35916	0.91
		A/W	-17.0	-16.5	-21.5	-12.7	-6.13845	-0.00602556	-0.361227	0.91
8	1.0	vor	243	247	180	188	97.0723	0.0568097	3.02011	1.43/7.23
		Q	-61.9	-66.3	-111	-75.6	-27.7597	-0.028154	-0.894316	1.37
		A/W	-61.7	-66.3	-111	-74.9	-27.7146	-0.0280119	-0.892473	1.38

Tabelle 6.5: Einige ausgewählte Daten zu den unterschiedlichen Arten der Parametrisierung der Rotation: Quaternionen (Q) und Achse/Winkel-Darstellung (A/W).

- Die Szenen 4 und 8 ($\sigma = 0.7$ Pixel) wurden gewählt, da in diesen beiden Fällen die Achse/Winkel-Darstellung wesentlich besser als die Quaternionen-Parametrisierung ist.

- Bei den Szenen 6 und 8 mit $\sigma = 1.0$ Pixel war die vor der Optimierung laufende Selbstkalibrierung sehr schlecht, was sich im großen Anstieg des Rückprojektionsfehlers vor der Optimierung und nach der Orthogonalisierung der Rotationsmatrix zeigt. Dies bedeutet insbesondere, dass die Fehler in den einzelnen Parametern vor der Optimierung sehr groß und damit die Startwerte sehr schlecht sind. In Szene 6 zeigt sich dies beispielsweise in einem Zuwachs des Fehlers in der Rotation nach dem Bündelausgleich, wobei dieser Zuwachs bei den Quaternionen ca. 1% geringer ausfällt als bei der Achse/Winkel-Darstellung.

Betrachtet man die Ergebnisse der Versuche, so kann man nicht sagen, dass eine der beiden Parametrisierungen immer besser ist als die andere. Wird nur auf den Rückprojektionsfehler Wert gelegt, so gibt es zwischen den beiden Varianten praktisch keinen Unterschied: Nach 20 Iterationen ist der Fehler immer in etwa gleich groß.

In wenigen Fällen sind mit einer Parametrisierung nach der Optimierung *alle* Parameter besser als bei der anderen, wobei dies einmal für die Quaternionen und ein anderes Mal für die Achse/Winkel-Darstellung gilt. Abhängigkeiten von der verwendeten Szene oder dem Rauschen auf den Bildpunkten konnten nicht festgestellt werden.
Betrachtet man die einzelnen Parameter, so ergibt sich folgendes Bild[10]:

Rotation. In 70,8% der Fälle ist die Verringerung des Fehlers in der Rotation bei Verwendung von Quaternionen größer als bei Verwendung der Achse/Winkel-Repräsentation. Insgesamt ist bei beiden Verfahren die Änderung prozentual betrachtet recht groß, in vielen

[10] die Angaben in Prozent beziehen sich auf die Gesamtzahl von 24 Versuchen.

Szene	σ	Änderung $\boldsymbol{R}$ (Q)	Änderung $\boldsymbol{R}$ (A/W)	Änderung 3-D (Q)	Änderung 3-D (A/W)
1	0.3	-74.55%	-74.39%	-59.03%	-58.99%
2	1.0	-57.75%	-51.98%	-51.26%	-49.88%
4	0.7	-39.49%	-47.54%	-45.68%	-54.29%
6	1.0	+10.45%	+11.48%	-4.70%	-3.01%
8	0.7	-66.26%	-71.15%	-53.80%	-54.11%
8	1.0	-49.56%	-49.31%	-29.61%	-29.55%

Tabelle 6.6: Die prozentualen Veränderungen des Fehlers in der Rotation sowie auf den 3-D-Punkten zu den Daten in Tabelle 6.5

Fällen über 50%. Die Unterschiede zwischen den beiden variieren von unter 1% bis um die 5%, wobei durchaus auch die Achse/Winkel-Darstellung um diesen Betrag besser sein kann als die Quaternionen. Vergleiche dazu auch Tabelle 6.6.

3-D-Punkte. Betrachtet man die 3-D-Punkte allein, so ergibt sich kein eindeutiger Vorteil für eines der beiden Verfahren. In jeweils etwa der Hälfte der Fälle war die Änderung des Fehlers beim einen größer als beim anderen, auch hier sind die Unterschiede relativ gering. Vergleiche auch hierzu Tabelle 6.6. Große Unterschiede, wie z. B. in Szene 4, sind nicht die Regel, kommen aber vor.

Translation. Auch für die Translation gilt: Beide Verfahren sind in etwa gleich gut, in 50% der Fälle sind die Quaternionen besser, in den anderen 50% die Achse/Winkel-Darstellung.

f_x/f_y. Die Brennweiten wurden zusammen betrachtet, da meist gilt: Ist der Fehler nach der Optimierung in f_x bei einer Darstellung geringer, so auch der in f_y. Nur in zwei Fällen war die Veränderung des Fehlers in f_x bei den Quaternionen und f_y bei der Achse/Winkel-Darstellung größer. In den Verbleibenden sind beide Repräsentationen ungefähr gleich gut.

u_o/v_0. Auch u_o und v_0 wurden zusammen betrachtet, da sie sich ähnlich verhalten wie die Brennweiten. Hier waren in 50% der Fälle die Quaternionen besser, in einem Drittel der Fälle war es die Achse/Winkel-Darstellung. Bei den restlichen sind die Ergebnisse in etwa gleich.

Hierbei ist zu beachten, dass die Unterschiede zwischen den resultierenden Fehlern oft nur sehr gering sind, d. h. selbst wenn eines der beiden Verfahren besser ist, muss dies nicht gravierend sein. Exemplarisch sind dazu in Tabelle 6.6 die prozentualen Änderungen im Fehler in der Rotation sowie in den 3-D-Punkten zu den Werten in Tabelle 6.5 zusammengestellt.

Fazit: Eine eindeutige Empfehlung für eine der beiden Parametrisierungen kann nicht gegeben werden. Für den Anwendungsfall der Lichtfeldrekonstruktion ist der Rückprojektionsfehler wichtig, hier verhalten sich beide in etwa gleich. Legt man besonderen Wert auf eine Verringerung des Fehlers bei der Rotation, so sollten Quaternionen verwendet werden, auch wenn nicht in allen Fällen garantiert ist, dass diese tatsächlich ein besseres Ergebnis liefern als die Achse/Winkel-Darstellung.

6.2.4 Korrektur von Linsenverzerrungen

In diesem Abschnitt geht es um die Korrektur von Linsenverzerrungen, wie sie in Abschnitt 5.2 beschrieben wurde. Insbesondere wurden folgende Fragen untersucht:

- Wie gut funktioniert die Optimierung, wenn im Bild Verzerrungen vorhanden sind und diese mitoptimiert werden?
- Was passiert, wenn Verzerrungen vorhanden sind, diese aber nicht in der Optimierung berücksichtigt werden?
- Wie verhält sich der Bündelausgleich in Bildern, in denen keine Linsenverzerrungen da sind, die zugehörigen Parameter aber trotzdem mitoptimiert werden?

Um die Versuche durchführen zu können, wurden die Linsenverzerrungen simuliert wie in Abschnitt 5.2.4 beschrieben. Dadurch waren die zu schätzenden Verzerrungsparameter κ_i bekannt, und es konnte für diese zusätzlich die Veränderung des mittleren quadratischen Fehlers (in mm^{-2}) über alle Aufnahmen nach der Optimierung berechnet werden.

Für die Versuche wurden die Szenen 1 (rotatorisch) und 7 (translatorisch), jeweils mit $\sigma = 0.3$ Pixel verwendet, was in etwa dem Schätzwert der Standardabweichung in den realen Bildern entspricht (siehe Abschnitt 6.3). Die Verzerrungen wurden mit $\kappa = \pm 0.01, \pm 0.1$ und ± 0.2 mm^{-2} simuliert, wobei dieser Parameter in jedem Bild der Folge gleich war.[11] Dabei wurden zuerst die Bildpunkte verzerrt, anschließend wurde das normalverteilte Rauschen addiert. Wie bereits vorher erfolgte der Abbruch der Optimierung nach 20 Iterationen, für die Parametrisierung der Rotation wurden Quaternionen verwendet.

Die Ergebnisse der Verzerrungssimulation sind für Szene 1 in Tabelle 6.7 sowie für Szene 7 in Tabelle 6.8 zusammengefasst. In Tabelle 6.9 stehen die Ergebnisse für die bereits in den vorangegangenen Abschnitten verwendeten Daten, d. h. für die unverzerrten Szenen, bei denen dennoch in der Optimierung eine Korrektur des Parameters κ erfolgte. Die vollständige Tabelle findet man in Anhang D.4.

Die Rechenzeit pro Iteration stieg bei Korrektur von κ von ca. 45 Sekunden auf ca. 70 Sekunden.

Zu beachten ist, dass Linsenverzerrungen erst in der Optimierung berücksichtigt werden; das vorher laufende lineare Verfahren, das die initiale Rekonstruktion liefert, geht von der Annahme aus, dass keine Verzerrungen vorhanden sind. Damit werden die Startwerte und der Rückprojektionsfehler mit zunehmender Verzerrung schlechter.
Die Ergebnisse auf den verzerrten Bildern (Tabellen 6.7 und 6.8) sehen wie folgt aus:

- Die Parameter κ_i in den einzelnen Bildern werden praktisch immer sehr gut geschätzt, was sich in einem großen Rückgang des mittleren quadratischen Fehlers in diesem Parameter in fast allen Versuchen äußert. In nur zwei Fällen nimmt der Fehler nicht ab sondern steigt stattdessen an (Szene 7, $\kappa = \pm 0.01$ mm^{-2}). Im überwiegenden Teil sinkt der Fehler auf einen Wert nahe bei null.
- Bei geringen Verzerrungen ($\kappa = \pm 0.01$ mm^{-2}) macht es keinen großen Unterschied, ob man diese mitoptimiert oder nicht: Die Rückprojektionsfehler unterscheiden sich praktisch nicht (erst in der dritten Nachkommastelle) und die Veränderungen der Fehler in den einzelnen Parametern sind manchmal mit und manchmal ohne Optimierung größer. Allerdings dauert die Optimierung mit Verzerrungskorrektur wesentlich länger.
- Bei starken Verzerrungen ab $|\kappa| = 0.1$ mm^{-2} bemerkt man große Unterschiede zwischen den beiden Verfahren. Zunächst ist im Vergleich zu den geringen Verzerrungen ein starker

[11] Optimiert wurden natürlich trotzdem die κ_i der einzelnen Aufnahmen unabhängig voneinander.

κ		κ	f_x	f_y	u_0	v_0	$\boldsymbol{t}$	$\boldsymbol{R}$	3-D	Rückproj.	$\hat{\sigma}$
0.01	vor	0.01	2.38	2.24	8.79	9.06	0.748	0.00310	0.232	0.447/0.482	
	ohne	0	0.842	1.23	-4.72	-5.50	0.252	-0.00174	-0.124	0.396	0.29
	mit	-0.00293	0.508	0.893	-4.71	-6.32	0.155	-0.00191	-0.125	0.394	0.29
-0.01	vor	0.01	6.42	6.39	6.74	6.80	1.98	0.00214	0.256	0.460/0.489	
	ohne	0	-2.95	-2.79	-2.30	-3.26	-0.915	-0.000870	-0.162	0.401	0.29
	mit	-0.00402	-1.41	-1.30	-3.86	-5.15	-0.489	-0.00128	-0.159	0.398	0.29
0.1	vor	0.1	6.98	7.32	35.3	41.4	1.27	0.0133	0.508	0.714/0.779	
	ohne	0	0.537	0.132	-5.27	-15.0	0.919	-0.00326	-0.302	0.504	0.37
	mit	-0.0909	-4.78	-5.23	-30.6	-37.4	-0.641	-0.0117	-0.399	0.405	0.29
-0.1	vor	0.1	12.8	13.0	39.3	46.2	2.56	0.0149	0.620	0.797/1.28	
	ohne	0	1.40	1.20	-9.88	-20.8	0.736	-0.00467	-0.388	0.549	0.40
	mit	-0.0902	-8.82	-9.46	-37.0	-43.3	-1.42	-0.0139	-0.470	0.399	0.29
0.2	vor	0.2	18.4	19.7	66.2	78.0	5.87	0.0250	0.972	1.16/1.23	
	ohne	0	-2.96	-3.50	-7.98	-26.6	-0.427	-0.00557	-0.568	0.708	0.51
	mit	-0.191	-6.56	-7.72	-42.2	-63.7	-2.45	-0.0184	-0.727	0.416	0.30
-0.2	vor	0.2	30.8	32.6	77.5	86.5	3.19	0.0295	1.19	1.47/3.44	
	ohne	0	9.37	6.25	-20.4	-38.1	5.92	-0.00893	-0.664	0.934	0.68
	mit	-0.191	-22.6	-24.1	-71.9	-81.9	-0.528	-0.0277	-0.869	0.390	0.28

Tabelle 6.7: Szene 1: Korrektur von Linsenverzerrungen, $\sigma = 0.3$ Pixel.

κ		κ	f_x	f_y	u_0	v_0	$\boldsymbol{t}$	$\boldsymbol{R}$	3-D	Rückproj.	$\hat{\sigma}$
0.01	vor	0.01	9.87	10.4	8.90	15.2	3.36	0.00441	0.394	0.431/0.471	
	ohne	0	-3.59	-3.78	-4.65	-8.14	-1.27	-0.00259	-0.229	0.401	0.29
	mit	0.00963	-3.00	-3.18	-3.80	-5.72	-1.09	-0.00187	-0.224	0.399	0.29
-0.01	vor	0.01	18.3	19.3	8.97	12.3	6.60	0.00368	0.412	0.40/0.470	
	ohne	0	-4.49	-5.67	-2.66	-3.75	-1.85	-0.00101	-0.174	0.395	0.29
	mit	0.00163	-4.51	-5.73	-3.76	-6.30	-1.86	-0.00158	-0.173	0.394	0.29
0.1	vor	0.1	12.6	11.0	44.0	31.4	4.35	0.0135	0.717	0.595/0.667	
	ohne	0	-3.93	-2.29	-16.5	-7.02	-1.56	-0.00464	-0.440	0.457	0.33
	mit	-0.0847	-3.00	-1.21	-30.8	-21.1	-1.15	-0.00963	-0.506	0.407	0.30
-0.1	vor	0.1	19.9	19.5	45.0	35.4	7.89	0.0139	0.587	0.575/0.844	
	ohne	0	-14.2	-13.9	-19.2	-12.9	-5.79	-0.00552	-0.343	0.464	0.34
	mit	-0.0807	-7.61	-6.96	-34.4	-27.5	-3.63	-0.0108	-0.351	0.394	0.29
0.2	vor	0.2	56.8	54.0	76.6	79.5	17.8	0.0283	1.29	0.785/0.901	
	ohne	0	-8.71	-5.85	-14.2	-16.4	-2.67	-0.00589	-0.386	0.589	0.43
	mit	-0.185	-4.63	-1.69	2.29	7.44	-0.778	0.00121	-0.512	0.425	0.31
-0.2	vor	0.2	57.8	58.3	109	84.4	24.8	0.0327	1.76	0.929/3.04	
	ohne	0	-37.0	-37.9	-54.1	-36.6	-16.7	-0.0151	-1.01	0.673	0.49
	mit	-0.144	-4.57	-5.03	-72.5	-57.6	-5.81	-0.0226	-0.806	0.469	0.34

Tabelle 6.8: Szene 7: Korrektur von Linsenverzerrungen, $\sigma = 0.3$ Pixel.

Anstieg des Rückprojektionsfehlers vor der Optimierung zu beobachten. Ohne Korrektur sinkt dieser durch den Bündelausgleich auch nicht sehr weit ab. Die Linsenverzerrungen wirken hier wie ein Rauschen auf den Bildpunkten mit höherem σ als dem tatsächlich verwendeten von 0.3 Pixel, was man auch an dem Schätzwert $\hat{\sigma}$ nach der Optimierung sieht. Mit Verzerrungskorrektur hingegen sinkt der Rückprojektionsfehler genauso weit ab wie bei Verwendung von unverzerrten Bildern. Auch der Schätzwert $\hat{\sigma}$ entspricht wieder dem des tatsächlich verwendeten Rauschens. In der überwiegenden Zahl der Versuche schlägt

Szene		κ	f_x	f_y	u_0	v_0	$\boldsymbol{t}$	$\boldsymbol{R}$	3-D	Rückproj.
1	vor	0	8.60	8.13	9.96	4.37	2.21	0.00269973	0.243482	0.45/0.49
	ohne	0	-4.09	-3.28	-7.69	-2.74	-0.883	-0.00201266	-0.14373	0.400
	mit	0.00656	-3.53	-2.74	-7.16	-2.35	-0.744	-0.00176292	-0.141145	0.400
7	vor	0	14.7	15.5	12.7	9.67	4.99	0.00387035	0.320034	0.45/0.48
	ohne	0	-3.27	-4.05	-7.08	-4.73	-1.13	-0.00201028	-0.145408	0.395
	mit	0.0198	-3.26	-3.97	-6.51	-4.62	-1.08	-0.00183659	-0.141613	0.393
8	vor	0	37.8	37.9	31.1	33.2	13.4	0.010902	0.622307	0.43/0.78
	ohne	0	-5.81	-6.36	-14.2	-12.0	-2.58	-0.00454095	-0.204071	0.406
	mit	0.0156	-6.52	-7.13	-14.4	-12.1	-2.77	-0.0047505	-0.205862	0.405

Tabelle 6.9: Szenen 1, 7 und 8: Korrektur von Linsenverzerrungen, obwohl keine vorliegen, $\sigma = 0.3$ Pixel.

sich dies auch auf die Fehler in den einzelnen Parametern nieder.

Der Rückprojektionsfehler für Szene 1 in Abhängigkeit von κ ist in Bild 6.14(a) dargestellt, derjenige für Szene 7 in Bild 6.14(b). Man sieht deutlich, wie der Fehler mit Korrektur auch bei starken Verzerrungen immer auf in etwa den gleichen Wert absinkt, während dies ohne Korrektur nicht der Fall ist. Ebenfalls zu sehen ist die Zunahme des Rückprojektionsfehler, den das lineare Verfahren liefert.

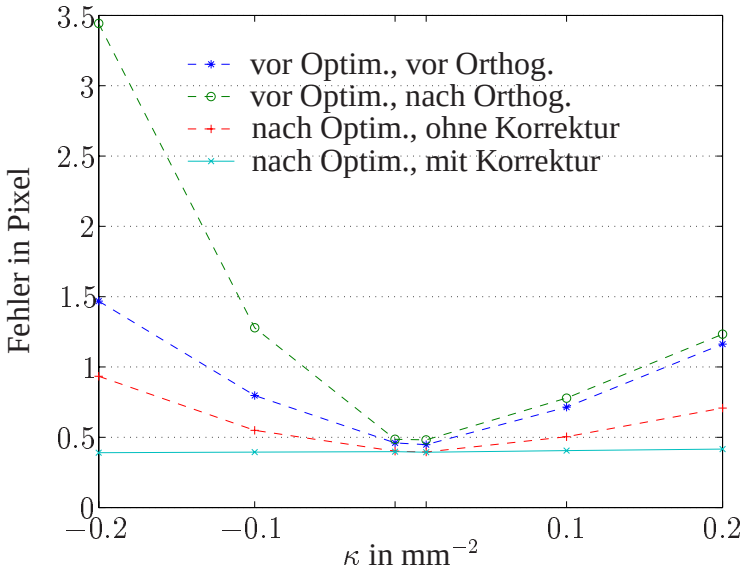
In den Versuchen mit denjenigen Szenen, die aus unverzerrten Bildern bestehen, bei denen die Korrektur aber während der Optimierung dennoch durchgeführt wurde, ergibt sich folgendes Bild (vergleiche Tabelle 6.9):

- Der Rückprojektionsfehler ist mit und ohne Korrektur in etwa gleich groß, mit einem leichten Vorteil für den Fall, bei dem die Verzerrungen korrigiert werden. Allerdings zeigen sich Unterschiede erst in der dritten Nachkommastelle.

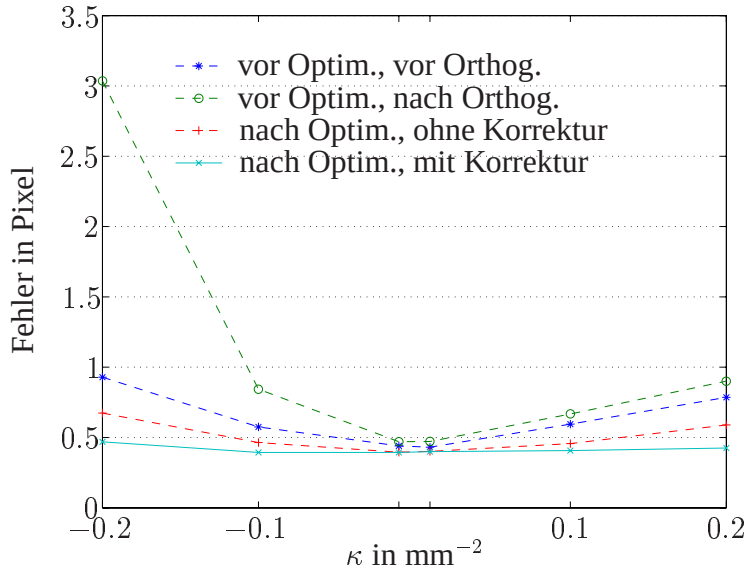
- Die Veränderungen des Fehlers in den einzelnen Parametern sind in den meisten Fällen ohne die Korrektur besser. Der Fehler in den κ_i kann hier natürlich nur schlechter werden, da in allen Bildern $\kappa_i = 0$ mm^{-2} gilt und dies auch vom linearen Verfahren so angenommen wird. Der Fehler vor der Optimierung ist also immer null. Zu beobachten ist, dass in einigen Fällen relativ große Fehler in den κ_i auftreten (z. B. Szene 7): Der Fehler entspricht nach der Optimierung in etwa dem bei einer tatsächlichen Verzerrung von $\kappa_i = 0.02$ mm^{-2}.

Fazit: Insgesamt gesehen funktioniert die Korrektur sehr gut. In *allen* betrachteten Fällen ist der Rückprojektionsfehler bei Mitoptimierung der Linsenverzerrungen genauso groß oder geringer als ohne, selbst dann, wenn gar keine Verzerrungen vorliegen. Auch die Fehler in den einzelnen Parametern werden mit Korrektur kleiner als ohne, wenn tatsächlich verzerrte Bilder vorliegen. Nur wenn möglichst gute Kameraparameter benötigt werden und keine Verzerrungen vorhanden sind, sollte die Korrektur unterbleiben. Zu bedenken ist jedoch der Anstieg in der benötigten Rechenzeit, welcher gerade bei realen Szenen[12] sehr groß ist.

[12] siehe Abschnitt 6.3.



