UNIVERSITÄT
KOBLENZ · LANDAU
Fachbereich 4: Informatik

3D-Rekonstruktion aus monokularen Bilderserien

Bachelorarbeit
zur Erlangung des Grades
Bachelor of Science
im Studiengang Computervisualistik

vorgelegt von

Stephan Manthe

Betreuer: Dipl.-Inform. Frank Neuhaus, Institut für Computervisualistik, Fachbereich Informatik, Universität Koblenz-Landau
Erstgutachter: Prof. Dr.-Ing. Dietrich Paulus, Institut für Computervisualistik, Fachbereich Informatik, Universität Koblenz-Landau
Zweitgutachter: Dipl.-Inform. Frank Neuhaus, Institut für Computervisualistik, Fachbereich Informatik, Universität Koblenz-Landau

Koblenz, im September 2