(a) Szene 1



(b) Szene 7

Bild 6.14: Der Rückprojektionsfehler für die Szenen 1 (a) und 7 (b) in Abhängigkeit von der Stärke der Verzerrung. Ohne Korrektur steigt der Fehler bei starken Verzerrungen an, während er mit Korrektur auf gleichem Niveau bleibt.

6.2.5 Verhalten unabhängig von der vorangegangenen Rekonstruktion

Bisher wurden nur Simulationen betrachtet, bei denen die Startwerte für den Bündelausgleich aus dem vorher laufenden linearen Verfahren zur 3-D-Rekonstruktion stammen. In der praktischen Anwendung wird dies in der einen oder anderen Form auch immer der Fall sein. Aller-

dings besteht damit immer eine Abhängigkeit vom vorher verwendeten Verfahren: Je nachdem, ob dieses gute oder schlechte Startwerte liefert, wird auch der Bündelausgleich mehr oder weniger gut gelingen. Es soll nun versucht werden, die Leistungsfähigkeit des Bündelausgleichs unabhängig davon zu beurteilen, was allerdings sehr schwierig ist, da bereits eine geringe Veränderung der Kameraparameter den Rückprojektionsfehler stark ansteigen lässt. Insbesondere soll in den beiden folgenden Abschnitten betrachtet werden,

- wie sich der Bündelausgleich verhält, wenn als Startwerte verrauschte Kameraparameter und 3-D-Punkte verwendet werden und *keine* projektive Verzerrung vorliegt und,
- wie groß der Selbstkalibrierungseffekt des Bündelausgleichs ist.

Dazu wurden die gleichen acht Szenen wie vorher verwendet, wobei an Stelle der Rekonstruktion durch das lineare Verfahren die tatsächlichen Projektionsmatrizen und 3-D-Punkte benutzt wurden. Wie diese für die jeweilige Untersuchung verändert wurden, wird im zugehörigen Abschnitt erläutert.

Die für eine bestimmte Szene verwendeten Bildpunkte waren in allen Experimenten gleich, auf ihnen lag ein normalverteiltes Rauschen mit $\sigma = 0.3$ Pixel. Wie vorher wurden zur Parametrisierung der Rotation Quaternionen benutzt und die Optimierung nach 20 Iterationen abgebrochen.

Verrauschte Parameter

Für die im Folgenden dargestellten Ergebnisse wurden die tatsächlichen Projektionsmatrizen und 3-D-Punkte der jeweiligen Szene hergenommen und die einzelnen Kameraparameter und Koordinaten der 3-D-Punkte verrauscht. Eine projektive Verzerrung der Szene war *nicht* vorhanden, weshalb in den Tabellen beim Rückprojektionsfehler vor der Optimierung immer nur ein Wert angegeben ist; im Gegensatz zu den vorher durchgeführten Simulationen ändert sich der Fehler durch die Orthogonalisierung der Rotationsmatrix nicht.

Für die Veränderung der Kameraparameter wurden Zufallszahlen aus einer Normalverteilung mit Standardabweichung σ_P verwendet, welche gewichtet und anschließend zum jeweiligen Parameter addiert wurden. Die Gewichtung ist notwendig, da die Kameraparameter in sehr unterschiedlichen Größenordnungen liegen (z. B. ca. 1000 Pixel für die Brennweiten und im Bereich von -1 bis $+1$ für die Elemente der Quaternionen). Als Gewichtungsfaktor wurde 1% des Parameterwerts gewählt, mit Ausnahme von u_0 und v_0, welche beide gleich null waren. Hier und bei den Koordinaten der 3-D-Punkte wurden die generierten Zufallszahlen ohne Gewichtung addiert.

Für σ_P wurden die Werte $0, 0.1, 0.2, 0.3, 0.6$ und 1.0 benutzt. Die vollständigen Ergebnisse der Simulationen befinden sich in Anhang D.5.1. Daraus wurde folgende Auswahl getroffen: In Tabelle 6.10 sind die Fehler nach der Optimierung für alle Szenen zusammengestellt, wenn man $\sigma_P = 0$ wählt. Das bedeutet, die Optimierung wurde mit den richtigen Werten gestartet, eine vorherige Veränderung der Kameraparameter oder der 3-D-Punkte erfolgte *nicht*. Der angegebene Rückprojektionsfehler vor der Optimierung entsteht somit ausschließlich durch das Rauschen auf den Bildpunkten. Wenn während der Optimierung überhaupt etwas verändert wird, so nur vom richtigen Wert weg, der Fehler kann folglich nur ansteigen. Man kann beobachten, dass der Rückprojektionsfehler durch den Bündelausgleich in allen acht Szenen noch um ca. 0.02 Pixel sinkt und dass sich alle Parameter von den richtigen Werten weg bewegen.

Szene		f_x	f_y	u_0	v_0	$\boldsymbol{t}$	$\boldsymbol{R}$	3-D	Rückproj.	$\hat{\sigma}$
1	vor	0	0	0	0	0	0	0	0.428	
	And.	1.69	1.745	0.498	0.509	0.47257	0.000237298	0.0874587	0.402	0.29
2	vor	0	0	0	0	0	0	0	0.416	
	And.	1.49	1.51	0.395	0.339	0.380442	0.000175896	0.0821289	0.392	0.28
3	vor	0	0	0	0	0	0	0	0.428	
	And.	1.76	1.90	0.297	0.309	0.538416	0.000145087	0.162121	0.404	0.29
4	vor	0	0	0	0	0	0	0	0.422	
	And.	1.81	1.84	0.432	0.341	0.526303	0.000180275	0.144702	0.400	0.29
5	vor	0	0	0	0	0	0	0	0.433	
	And.	0.343	0.398	0.154	0.205	0.178931	0.000132436	0.179012	0.409	0.30
6	vor	0	0	0	0	0	0	0	0.423	
	And.	0.618	0.531	0.190	0.199	0.171784	0.000147863	0.164417	0.400	0.29
7	vor	0	0	0	0	0	0	0	0.422	
	And.	0.903	0.780	0.485	0.404	0.252635	0.000203842	0.0983554	0.397	0.29
8	vor	0	0	0	0	0	0	0	0.426	
	And.	0.765	0.661	0.292	0.347	0.242724	0.000179056	0.106385	0.402	0.29

Tabelle 6.10: Ergebnisse der Optimierung bei Verwendung der tatsächlichen Projektionsmatrizen und 3-D-Punkte als Startwerte für den Bündelausgleich.

Daraus kann der Schluss gezogen werden, dass der Bündelausgleich einen Bias hat, die Schätzwerte der Parameter nach der Optimierung sind also verzerrt und liegen etwas von den wahren Werten entfernt. Dafür, dass dies trotz der durchgeführten erwartungstreuen Maximum-Likelihood-Schätzung der Fall ist, kommen mehrere Gründe in Frage:

- Zunächst entspricht der Bündelausgleich nur dann einer Maximum-Likelihood-Schätzung, wenn das Rauschen auf den Bildpunkten normalverteilt, mittelwertfrei und isotrop ist. Es ist somit als Erstes zu prüfen, ob diese Bedingungen überhaupt erfüllt sind, was bei einer echten Bildfolge nicht unbedingt der Fall sein muss. Da das Rauschen hier mittels eines Zufallszahlengenerators simuliert wurde, sollte das an dieser Stelle jedoch keine Probleme bereiten.

- Maximum-Likelihood-Schätzung ist nur *in erster Näherung* erwartungstreu [Kan96], ein geringer Bias wird daher immer vorhanden sein.

- Auch bei einem erwartungstreuen Schätzer werden die wahren Werte nur asymptotisch erreicht, wenn die Anzahl der Daten, also der Bildpunkte, gegen Unendlich geht.

In diesem Zusammenhang sind auch die beiden anderen Tabellen in diesem Abschnitt interessant. Dort sind die Ergebnisse für die Szenen 1 (rotatorisch, Tabelle 6.11) und 5 (translatorisch, Tabelle 6.12), jeweils mit verschieden stark veränderten Parametern, abgedruckt.

Abweichend von allen anderen Tabellen in dieser Arbeit wurden hier nicht nur die Veränderungen der Fehler nach der Optimierung angegeben, sondern auch deren Wert, da man so leichter erkennen kann, wohin das Verfahren konvergiert ist. Je nach Größe von σ_P stieg der Rückprojektionsfehler an, wobei der Anstieg von einer Verdopplung bei $\sigma_P = 0.1$ bis zum über 30ig-fachen bei $\sigma_P = 1$ reicht[13]. Dabei fällt zunächst auf: Wie groß der Rückprojektionsfehler auch wurde, er sank durch die Optimierung immer wieder auf Werte um 0.4 Pixel ab.

[13] siehe dazu z. B. auch Szene 4 ($\sigma_P = 1$) in Tabelle D.39 im Anhang, wo der Fehler auf 15.8 Pixel anstieg.

σ_P		f_x	f_y	u_0	v_0	$\boldsymbol{t}$	$\boldsymbol{R}$	3-D	Rückproj.
0.1	vor	0.860	1.13	0.00112	0.00114	0.254753	0.000262038	0.164103	1.19
	nach	1.63	1.79	0.309	0.369	0.476807	0.000181166	0.0860962	0.402
	Änd.	0.771	0.660	0.307	0.368	0.222054	-8.08726e-05	-0.0780065	-0.788
0.2	vor	2.42	2.30	0.00177	0.00201	0.793596	0.000653836	0.344356	2.03
	nach	1.74	2.02	0.463	0.489	0.529377	0.000230122	0.0891356	0.402
	Änd.	-0.678	-0.283	0.461	0.487	-0.264219	-0.000423714	-0.25522	-1.628
0.3	vor	3.64	2.43	0.00320	0.00258	0.800516	0.00117352	0.553162	5.00
	nach	1.71	2.04	0.440	0.485	0.53662	0.000250402	0.0894364	0.402
	Änd.	-1.93	-0.392	0.436	0.483	-0.263896	-0.00092312	-0.463726	-4.598
0.6	vor	6.08	6.79	0.00437	0.00654	1.77068	0.00220986	1.00501	7.84
	nach	1.52	1.53	0.648	0.596	0.454354	0.000354132	0.114113	0.403
	Änd.	-4.56	-5.26	0.644	0.589	-1.31633	-0.00185573	-0.890896	-7.437
1.0	vor	11.6	7.87	0.0101	0.00746	3.18943	0.00283297	1.74166	12.9
	nach	2.19	2.53	0.692	0.515	0.702098	0.000305696	0.100138	0.402
	Änd.	-9.41	-5.34	0.682	0.507	-2.48733	-0.00252727	-1.64152	-12.498

Tabelle 6.11: Szene 1: Verrauschte Parameter.

σ_P		f_x	f_y	u_0	v_0	$\boldsymbol{t}$	$\boldsymbol{R}$	3-D	Rückproj.
0.1	vor	0.849	0.967	0.00103	0.000895	0.294971	0.000346911	0.164988	0.886
	nach	1.14	1.09	0.164	0.177	0.29751	0.000155466	0.175277	0.409
	Änd.	0.293	0.122	0.163	0.176	0.00253881	-0.000191445	0.0102889	-0.477
0.2	vor	2.26	2.19	0.00193	0.00211	0.833876	0.000442003	0.33494	1.43
	nach	2.78	2.85	0.348	0.184	0.763251	0.000245865	0.180276	0.410
	Änd.	0.520	0.665	0.346	0.182	-0.0706243	-0.000196138	-0.154664	-1.02
0.3	vor	3.55	3.02	0.00210	0.00249	0.915256	0.000997179	0.4983	1.83
	nach	2.65	2.57	0.448	0.289	0.72574	0.000217564	0.185666	0.409
	And.	-0.900	-0.450	0.446	0.286	-0.189516	-0.000779615	-0.312634	-1.421
0.6	vor	5.67	7.48	0.00536	0.00731	1.97023	0.0017121	1.05039	3.55
	nach	5.06	5.01	0.816	0.673	1.6311	0.00041829	0.285539	0.419
	Änd.	-0.607	-2.47	0.811	0.666	-0.339127	-0.00129381	-0.764849	-3.131
1.0	vor	7.54	14.0	0.00844	0.00899	3.13748	0.0030505	1.74894	7.13
	nach	7.96	7.89	0.763	0.705	2.08865	0.000428611	0.202812	0.437
	Änd.	0.415	-6.14	0.754	0.696	-1.04883	-0.00262189	-1.54612	-6.693

Tabelle 6.12: Szene 5: Verrauschte Parameter.

Bei den Fehlern in den Parametern zeigt sich folgendes Verhalten:

- Bei nur kleiner Veränderung der Parameter, also kleinem σ_P, werden die einzelnen Parameter zum überwiegenden Teil durch die Optimierung nicht besser, sondern schlechter.
- Erst bei größerem σ_P wird in den meisten Parametern eine Verbesserung erreicht, welche manchmal relativ groß, manchmal aber auch nur sehr gering ausfällt.
- Eine Ausnahme sind u_0 und v_0. Diese liegen immer recht nahe am wahren Wert, eine Abnahme des Fehlers in diesen beiden Parametern konnte niemals beobachtet werden.
- Der Fehler in der Rotation sowie in den 3-D-Punkten fällt außer bei $\sigma_P = 0$ immer, mit einer einzigen Ausnahme in Szene 5.

Szene		f_x	f_y	u_0	v_0	$\boldsymbol{t}$	$\boldsymbol{R}$	3-D	Rückproj.	$\hat{\sigma}$
1	vor	41.3	38.5	46.6	85.0	11.6	0.0234	1.12	0.428/1.87	
	And.	18.0	16.0	-11.8	-0.770	5.46	-0.00251	0.0124	0.937	0.68
2	vor	25.7	24.7	36.1	34.1	6.80	0.0117	0.612	0.416/0.773	
	And.	-11.0	-9.83	-15.4	-14.9	-2.78	-0.00505	-0.255	0.445	0.32
3	vor	44.6	44.4	34.9	33.1	13.7	0.0114	0.704	0.428/0.943	
	And.	-6.29	-5.97	-6.26	-10.0	-2.07	-0.00320	-0.0203	0.487	0.35
4	vor	36.7	37.3	98.8	62.7	12.9	0.0281	1.27	0.422/2.21	
	And.	36.6	34.8	-0.763	-2.68	10.8	-0.00344	0.241	0.875	0.63
5	vor	61.0	58.8	101	43.6	18.0	0.0261	1.33	0.433/2.41	
	And.	-0.105	1.05	-0.272	-4.76	2.96	0.00706	1.97	0.903	0.66
6	vor	85.6	73.5	118	72.5	16.1	0.0373	1.90	0.423/3.12	
	And.	-6.11	-0.122	-0.413	2.56	1.15	0.00252	0.974	1.05	0.76
7	vor	57.0	62.8	55.3	85.4	21.3	0.0246	1.07	0.422/1.23	
	And.	0.563	-2.99	-16.0	-37.7	-0.114	-0.00951	-0.292	0.530	0.38
8	vor	35.1	34.3	64.3	34.7	14.0	0.0185	0.703	0.426/0.860	
	And.	0.0914	0.394	-28.4	-5.97	0.0460	-0.00693	-0.235	0.499	0.36

Tabelle 6.13: Selbstkalibrierungseffekt beim Bündelausgleich.

Selbstkalibrierung

Bereits in den vorangegangenen Abschnitten zeigte sich, dass der Bündelausgleich auch als Selbstkalibrierung bzw. als Optimierung derselben aufgefasst werden kann. Eine misslungene lineare Selbstkalibrierung äußerte sich in einem mehr oder weniger starken Anstieg des Rückprojektionsfehlers durch die Orthogonalisierung der Rotationsmatrix. Es wurde nun versucht, eine schlechte Selbstkalibrierung kontrolliert zu erzeugen.

Als Ausgangspunkt dienten wiederum die korrekten Projektionsmatrizen und 3-D-Punkte. Diese wurden mit Hilfe einer zufällig gewählten projektiven Transformation (leicht) verzerrt, anschließend wurde der Bündelausgleich gestartet. Man hat somit eine Ausgangssituation, in der eine *exakte* projektive Rekonstruktion vorliegt. Durch den Bündelausgleich soll diese hin zu einer euklidischen optimiert werden, welche mit den tatsächlichen Verhältnissen möglichst gut übereinstimmt.

Dies bedeutet für die folgenden Tabellen: Der angegebene Rückprojektionsfehler vor der Optimierung und vor der Orthogonalisierung entsteht einzig aus dem Rauschen auf den Bildpunkten. Der Anstieg des Fehlers durch die Orthogonalisierung ist wiederum ein Maß für den Grad der projektiven Verzerrung. Jede der acht Szenen wurde fünfmal durchgerechnet, jedesmal mit einer anderen projektiven Transformation. Eine Auswahl aus den Ergebnissen ist in Tabelle 6.13 abgedruckt, die vollständigen Daten befinden sich Anhang D.5.2.

Die projektive Transformation wurde erzeugt, indem zu jedem Element der 4×4 Einheitsmatrix eine Zufallszahl aus einer Normalverteilung unter Verwendung einer Standardabweichung von 10^{-4} addiert wurde. Trotz dieser nur geringen Änderung wurden die Fehler in den Parametern relativ groß.

Übrigens: Wendet man das normalerweise vorher laufende lineare Verfahren zur Selbstkalibrierung auf die projektiv verzerrte Rekonstruktion an, so liefert dieses hier einen Fehler in den einzelnen Parametern, die im Bereich von 10^{-2} liegen. Dass dies so gut funktioniert liegt daran, dass bereits eine exakte projektive Rekonstruktion vorliegt und das Verfahren bei der Bestimmung der benötigten Transformation nicht mit den verrauschten Bildpunkten arbeitet, sondern

(a) Szene 1: Karten

(b) Szene 2: Tuch

Bild 6.15: Die jeweils ersten Bilder der Aufnahmen der echten Szenen.

ausschließlich mit den Projektionsmatrizen.

Zu den Ergebnissen ist zu sagen, dass der Anstieg des Rückprojektionsfehlers durch die Orthogonalisierung in Rahmen dessen liegt, was auch bei den vorherigen Simulationen zu beobachten war, bei denen das lineare Verfahren die Startwerte lieferte. Im Gegensatz zum vorherigen Abschnitt, wo der Rückprojektionsfehler durch die Optimierung wieder auf die beim vorhandenen Rauschen zu erwartende Größe (um 0.4 Pixel) sank, war das bei den hier vorgestellten Versuchen zur Selbstkalibrierung nicht der Fall. In vielen Fällen war der Fehler nach der Optimierung noch ca. doppelt so groß.

Die Veränderung des Fehlers in den einzelnen Parametern ist schwierig zu beurteilen: Es kann alles beobachtet werden, von einer großen Abnahme bis zu einer großen Zunahme des Fehlers oder auch nur sehr geringe Veränderungen.

Insgesamt gesehen lässt sich nur sagen, dass das Verfahren manchmal die Parameter in die richtige Richtung verändert und manchmal nicht. Der einzige Vorteil, den der Bündelausgleich im Bezug auf die Selbstkalibrierung zu bieten scheint ist, dass er aus der näherungsweisen euklidischen Rekonstruktion des vorhergehenden linearen Verfahrens eine exakt euklidische machen kann. Wenn die vorhergehende Selbstkalibrierung allerdings relativ schlecht ist, so kann auch der Bündelausgleich daran nicht mehr viel verändern.

6.3 Experimente mit realen Daten

Dieser Abschnitt beschreibt die Ergebnisse bei Anwendung des Bündelausgleichs auf echte Bildfolgen. Die wahren Werte der zu schätzenden Parameter waren dabei nicht bekannt, daher kann hier nur die Veränderung des Rückprojektionsfehlers während der Optimierung betrachtet werden. Für den Anwendungsfall Lichtfeldrekonstruktion ist dies aber der entscheidende Wert. Jede Szene wurde zweimal durchgerechnet, jeweils mit 10 Iterationen: Einmal ohne Korrektur von Linsenverzerrungen und einmal mit Korrektur. Hierbei ist anzumerken, dass sichtbare Verzerrungen nicht auftreten; die Ergebnisse der Simulation zeigen jedoch, dass eine Korrektur der Linsenverzerrungen zumindest nicht schadet, wenn sie auch die Rechenzeit beeinflusst.
Die folgenden beiden Szenen wurden verwendet:

#Iterationen	Karten		Tuch	
	ohne	mit	ohne	mit
vorher (vor Orthog.)	0.4692580	0.4692580	0.3745770	0.3745770
vorher (nach Orthog.)	0.5546684	0.5546684	0.4771462	0.4771462
1	0.4276170	0.4278811	0.3449298	0.3438074
2	0.4224856	0.4222467	0.3409849	0.3395271
3	0.4205521	0.4201426	0.3393406	0.3379993
4	0.4198513	0.4193886	0.3385425	0.3373337
5	0.4191159	0.4191081	0.3382206	0.3370689
6	0.4189686	0.4185567	0.3381399	0.3369981
7	0.4188352	0.4183285	0.3379784	0.3368725
8	0.4185365	0.4179466	0.3378770	0.3368650
9	0.4183919	0.4177388	0.3378590	0.3368466
10	0.4182988	0.4177162	0.3378509	0.3368279

Tabelle 6.14: Echte Szenen: Veränderung des Rückprojektionsfehlers in Abhängigkeit von der Anzahl der Iterationen. Der erste Wert der Tabelle („vorher (vor Orthog.)“) gibt den Fehler vor der Optimierung und *vor* der Orthogonalisierung der Rotationsmatrix an, der zweite Wert („vorher (nach Orthog.)“) den Fehler vor der Optimierung und *nach* der Orthogonalisierung der Rotationsmatrix. Die Spalten „ohne“ bzw. „mit“ geben die Fehler ohne bzw. mit Korrektur von Linsenverzerrungen an.

Szene 1: Karten. Besteht aus 1429 3-D-Punkten und 31 Aufnahmen, siehe auch Bild 6.15(a).

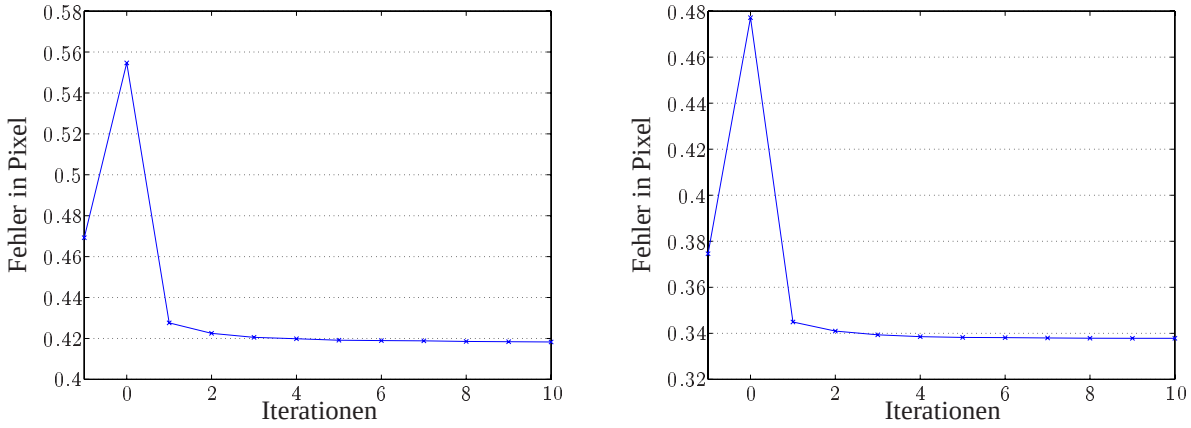
Szene 2: Tuch. Besteht aus 905 3-D-Punkten und 50 Aufnahmen, siehe auch Bild 6.15(b).

Alle 3-D-Punkte waren in allen Aufnahmen sichtbar, Ausreißer in den Punktkorrespondenzen wurden vor der Berechnung automatisch entfernt.

Die Ergebnisse der Berechnungen sind in Tabelle 6.14 zusammengestellt. Zu beobachten ist, dass der Rückprojektionsfehler durch die Orthogonalisierung der Rotationsmatrix in beiden Szenen zunächst um ca. 0.1 Pixel ansteigt, wobei der Fehler in der Tuch-Szene von Anfang an geringer ist als in der Karten-Szene. In beiden Fällen sinkt der Fehler bereits nach der ersten Iteration wieder unter den Wert vor der Orthogonalisierung. Bereits nach wenigen Iterationen verändert er sich kaum noch: In der Karten-Szene bleibt die zweite Nachkommastelle in beiden Varianten schon nach vier Iterationen stabil, in der Tuch-Szene sogar noch früher, nämlich nach zwei (mit Verzerrungskorrektur) bzw. drei (ohne Verzerrungskorrektur) Iterationen. Eine graphische Darstellung des Rückprojektionsfehlers in Abhängigkeit von der Anzahl der durchgeführten Iterationen ist in Bild 6.16 zu sehen. In Bild 6.17 sieht man eine Darstellung der rekonstruierten Szenen und Kamerapositionen.

Bild 6.18 zeigt die Positionen der Kameras in der Tuch-Szene vor und nach der Optimierung. Deutlich erkennbar ist der glattere Verlauf nach der Optimierung. Dies zeigt sich besonders im linken und rechten Drittel des Bildes, wo vor dem Bündelausgleich eine große Veränderung der Kameraposition von einer Aufnahme zur nächsten zu sehen ist.

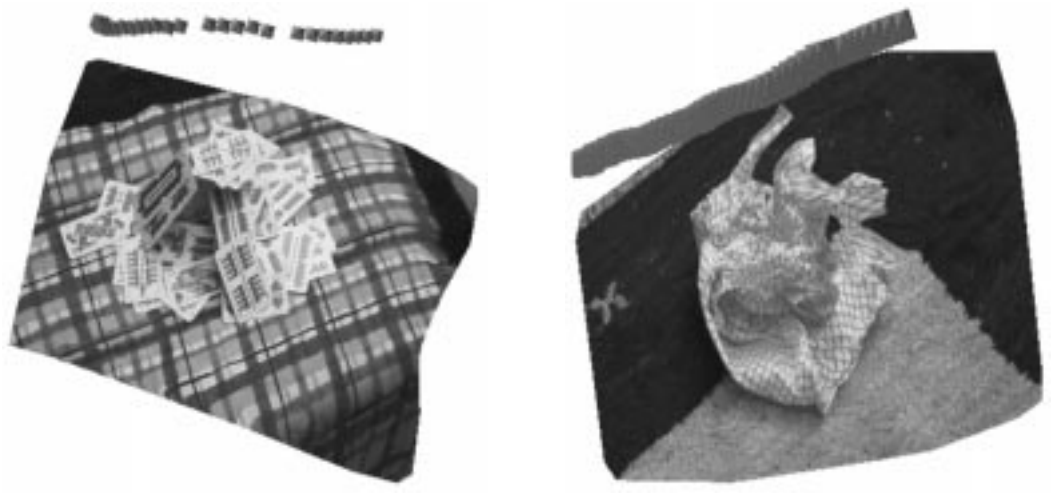
Bezüglich der Korrektur von Linsenverzerrungen bestätigt sich das Ergebnis der Simulation: Selbst wenn praktisch keine vorhanden sind, wird der Rückprojektionsfehler mit Verzerrungskorrektur etwas kleiner als ohne. Allerdings wirkt sich dies erst in der dritten Nachkommastelle aus: Bei der Karten-Szene beträgt der Unterschied nach 10 Iterationen 0.0006 Pixel, bei der Tuch-Szene 0.001 Pixel. Dies ist vernachlässigbar klein, insbesondere wenn man die Unterschiede in den Rechenzeiten bei den beiden Varianten betrachtet.



(a) Rückprojektionsfehler: Karten

(b) Rückprojektionsfehler: Tuch

Bild 6.16: Veränderung des Rückprojektionsfehlers der echten Szenen in Abhängigkeit von der Anzahl der Iterationen. Bereits nach wenigen Iterationen verändert sich der Fehler kaum noch. Der Wert bei „-1“ gibt den Fehler vor der Optimierung und *vor* der Orthogonalisierung der Rotationsmatrix an, der Wert bei „0“ den Fehler vor der Optimierung und *nach* der Orthogonalisierung der Rotationsmatrix. Dies ist der Startwert für den Bündelausgleich. Dargestellt sind die Fehler bei der Optimierung ohne Korrektur von Linsenverzerrungen.



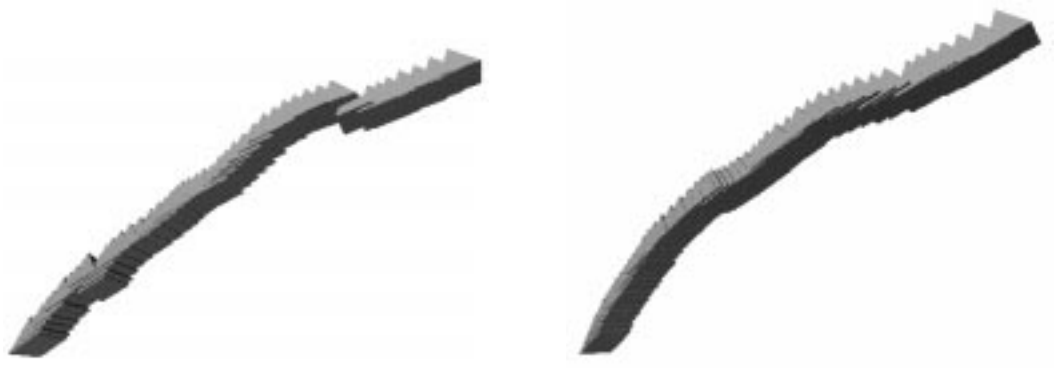
(a) Rekonstruktion Szene 1: Karten

(b) Rekonstruktion Szene 2: Tuch

Bild 6.17: Rekonstruktionen der beiden realen Szenen mit Kamerapositionen.

Bei der Karten-Szene beträgt die Rechenzeit für 10 Iterationen ohne Verzerrungskorrektur ca. 3 Stunden, also etwa 18 Minuten für eine Levenberg-Marquardt-Iteration. Mit Verzerrungskorrektur kommt man auf eine Gesamtzeit von 8 Stunden, das sind ca. 48 Minuten pro Iteration.

Die Berechnung der Tuch-Szene benötigt ohne Verzerrungskorrektur ca. 5 Stunden (30 Minuten pro Iteration), mit Korrektur dagegen 12 Stunden und 45 Minuten (ca. 76 Minuten pro



(a) vor der Optimierung (b) nach der Optimierung

Bild 6.18: Kamerapositionen der Tuch-Szene vor (a) und nach (b) der Optimierung.

Iteration). Alle Zeitangaben beziehen sich auf einen PC mit Intel Celeron-Prozessor mit 366 MHz.

Die Schätzwerte für die Standardabweichung des angenommenen normalverteilten Rauschens auf den Bildpunkten betragen $\hat{\sigma}_K = 0.30$ Pixel für die Karten- und $\hat{\sigma}_T = 0.24$ Pixel für die Tuch-Szene.

Kapitel 7

Zusammenfassung und Ausblick

In der vorliegenden Arbeit wurde ein nichtlineares Verfahren zur Minimierung des Rückprojektionsfehlers bei der Rekonstruktion von Kameraparametern und 3-D-Punkten aus Bildfolgen betrachtet. Die Minimierung des Rückprojektionsfehlers wird auch als Bündelausgleich bezeichnet. Dieses Optimierungskriterium wurde gewählt, da es damit möglich ist, alle Aufnahmen und alle 3-D-Punkte der Szene zusammen zu optimieren. Es wird also immer die maximal vorhandene Information verwendet und das Verfahren wird mit zunehmender Anzahl an Daten stabiler. Zudem entspricht die Minimierung des Rückprojektionsfehlers einer Maximum-Likelihood-Schätzung der Parameter, wenn man von Bildpunkten ausgeht, auf denen ein normalverteiltes, mittelwertfreies und isotropes Rauschen liegt. Damit ist der Bündelausgleich in einem statistischen Sinne optimal, denn die Kovarianzmatrix der zu schätzenden Parameter erreicht die Cramér-Rao-Schranke, d. h. kleinere Kovarianzen können nicht erreicht werden.

Für die Optimierung muss eine geeignete Parametrisierung der Projektionsmatrizen gefunden werden, da man bei direkter Verwendung der Matrixelemente eine projektive und keine euklidische Rekonstruktion erhalten würde. Für alle Parameter außer der Rotation wird dabei eine einfache additive Veränderung während der Optimierung verwendet. Bei der Rotation gibt es mehrere Alternativen, wobei Eulerwinkel nicht weiter betrachtet wurden, da diese keine gute Parametrisierung für Rotationen sind. Üblicherweise wird die Darstellung als Rotation um eine Achse mit einem bestimmten Winkel gewählt. Diese ist einfach zu handhaben, da man genau drei Parameter für die drei Freiheitsgrade der Rotation hat. Eine weitere Möglichkeit zur Darstellung von Rotationen im Raum sind die Quaternionen. Diese werden in der Literatur im Zusammenhang mit dem Bündelausgleich praktisch nicht verwendet, obwohl sie sich in vielen Bereichen als sehr erfolgreich erwiesen haben. Ein Grund hierfür könnte sein, dass ein Quaternion vier Elemente aber nur drei Freiheitsgrade hat, wodurch eine einfache additive Veränderung dieser vier Elemente nicht sehr Erfolg versprechend ist. Daher wurde in dieser Arbeit ein neuer Ansatz vorgestellt, welcher es erlaubt, die Änderungen in einem Quaternion mit nur drei Parametern so durchzuführen, dass auch bei beliebig großen Veränderungen aus einem Quaternion mit Norm eins wieder ein solches entsteht. Wie sich zeigt, ist die so gewonnene minimale Parametrisierung einfach zu handhaben und sie erfordert in der Praxis auch keinen wesentlich höheren Implementierungsaufwand als die Verwendung der Achse/Winkel-Darstellung. Simulationen ergaben, dass sich die Ergebnisse bei den beiden Varianten nicht erheblich unterscheiden. Insbesondere hat die verwendete Darstellung der Rotation keinen Einfluss auf den für das Lichtfeld wichtigen Rückprojektionsfehler. Betrachtet man nur den Fehler in der Rotation nach der Optimierung, so zeigt sich, dass Quaternionen hier oft besser sind als die Achse/Winkel-

Darstellung, wobei die Unterschiede aber meist recht gering ausfallen. Bezüglich der übrigen Parameter kann keine eindeutige Empfehlung für oder gegen die Quaternionen gegeben werden.

Als zu optimierende Funktion wurde an Stelle der gleichzeitigen Optimierung von Struktur und Kameraparametern der eingebettete Bündelausgleich verwendet, welcher aus dem ursprünglichen Kriterium durch geschickte Umformulierung des Problems hervorgeht. Dadurch ergibt sich neben einer Reduktion der Dimension des Suchraums auch eine wesentlich geringere Zeitkomplexität für die Optimierung.

Als lokales nichtlineares Optimierungsverfahren wurde der Gauß-Newton-Algorithmus gewählt, wegen der größeren numerischen Stabilität zusammen mit der Levenberg-Marquardt-Erweiterung. Die bei diesem Verfahren auftretenden Jacobi-Matrizen der ersten Ableitungen der Fehlerfunktion haben beim vorliegenden Problem eine einfache Blockstruktur, die eine weitere Ersparnis an Rechenzeit zur Folge hat. Beim eingebetteten Bündelausgleich erhält man eine Reduktion von $O(nm^3)$ auf $O(nm)$, wobei n die Anzahl der 3-D-Punkte und m die Anzahl der Aufnahmen ist. Für die Behandlung von Verdeckungen bzw. 3-D-Punkten, die nicht über die gesamte Sequenz hinweg verfolgt werden können, wurde das Gauß-Newton-Verfahren so angepasst, dass es mit variablen Blockgrößen in der Jacobi-Matrix arbeiten kann.

Weiterhin wurden zwei Erweiterungen des Bündelausgleichs vorgestellt: Die unterschiedliche Gewichtung des Rückprojektionsfehlers bei den einzelnen Bildpunkten sowie die Berücksichtigung von Linsenverzerrungen. Die Gewichtung des Fehlers ist nützlich, da beim zur Ermittlung von Punktkorrespondenzen verwendeten Verfahren die Genauigkeit bei der Lokalisation der Punkte vom Anfang der Bildfolge zum Ende hin abnimmt. Wie die genaue Gewichtung aussehen sollte, hängt vom Verhalten dieses Verfahrens ab; da darüber noch keine Daten vorlagen, wurde auf eine nähere Betrachtung der Fehlergewichtung in dieser Arbeit verzichtet. Dies sollte ein Gegenstand zukünftiger Untersuchungen sein. Insbesondere dann, wenn der Anstieg des Fehlers in der Lokalisation vom Anfang zum Ende der Sequenz hin groß ist, sollte sich durch die Gewichtung eine bessere Bildqualität beim Lichtfeld erreichen lassen.

Die Korrektur von Linsenverzerrungen wird in der Literatur im Zusammenhang mit dem Bündelausgleich recht selten betrachtet. Aussagen darüber, wie gut die damit erzielbaren Ergebnisse sind, lagen nicht vor. Für die Korrektur wurde in jeder Aufnahme ein zusätzlicher Parameter eingeführt, mit dem radiale Verzerrungen in erster Näherung modelliert werden können. Dies sollte auch für die Anwendung auf echte Bildfolgen genügen. In den Simulationen zeigte sich, dass die Korrektur beim Bündelausgleich sehr gut funktioniert, obwohl das vorhergehende lineare Verfahren zur Rekonstruktion (und auch die Selbstkalibrierung) von unverzerrten Bildern ausgeht. Die Korrektur kommt somit allein durch den Bündelausgleich zustande. Der einzige gravierende Nachteil besteht darin, dass sich die Rechenzeit gegenüber der Optimierung ohne Verzerrungen mehr als verdoppelt. Hier sind weitere Experimente, insbesondere mit realen Szenen, nötig, um zu entscheiden, wie stark die Verzerrungen bei einer echten Kamera tatsächlich sind und ob sich der zusätzliche Zeitaufwand für die Korrektur lohnt.

Da ein nichtlineares lokales Optimierungsverfahren eingesetzt wird, benötigt man Startwerte für die zu schätzenden Parameter. Diese stammen aus einem linearen Faktorisierungsalgorithmus, welcher eine projektive Rekonstruktion liefert. Im Anschluss daran wird ein ebenfalls linearer Algorithmus zur Selbstkalibrierung angewandt, welcher aus der projektiven eine euklidische Rekonstruktion macht; diese wird zusätzlich durch eine nichtlineare Optimierung verbessert. Je nach der Qualität der Daten gelingt die Selbstkalibrierung mehr oder weniger gut, eine geringe projektive Verzerrung ist immer vorhanden. Bei der Berechnung der Startwerte aus den Projektionsmatrizen wird eine exakt euklidische Rekonstruktion erzwungen, indem die Ma-

trizen unter der Annahme eines rechtwinkligen Koordinatensystems in ihre Bestandteile zerlegt werden. Die so entstehenden Rotationsmatrizen sind wegen der nur näherungsweisen euklidischen Rekonstruktion zunächst gar keine Rotationsmatrizen, d. h. sie sind im Allgemeinen nicht orthogonal. Daher werden sie unter Verwendung der Singulärwertzerlegung in die nächstliegenden „echten“ Rotationsmatrizen übergeführt. Die Folge dieses Schrittes ist ein Anstieg im Rückprojektionsfehler, der durchaus relativ groß ausfallen kann, bei den realen Bildfolgen z. B. betrug er ca. 25%. Diese Zunahme des Rückprojektionsfehlers wird durch den Bündelausgleich jedoch wieder korrigiert, meist ist der Fehler bereits nach dem ersten Iterationsschritt auf einen Wert unterhalb desjenigen vor der Orthogonalisierung gesunken. Dass die Rekonstruktion während der gesamten Optimierung euklidisch bleibt, ist durch die gewählte Parametrisierung gewährleistet.

Die Zunahme des Fehlers durch die Orthogonalisierung kann als grobes Maß für die Anzahl der nötigen Iterationen verwendet werden, d. h. bei einem geringen Anstieg genügen weniger Iterationen, bis eine gute Konvergenz der Parameter erreicht ist. Nach den Simulationen zu urteilen sollten dafür 10–20 Stück ausreichend sein, wobei sich der Rückprojektionsfehler bei zunehmender Zahl an Iterationen nicht mehr so stark verändert wie die einzelnen Parameter. Auch bei höherem Rauschen auf den Bildpunkten sollten mehr Iterationen durchgeführt werden, als grobes Maß für die Standardabweichung des Rauschens kann der Rückprojektionsfehler vor der Orthogonalisierung dienen.

Die untersuchten echten Bildfolgen legen den Schluss nahe, dass bei Verwendung von mehr Punkten und mehr Aufnahmen als in den Simulationen das Verfahren schneller konvergiert. Insbesondere der Rückprojektionsfehler war schon nach ca. 5 Iterationen relativ stabil, für praktische Anwendungen dürften also 5–10 Iterationen durchaus genügen. Interessant wären hier weitere Experimente mit echten oder künstlich erzeugten Bildfolgen, bei denen die wahren Werte der zu schätzenden Parameter bekannt sind. Will man die Simulationen als Richtwerte für das Verhalten bei echten Bildfolgen verwenden, so sollte man diejenigen betrachten, bei denen ein Rauschen mit $\sigma = 0.3$ Pixel auf den Bildpunkten lag. Die Schätzwerte für σ bei den echten Szenen lagen nämlich etwa in diesem Bereich.

Insgesamt gesehen liefert der Bündelausgleich gute Ergebnisse: Die Abnahme des Rückprojektionsfehlers betrug meist ca. 10%, bezogen auf den Wert *vor* der Orthogonalisierung der Rotationsmatrizen. Die Abnahme bezogen auf den Wert nach der Orthogonalisierung ist – je nach Anstieg des Fehlers – teilweise wesentlich größer. Wenn es, wie beim Lichtfeld, in erster Linie auf den Rückprojektionsfehler ankommt, sollten Vergleiche allerdings mit dem erstgenannten Wert durchgeführt werden, da man diesen Fehler bereits vor der Optimierung vorliegen hat. Die Abnahme des Fehlers in der Rotation betrug meist über 50%; bei den 3-D-Punkten liegt der Fall ähnlich, insbesondere wenn die Startwerte nicht zu schlecht sind. Bei der Translation und den intrinsischen Kameraparametern ist das Verhalten uneinheitlich. In einigen Fällen konnte auch eine Zunahme des Fehlers bei einzelnen Parametern beobachtet werden.

Die Nachteile des Bündelausgleichs ergeben sich in erster Linie daraus, dass er eine lokales nichtlineares Optimierungsverfahren ist. Hier ist zunächst die benötigte Rechenzeit zu nennen, welche selbst für kleine Sequenzen sehr hoch ist. Weiterhin werden möglichst gute Startwerte benötigt. Bei schlechten Startwerten, welche sich oft aus einer nicht gelungenen Selbstkalibrierung ergeben, erhält man zwar Projektionsmatrizen, die die für eine euklidische Rekonstruktion notwendigen Bedingungen (rechtwinklige Bildkoordinatensysteme und orthogonale Rotationsmatrizen) erfüllen, der Fehler in den Parametern ist jedoch auch nach der Optimierung noch relativ groß. Das Verfahren läuft eben immer ins nächste lokale Minimum. Die Ergebnisse

der Auswertungen unabhängig von den vorhergehenden linearen Methoden zur Rekonstruktion bzw. zur Selbstkalibrierung und insbesondere deren Übertragbarkeit auf die praktische Anwendung sind schwierig zu bewerten. Festgestellt werden konnte aber ein Bias, d. h. die beim Bündelausgleich entstehenden Schätzwerte für die Parameter liegen immer etwas von den wahren Werten entfernt. Hier ist ein Ansatzpunkt für weitere Arbeiten; es sollte beispielsweise untersucht werden, in wie weit sich der Bias bei Verwendung größerer Datenmengen verändert und wie er korrigiert werden könnte.

In den Simulationen wurden die Fehler in den Parametern durch Ermittlung der optimalen Ähnlichkeitstransformation berechnet. Es bietet sich an, weitere Vergleiche nach Anwendung der optimalen projektiven Transformation durchzuführen, um festzustellen, wie stark die projektiven Verzerrungen nach der Optimierung noch sind.

Literaturverzeichnis

[AP95] Azarbayejani, A., Pentland, A. *Recursive Estimation of Motion, Structure, and Focal Length*. In *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Bd. 17(6):S. 562–575, 1995 19, A

[DS83] Dennis, J., Schnabel, R. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice Hall, Englewood Cliffs, NJ, 1983 7, 4.3, 4.3.2, 4.3.3

[Fau93] Faugeras, O. *Three-Dimensional Computer Vision: A Geometric Viewpoint*. MIT Press, 1993 3, 5, 17, 18, A, A

[Fua99] Fua, P. *Using Model-Driven Bundle-Adjustment to Model Heads from Raw Video Sequences*. In *Proceedings of the 7th International Conference on Computer Vision (ICCV)*, Bd. 1, S. 46–53. IEEE Computer Society Press, Corfu, Sept. 1999 7

[FZ98] Fitzgibbon, A. W., Zisserman, A. *Automatic Camera Recovery for Closed or Open Image Sequences*. In *Proc. European Conference on Computer Vision*, S. 311–326. Springer-Verlag, Juni 1998 7

[GGSC96] Gortler, S., Grzeszczuk, R., Szeliski, R., Cohen, M. *The Lumigraph*. In *Computer Graphics (Proceedings of Siggraph 96)*, S. 43–54. Addison-Wesley, New Orleans, Louisiana, Aug. 1996 1.1, 1

[HÅ97] Heyden, A., Åström, K. *Euclidean Reconstruction from Image Sequences with Varying and Unknown Focal Length and Principal Point*. In *Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR '97)*, S. 438–443. Puerto Rico, Juni 1997 7, 1, 9, 4.2, 17, 4.3, 4.3.1, 4.3.2, 4.3.3, 6.2

[HÅ99] Heyden, A., Åström, K. *Flexible Calibration: Minimal Cases for Auto-calibration*. In *Proceedings of the 7th International Conference on Computer Vision (ICCV)*, Bd. 1, S. 350–355. IEEE Computer Society Press, Corfu, Sept. 1999 7, 17, 4.3

[Har94] Hartley, R. *Euclidian Reconstruction from Uncalibrated Views*, Bd. 825 von *Lecture notes in Computer Science*, S. 237–256. Springer-Verlag, 1994 7, 1, 4.2.2, 4.3, 4.3.1, 4.3.2, 4.3.3, 4.3.4, 5.1, 32

[Har97a] Hartley, R. *In Defense of the Eight-Point Algorithm*. In *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Bd. 19(6):S. 580–593, 1997 5

[Har97b] Hartley, R. *Lines and Points in Three Views and the Trifocal Tensor*. In *International Journal of Computer Vision*, Bd. 22(2):S. 125–140, 1997 5

[Har98] Hartley, R. *Minimizing Algebraic Error in Geometric Estimation Problems*. In *Proceedings of the International Conference on Computer Vision (ICCV)*, S. 469–476. Bombay, 1998 7, 3.1

[HKP+99] Heigl, B., Koch, R., Pollefeys, M., Denzler, J., Van Gool, L. *Plenoptic Modeling and Rendering from Image Sequences Taken by a Hand-Held Camera*. In *Mustererkennung 1999*, Förstner, W., Buhmann, J., Gaber, A., Faber, P., Hg., S. 94–101. Springer-Verlag, Heidelberg, Sept. 1999 1.1, 1, 2

[HN99] Heigl, B., Niemann, H. *Camera Calibration from Extended Image Sequences for Lightfield Reconstruction*. In *Proceedings of Vision, Modeling, and Visualization*, S. 43–50. Erlangen, Germany, Nov. 1999 5, 1.3, 2.3, 4.1, 4.2, 12, 4.4, 35, 6.2

[HPN99] Hornegger, J., Paulus, D., Niemann, H. *Probabilistic Modeling in Computer Vision*, Bd. 2 von *Handbook of Computer Vision and Applications*, Kap. 26, S. 817–854. Academic Press, 1999 14

[HT99] Hornegger, J., Tomasi, C. *Representation Issues in the ML Estimation of Camera Motion*. In *Proceedings of the 7th International Conference on Computer Vision (ICCV)*, S. 640–647. IEEE Computer Society Press, Corfu, Sept. 1999 7, 17, 17, 18

[Kan93] Kanatani, K. *Geometric Computation for Machine Vision*, Bd. 37 von *Oxford Engineering Series*. Oxford University Press, Oxford, 1993 7, 2.3

[Kan96] Kanatani, K. *Statistical Optimization for Geometric Computation: Theory and Practice*, Bd. 18 von *Machine Intelligence and Pattern Recognition*. Elsevier Science, Amsterdam, 1996 7, 3.2.1, 3.2.1, 3.2.1, 3.2.2, 9, 6.2.5

[KDRS98] Köthe, U., Diener, H., Ristow, B., Schreyer, M. *Acquisition, Reconstruction and Simulation of Real Objects*. IEEE Industrial Electronics Society, Aachen, Germany, 1998 5.2.1

[KM98] Kanade, T., Morris, D. *Factorization Methods for Structure from Motion*. In *Phil. Trans. R. Soc. Lond. A*, Bd. 356:S. 1153–1173, 1998 5

[Kre91] Kreyszig, E. *Statistische Methoden und ihre Anwendung*. Vandenhoeck und Ruprecht, Göttingen, 7 Aufl., 1991 3.2.2

[LF97] Luong, Q.-T., Faugeras, O. *Camera Calibration, Scene Motion and Structure recovery from point correspondences and fundamental matrices*. In *International Journal of Computer Vision*, Bd. 22(3):S. 261–289, 1997 5

[LH81] Longuet-Higgins, H. *A Computer Algorithm for Reconstructing a Scene from Two Projections*. In *Nature*, Bd. 293:S. 133–135, 1981 3

[LH96] Levoy, M., Hanrahan, P. *Light Field Rendering*. In *Computer Graphics (Proceedings of Siggraph 96)*, S. 31–42. Addison-Wesley, New Orleans, Louisiana, Aug. 1996 1.1, 1

[McL99] McLauchlan, P. *Gauge Invariance in Projective 3D Reconstruction*. In *Proceedings of the IEEE Workshop on Multi-View Modeling & Analysis of Visual Scenes*. Fort Collins, Colorado, Juni 1999 7, 4.3, 5.1, 6.2

[MT96] Mohr, R., Triggs, B. *Projective Geometry for Image Analysis*. In *Int. Symp. Photogrammetry and Remote Sensing*. Juli 1996. Tutorial 2.3

[Nie90] Niemann, H. *Pattern Analysis and Understanding*, Bd. 4 von *Series in Information Sciences*. Springer–Verlag, Berlin, Heidelberg, 1990 3, 5.2.1

[PK94] Poelman, C., Kanade, T. *A Paraperspective Factorization Method for Shape and Motion Recovery*. In *Proceedings 3rd European Conference on Computer Vision*, Bd. 2, S. 97–108. Stockholm, Sweden, 1994 5

[PKV98] Pollefeys, M., Koch, R., Van Gool, L. *Self-Calibration and Metric Reconstruction in spite of Varying and Unknown Internal Camera Parameters*. In *Proceedings of the International Conference on Computer Vision (ICCV)*, S. 90–95. Bombay, 1998 6, 2.3, 2.3, 2.3, 2.3, 1

[Pol99] Pollefeys, M. *Self-Calibration and Metric 3D Reconstruction from Uncalibrated Image Sequences*. Dissertation, Katholieke Universiteit Leuven – Faculteit Toegepaste Wetenschappen, 1999 2.3, 2.3, 2.3, 5.2.1

[PTVF92] Press, W., Teukolsky, S., Vetterling, W., Flannery, B. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 2 Aufl., 1992 7, 4.3, 4.3.3, 15, 48, 2

[Rao73] Rao, C. *Linear Statistical Inference and Its Applications*. Wiley, New York, 2 Aufl., 1973 3.2.1

[Ris99] Ristow, B. *Modelling Real World Objects Combining Photogrammetric Reconstruction and Semantic Description*. In *Proceedings of Vision, Modeling, and Visualization*, S. 101–108. Erlangen, Germany, Nov. 1999 7

[Sch93] Schwarz, H. *Numerische Mathematik*. Teubner-Verlag, Stuttgart, 1993 7, 4.3, 4.3.2

[SK93] Szeliski, R., Kang, S. *Recovering 3D Shape and Motion from Image Streams using Non-Linear Least Squares*. In *Proceedings of the Computer Vision and Pattern Recognition (CVPR)*, S. 752–753. Los Alamitos, California, Juni 1993 17

[SK97] Szeliski, R., Kang, S. *Shape Ambiguities in Structure From Motion*. In *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Bd. 19(5):S. 506–512, 1997 5.1, 32

[SKZ99] Shum, H.-Y., Ke, Q., Zhang, Z. *Efficient Bundle Adjustment with Virtual Key Frames: A Hierarchical Approach to Multi-Frame Structure from Motion*. In *Proceedings of the Computer Vision and Pattern Recognition (CVPR)*, Bd. 2, S. 538–543. Fort Collins, Colorado, Juni 1999 7, 1, 17, 4.3.1, 22

[Sla80] Slama, C. (Hg.). *Manual of Photogrammetry*. American Society of Photogrammetry, Falls Church, VA, 4 Aufl., 1980 7, 4.1, 4.3.4, 5.1, 34, 34

[ST94] Shi, J., Tomasi, C. *Good Features to Track.* In *Proceedings of Computer Vision and Pattern Recognition*, S. 593–600. IEEE Computer Society Press, Seattle, Washington, Juni 1994 3, 5.1

[ST96] Sturm, P., Triggs, B. *A Factorization based Algorithm for Multiple-Image Projective Structure from Motion.* In *Proceedings European Conference on Computer Vision*, S. 709–720. Springer-Verlag, 1996 5

[ST98] Szeliski, R., Torr, P. *Geometrically Constrained Structure from Motion: Points on Planes*. In *European Workshop on 3D Structure from Multiple Images of Large-Scale Environments (SMILE)*, S. 171–186. Freiburg, Germany, Juni 1998 7, 17, 4.3.1

[TD97] Trefethen, L., D. Bau, III. *Numerical Linear Algebra*. SIAM, 1997 7, 26, 48, 49

[TK92] Tomasi, C., Kanade, T. *Shape and Motion from Image Streams under Orthography: A Factorization Method.* In *International Journal of Computer Vision*, Bd. 9(2):S. 137–153, 1992 5

[Tsa87] Tsai, R. *A versatile Camera Calibration Technique for High-Accuracy 3D Machine Vision Metrology Using off-the-shelf TV Cameras and Lenses.* In *IEEE Journal on Robotics and Automation*, Bd. 3(4):S. 323–344, Aug. 1987 34

[TV98] Trucco, E., Verri, A. *Introductory Techniques for 3-D Computer Vision.* Addison-Wesley, 1998 3, 5, 2.2, 2.3, 4.2.2, 17, 5.2.1, 34, 34, 5.2.3, 37, A, C, C, C

[WAH93] Weng, J., Ahuja, N., Huang, T. *Optimal Motion and Structure Estimation.* In *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Bd. 15(9):S. 864–884, 1993 7, 3.2.1, 3.2.1, 3.2.2, 9, 12, 5.1

[WW92] Watt, A., Watt, M. *Advanced Animation and Rendering Techniques*. Addison-Wesley, 1992 18

[XZ96] Xu, G., Zhang, Z. *Epipolar Geometry in Stereo, Motion and Object Recognition, A Unified Approach*. Kluwer Academic Publishers, 1996 2.2

[Zha96] Zhang, Z. *On the Epipolar Geometry Between Two Images With Lens Distortion.* In *Proceedings International Conference Pattern Recognition (ICPR)*, S. 407–411. Wien, Aug. 1996 5.2.1, 34, 34, 34, 34, 34, 5.2.4, 5.2.4

[Zha98a] Zhang, Z. *Determining the Epipolar Geometry and its Uncertainty: A Review*. In *International Journal of Computer Vision*, Bd. 27(2):S. 161–195, 1998 5, 4.3.1

[Zha98b] Zhang, Z. *On the Optimization Criteria Used in Two-View Motion Analysis.* In *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Bd. 20(7):S. 717–729, 1998 7, 3.1, 3.1

Anhang A

Quaternionen

Dieser Anhang bezieht sich auf die Parametrisierung der Rotation mit Quaternionen in Abschnitt 4.2.2. Zunächst soll erläutert werden, was Quaternionen eigentlich sind – nämlich Zahlen. Ähnlich wie man von den reellen zu den komplexen Zahlen kommt, kommt man von den komplexen Zahlen zu den Quaternionen. Und genau, wie beim Übergang auf die komplexen Zahlen eine Eigenschaft verloren geht, nämlich die Anordnung, verliert man bei den Quaternionen das Kommutativgesetz der Multiplikation.

Definition. Ein Quaternion $\boldsymbol{h}$ ist wie folgt definiert [Fau93]:

$$\boldsymbol{h} = w + xi + yj + zk \qquad\qquad w, x, y, z \in \mathbb{R} \quad . \tag{A.1}$$

An Stelle von nur einer imaginären Einheit i bei den komplexen Zahlen, hat man bei den Quaternionen deren drei: i, j, k. Multiplikation und Addition werden ähnlich wie bei den komplexen Zahlen einfach komponentenweise ausgeführt, wobei gilt

$$\begin{aligned} i^2 = j^2 = k^2 = -1 \quad &, \\ ij = -ji = k \quad &, \\ jk = -kj = i \quad &, \\ ki = -ik = j \quad &. \end{aligned} \tag{A.2}$$

Oft wird abkürzend die Tupelschreibweise gewählt: $\boldsymbol{h} = (w, x, y, z)$ oder auch $\boldsymbol{h} = (w, \boldsymbol{v})$ wobei in $\boldsymbol{v}$ die Imaginärteile zusammengefasst sind.
Hier die Rechenregeln für Quaternionen:

1. Die Assoziativgesetze sind gültig,
2. das Kommutativgesetz bzgl. der Multiplikation gilt *nicht*: $\boldsymbol{h}_1\boldsymbol{h}_2 \neq \boldsymbol{h}_2\boldsymbol{h}_1 \Rightarrow$ Quaternionen sind kein Körper mehr,
3. Konjugierte: $\overline{\boldsymbol{h}} = w - xi - yj - zk$,
4. Norm: $|\boldsymbol{h}| = \sqrt{\boldsymbol{h} \cdot \overline{\boldsymbol{h}}} = \sqrt{w^2 + x^2 + y^2 + z^2}$,
5. Einheitsquaternionen: $|\boldsymbol{h}| = 1 \Rightarrow \boldsymbol{h}^{-1} = \overline{\boldsymbol{h}}$.

Darstellung von Rotationen. Kommen wir nun zu der hier vorliegenden Anwendung von Quaternionen, der Darstellung von Rotationen im Raum. Auch hier lässt sich eine Analogie zu den komplexen Zahlen herstellen, da diese Drehstreckungen in der Ebene definieren. Verwendet man nur komplexe Zahlen mit Norm eins, so erhält man Drehungen in der Ebene. Genauso definieren Einheitsquaternionen Rotationen im dreidimensionalen Raum:
Es seien

- $\boldsymbol{p}$ ein zu rotierender 3-D-Punkt,
- $\boldsymbol{a}$ die Rotationsachse, $|\boldsymbol{a}| = 1$,
- Θ der Rotationswinkel.

Definiere:

- Quaternion $\boldsymbol{h} = \left(\cos\frac{\Theta}{2}, \sin\frac{\Theta}{2} \cdot \boldsymbol{a}\right)$,
- Quaternion $\boldsymbol{p}' = (0, \boldsymbol{p})$.

Dann gilt:

$$\boldsymbol{p}'_{rot} = \boldsymbol{h} \cdot \boldsymbol{p}' \cdot \overline{\boldsymbol{h}} \quad , \tag{A.3}$$

wobei $\boldsymbol{p}'_{rot}$ das Quaternion ist, welches dem rotierten Punkt entspricht.

Zu beachten ist, dass die Darstellung einer Rotation als Quaternion nicht eindeutig ist, d. h. die beiden Quaternionen $\boldsymbol{h}_1 = \left(\cos\frac{\Theta}{2}, \sin\frac{\Theta}{2} \cdot \boldsymbol{a}\right)$ und $\boldsymbol{h}_2 = \left(-\cos\frac{\Theta}{2}, -\sin\frac{\Theta}{2} \cdot \boldsymbol{a}\right)$ beschreiben die gleiche Rotation. Dies ergibt sich direkt aus der Achse/Winkel-Darstellung, da eine Rotation um die Achse $\boldsymbol{a}$ mit dem Winkel Θ das Gleiche ist, wie eine Rotation um die Achse $-\boldsymbol{a}$ mit dem Winkel $2\pi - \Theta$. Welches der beiden Quaternionen für die Darstellung der Rotation verwendet wird, ist für die Berechnungen egal. Aufpassen muss man allerdings, wenn man den Abstand zwischen zwei Quaternionen berechnen will, wie dies für die Ermittlung des Fehlers in der Rotation in Kapitel 6 geschieht.

Umrechnung Quaternion – Rotationsmatrix. Bei Verwendung von Quaternionen zur Repräsentation von Rotationen benötigt man des Öfteren die Umrechnung von einem Quaternion zu einer Rotationsmatrix und umgekehrt. Zuerst der Weg vom Quaternion $\boldsymbol{h} = (h_0, h_1, h_2, h_3)$ zur Matrix $\boldsymbol{R}$ [Fau93, AP95]:

$$\boldsymbol{R} = \begin{pmatrix} h_0^2 + h_1^2 - h_2^2 - h_3^2 & 2(h_1h_2 - h_0h_3) & 2(h_1h_3 + h_0h_2) \\ 2(h_1h_2 + h_0h_3) & h_0^2 - h_1^2 + h_2^2 - h_3^2 & 2(h_2h_3 - h_0h_1) \\ 2(h_1h_3 - h_0h_2) & 2(h_2h_3 + h_0h_1) & h_0^2 - h_1^2 - h_2^2 + h_3^2 \end{pmatrix} \quad . \tag{A.4}$$

Die Berechnung eines Quaternions aus einer Rotationsmatrix funktioniert wie folgt [TV98]:

1. Berechnung von Rotationsachse und -winkel aus der Rotationsmatrix $\boldsymbol{R}$.
   Diese erhält man über die Eigenwerte und -vektoren von $\boldsymbol{R}$: $\boldsymbol{R}$ hat die drei Eigenwerte 1 und $\cos\theta \pm i\sin\theta$. Die Rotationsachse entspricht dem zum Eigenwert 1 gehörenden Eigenvektor $\boldsymbol{a}$, welcher für die Quaternionendarstellung auf Länge eins normiert werden muss. Der Rotationswinkel θ lässt sich dann aus einem der beiden verbleibenden Eigenwerte ermitteln. Zu beachten ist, dass die Richtung der Achse, für die man von vorneherein zwei Möglichkeiten hat, konsistent zum Winkel gewählt werden muss. Die Konsistenz der beiden Werte kann durch Einsetzen in die Rodrigues-Formel (4.13) überprüft werden.

2. Sind Rotationsachse $\boldsymbol{a}$ (mit $|\boldsymbol{a}| = 1$) und Rotationswinkel θ gegeben, so erhält man das dazugehörende Quaternion durch

$$\boldsymbol{h} = \left(\cos\frac{\theta}{2},\ \sin\frac{\theta}{2} \cdot \boldsymbol{a}\right) \quad . \tag{A.5}$$

Anhang B

Singulärwertzerlegung

Die hier beschriebene Singulärwertzerlegung wird im Verlauf der Arbeit an verschiedenen Stellen verwendet: Zur Bildung der (Pseudo-)Inversen einer Matrix (Abschnitt 4.3), zur Orthogonalisierung von Rotationsmatrizen (Abschnitt 6.1) oder zur Berechnung einer orthogonalen Basis für die bei der Quaternionenparametrisierung zu bestimmende Tangentialebene (Abschnitt 4.2.2).

Die Singulärwertzerlegung[1] [TD97, PTVF92] ist ein Verfahren der numerischen linearen Algebra, welches sehr vielseitig einsetzbar ist, so beispielsweise zum

- Lösen von überbestimmten Gleichungssystemen,
- Sicherstellen von Rangkriterien für Matrizen,
- Erzwingen der Orthogonalität einer Matrix,
- Berechnen der Pseudoinversen einer Matrix,
- Bestimmen einer orthogonalen Basis eines Raums, wenn bereits eine beliebige Basis gegeben ist.

Zudem ist die Singulärwertzerlegung numerisch viel besser konditioniert als z. B. die Gauß-Elimination.

Auf die eigentliche Berechnung der SVD soll hier nicht eingegangen werden; Näheres findet man in der o. g. Literatur. Es sollen vielmehr nur die Eigenschaften der SVD und deren Anwendungen geschildert werden. Der übersichtlicheren Notation wegen beschränken wir uns hier auf $m \times n$-Matrizen mit $m \geq n$, welche in der vorliegenden Arbeit auch ausschließlich verwendet werden.

Die Art der Darstellung entspricht der in [TD97] als *reduzierte* SVD bezeichneten Version, gegenüber der *vollen* SVD. Die Unterschiede zwischen den beiden Varianten bestehen insbesondere in der Größe der bei der Zerlegung entstehenden Matrizen. Für Details siehe [TD97]. Auch die Version in [PTVF92] entspricht der reduzierten SVD; diese wird in der Praxis üblicherweise verwendet. Sie wird im Folgenden nur noch mit *SVD* bezeichnet.
Der zentrale Satz der Singulärwertzerlegung lautet: *Jede* $m \times n$-Matrix $\boldsymbol{A}$ lässt sich zerlegen in ein Produkt

$$\boldsymbol{A} = \boldsymbol{U}\,\boldsymbol{\Sigma}\,\boldsymbol{V}^T \quad . \tag{B.1}$$

[1] engl.: Singular Value Decomposition (SVD)

Diese Zerlegung ist nicht eindeutig. Für die entstehenden Matrizen gilt:

- $\boldsymbol{U} \in \mathbb{R}^{\mathrm{m}\times\mathrm{n}}$, die Spaltenvektoren von $\boldsymbol{U}$ sind orthonormal: $\boldsymbol{U}^T\boldsymbol{U} = \boldsymbol{I}$,
- $\boldsymbol{V} \in \mathbb{R}^{\mathrm{n}\times\mathrm{n}}$ ist quadratisch und sowohl Spalten- als auch Zeilenvektoren sind orthonormal: $\boldsymbol{V}^T\boldsymbol{V} = \boldsymbol{V}\boldsymbol{V}^T = \boldsymbol{I}$,
- $\boldsymbol{\Sigma} \in \mathbb{R}^{\mathrm{n}\times\mathrm{n}}$ ist eine quadratische Diagonalmatrix.

Für $\boldsymbol{\Sigma}$ gilt:

$$\boldsymbol{\Sigma} = \begin{pmatrix} \sigma_1 & 0 & 0 & 0 \\ 0 & \sigma_2 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & \sigma_n \end{pmatrix} \quad . \tag{B.2}$$

Die σ_i werden als *Singulärwerte* bezeichnet. Sie haben folgende Eigenschaft:

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0 \quad . \tag{B.3}$$

Generell gilt, dass Singulärwerte, welche fast null sind (also unterhalb eines Schwellwerts[2] liegen), auf exakt null gesetzt werden sollten. Dadurch erreicht man eine höhere numerische Stabilität. Die Konditionszahl C ist nämlich definiert als

$$C = \frac{\sigma_1}{\sigma_k} \quad , \tag{B.4}$$

wobei σ_k der kleinste positive (d. h. der letzte von null verschiedene) Singulärwert ist. Durch das Nullsetzen von Singulärwerten, die sowieso schon nahe bei Null liegen, erhält man damit eine Verbesserung der Konditionszahl um ganze Größenordnungen!
Nun zu den Anwendungen dieser Zerlegung:

Rang einer Matrix. Der Rang einer Matrix $\boldsymbol{A}$ entspricht der Anzahl der von null verschiedenen Singulärwerte. Dies hat folgende Konsequenzen: Bei Matrizen, die durch ein numerisches Verfahren entstanden sind und vorgegebene Rangkriterien erfüllen müssen, kann überprüft werden, ob diese auch tatsächlich erfüllt sind. Zudem kann man den Rang einer Matrix erzwingen, indem diejenigen Singulärwerte exakt gleich null gesetzt werden, die schon näherungsweise null sind. Anschließend werden die drei Matrizen wieder zusammenmultipliziert.

Nullraum einer Matrix. Der Nullraum wird aufgespannt von den Spaltenvektoren $\boldsymbol{v}_i$ von $\boldsymbol{V}$, wobei sich i aus denjenigen Singulärwerten ergibt, für die $\sigma_i = 0$ gilt.

Bildraum einer Matrix. Der Bildraum wird aufgespannt von den Spaltenvektoren $\boldsymbol{u}_i$ von $\boldsymbol{U}$, mit $\sigma_i \neq 0$.

Berechnung einer orthonormalen Basis. Gegeben sei eine beliebige Basis eines n-dimensionalen Unterraums des $\mathbb{R}^{\mathrm{m}}$, die Basisvektoren sollen in der $m \times n$ Matrix $\boldsymbol{A}$ zusammengefasst sein. Berechnet man die SVD von $\boldsymbol{A}$, so ergeben die Spaltenvektoren der Matrix $\boldsymbol{U}$ eine orthonormale Basis, die den gleichen Raum aufspannt wie $\boldsymbol{A}$.

[2] zur Wahl des Schwellwerts siehe [PTVF92].

Berechnung der Pseudoinversen. Hierbei ist wie folgt vorzugehen:

- Berechne $\boldsymbol{A} = \boldsymbol{U}\,\boldsymbol{\Sigma}\,\boldsymbol{V}^T$.
- Bilde die Matrix $\boldsymbol{\Sigma}'$, indem alle von null verschiedenen Singulärwerte durch ihren Kehrwert ersetzt werden, also $\sigma_i \rightarrow \frac{1}{\sigma_i}$.
- Berechne die Pseudoinverse durch $\boldsymbol{A}^+ = \boldsymbol{V}\,\boldsymbol{\Sigma}'\,\boldsymbol{U}^T$.

Orthogonalisierung einer Matrix Die Orthogonalisierung kann eingesetzt werden, wenn beispielsweise eine Rotationsmatrix mit einem numerischen Verfahren bestimmt wird. Man erhält dann eine Matrix, die „fast“ (numerisch) orthogonal ist, möchte aber eine exakt orthogonale Matrix. Dies kann mit Hilfe der SVD wie folgt erreicht werden:

- Sei $\boldsymbol{A} = \boldsymbol{U}\,\boldsymbol{\Sigma}\,\boldsymbol{V}^T$ numerisch orthogonal,
- dann sind alle Singulärwerte näherungsweise gleich eins. Diese werden auf eins gesetzt; dadurch entsteht an Stelle von $\boldsymbol{\Sigma}$ die $n \times n$ Einheitsmatrix.
- Berechne $\boldsymbol{A}' = \boldsymbol{U}\,\boldsymbol{V}^T$,
- $\boldsymbol{A}'$ ist dann exakt orthogonal.

$\boldsymbol{A}'$ ist die im Sinne der Frobeniusnorm nächstliegende orthogonale Matrix.

Der Aufwand zur Berechnung der SVD einer $m \times n$ Matrix mit $m \geq n$ beträgt $O(mn^2)$ [TD97].

Anhang C

Berechnung der Kameraparameter aus einer Projektionsmatrix

Hier wird erläutert, wie aus einer gegebenen Projektionsmatrix $\boldsymbol{P}$ die Kameraparameter berechnet werden können, wenn man annimmt, dass die Achsen des Bildkoordinatensystems senkrecht aufeinander stehen. Dies wird verwendet, um aus den Projektionsmatrizen, welche aus dem linearen Verfahren zur 3-D-Rekonstruktion stammen, die Startwerte für den Bündelausgleich zu berechnen; Näheres zu den dabei auftretenden Problemen wird in Abschnitt 6.1 beschrieben.

Dieser Abschnitt richtet sich im Wesentlichen nach [TV98], wobei das dort beschriebene Verfahren allerdings nicht direkt verwendet werden kann. Daher ist hier die angepasste Version erläutert. Der Unterschied besteht in der Verwendung der extrinsischen Kameraparameter. In dieser Arbeit wurde für die Projektionsmatrizen folgendes Modell zugrunde gelegt:

$$\boldsymbol{P} = \boldsymbol{K}\,\boldsymbol{R}^T(\boldsymbol{I}_3|-\boldsymbol{t}) \quad . \tag{C.1}$$

Dabei geben die Rotationsmatrix $\boldsymbol{R}$ und der Translationsvektor $\boldsymbol{t}$ die Bewegung *der Kamera* an. In [TV98] wird dagegen ein Modell verwendet, bei dem $\boldsymbol{R}$ und $\boldsymbol{t}$ die Bewegung *der 3-D-Szenenpunkte* beschreiben:

$$\boldsymbol{P} = \boldsymbol{K}\,(\boldsymbol{R}|\,\boldsymbol{t}) \quad . \tag{C.2}$$

Wie die einzelnen Kameraparameter aus der Darstellung in Formel (C.2) ermittelt werden können, steht in [TV98]. Wie dies mit Formel (C.1) geschieht, folgt nun.
Durch Ausmultiplizieren von Formel (C.1) ergibt sich

$$\boldsymbol{P} = \begin{pmatrix} f_x r_{11} + u_0 r_{13} & f_x r_{21} + u_0 r_{23} & f_x r_{31} + u_0 r_{33} & -f_x \boldsymbol{r}_{\cdot 1}^T \boldsymbol{t} - u_0 \boldsymbol{r}_{\cdot 3}^T \boldsymbol{t} \\ f_y r_{12} + v_0 r_{13} & f_y r_{22} + v_0 r_{23} & f_y r_{32} + v_0 r_{33} & -f_y \boldsymbol{r}_{\cdot 2}^T \boldsymbol{t} - v_0 \boldsymbol{r}_{\cdot 3}^T \boldsymbol{t} \\ r_{13} & r_{23} & r_{33} & -\boldsymbol{r}_{\cdot 3}^T \boldsymbol{t} \end{pmatrix} \tag{C.3}$$

mit

$$\boldsymbol{r}_{\cdot i}^T = \begin{pmatrix} r_{1i} & r_{2i} & r_{3i} \end{pmatrix} \quad . \tag{C.4}$$

Da eine Projektionsmatrix $\boldsymbol{P}$ nur bis auf einen Skalierungsfaktor γ eindeutig ist, muss dieser zunächst berechnet werden. Bezeichnet man die Elemente der (noch nicht normalisierten) Projektionsmatrix mit p_{ij},so erhält man den Skalierungsfaktor aus den ersten drei Elementen der letzten Zeile von $\boldsymbol{P}$, welche nur Elemente der Rotationsmatrix enthält. Da eine Rotationsmatrix orthonormal ist, gilt

$$\sqrt{p_{31}^2 + p_{32}^2 + p_{33}^2} = |\gamma| \sqrt{r_{13}^2 + r_{23}^2 + r_{33}^2} = |\gamma| \quad . \tag{C.5}$$

Man erhält so natürlich nur den absoluten Wert von γ, nicht dessen Vorzeichen. Dieses kann aber aus der Projektionsmatrix allein nicht bestimmt werden. Bei der vorliegenden Anwendung ist auch nur wichtig, dass die Projektionsmatrizen passend zu den 3-D-Punkten gewählt werden, da sonst die Kameras von der Szene weg blicken oder die Szene insgesamt gespiegelt ist. Darauf soll hier jedoch nicht näher eingegangen werden, da die Konsistenz bereits vor dem Bündelausgleich gewährleistet wird. Das bedeutet folglich: Hier ist nur der Betrag des Skalierungsfaktors wichtig, Vorzeichen werden keine verändert.

Von nun an sei die Projektionsmatrix $\boldsymbol{P}$ durch Division jedes Elements durch γ normalisiert, p_{ij} sei ein Element dieser normalisierten Matrix. Dann kann man folgende, im weiteren hilfreiche, Abkürzungen einführen:

$$\begin{aligned} \boldsymbol{b}_1 &= \begin{pmatrix} p_{11} & p_{12} & p_{13} \end{pmatrix}^T \quad , \\ \boldsymbol{b}_2 &= \begin{pmatrix} p_{21} & p_{22} & p_{23} \end{pmatrix}^T \quad , \\ \boldsymbol{b}_3 &= \begin{pmatrix} p_{31} & p_{32} & p_{33} \end{pmatrix}^T \quad , \\ \boldsymbol{b}_4 &= \begin{pmatrix} p_{14} & p_{24} & p_{24} \end{pmatrix}^T \quad . \end{aligned} \tag{C.6}$$

Der Hauptpunkt (u_0, v_0) ergibt sich durch

$$u_0 = \boldsymbol{b}_1^T \boldsymbol{b}_3 \quad , \qquad v_0 = \boldsymbol{b}_2^T \boldsymbol{b}_3 \quad . \tag{C.7}$$

Damit kann man die Brennweiten berechnen:

$$f_x = \sqrt{\boldsymbol{b}_1^T \boldsymbol{b}_1 - u_0^2} \quad , \qquad f_y = \sqrt{\boldsymbol{b}_2^T \boldsymbol{b}_2 - v_0^2} \quad . \tag{C.8}$$

Die Rotationsmatrix ergibt sich aus

$$\begin{aligned} \boldsymbol{r}_{\cdot 1} &= \frac{1}{f_x} \boldsymbol{b}_1 - \frac{u_0}{f_x} \boldsymbol{b}_3 \quad , \\ \boldsymbol{r}_{\cdot 2} &= \frac{1}{f_y} \boldsymbol{b}_2 - \frac{v_0}{f_y} \boldsymbol{b}_3 \quad , \\ \boldsymbol{r}_{\cdot 3} &= \boldsymbol{b}_3 \quad . \end{aligned} \tag{C.9}$$

Den Translationsvektor erhält man aus der Lösung des folgenden Gleichungssystems:

$$\begin{pmatrix} \boldsymbol{b}_1^T \\ \boldsymbol{b}_2^T \\ \boldsymbol{b}_3^T \end{pmatrix} \boldsymbol{t} = -\boldsymbol{b}_4 \quad . \tag{C.10}$$

Anhang D

Tabellen

In diesem Anhang befinden sich sämtliche Tabellen zu den Simulationen, die in Kapitel 6 beschrieben sind. Aus diesen Daten wurden jeweils einige ausgewählt, auf welche im zugehörigen Abschnitt genauer eingegangen wird.

D.1 Anzahl der nötigen Iterationsschritte

Hier befinden sich die Tabellen zu Abschnitt 6.2.1

#Iter	f_x	f_y	u_0	v_0	$\boldsymbol{t}$	$\boldsymbol{R}$	3-D	Rückproj.
0	8.60	8.13	9.96	4.37	2.21369	0.00269973	0.243482	0.45/0.49
1	-0.189	0.168	-2.50	-0.0402	0.0364844	-0.000585481	-0.086026	0.407386
2	-0.466	-0.0420	-3.31	-0.471	-0.0154891	-0.000802894	-0.095357	0.405092
3	-0.803	-0.321	-3.95	-0.755	-0.0925322	-0.000967943	-0.102567	0.404078
4	-1.14	-0.608	-4.47	-1.02	-0.169569	-0.00110851	-0.108743	0.403375
5	-1.44	-0.882	-4.91	-1.26	-0.243064	-0.00122831	-0.114066	0.40284
7	-2.00	-1.37	-5.63	-1.68	-0.376945	-0.00143381	-0.122566	0.40207
10	-2.67	-1.97	-6.43	-2.13	-0.536825	-0.00166201	-0.131515	0.401348
15	-3.50	-2.73	-7.24	-2.57	-0.738703	-0.00190034	-0.140003	0.4007
20	-4.09	-3.28	-7.69	-2.74	-0.882988	-0.00201266	-0.14373	0.400397
30	-4.85	-4.00	-7.91	-2.83	-1.07228	-0.00207101	-0.145717	0.40017

Tabelle D.1: Szene 1: Unterschiedliche Anzahl an Iterationen, $\sigma = 0.3$ Pixel.

#Iter	f_x	f_y	u_0	v_0	$\boldsymbol{t}$	$\boldsymbol{R}$	3-D	Rückproj.
0	24.1	24.3	24.5	13.8	6.79199	0.00725524	0.760749	1.46/1.53
1	-4.33	-4.33	-3.96	-2.13	-1.14945	-0.00135193	-0.269379	1.34521
2	-5.15	-5.24	-6.00	-2.86	-1.47002	-0.00185843	-0.28602	1.33265
3	-5.50	-5.73	-7.47	-3.59	-1.56185	-0.00225747	-0.296789	1.33012
4	-5.76	-6.02	-8.78	-4.30	-1.63331	-0.00261829	-0.306065	1.32845
5	-5.92	-6.26	-9.78	-4.96	-1.66401	-0.00291308	-0.314564	1.32737
7	-6.11	-6.42	-10.6	-5.66	-1.72446	-0.00317600	-0.324978	1.32642
10	-6.20	-6.60	-11.6	-6.07	-1.76019	-0.00343181	-0.335567	1.32562
15	-6.65	-7.13	-14.1	-7.19	-1.87896	-0.00412382	-0.355963	1.32386
20	-6.62	-7.11	-15.1	-7.29	-1.84025	-0.00442061	-0.368246	1.32326
30	-6.63	-7.09	-15.5	-7.31	-1.84873	-0.00451304	-0.376605	1.32312

Tabelle D.2: Szene 1: Unterschiedliche Anzahl an Iterationen, $\sigma = 1.0$ Pixel.

#Iter	f_x	f_y	u_0	v_0	$\boldsymbol{t}$	$\boldsymbol{R}$	3-D	Rückproj.
0	13.5	12.8	11.0	9.88	3.80241	0.00332699	0.416231	0.48/0.58
1	-1.79	-1.85	-0.595	-0.893	-0.417459	-0.000176566	-0.101558	0.406833
2	-2.67	-2.65	-1.63	-1.50	-0.618445	-0.000457643	-0.120262	0.401815
3	-3.28	-3.15	-2.42	-2.29	-0.768927	-0.000709791	-0.13525	0.39897
4	-3.81	-3.63	-3.09	-2.81	-0.900866	-0.000909012	-0.147784	0.397033
5	-4.25	-4.01	-3.65	-3.30	-1.01233	-0.00107932	-0.158296	0.395546
7	-4.98	-4.64	-4.53	-4.01	-1.19907	-0.00133725	-0.175435	0.393551
10	-5.78	-5.32	-5.41	-4.74	-1.40713	-0.00158957	-0.19476	0.391914
15	-6.59	-6.03	-6.27	-5.59	-1.63158	-0.00184628	-0.216323	0.390723
20	-7.09	-6.47	-6.85	-6.12	-1.77999	-0.00201486	-0.231629	0.390242
30	-7.63	-6.94	-7.43	-6.73	-1.9532	-0.00219233	-0.251369	0.389937

Tabelle D.3: Szene 2: Unterschiedliche Anzahl an Iterationen, $\sigma = 0.3$ Pixel.

#Iter	f_x	f_y	u_0	v_0	$\boldsymbol{t}$	$\boldsymbol{R}$	3-D	Rückproj.
0	36.0	36.5	38.9	26.1	9.57001	0.0110419	0.984122	1.50/1.86
1	-0.468	-0.546	-3.15	-6.13	-0.602256	-0.00127828	-0.216449	1.35796
2	-1.20	-1.25	-6.34	-8.17	-0.716287	-0.00224619	-0.259225	1.33808
3	-1.70	-1.70	-9.21	-9.35	-0.816262	-0.00301212	-0.295405	1.33191
4	-2.54	-2.61	-10.0	-10.2	-0.953565	-0.00329516	-0.318416	1.32975
5	-2.78	-2.85	-12.2	-11.3	-0.999159	-0.00390299	-0.343103	1.32553
7	-3.81	-4.04	-14.3	-12.7	-1.1803	-0.0045624	-0.38257	1.32181
10	-4.84	-4.97	-17.2	-12.7	-1.3944	-0.00513809	-0.422635	1.31879
15	-6.30	-6.64	-20.6	-14.4	-1.70133	-0.00600696	-0.470187	1.31546
20	-7.89	-8.12	-21.8	-15.2	-1.97096	-0.00637703	-0.504492	1.31433
30	-8.34	-8.63	-22.6	-15.3	-2.03998	-0.00640388	-0.516742	1.31299

Tabelle D.4: Szene 2: Unterschiedliche Anzahl an Iterationen, $\sigma = 1.0$ Pixel.

#Iter	f_x	f_y	u_0	v_0	$\boldsymbol{t}$	$\boldsymbol{R}$	3-D	Rückproj.
0	19.3	19.8	11.2	12.3	5.90368	0.00459025	0.778891	0.50/0.59
1	-1.01	-0.893	-2.54	-2.61	-0.352976	-0.000990341	-0.346264	0.417514
2	-1.39	-1.21	-3.55	-2.70	-0.43612	-0.00117656	-0.378829	0.40921
3	-1.37	-1.05	-3.67	-3.69	-0.425221	-0.00138256	-0.398861	0.405811
4	-1.34	-1.04	-3.88	-4.08	-0.419134	-0.0015487	-0.410277	0.404423
5	-1.28	-0.997	-4.03	-4.34	-0.411231	-0.00165106	-0.417656	0.403919
7	-1.16	-0.881	-4.33	-5.18	-0.381681	-0.00191497	-0.428137	0.402999
10	-1.10	-0.885	-4.49	-5.43	-0.372579	-0.00204999	-0.433749	0.402541
15	-1.07	-0.891	-4.73	-6.11	-0.379931	-0.00227117	-0.441893	0.402067
20	-1.05	-0.924	-4.76	-6.17	-0.383144	-0.00232705	-0.444368	0.401944
30	-1.06	-0.982	-5.09	-7.19	-0.391462	-0.00261127	-0.449118	0.401479

Tabelle D.5: Szene 3: Unterschiedliche Anzahl an Iterationen, $\sigma = 0.3$ Pixel.

#Iter	f_x	f_y	u_0	v_0	$\boldsymbol{t}$	$\boldsymbol{R}$	3-D	Rückproj.
0	32.9	32.8	22.7	20.8	9.44783	0.00895585	1.99887	1.74/1.86
1	-19.8	-19.9	-5.47	-2.57	-4.64533	-0.00199752	-0.845837	1.42909
2	-24.3	-24.3	-7.70	-4.35	-5.43451	-0.00290136	-0.875107	1.34848
3	-25.6	-25.5	-9.72	-6.11	-5.69087	-0.00366099	-0.892583	1.3436
4	-26.3	-26.1	-10.8	-7.43	-5.8331	-0.0041861	-0.910031	1.3412
5	-26.8	-26.5	-12.1	-8.59	-5.98865	-0.00466943	-0.927471	1.33991
7	-27.3	-26.8	-13.4	-10.2	-6.19151	-0.00518442	-0.96023	1.3378
10	-27.4	-27.1	-14.4	-12.0	-6.23882	-0.0056941	-0.983573	1.33572
15	-27.4	-27.1	-15.2	-11.9	-6.26218	-0.00597865	-1.00637	1.3349
20	-27.2	-27.0	-15.5	-12.7	-6.37081	-0.00612914	-1.04459	1.33437
30	-27.3	-27.0	-15.6	-13.3	-6.47865	-0.00634682	-1.09844	1.33372

Tabelle D.6: Szene 3: Unterschiedliche Anzahl an Iterationen, $\sigma = 1.0$ Pixel.

#Iter	f_x	f_y	u_0	v_0	$\boldsymbol{t}$	$\boldsymbol{R}$	3-D	Rückproj.
0	13.7	13.8	7.26	6.28	4.27971	0.00190502	0.536617	0.47/0.50
1	-1.41	-1.42	-1.77	-1.42	-0.356393	-0.000633048	-0.243471	0.403089
2	-1.96	-1.93	-1.99	-1.63	-0.505395	-0.000653689	-0.249905	0.400419
3	-2.29	-2.29	-2.32	-1.95	-0.601842	-0.000685527	-0.255432	0.399574
4	-2.61	-2.60	-2.60	-2.25	-0.714165	-0.000745732	-0.259173	0.399159
5	-2.89	-2.87	-2.77	-2.39	-0.789912	-0.00076524	-0.264091	0.398976
7	-3.26	-3.25	-3.08	-2.64	-0.913956	-0.0008251	-0.269643	0.398785
10	-3.72	-3.71	-3.40	-2.91	-1.05954	-0.000894048	-0.27832	0.398456
15	-4.23	-4.19	-3.73	-3.23	-1.22014	-0.000940113	-0.291207	0.398295
20	-4.37	-4.33	-3.82	-3.40	-1.27984	-0.000984255	-0.299628	0.398131
30	-4.39	-4.36	-3.91	-3.39	-1.31655	-0.000988096	-0.309838	0.398069

Tabelle D.7: Szene 4: Unterschiedliche Anzahl an Iterationen, $\sigma = 0.3$ Pixel.

#Iter	f_x	f_y	u_0	v_0	$\boldsymbol{t}$	$\boldsymbol{R}$	3-D	Rückproj.
0	108	108	29.6	36.4	34.1065	0.0112885	2.91836	1.51/1.71
1	-1.64	-1.09	-4.01	-5.01	-0.808236	-0.00279874	-0.397243	1.40721
2	-3.10	-2.55	-5.09	-6.97	-1.21572	-0.00370689	-0.450079	1.38680
3	-4.63	-4.04	-5.86	-8.67	-1.69095	-0.0042718	-0.493970	1.38139
4	-5.73	-5.22	-6.81	-10.1	-2.09375	-0.00484404	-0.527139	1.37432
5	-7.25	-6.65	-7.47	-11.3	-2.54339	-0.00529292	-0.550549	1.37018
7	-10.2	-9.63	-8.50	-13.1	-3.43532	-0.00574373	-0.601731	1.36536
10	-14.5	-13.8	-10.1	-15.7	-4.73940	-0.00621472	-0.680382	1.35778
15	-20.0	-19.2	-11.1	-17.9	-6.43737	-0.00651808	-0.791621	1.35266
20	-25.4	-24.7	-12.3	-19.5	-8.08524	-0.00692010	-0.874687	1.34952
30	-27.4	-26.6	-13.1	-20.3	-8.77004	-0.00716458	-0.960076	1.34809

Tabelle D.8: Szene 4: Unterschiedliche Anzahl an Iterationen, $\sigma = 1.0$ Pixel.

#Iter	f_x	f_y	u_0	v_0	$\boldsymbol{t}$	$\boldsymbol{R}$	3-D	Rückproj.
0	192	196	109	87.7	66.5255	0.0291149	4.57825	0.48/4.44
1	1.67	-5.25	-20.1	-16.9	-0.891702	-0.00378270	0.587198	0.786031
2	-5.14	-12.0	-29.2	-21.9	-2.51543	-0.00610847	0.624520	0.603373
3	-9.73	-16.8	-35.8	-25.3	-3.77054	-0.00772558	0.636412	0.542911
4	-12.9	-20.0	-39.7	-27.8	-4.62074	-0.00878622	0.641437	0.515833
5	-15.3	-22.4	-42.2	-29.8	-5.24467	-0.00957119	0.641501	0.501407
7	-18.7	-25.5	-45.4	-32.9	-6.19774	-0.0106630	0.627993	0.487830
10	-22.3	-28.9	-48.7	-35.8	-7.24284	-0.0118546	0.601573	0.476800
15	-27.4	-33.7	-52.3	-39.4	-8.80297	-0.0132733	0.545373	0.466094
20	-32.6	-38.6	-55.3	-42.5	-10.4277	-0.0144653	0.497950	0.458867
30	-42.1	-48.0	-60.4	-46.8	-13.4255	-0.0162071	0.407618	0.449171

Tabelle D.9: Szene 5: Unterschiedliche Anzahl an Iterationen, $\sigma = 0.3$ Pixel.

#Iter	f_x	f_y	u_0	v_0	$\boldsymbol{t}$	$\boldsymbol{R}$	3-D	Rückproj.
0	45.1	47.2	18.7	13.9	30.3697	0.00762509	8.95917	1.53/2.08
1	-2.27	-2.14	-1.87	0.730	-2.81416	-0.000754696	-1.23131	1.39205
2	-2.49	-2.06	-2.25	-0.0646	-3.10015	-0.00131719	-1.37901	1.34475
3	-2.33	-2.15	-3.54	-0.491	-3.31688	-0.00158908	-1.49596	1.33370
4	-2.03	-2.03	-3.24	-1.43	-3.45765	-0.0018235	-1.59157	1.32670
5	-1.76	-1.98	-3.59	-2.04	-3.57408	-0.00196921	-1.67755	1.32225
7	-1.07	-1.73	-4.11	-3.08	-3.79202	-0.00239502	-1.83853	1.31697
10	-0.266	-1.24	-3.75	-3.39	-4.00147	-0.00241332	-2.04267	1.31322
15	1.04	-0.334	-5.18	-4.31	-4.35629	-0.00269109	-2.34674	1.31018
20	2.21	0.765	-4.29	-4.46	-4.61299	-0.00269160	-2.61643	1.30834
30	3.90	2.23	-5.30	-4.92	-5.30667	-0.00262908	-3.15186	1.30559

Tabelle D.10: Szene 5: Unterschiedliche Anzahl an Iterationen, $\sigma = 1.0$ Pixel.

#Iter	f_x	f_y	u_0	v_0	$\boldsymbol{t}$	$\boldsymbol{R}$	3-D	Rückproj.
0	57.6	59.2	33.7	31.3	20.4984	0.0105836	2.21849	0.53/1.16
1	0.147	-2.26	-2.52	-6.40	-0.119895	-0.00103764	-0.0104744	0.479456
2	-1.84	-4.10	-5.27	-8.64	-0.52972	-0.00200706	-0.000578772	0.452702
3	-3.27	-5.67	-7.62	-10.7	-0.936082	-0.00278368	0.0137344	0.440712
4	-4.92	-7.15	-9.32	-12.9	-1.36049	-0.00342064	0.0252563	0.431298
5	-6.45	-8.76	-10.9	-13.9	-1.77111	-0.00388089	0.0368972	0.425855
7	-9.06	-11.4	-13.0	-16.0	-2.50609	-0.00455014	0.0507975	0.418615
10	-12.4	-14.8	-15.4	-18.3	-3.47482	-0.00529367	0.060941	0.412516
15	-17.0	-19.5	-17.8	-20.4	-4.74653	-0.00601606	0.0635578	0.407342
20	-20.1	-22.4	-19.3	-21.9	-5.56825	-0.00656161	0.0609395	0.404783
30	-23.9	-26.2	-21.2	-23.2	-6.63762	-0.00720282	0.0380275	0.402832

Tabelle D.11: Szene 6: Unterschiedliche Anzahl an Iterationen, $\sigma = 0.3$ Pixel.

#Iter	f_x	f_y	u_0	v_0	$\boldsymbol{t}$	$\boldsymbol{R}$	3-D	Rückproj.
0	340	346	171	128	91.9583	0.0623885	8.46183	1.86/3.95
1	0.630	-1.70	0.289	5.49	-0.253698	0.00463953	-0.827802	1.97482
2	1.42	-2.31	1.29	6.80	-0.336535	0.00547448	-0.954637	1.78797
3	1.92	-2.62	2.63	5.68	-0.286659	0.00579886	-1.00761	1.75875
4	2.16	-2.87	3.09	5.32	-0.273275	0.00603575	-1.02735	1.74739
5	2.39	-3.05	3.30	4.28	-0.20493	0.00621891	-0.999173	1.74074
7	2.08	-3.77	2.33	4.04	-0.208948	0.00640782	-0.946894	1.73003
10	1.56	-4.69	2.20	2.61	-0.13244	0.00664758	-0.837542	1.7194
15	0.266	-6.45	0.97	-0.521	0.130527	0.00664495	-0.595158	1.7064
20	-1.68	-8.46	-0.940	-1.50	0.173968	0.00652012	-0.397551	1.69661
30	-6.19	-13.12	-5.48	-6.25	0.602335	0.00537206	0.0942957	1.67995

Tabelle D.12: Szene 6: Unterschiedliche Anzahl an Iterationen, $\sigma = 1.0$ Pixel.

#Iter	f_x	f_y	u_0	v_0	$\boldsymbol{t}$	$\boldsymbol{R}$	3-D	Rückproj.
0	14.7	15.5	12.7	9.67	4.98782	0.00387035	0.320034	0.45/0.48
1	0.906	0.897	-1.87	-1.09	0.314842	-0.000481646	-0.0623608	0.401218
2	0.804	0.639	-2.30	-1.52	0.323593	-0.000635499	-0.0709698	0.398411
3	0.542	0.223	-2.74	-1.78	0.230504	-0.000759598	-0.078286	0.39759
4	0.274	-0.143	-3.09	-2.05	0.133183	-0.000864214	-0.0846176	0.397068
5	0.0125	-0.486	-3.51	-2.31	0.0326721	-0.000979691	-0.0904852	0.396694
7	-0.510	-1.13	-4.26	-2.77	-0.160668	-0.00118478	-0.100979	0.396173
10	-1.25	-1.96	-5.05	-3.40	-0.432925	-0.0014214	-0.114048	0.395643
15	-2.35	-3.12	-6.19	-4.16	-0.818137	-0.00175315	-0.131724	0.395076
20	-3.27	-4.05	-7.08	-4.73	-1.13315	-0.00201028	-0.145408	0.394703
30	-4.83	-5.60	-8.19	-5.45	-1.65574	-0.00233428	-0.164509	0.39429

Tabelle D.13: Szene 7: Unterschiedliche Anzahl an Iterationen, $\sigma = 0.3$ Pixel.

#Iter	f_x	f_y	u_0	v_0	$\boldsymbol{t}$	$\boldsymbol{R}$	3-D	Rückproj.
0	45.5	47.3	37.8	61.9	15.7021	0.0179498	1.42777	1.49/1.96
1	-1.54	-1.61	-7.86	-7.78	-0.188971	-0.00250348	-0.277445	1.47018
2	-3.71	-3.82	-7.06	-12.7	-0.821894	-0.00344445	-0.331965	1.40025
3	-6.17	-6.49	-12.5	-15.7	-1.81465	-0.00472496	-0.395216	1.38765
4	-7.30	-7.70	-12.7	-19.3	-2.19449	-0.00559850	-0.443210	1.38273
5	-8.27	-8.86	-13.8	-21.6	-2.64564	-0.00618471	-0.482048	1.37792
7	-9.51	-10.2	-21.5	-26.6	-3.14097	-0.00822925	-0.579054	1.36415
10	-11.5	-12.6	-24.1	-33.7	-3.94101	-0.0101518	-0.689919	1.35328
15	-14.0	-15.4	-25.9	-41.5	-4.88909	-0.0121632	-0.804288	1.34544
20	-16.9	-18.7	-25.5	-46.5	-6.02078	-0.0130695	-0.871111	1.34217
30	-19.7	-21.6	-25.7	-50.0	-7.03877	-0.0137066	-0.917980	1.34056

Tabelle D.14: Szene 7: Unterschiedliche Anzahl an Iterationen, $\sigma = 1.0$ Pixel.

#Iter	f_x	f_y	u_0	v_0	$\boldsymbol{t}$	$\boldsymbol{R}$	3-D	Rückproj.
0	37.8	37.9	31.1	33.2	13.4308	0.010902	0.622307	0.43/0.78
1	-0.508	-0.434	-2.25	-4.83	-0.695125	-0.00137319	-0.0411205	0.433592
2	-0.942	-0.945	-3.31	-6.00	-0.911170	-0.00175250	-0.0570776	0.424274
3	-1.35	-1.41	-4.31	-6.56	-1.06368	-0.00202747	-0.0720372	0.421553
4	-1.72	-1.87	-5.23	-6.98	-1.21578	-0.00225618	-0.0851671	0.419422
5	-2.03	-2.23	-6.24	-7.25	-1.31769	-0.00247262	-0.0977000	0.417414
7	-2.57	-2.87	-7.86	-7.90	-1.52784	-0.00283447	-0.119091	0.414675
10	-3.41	-3.80	-9.92	-9.26	-1.77775	-0.00343054	-0.149000	0.410955
15	-4.70	-5.23	-12.5	-10.7	-2.20215	-0.00404516	-0.181233	0.407689
20	-5.81	-6.36	-14.2	-12.0	-2.57616	-0.00454095	-0.204071	0.406048
30	-7.46	-8.13	-16.9	-14.0	-3.06208	-0.00529553	-0.235144	0.403866

Tabelle D.15: Szene 8: Unterschiedliche Anzahl an Iterationen, $\sigma = 0.3$ Pixel.

#Iter	f_x	f_y	u_0	v_0	$\boldsymbol{t}$	$\boldsymbol{R}$	3-D	Rückproj.
0	243	247	180	188	97.0723	0.0568097	3.02011	1.43/7.23
1	5.07	3.93	-10.6	-36.2	0.746562	-0.00616676	0.0154363	1.89948
2	0.861	-0.867	-22.8	-32.2	-1.25988	-0.00757201	-0.0487117	1.58487
3	-1.85	-4.32	-32.3	-34.0	-2.67105	-0.00921581	-0.102835	1.53122
4	-5.65	-8.96	-44.1	-31.1	-4.41247	-0.010428	-0.14726	1.51149
5	-8.80	-12.5	-49.9	-32.2	-5.75825	-0.0114213	-0.190535	1.49581
7	-14.9	-19.2	-61.8	-32.9	-8.48151	-0.0131426	-0.273635	1.47739
10	-23.8	-29.0	-75.2	-36.0	-12.1578	-0.0154221	-0.400477	1.4595
15	-39.9	-45.3	-91.2	-51.2	-19.0703	-0.0204039	-0.631261	1.43299
20	-61.9	-66.3	-111	-75.6	-27.7597	-0.028154	-0.894316	1.37497
30	-104	-108	-126	-109	-44.6396	-0.0356272	-1.23315	1.33151

Tabelle D.16: Szene 8: Unterschiedliche Anzahl an Iterationen, $\sigma = 1.0$ Pixel.

D.2 Einfluss des Rauschens auf den Bildpunkten

Hier befinden sich die Tabellen zu Abschnitt 6.2.2

σ	f_x	f_y	u_0	v_0	$\boldsymbol{t}$	$\boldsymbol{R}$	3-D	Rückproj.	$\hat{\sigma}$
0.1	3.09	2.94	3.12	2.08	0.860836	0.00096633	0.0881747	0.14/0.16	
	-1.12	-0.931	-1.88	-1.35	-0.29028	-0.000610078	-0.0506682	0.13	0.093
0.2	5.85	6.01	8.17	5.38	1.52297	0.00251597	0.199866	0.30/0.36	
	-4.38	-4.50	-6.48	-4.26	-1.08364	-0.00204613	-0.142022	0.26	0.19
0.3	8.60	8.13	9.96	4.37	2.21369	0.00269973	0.243482	0.45/0.49	
	-4.09	-3.28	-7.69	-2.74	-0.882988	-0.00201266	-0.14373	0.40	0.29
0.5	18.9	18.5	12.5	5.25	5.36	0.00350493	0.410793	0.75/0.78	
	-2.69	-2.06	-5.13	0.0843	-0.632547	-0.00126083	-0.14391	0.66	0.48
0.7	11.5	11.1	8.19	11.0	3.04455	0.00351761	0.537632	1.04/1.08	
	-8.15	-7.95	-4.67	-7.35	-2.08168	-0.00226627	-0.315344	0.92	0.67
1.0	24.1	24.3	24.5	13.8	6.79199	0.00725524	0.760749	1.46/1.53	
	-6.62	-7.11	-15.1	-7.29	-1.84025	-0.00442061	-0.368246	1.32	0.96
2.0	42.6	44.8	45.6	41.3	13.0532	0.0159312	1.58283	3.00/3.22	
	-26.7	-26.8	-28.9	-24.2	-7.90669	-0.00991456	-0.85271	2.64	1.91

Tabelle D.17: Szene 1: Normalverteiltes Rauschen verschiedener Stärke.

σ	f_x	f_y	u_0	v_0	$\boldsymbol{t}$	$\boldsymbol{R}$	3-D	Rückproj.	$\hat{\sigma}$
0.1	4.39	4.26	1.48	2.18	1.25102	0.000579603	0.0883764	0.15/0.15	
	-1.77	-1.65	-0.246	-1.15	-0.488486	-0.000210537	-0.0387795	0.13	0.095
0.2	6.09	5.96	7.10	4.94	1.52836	0.00205506	0.221961	0.32/0.37	
	-1.75	-1.65	-5.09	-2.79	-0.333042	-0.00142993	-0.13985	0.27	0.20
0.3	13.5	12.8	11.0	9.88	3.80241	0.00332699	0.416231	0.48/0.58	
	-7.09	-6.47	-6.85	-6.12	-1.77999	-0.00201486	-0.231629	0.39	0.28
0.5	9.60	9.97	15.9	11.4	2.61396	0.00440344	0.464959	0.78/0.89	
	-2.10	-2.44	-10.8	-5.68	-0.580662	-0.00255665	-0.29804	0.67	0.48
0.7	31.1	30.0	11.7	14.9	8.43694	0.00425736	0.626477	1.09/1.13	
	-13.9	-13.7	-4.68	-6.03	-3.75105	-0.00190308	-0.330934	0.91	0.66
1.0	36.0	36.5	38.9	26.1	9.57001	0.0110419	0.984122	1.50/1.86	
	-7.89	-8.12	-21.8	-15.2	-1.97096	-0.00637703	-0.504492	1.31	0.95
2.0	70.0	68.6	50.2	55.8	19.9397	0.0161926	1.85127	3.09/3.47	
	-12.4	-12.2	-28.8	-27.8	-3.15973	-0.00837949	-0.948882	2.59	1.88

Tabelle D.18: Szene 2: Normalverteiltes Rauschen verschiedener Stärke.

σ	f_x	f_y	u_0	v_0	$\boldsymbol{t}$	$\boldsymbol{R}$	3-D	Rückproj.	$\hat{\sigma}$
0.1	2.59	2.62	1.93	1.48	0.874766	0.000542909	0.238651	0.18/0.19	
	-0.86	-0.85	-1.28	-0.94	-0.292136	-0.000339836	-0.170242	0.13	0.097
0.2	10.5	11.2	12.8	13.3	3.75051	0.00505552	0.587208	0.36/0.50	
	-2.50	-3.50	-8.96	-9.06	-0.973064	-0.00352838	-0.303291	0.27	0.20
0.3	19.3	19.8	11.2	12.3	5.90368	0.00459025	0.778891	0.50/0.59	
	-1.05	-0.924	-4.76	-6.17	-0.383144	-0.00232705	-0.444368	0.40	0.29
0.5	31.4	30.2	18.4	19.3	10.1378	0.00592758	0.984493	0.82/0.89	
	-11.4	-10.4	-10.7	-12.6	-3.32513	-0.00350958	-0.360192	0.65	0.47
0.7	29.7	31.1	24.8	26.4	8.77143	0.0105598	1.4607	1.23/1.46	
	-24.5	-26.1	-14.2	-18.5	-7.16233	-0.00759657	-0.779516	0.94	0.68
1.0	32.9	32.8	22.7	20.8	9.44783	0.00895585	1.99887	1.74/1.86	
	-27.2	-27.0	-15.5	-12.7	-6.37081	-0.00612914	-1.04459	1.33	0.97
2.0	179	180	58.6	58.1	49.9129	0.0248215	4.54032	3.78/4.06	
	-29.5	-29.8	-20.6	-19.3	-8.80477	-0.0144073	-1.94658	2.69	1.95

Tabelle D.19: Szene 3: Normalverteiltes Rauschen verschiedener Stärke.

σ	f_x	f_y	u_0	v_0	$\boldsymbol{t}$	$\boldsymbol{R}$	3-D	Rückproj.	$\hat{\sigma}$
0.1	8.50	8.58	2.58	2.18	2.41829	0.000652609	0.205274	0.16/0.17	
	-2.31	-2.43	-1.05	-0.621	-0.68274	-0.000204335	-0.126744	0.13	0.095
0.2	6.93	6.62	5.34	8.80	2.30202	0.00268645	0.408378	0.33/0.41	
	-1.80	-1.64	-3.98	-7.42	-0.722118	-0.00225183	-0.25188	0.26	0.19
0.3	13.7	13.8	7.26	6.28	4.27971	0.00190502	0.536617	0.47/0.50	
	-4.37	-4.33	-3.82	-3.40	-1.27984	-0.000984255	-0.299628	0.40	0.29
0.5	57.4	56.7	17.6	20.1	18.0021	0.00656959	1.53945	0.79/0.91	
	-12.1	-11.2	-7.05	-10.6	-3.78094	-0.0038471	-0.511639	0.66	0.48
0.7	88.7	91.4	47.5	41.1	26.6822	0.0138212	1.78132	1.20/1.73	
	-12.4	-15.4	-12.6	-13.5	-3.73035	-0.00545732	-0.81368	0.98	0.71
1.0	108	108	29.6	36.4	34.1065	0.0112885	2.91836	1.51/1.71	
	-25.4	-24.7	-12.3	-19.5	-8.08524	-0.0069201	-0.874687	1.35	0.98
2.0	164	168	77.0	56.3	53.9878	0.0199919	4.5827	3.00/3.32	
	-29.8	-32.4	-31.0	-23.9	-9.13984	-0.00770092	-0.657717	2.63	1.91

Tabelle D.20: Szene 4: Normalverteiltes Rauschen verschiedener Stärke.

σ	f_x	f_y	u_0	v_0	$\boldsymbol{t}$	$\boldsymbol{R}$	3-D	Rückproj.	$\hat{\sigma}$
0.1	6.76	6.88	2.63	1.89	2.11889	0.000700558	0.235186	0.16/0.16	
	-1.04	-1.22	-0.498	-0.0717	-0.278572	-5.58657e-06	-0.0885836	0.13	0.094
0.2	13.1	13.2	8.03	6.11	4.63196	0.00239087	0.668304	0.34/0.38	
	0.136	-0.0601	-4.11	-2.15	-0.0451825	-0.00104413	-0.18228	0.26	0.19
0.3	192	196	109	87.7	66.5255	0.0291149	4.57825	0.48/4.44	
	-32.6	-38.6	-55.3	-42.5	-10.4277	-0.0144653	0.49795	0.46	0.33
0.5	377	387	210	172	127.644	0.0481796	5.1844	0.83/12.8	
	-99.3	-113	-111	-87.1	-32.6377	-0.0237607	1.51837	0.75	0.54
0.7	103	104	36.1	25.6	44.0299	0.0139408	8.23066	1.17/1.27	
	-2.14	-4.00	-7.52	-2.60	-2.67273	-0.00443786	-1.31346	0.94	0.68
1.0	45.1	47.2	18.7	13.9	30.3697	0.00762509	8.95917	1.53/2.08	
	2.21	0.765	-4.29	-4.46	-4.61299	-0.0026916	-2.61643	1.31	0.95
2.0	482	483	174	122	169.656	0.0836537	32.3138	3.33/4.07	
	1.86	-2.19	-5.95	-2.31	-1.83063	0.000628537	-1.5482	2.83	2.05

Tabelle D.21: Szene 5: Normalverteiltes Rauschen verschiedener Stärke.

σ	f_x	f_y	u_0	v_0	$\boldsymbol{t}$	$\boldsymbol{R}$	3-D	Rückproj.	$\hat{\sigma}$
0.1	51.7	53.6	32.8	25.1	16.6505	0.00888625	1.04164	0.18/0.85	
	-21.5	-23.3	-21.9	-16.6	-5.80511	-0.0056877	0.124504	0.15	0.11
0.2	13.4	13.1	6.07	13.4	2.95899	0.0024124	0.988553	0.33/0.46	
	-5.49	-5.49	-3.91	-9.57	-1.4352	-0.00176744	-0.522225	0.27	0.20
0.3	57.6	59.2	33.7	31.3	20.4984	0.0105836	2.21849	0.53/1.16	
	-20.1	-22.4	-19.3	-21.9	-5.56825	-0.00656161	0.0609395	0.40	0.29
0.5	265	275	200	192	87.0831	0.0497125	4.18922	0.86/12.5	
	-141	-151	-162	-160	-45.4503	-0.0402886	-1.11215	0.72	0.53
0.7	87.7	91.6	46.5	26.4	35.2261	0.0141139	5.94594	1.16/1.46	
	-9.71	-13.9	-13.4	-4.34	-2.45056	-0.00359651	-0.168214	0.95	0.69
1.0	340	346	171	128	91.9583	0.0623885	8.46183	1.86/3.95	
	-1.68	-8.46	-0.940	-1.50	0.173968	0.00652012	-0.397551	1.70	1.23
2.0	73.2	71.2	39.8	43.9	78.6348	0.0161048	24.6456	3.05/3.81	
	19.1	18.0	-3.13	-18.7	0.0108106	-0.00659151	-1.94901	2.61	1.89

Tabelle D.22: Szene 6: Normalverteiltes Rauschen verschiedener Stärke.

σ	f_x	f_y	u_0	v_0	$\boldsymbol{t}$	$\boldsymbol{R}$	3-D	Rückproj.	$\hat{\sigma}$
0.1	3.75	3.97	2.02	2.05	1.43082	0.000712568	0.0902733	0.15/0.15	
	-0.681	-0.995	-0.537	-0.472	-0.397488	-0.000141937	-0.0392904	0.13	0.096
0.2	11.2	11.7	6.34	11.0	4.33496	0.00324853	0.312367	0.30/0.36	
	-3.17	-3.64	-2.37	-6.30	-1.45613	-0.0017028	-0.1796	0.26	0.19
0.3	14.7	15.5	12.7	9.67	4.98782	0.00387035	0.320034	0.45/0.48	
	-3.27	-4.05	-7.08	-4.73	-1.13315	-0.00201028	-0.145408	0.39	0.29
0.5	32.1	33.9	15.2	28.5	11.4042	0.00790888	0.667514	0.73/0.83	
	-12.2	-13.4	-4.54	-11.7	-4.48034	-0.00318831	-0.336339	0.66	0.48
0.7	20.8	22.9	9.31	16.6	8.79317	0.00505024	0.848373	1.01/1.05	
	-0.229	-2.40	2.34	-6.78	-1.13534	-0.00107866	-0.379796	0.93	0.67
1.0	45.5	47.3	37.8	61.9	15.7021	0.0179498	1.42777	1.49/1.96	
	-16.9	-18.7	-25.5	-46.5	-6.02078	-0.0130695	-0.871111	1.34	0.97
2.0	76.3	76.5	30.1	20.7	29.8411	0.0102848	1.71714	2.83/2.87	
	-26.7	-25.5	-8.62	1.62	-10.9875	-0.00229837	-0.626682	2.61	1.89

Tabelle D.23: Szene 7: Normalverteiltes Rauschen verschiedener Stärke.

σ	f_x	f_y	u_0	v_0	$\boldsymbol{t}$	$\boldsymbol{R}$	3-D	Rückproj.	$\hat{\sigma}$
0.1	3.09	3.01	2.65	1.62	1.04537	0.000795802	0.079257	0.14/0.15	
	-0.684	-0.680	-1.21	-0.108	-0.241711	-0.000312591	-0.0302742	0.13	0.096
0.2	5.07	4.80	4.97	10.6	1.83566	0.00285408	0.267166	0.29/0.32	
	-2.55	-2.31	-2.26	-7.34	-1.01724	-0.00184409	-0.15166	0.26	0.19
0.3	37.8	37.9	31.1	33.2	13.4308	0.010902	0.622307	0.43/0.78	
	-5.81	-6.36	-14.2	-12.0	-2.57616	-0.00454095	-0.204071	0.41	0.29
0.5	28.8	28.5	13.5	24.3	9.93676	0.00693549	0.516686	0.72/0.77	
	-1.94	-1.60	-2.95	-11.4	-0.319274	-0.00290833	-0.151577	0.66	0.48
0.7	27.7	27.0	28.6	19.7	9.38349	0.00846833	0.667536	1.01/1.11	
	-16.7	-16.1	-20.2	-11.3	-6.00133	-0.00561137	-0.35916	0.91	0.66
1.0	243	247	180	188	97.0723	0.0568097	3.02011	1.43/7.23	
	-61.9	-66.3	-111	-75.6	-27.7597	-0.028154	-0.894316	1.37	1.0
2.0	74.9	74.0	101	71.2	25.5685	0.0308874	2.34341	2.90/4.82	
	-12.1	-13.0	-66.3	-14.5	-6.50185	-0.0156181	-1.02892	2.72	1.97

Tabelle D.24: Szene 8: Normalverteiltes Rauschen verschiedener Stärke.

D.3 Unterschiedliche Parametrisierung der Rotation

Hier befinden sich die Tabellen zu Abschnitt 6.2.3

σ		f_x	f_y	u_0	v_0	$\boldsymbol{t}$	$\boldsymbol{R}$	3-D	Rückproj.	$\hat{\sigma}$
0.3	vor	8.60	8.13	9.96	4.37	2.21369	0.00269973	0.243482	0.45/0.49	
	Q	-4.09	-3.28	-7.69	-2.74	-0.882988	-0.00201266	-0.14373	0.40	0.29
	A/W	-4.10	-3.28	-7.65	-2.75	-0.885005	-0.00200837	-0.14363	0.40	0.29
0.7	vor	11.5	11.1	8.19	11.0	3.04455	0.00351761	0.537632	1.04/1.08	
	Q	-8.15	-7.95	-4.67	-7.35	-2.08168	-0.00226627	-0.315344	0.92	0.67
	A/W	-8.34	-8.18	-4.60	-7.30	-2.14552	-0.00225269	-0.315123	0.92	0.67
1.0	vor	24.1	24.3	24.5	13.8	6.79199	0.00725524	0.760749	1.46/1.53	
	Q	-6.62	-7.11	-15.1	-7.29	-1.84025	-0.00442061	-0.368246	1.32	0.96
	A/W	-6.24	-6.74	-15.1	-6.96	-1.73122	-0.00437492	-0.365443	1.32	0.96

Tabelle D.25: Szene 1: Quaternionen – Achse/Winkel.

σ		f_x	f_y	u_0	v_0	$\boldsymbol{t}$	$\boldsymbol{R}$	3-D	Rückproj.	$\hat{\sigma}$
0.3	vor	13.5	12.8	11.0	9.88	3.80241	0.00332699	0.416231	0.48/0.58	
	Q	-7.09	-6.47	-6.85	-6.12	-1.77999	-0.00201486	-0.231629	0.39	0.28
	A/W	-7.06	-6.44	-6.85	-6.16	-1.77052	-0.00202026	-0.231385	0.39	0.28
0.7	vor	31.1	30.0	11.7	14.9	8.43694	0.00425736	0.626477	1.09/1.13	
	Q	-13.9	-13.7	-4.68	-6.03	-3.75105	-0.00190308	-0.330934	0.91	0.66
	A/W	-11.2	-11.2	-4.20	-4.36	-3.02155	-0.00155563	-0.303192	0.91	0.66
1.0	vor	36.0	36.5	38.9	26.1	9.57001	0.0110419	0.984122	1.50/1.86	
	Q	-7.89	-8.12	-21.8	-15.2	-1.97096	-0.00637703	-0.504492	1.31	0.95
	A/W	-7.86	-7.83	-19.2	-15.0	-1.97941	-0.00573987	-0.490924	1.32	0.95

Tabelle D.26: Szene 2: Quaternionen – Achse/Winkel.

σ		f_x	f_y	u_0	v_0	$\boldsymbol{t}$	$\boldsymbol{R}$	3-D	Rückproj.	$\hat{\sigma}$
0.3	vor	19.3	19.8	11.2	12.3	5.90368	0.00459025	0.778891	0.50/0.59	
	Q	-1.05	-0.924	-4.76	-6.17	-0.383144	-0.00232705	-0.444368	0.40	0.29
	A/W	-1.41	-1.32	-5.01	-6.71	-0.461889	-0.0024852	-0.447069	0.40	0.29
0.7	vor	29.7	31.1	24.8	26.4	8.77143	0.0105598	1.4607	1.23/1.46	
	Q	-24.5	-26.1	-14.2	-18.5	-7.16233	-0.00759657	-0.779516	0.94	0.68
	A/W	-24.7	-26.3	-16.5	-20.7	-7.29531	-0.00837569	-0.812403	0.94	0.68
1.0	vor	32.9	32.8	22.7	20.8	9.44783	0.00895585	1.99887	1.74/1.86	
	Q	-27.2	-27.0	-15.5	-12.7	-6.37081	-0.00612914	-1.04459	1.33	0.97
	A/W	-27.2	-27.1	-15.5	-12.6	-6.39037	-0.00610905	-1.06381	1.33	0.97

Tabelle D.27: Szene 3: Quaternionen – Achse/Winkel.

σ		f_x	f_y	u_0	v_0	$\boldsymbol{t}$	$\boldsymbol{R}$	3-D	Rückproj.	$\hat{\sigma}$
0.3	vor	13.7	13.8	7.26	6.28	4.27971	0.00190502	0.536617	0.47/0.50	
	Q	-4.37	-4.33	-3.82	-3.40	-1.27984	-0.000984255	-0.299628	0.40	0.29
	A/W	-3.40	-3.37	-3.67	-3.32	-0.9933	-0.000952536	-0.287074	0.40	0.29
0.7	vor	88.7	91.4	47.5	41.1	26.6822	0.0138212	1.78132	1.20/1.73	
	Q	-12.4	-15.4	-12.6	-13.5	-3.73035	-0.00545732	-0.81368	0.98	0.71
	A/W	-14.0	-17.3	-19.4	-16.6	-4.64984	-0.00657011	-0.967142	0.96	0.70
1.0	vor	108	108	29.6	36.4	34.1065	0.0112885	2.91836	1.51/1.71	
	Q	-25.4	-24.7	-12.3	-19.5	-8.08524	-0.0069201	-0.874687	1.35	0.98
	A/W	-23.6	-22.8	-12.5	-19.4	-7.72696	-0.00682723	-0.94095	1.35	0.99

Tabelle D.28: Szene 4: Quaternionen – Achse/Winkel.

σ		f_x	f_y	u_0	v_0	$\boldsymbol{t}$	$\boldsymbol{R}$	3-D	Rückproj.	$\hat{\sigma}$
0.3	vor	192	196	109	87.7	66.5255	0.0291149	4.57825	0.48/4.44	
	Q	-32.6	-38.6	-55.3	-42.5	-10.4277	-0.0144653	0.49795	0.46	0.33
	A/W	-34.0	-40.0	-55.6	-42.5	-10.8239	-0.014387	0.495124	0.46	0.33
0.7	vor	103	104	36.1	25.6	44.0299	0.0139408	8.23066	1.17/1.27	
	Q	-2.14	-4.00	-7.52	-2.60	-2.67273	-0.00443786	-1.31346	0.94	0.68
	A/W	-2.07	-4.04	-8.06	-2.44	-2.63816	-0.00441323	-1.29443	0.94	0.68
1.0	vor	45.1	47.2	18.7	13.9	30.3697	0.00762509	8.95917	1.53/2.08	
	Q	2.21	0.765	-4.29	-4.46	-4.61299	-0.0026916	-2.61643	1.31	0.95
	A/W	3.27	1.61	-4.99	-4.07	-4.45546	-0.00257544	-2.65749	1.31	0.95

Tabelle D.29: Szene 5: Quaternionen – Achse/Winkel.

σ		f_x	f_y	u_0	v_0	$\boldsymbol{t}$	$\boldsymbol{R}$	3-D	Rückproj.	$\hat{\sigma}$
0.3	vor	57.6	59.2	33.7	31.3	20.4984	0.0105836	2.21849	0.53/1.16	
	Q	-20.1	-22.4	-19.3	-21.9	-5.56825	-0.00656161	0.0609395	0.40	0.29
	A/W	-17.7	-20.1	-17.5	-20.5	-4.80341	-0.00591668	0.13751	0.41	0.30
0.7	vor	87.7	91.6	46.5	26.4	35.2261	0.0141139	5.94594	1.16/1.46	
	Q	-9.71	-13.9	-13.4	-4.34	-2.45056	-0.00359651	-0.168214	0.95	0.69
	A/W	-9.56	-13.8	-12.8	-4.17	-2.48699	-0.00348508	-0.197346	0.95	0.69
1.0	vor	340	346	171	128	91.9583	0.0623885	8.46183	1.86/3.95	
	Q	-1.68	-8.46	-0.940	-1.50	0.173968	0.00652012	-0.397551	1.70	1.23
	A/W	-2.52	-9.48	0.281	-1.50	0.0102833	0.00716027	-0.254497	1.70	1.23

Tabelle D.30: Szene 6: Quaternionen – Achse/Winkel.

σ		f_x	f_y	u_0	v_0	$\boldsymbol{t}$	$\boldsymbol{R}$	3-D	Rückproj.	$\hat{\sigma}$
0.3	vor	14.7	15.5	12.7	9.67	4.98782	0.00387035	0.320034	0.45/0.48	
	Q	-3.27	-4.05	-7.08	-4.73	-1.13315	-0.00201028	-0.145408	0.39	0.29
	A/W	-3.23	-4.01	-6.95	-4.66	-1.11664	-0.00197918	-0.144794	0.39	0.29
0.7	vor	20.8	22.9	9.31	16.6	8.79317	0.00505024	0.848373	1.01/1.05	
	Q	-0.229	-2.40	2.34	-6.78	-1.13534	-0.00107866	-0.379796	0.93	0.67
	A/W	-0.668	-2.89	1.73	-7.09	-1.30637	-0.00123511	-0.388632	0.93	0.67
1.0	vor	45.5	47.3	37.8	61.9	15.7021	0.0179498	1.42777	1.49/1.96	
	Q	-16.9	-18.7	-25.5	-46.5	-6.02078	-0.0130695	-0.871111	1.34	0.97
	A/W	-17.2	-18.9	-25.0	-46.4	-6.01509	-0.0129724	-0.872486	1.34	0.97

Tabelle D.31: Szene 7: Quaternionen – Achse/Winkel.

σ		f_x	f_y	u_0	v_0	$\boldsymbol{t}$	$\boldsymbol{R}$	3-D	Rückproj.	$\hat{\sigma}$
0.3	vor	37.8	37.9	31.1	33.2	13.4308	0.010902	0.622307	0.43/0.78	
	Q	-5.81	-6.36	-14.2	-12.0	-2.57616	-0.00454095	-0.204071	0.41	0.29
	A/W	-5.77	-6.39	-14.5	-12.0	-2.55221	-0.00460326	-0.203813	0.41	0.29
0.7	vor	27.7	27.0	28.6	19.7	9.38349	0.00846833	0.667536	1.01/1.11	
	Q	-16.7	-16.1	-20.2	-11.3	-6.00133	-0.00561137	-0.35916	0.91	0.66
	A/W	-17.0	-16.5	-21.5	-12.7	-6.13845	-0.00602556	-0.361227	0.91	0.66
1.0	vor	243	247	180	188	97.0723	0.0568097	3.02011	1.43/7.23	
	Q	-61.9	-66.3	-111	-75.6	-27.7597	-0.028154	-0.894316	1.37	1.0
	A/W	-61.7	-66.3	-111	-74.9	-27.7146	-0.0280119	-0.892473	1.38	1.0

Tabelle D.32: Szene 8: Quaternionen – Achse/Winkel.

D.4 Korrektur von Linsenverzerrungen

Hier befinden sich die Tabellen zu Abschnitt 6.2.4.

κ		κ	f_x	f_y	u_0	v_0	$\boldsymbol{t}$	$\boldsymbol{R}$	3-D	Rückproj.	$\hat{\sigma}$
0.01	vor	0.01	2.38	2.24	8.79	9.06	0.748	0.00310	0.232	0.447/0.482	
	ohne	0	0.842	1.23	-4.72	-5.50	0.252	-0.00174	-0.124	0.396	0.29
	mit	-0.00292	0.508	0.893	-4.71	-6.32	0.155	-0.00191	-0.125	0.394	0.29
-0.01	vor	0.01	6.42	6.39	6.74	6.80	1.98	0.00214	0.256	0.460/0.489	
	ohne	0	-2.95	-2.79	-2.30	-3.26	-0.915	-0.000870	-0.162	0.401	0.29
	mit	-0.00402	-1.41	-1.30	-3.86	-5.15	-0.489	-0.00128	-0.159	0.398	0.29
0.1	vor	0.1	6.98	7.32	35.3	41.4	1.27	0.0133	0.508	0.714/0.779	
	ohne	0	0.537	0.132	-5.27	-15.0	0.919	-0.00326	-0.302	0.504	0.37
	mit	-0.0909	-4.78	-5.23	-30.6	-37.4	-0.641	-0.0117	-0.399	0.405	0.29
-0.1	vor	0.1	12.8	13.0	39.3	46.2	2.56	0.0149	0.620	0.797/1.28	
	ohne	0	1.40	1.20	-9.88	-20.8	0.736	-0.00467	-0.388	0.549	0.40
	mit	-0.0902	-8.82	-9.46	-37.0	-43.3	-1.42	-0.0139	-0.470	0.399	0.29
0.2	vor	0.2	18.4	19.7	66.2	78.0	5.87	0.0250	0.972	1.16/1.23	
	ohne	0	-2.96	-3.50	-7.98	-26.6	-0.427	-0.00557	-0.568	0.708	0.51
	mit	-0.191	-6.56	-7.72	-42.2	-63.7	-2.45	-0.0184	-0.727	0.416	0.30
-0.2	vor	0.2	30.8	32.6	77.5	86.5	3.19	0.0295	1.19	1.47/3.44	
	ohne	0	9.37	6.25	-20.4	-38.1	5.92	-0.00893	-0.664	0.934	0.68
	mit	-0.191	-22.6	-24.1	-71.9	-81.9	-0.528	-0.0277	-0.869	0.390	0.28

Tabelle D.33: Szene 1: Korrektur von Linsenverzerrungen, $\sigma = 0.3$ Pixel.

κ		κ	f_x	f_y	u_0	v_0	$\boldsymbol{t}$	$\boldsymbol{R}$	3-D	Rückproj.	$\hat{\sigma}$
0.01	vor	0.01	9.87	10.4	8.90	15.2	3.36	0.00441	0.394	0.431/0.471	
	ohne	0	-3.59	-3.78	-4.65	-8.14	-1.27	-0.00259	-0.229	0.401	0.29
	mit	0.00963	-3.00	-3.18	-3.80	-5.72	-1.09	-0.00187	-0.224	0.399	0.29
-0.01	vor	0.01	18.3	19.3	8.97	12.3	6.60	0.00368	0.412	0.40/0.470	
	ohne	0	-4.49	-5.67	-2.66	-3.75	-1.85	-0.00101	-0.174	0.395	0.29
	mit	0.00163	-4.51	-5.73	-3.76	-6.30	-1.86	-0.00158	-0.173	0.394	0.29
0.1	vor	0.1	12.6	11.0	44.0	31.4	4.35	0.0135	0.717	0.595/0.667	
	ohne	0	-3.93	-2.29	-16.5	-7.02	-1.56	-0.00464	-0.440	0.457	0.33
	mit	-0.0847	-3.00	-1.21	-30.8	-21.1	-1.15	-0.00963	-0.506	0.407	0.30
-0.1	vor	0.1	19.9	19.5	45.0	35.4	7.89	0.0139	0.587	0.575/0.844	
	ohne	0	-14.2	-13.9	-19.2	-12.9	-5.79	-0.00552	-0.343	0.464	0.34
	mit	-0.0807	-7.61	-6.96	-34.4	-27.5	-3.63	-0.0108	-0.351	0.394	0.29
0.2	vor	0.2	56.8	54.0	76.6	79.5	17.8	0.0283	1.29	0.785/0.901	
	ohne	0	-8.71	-5.85	-14.2	-16.4	-2.67	-0.00589	-0.386	0.589	0.43
	mit	-0.185	-4.63	-1.69	2.29	7.44	-0.778	0.00121	-0.512	0.425	0.31
-0.2	vor	0.2	57.8	58.3	109	84.4	24.8	0.0327	1.76	0.929/3.04	
	ohne	0	-37.0	-37.9	-54.1	-36.6	-16.7	-0.0151	-1.01	0.673	0.49
	mit	-0.144	-4.57	-5.03	-72.5	-57.6	-5.81	-0.0226	-0.806	0.469	0.34

Tabelle D.34: Szene 7: Korrektur von Linsenverzerrungen, $\sigma = 0.3$ Pixel.

Szene		κ	f_x	f_y	u_0	v_0	$\boldsymbol{t}$	$\boldsymbol{R}$	3-D	Rückproj.
1	vor	0	8.60	8.13	9.96	4.37	2.21	0.00270	0.243	0.45/0.49
	ohne	0	-4.09	-3.28	-7.69	-2.74	-0.883	-0.00201	-0.144	0.400
	mit	0.00656	-3.53	-2.74	-7.16	-2.35	-0.744	-0.00176	-0.141	0.400
2	vor	0	13.5	12.8	11.0	9.88	3.80	0.00333	0.416	0.48/0.58
	ohne	0	-7.09	-6.47	-6.85	-6.12	-1.78	-0.00201	-0.232	0.390
	mit	0.0118	-7.22	-6.62	-6.68	-5.02	-1.86	-0.00186	-0.232	0.389
3	vor	0	19.3	19.8	11.2	12.3	5.90	0.00459	0.779	0.50/0.59
	ohne	0	-1.05	-0.924	-4.76	-6.17	-0.383	-0.00233	-0.444	0.402
	mit	0.0115	-0.759	-0.657	-4.78	-6.63	-0.271	-0.00229	-0.448	0.400
4	vor	0	13.7	13.8	7.26	6.28	4.28	0.00191	0.537	0.47/0.50
	ohne	0	-4.37	-4.33	-3.82	-3.40	-1.28	-0.000984	-0.300	0.398
	mit	0.00750	-4.07	-4.03	-3.24	-3.56	-1.18	-0.000825	-0.293	0.397
5	vor	0	192	196	109	87.7	66.5	0.0291	4.58	0.48/4.44
	ohne	0	-32.6	-38.6	-55.3	-42.5	-10.4	-0.0145	0.498	0.459
	mit	0.0269	-27.9	-34.0	-61.5	-47.2	-9.23	-0.0159	0.494	0.454
6	vor	0	57.6	59.2	33.7	31.3	20.5	0.0106	2.22	0.53/1.16
	ohne	0	-20.1	-22.4	-19.3	-21.9	-5.57	-0.00656	0.0609	0.405
	mit	0.00967	-18.9	-21.2	-18.8	-22.6	-5.16	-0.00647	0.0863	0.404
7	vor	0	14.7	15.5	12.7	9.67	4.99	0.00387	0.320	0.45/0.48
	ohne	0	-3.27	-4.05	-7.08	-4.73	-1.13	-0.00201	-0.145	0.395
	mit	0.0198	-3.26	-3.97	-6.51	-4.62	-1.08	-0.00184	-0.142	0.393
8	vor	0	37.8	37.9	31.1	33.2	13.4	0.0109	0.622	0.43/0.78
	ohne	0	-5.81	-6.36	-14.2	-12.0	-2.58	-0.00454	-0.204	0.406
	mit	0.0156	-6.52	-7.13	-14.4	-12.1	-2.77	-0.00475	-0.206	0.405

Tabelle D.35: Szenen 1–8: Korrektur von Linsenverzerrungen obwohl keine vorliegen, $\sigma = 0.3$ Pixel.

D.5 Verhalten unabhängig von der vorangegangenen Rekonstruktion

Hier befinden sich die Tabellen zu Abschnitt 6.2.5

D.5.1 Verrauschte Parameter

σ_P		f_x	f_y	u_0	v_0	$\boldsymbol{t}$	$\boldsymbol{R}$	3-D	Rückproj.	$\hat{\sigma}$
0.0	vor	0	0	0	0	0	0	0	0.428	
	And.	1.69	1.75	0.498	0.509	0.473	0.000237298	0.0874587	0.402	0.29
0.1	vor	0.860	1.13	0.00112	0.00114	0.255	0.000262038	0.164103	1.19	
	And.	0.771	0.660	0.307	0.368	0.222	-8.08726e-05	-0.0780065	0.402	0.29
0.2	vor	2.42	2.30	0.00177	0.00201	0.794	0.000653836	0.344356	2.03	
	And.	-0.678	-0.283	0.461	0.487	-0.264	-0.000423714	-0.25522	0.402	0.29
0.3	vor	3.64	2.43	0.00320	0.00258	0.801	0.00117352	0.553162	5.00	
	And.	-1.93	-0.392	0.436	0.483	-0.264	-0.00092312	-0.463726	0.402	0.29
0.6	vor	6.08	6.79	0.00437	0.00654	1.77	0.00220986	1.00501	7.84	
	And.	-4.56	-5.26	0.644	0.589	-1.32	-0.00185573	-0.890896	0.403	0.29
1.0	vor	11.6	7.87	0.0101	0.00746	3.19	0.00283297	1.74166	12.9	
	And.	-9.41	-5.34	0.682	0.507	-2.49	-0.00252727	-1.64152	0.402	0.29

Tabelle D.36: Szene 1: Verrauschte Parameter.

σ_P		f_x	f_y	u_0	v_0	$\boldsymbol{t}$	$\boldsymbol{R}$	3-D	Rückproj.	$\hat{\sigma}$
0.0	vor	0	0	0	0	0	0	0	0.416	
	And.	1.49	1.51	0.395	0.339	0.380	0.000175896	0.0821289	0.392	0.28
0.1	vor	1.01	1.04	0.00103	0.000905	0.254	0.00032871	0.168643	1.35	
	And.	0.642	0.591	0.421	0.459	0.170	-0.000152131	-0.0828798	0.393	0.28
0.2	vor	1.89	2.48	0.00180	0.00203	0.596	0.000741	0.328715	2.79	
	And.	-0.313	-0.935	0.261	0.506	-0.184	-0.000550443	-0.243822	0.393	0.28
0.3	vor	3.60	2.53	0.00366	0.00303	1.06	0.000953044	0.504261	3.88	
	And.	-1.94	-0.905	0.669	0.624	-0.601	-0.000683549	-0.411189	0.392	0.28
0.6	vor	6.60	6.54	0.00707	0.00605	1.90	0.00141237	1.03143	5.26	
	And.	-4.71	-4.72	0.367	0.520	-1.31	-0.0011936	-0.910241	0.393	0.29
1.0	vor	11.3	8.81	0.0101	0.0106	2.29	0.00277561	1.63008	11.2	
	And.	-9.37	-6.81	0.634	0.393	-1.74	-0.00253202	-1.53985	0.393	0.29

Tabelle D.37: Szene 2: Verrauschte Parameter.

σ_P		f_x	f_y	u_0	v_0	$\boldsymbol{t}$	$\boldsymbol{R}$	3-D	Rückproj.	$\hat{\sigma}$
0.0	vor	0	0	0	0	0	0	0	0.428	
	Änd.	1.76	1.90	0.297	0.309	0.538	0.000145087	0.162121	0.404	0.29
0.1	vor	1.13	0.902	0.000894	0.000950	0.289	0.000398219	0.182523	1.66	
	Änd.	0.944	1.38	0.379	0.520	0.350	-0.000183498	-0.0177497	0.403	0.29
0.2	vor	1.77	1.57	0.00191	0.00230	0.535	0.000666253	0.314953	2.87	
	Änd.	0.311	0.678	0.271	0.280	-8e-06	-0.00037622	-0.0931763	0.404	0.29
0.3	vor	2.75	2.31	0.00301	0.00258	0.663	0.000911216	0.514053	3.78	
	Änd.	-1.19	-0.632	0.339	0.332	-0.127	-0.000605298	-0.281837	0.404	0.29
0.6	vor	6.68	5.76	0.00534	0.00487	1.69	0.00245697	0.944314	9.19	
	Änd.	-4.98	-3.94	0.268	0.288	-1.04	-0.00206461	-0.646599	0.404	0.29
1.0	vor	8.40	9.30	0.0109	0.0110	2.69	0.00350562	1.66824	12.4	
	Änd.	-2.89	-3.54	0.397	0.445	-0.441	-0.00307265	-1.28699	0.404	0.29

Tabelle D.38: Szene 3: Verrauschte Parameter.

σ_P		f_x	f_y	u_0	v_0	$\boldsymbol{t}$	$\boldsymbol{R}$	3-D	Rückproj.	$\hat{\sigma}$
0.0	vor	0	0	0	0	0	0	0	0.422	
	Änd.	1.81	1.84	0.432	0.341	0.526	0.000180275	0.144702	0.400	0.29
0.1	vor	0.860	1.20	0.000986	0.000820	0.321	0.000322857	0.181059	1.26	
	Änd.	0.957	0.624	0.417	0.387	0.216	-0.000113196	-0.03505	0.400	0.29
0.2	vor	2.73	2.75	0.00173	0.00208	0.624	0.000631219	0.354496	2.87	
	Änd.	-0.932	-0.955	0.253	0.307	-0.0773	-0.000448784	-0.198855	0.400	0.29
0.3	vor	3.02	3.19	0.00267	0.00288	0.940	0.00130576	0.478181	5.32	
	Änd.	-1.40	-1.55	0.456	0.485	-0.481	-0.00109876	-0.332552	0.400	0.29
0.6	vor	7.34	6.92	0.00642	0.00678	1.82	0.00234131	1.04724	9.43	
	And.	-5.36	-4.90	0.832	0.337	-1.26	-0.00202904	-0.889064	0.400	0.29
1.0	vor	8.54	10.9	0.00683	0.00937	3.89	0.00365893	1.70719	15.8	
	Änd.	-6.95	-9.29	0.500	0.403	-3.31	-0.00337136	-1.50397	0.400	0.29

Tabelle D.39: Szene 4: Verrauschte Parameter.

σ_P		f_x	f_y	u_0	v_0	$\boldsymbol{t}$	$\boldsymbol{R}$	3-D	Rückproj.	$\hat{\sigma}$
0.0	vor	0	0	0	0	0	0	0	0.433	
	Änd.	0.343	0.398	0.154	0.205	0.179	0.000132436	0.179012	0.409	0.30
0.1	vor	0.849	0.967	0.00103	0.000895	0.295	0.000346911	0.164988	0.886	
	Änd.	0.293	0.122	0.163	0.176	0.00254	-0.000191445	0.0102889	0.409	0.30
0.2	vor	2.26	2.19	0.00193	0.00211	0.834	0.000442003	0.33494	1.43	
	Änd.	0.520	0.665	0.346	0.182	-0.0706	-0.000196138	-0.154664	0.410	0.30
0.3	vor	3.55	3.02	0.00210	0.00249	0.915	0.000997179	0.4983	1.83	
	Änd.	-0.900	-0.450	0.446	0.286	-0.190	-0.000779615	-0.312634	0.409	0.30
0.6	vor	5.67	7.48	0.00536	0.00731	1.97	0.0017121	1.05039	3.55	
	Änd.	-0.607	-2.47	0.811	0.666	-0.339	-0.00129381	-0.764849	0.419	0.30
1.0	vor	7.54	14.0	0.00844	0.00899	3.14	0.0030505	1.74894	7.13	
	Änd.	0.415	-6.14	0.754	0.696	-1.05	-0.00262189	-1.54612	0.437	0.32

Tabelle D.40: Szene 5: Verrauschte Parameter.

σ_P		f_x	f_y	u_0	v_0	$\boldsymbol{t}$	$\boldsymbol{R}$	3-D	Rückproj.	$\hat{\sigma}$
0.0	vor	0	0	0	0	0	0	0	0.423	
	Änd.	0.618	0.531	0.190	0.199	0.172	0.000147863	0.164417	0.400	0.29
0.1	vor	0.992	1.59	0.000919	0.00120	0.336	0.000248448	0.168738	0.913	
	Änd.	0.359	-0.343	0.196	0.199	0.00662	-9.92833e-05	-0.00466505	0.400	0.29
0.2	vor	1.88	2.61	0.00194	0.00164	0.460	0.000495793	0.332997	1.27	
	Änd.	-0.165	-0.862	0.373	0.513	0.0404	-0.00024217	-0.163383	0.400	0.29
0.3	vor	3.45	3.49	0.00293	0.00313	0.858	0.000790136	0.538698	2.64	
	And.	-1.35	-1.24	0.458	0.371	-0.0661	-0.000539231	-0.293716	0.401	0.29
0.6	vor	3.66	5.56	0.00422	0.00560	1.46	0.00187431	0.990485	4.71	
	Änd.	0.291	-1.73	0.476	0.687	-0.338	-0.00152076	-0.788734	0.407	0.30
1.0	vor	10.7	11.4	0.00728	0.0111	2.53	0.00281145	1.58936	6.45	
	Änd.	-4.09	-4.91	0.402	0.588	-0.635	-0.00229757	-1.27147	0.419	0.30

Tabelle D.41: Szene 6: Verrauschte Parameter.

σ_P		f_x	f_y	u_0	v_0	$\boldsymbol{t}$	$\boldsymbol{R}$	3-D	Rückproj.	$\hat{\sigma}$
0.0	vor	0	0	0	0	0	0	0	0.422	
	Änd.	0.903	0.780	0.485	0.404	0.253	0.000203842	0.0983554	0.397	0.29
0.1	vor	1.06	0.786	0.00109	0.00102	0.392	0.000332786	0.168878	1.25	
	Änd.	0.0717	0.217	0.464	0.489	-0.0203	-0.000124701	-0.0674907	0.397	0.29
0.2	vor	2.01	1.96	0.00195	0.00186	0.466	0.000674808	0.371308	1.78	
	Änd.	-0.313	-0.140	0.412	0.442	0.117	-0.000464859	-0.26952	0.397	0.29
0.3	vor	2.87	3.21	0.00271	0.00232	0.922	0.000933864	0.536844	2.88	
	And.	-0.553	-0.896	0.607	0.696	-0.0994	-0.000603493	-0.421858	0.396	0.29
0.6	vor	5.76	6.56	0.00624	0.00561	2.08	0.00170224	1.00764	6.17	
	Änd.	-1.36	-2.39	0.677	0.990	-0.603	-0.00133453	-0.889076	0.399	0.29
1.0	vor	7.23	11.1	0.00937	0.0102	3.35	0.00295521	1.66004	11.2	
	Änd.	0.194	-3.76	0.979	1.08	-0.913	-0.00248001	-1.52473	0.406	0.29

Tabelle D.42: Szene 7: Verrauschte Parameter.

σ_P		f_x	f_y	u_0	v_0	$\boldsymbol{t}$	$\boldsymbol{R}$	3-D	Rückproj.	$\hat{\sigma}$
0.0	vor	0	0	0	0	0	0	0	0.426	
	Änd.	0.765	0.661	0.292	0.347	0.243	0.000179056	0.106385	0.402	0.29
0.1	vor	0.956	0.981	0.000830	0.00106	0.342	0.00030608	0.163263	0.873	
	Änd.	0.135	0.0313	0.228	0.321	-0.00518	-0.000129431	-0.0567454	0.403	0.29
0.2	vor	1.95	2.01	0.00201	0.00190	0.684	0.0006895	0.342145	1.60	
	Änd.	0.128	-0.0562	0.836	0.668	-0.0617	-0.000344506	-0.234189	0.402	0.29
0.3	vor	2.55	3.12	0.00288	0.00213	1.31	0.000953367	0.481613	2.57	
	Änd.	0.660	0.0412	0.610	0.516	-0.254	-0.000683861	-0.373184	0.406	0.29
0.6	vor	6.72	7.71	0.00502	0.00673	2.29	0.00152914	1.0574	5.55	
	Änd.	-2.21	-3.16	0.521	0.707	-0.521	-0.00125783	-0.947239	0.406	0.29
1.0	vor	9.50	9.98	0.0109	0.0107	3.24	0.00270812	1.75341	10.8	
	Änd.	-2.37	-2.96	0.967	0.704	-0.914	-0.00237203	-1.64011	0.412	0.30

Tabelle D.43: Szene 8: Verrauschte Parameter.

D.5.2 Selbstkalibrierung

	f_x	f_y	u_0	v_0	$\boldsymbol{t}$	$\boldsymbol{R}$	3-D	Rückproj.	$\hat{\sigma}$
vor	24.0	23.6	26.0	46.3	6.55162	0.0129897	0.641065	0.428/1.02	
Änd.	7.73	6.76	-3.00	-0.962	2.62998	-0.000899298	0.0318594	0.617	0.45
vor	41.2	42.5	59.0	48.5	12.2941	0.0183597	0.919137	0.428/1.62	
Änd.	4.49	7.99	-15.4	-30.1	1.87166	-0.00808295	-0.177887	0.544	0.39
vor	36.5	33.3	62.5	58.6	10.4249	0.0209391	1.04411	0.428/1.72	
Änd.	1.05	5.12	-0.990	-24.8	1.22482	-0.00419473	-0.124317	0.733	0.53
vor	41.3	38.5	46.6	85.0	11.6185	0.0234247	1.12168	0.428/1.87	
Änd.	18.0	16.0	-11.8	-0.770	5.45923	-0.00251345	0.0124091	0.937	0.68
vor	22.8	20.5	33.1	32.3	5.86078	0.0115331	0.596901	0.428/0.925	
Änd.	-5.49	-5.40	-0.991	-13.3	-1.22463	-0.00238251	-0.0858508	0.515	0.37

Tabelle D.44: Szene 1: Selbstkalibrierungseffekt des Bündelausgleichs.

	f_x	f_y	u_0	v_0	$\boldsymbol{t}$	$\boldsymbol{R}$	3-D	Rückproj.	$\hat{\sigma}$
vor	25.7	24.7	36.1	34.1	6.80027	0.0117269	0.612026	0.416/0.773	
Änd.	-11.0	-9.83	-15.4	-14.9	-2.78463	-0.00504536	-0.255484	0.445	0.32
vor	78.4	80.5	89.3	92.1	23.0485	0.0295046	1.5028	0.416/1.99	
Änd.	-4.47	-7.31	-24.0	-4.78	-1.425	-0.00401148	0.0616582	0.740	0.54
vor	16.4	16.9	23.8	19.1	4.3213	0.00726101	0.379075	0.416/0.571	
Änd.	-2.02	-1.84	-1.12	-2.43	-0.43966	-0.000544102	-0.00711423	0.442	0.32
vor	22.8	24.0	38.8	39.7	6.22336	0.0131792	0.631898	0.416/0.725	
Änd.	-5.87	-6.12	-9.82	-4.61	-1.38813	-0.00238399	-0.0725427	0.476	0.35
vor	29.5	30.1	51.1	38.8	7.88872	0.015146	0.762815	0.416/1.20	
Änd.	-1.13	-0.328	-19.0	-19.6	0.232415	-0.00644804	-0.231244	0.497	0.36

Tabelle D.45: Szene 2: Selbstkalibrierungseffekt des Bündelausgleichs.

	f_x	f_y	u_0	v_0	$\boldsymbol{t}$	$\boldsymbol{R}$	3-D	Rückproj.	$\hat{\sigma}$
vor	7.51	6.75	18.8	16.7	2.11842	0.00641699	0.265783	0.428/0.486	
Änd.	-1.82	-1.67	-3.50	-0.630	-0.520262	-0.000638891	0.0247056	0.419	0.30
vor	37.6	37.6	12.4	14.5	11.2272	0.00393993	0.5111	0.428/0.475	
Änd.	3.66	4.08	-0.135	-0.165	0.991833	-0.000240833	0.0559248	0.415	0.30
vor	44.6	44.4	34.9	33.1	13.7429	0.0114203	0.70364	0.428/0.943	
Änd.	-6.29	-5.97	-6.26	-10.0	-2.07418	-0.00319607	-0.0202644	0.487	0.35
vor	70.9	67.7	58.6	70.9	22.4855	0.0213702	1.33122	0.428/1.91	
Änd.	25.3	28.0	-1.50	-2.49	6.56857	-0.00180918	0.168355	0.795	0.58
vor	21.5	21.1	25.3	26.8	6.58001	0.00914557	0.462799	0.428/0.754	
Änd.	-2.17	-1.89	-4.26	-7.42	-0.609173	-0.00236284	0.0300847	0.457	0.33

Tabelle D.46: Szene 3: Selbstkalibrierungseffekt des Bündelausgleichs.

	f_x	f_y	u_0	v_0	$\boldsymbol{t}$	$\boldsymbol{R}$	3-D	Rückproj.	$\hat{\sigma}$
vor	36.7	37.3	98.8	62.7	12.865	0.0281496	1.26911	0.422/2.21	
Änd.	36.6	34.8	-0.763	-2.68	10.7725	-0.0034406	0.241102	0.875	0.63
vor	93.6	93.3	36.8	30.4	26.8374	0.00975353	0.899746	0.422/0.773	
Änd.	-11.3	-10.6	-10.3	-3.65	-3.70184	-0.00338968	-0.0118376	0.454	0.33
vor	44.4	45.4	48.9	59.1	11.202	0.0196399	0.980893	0.422/1.64	
Änd.	-16.6	-16.7	-0.970	-1.68	-3.08632	-0.00450198	0.247192	0.745	0.54
vor	14.0	14.5	15.0	22.1	3.71817	0.00666054	0.314896	0.422/0.640	
Änd.	-3.94	-4.05	-0.881	-0.550	-0.692879	-0.00123269	0.130006	0.466	0.34
vor	57.8	58.0	33.6	35.6	15.7792	0.0123842	0.809673	0.422/0.986	
Änd.	-10.3	-9.85	-0.850	-1.15	-2.59997	-0.00293661	0.14776	0.551	0.40

Tabelle D.47: Szene 4: Selbstkalibrierungseffekt des Bündelausgleichs.

	f_x	f_y	u_0	v_0	$\boldsymbol{t}$	$\boldsymbol{R}$	3-D	Rückproj.	$\hat{\sigma}$
vor	61.0	58.8	101	43.6	17.9755	0.0261124	1.32788	0.433/2.41	
Änd.	-0.105	1.05	-0.272	-4.76	2.95643	0.00705659	1.97013	0.903	0.66
vor	56.0	56.3	42.2	47.9	17.3749	0.0142749	0.838109	0.433/1.03	
Änd.	1.67	0.721	-8.54	-0.373	0.462087	-0.00189815	0.0994127	0.62	0.45
vor	56.2	56.0	52.0	43.5	17.1969	0.0154169	0.896579	0.433/1.19	
Änd.	2.02	1.03	-0.462	-17.7	-0.159058	-0.00107353	0.273278	0.593	0.43
vor	17.8	18.2	10.2	21.8	5.04912	0.00595276	0.322596	0.433/0.573	
Änd.	-0.0116	0.0515	-0.441	-0.274	0.400148	-0.000651224	0.205753	0.452	0.33
vor	41.8	42.6	23.3	26.9	12.3818	0.00828576	0.65575	0.433/0.687	
Änd.	0.259	0.0537	-5.42	-12.4	0.537211	-0.00256634	0.254189	0.444	0.32

Tabelle D.48: Szene 5: Selbstkalibrierungseffekt des Bündelausgleichs.

	f_x	f_y	u_0	v_0	$\boldsymbol{t}$	$\boldsymbol{R}$	3-D	Rückproj.	$\hat{\sigma}$
vor	66.3	65.4	21.9	41.0	20.2084	0.00953096	0.822791	0.423/1.19	
Änd.	1.09	1.21	-1.76	-20.0	0.287049	-0.00440694	0.0891776	0.479	0.35
vor	42.2	43.2	80.4	47.2	14.9508	0.0217852	1.12116	0.423/2.39	
Änd.	1.26	-0.0359	-0.484	-10.2	1.31089	0.00346724	1.26579	0.700	0.51
vor	23.8	23.3	26.9	48.5	4.48803	0.0143008	0.693235	0.423/1.40	
Änd.	0.164	-1.34	1.28	-0.371	1.03573	-0.00201632	0.498626	0.712	0.52
vor	48.2	47.9	54.4	42.4	14.1473	0.0169443	0.996603	0.423/1.10	
Änd.	0.369	-0.685	-12.6	-1.12	0.716152	0.000313242	0.793429	0.622	0.45
vor	85.6	73.5	118	72.5	16.0909	0.0373262	1.89736	0.423/3.12	
Änd.	-6.11	-0.122	-0.413	2.56	1.15201	0.00251809	0.973995	1.05	0.76

Tabelle D.49: Szene 6: Selbstkalibrierungseffekt des Bündelausgleichs.

	f_x	f_y	u_0	v_0	$\boldsymbol{t}$	$\boldsymbol{R}$	3-D	Rückproj.	$\hat{\sigma}$
vor	26.0	28.5	27.3	36.3	9.2772	0.0111382	0.512291	0.422/0.576	
Änd.	-1.83	-3.49	-7.38	-15.5	-0.416961	-0.00413314	-0.154454	0.422	0.31
vor	37.0	36.5	23.2	23.8	12.1631	0.00802949	0.480833	0.422/0.526	
Änd.	-0.106	0.214	-0.421	-0.903	0.0567206	-0.000261656	-0.0140199	0.422	0.31
vor	57.0	62.8	55.3	85.4	21.2866	0.0245726	1.07039	0.422/1.23	
Änd.	0.563	-2.99	-16.0	-37.7	-0.113916	-0.00950721	-0.291609	0.530	0.38
vor	78.8	77.3	41.4	56.4	25.4907	0.0174883	1.10143	0.422/0.833	
Änd.	-3.84	-2.55	-10.8	-0.631	-1.44032	-0.00243269	-0.0956404	0.53	0.38
vor	25.1	24.4	11.2	12.8	8.08121	0.00414721	0.296763	0.422/0.437	
Änd.	-0.145	0.206	-1.27	-0.225	-0.0512972	-0.000323338	0.00570609	0.400	0.29

Tabelle D.50: Szene 7: Selbstkalibrierungseffekt des Bündelausgleichs.

	f_x	f_y	u_0	v_0	$\boldsymbol{t}$	$\boldsymbol{R}$	3-D	Rückproj.	$\hat{\sigma}$
vor	59.1	57.7	38.4	86.8	22.5064	0.0237249	1.01091	0.426/1.36	
Änd.	0.103	0.908	0.415	-0.189	-0.304015	-0.0012155	0.0839276	0.576	0.42
vor	35.1	34.3	64.3	34.7	13.9728	0.0184507	0.702906	0.426/0.860	
Änd.	0.0914	0.394	-28.4	-5.97	0.0459775	-0.0069319	-0.23467	0.499	0.36
vor	77.6	81.6	94.2	53.5	29.6281	0.02642	1.1558	0.426/1.65	
Änd.	4.26	0.0676	-0.893	-13.6	0.337456	-0.00224772	0.00495408	0.731	0.53
vor	13.8	13.4	18.9	13.8	4.22497	0.00618296	0.290633	0.426/0.534	
Änd.	-1.21	-0.705	-0.919	-4.72	-0.132278	-0.00126128	-0.0205657	0.422	0.31
vor	90.4	91.4	71.6	50.4	32.6981	0.0204623	1.15664	0.426/1.14	
Änd.	0.828	2.17	-22.0	-1.60	0.568284	-0.00410299	-0.0412805	0.565	0.41

Tabelle D.51: Szene 8: Selbstkalibrierungseffekt des Bündelausgleichs.

Index

Bündelausgleich, 29–31, 34, 39, 40, 43

Quaternionen, 33, 93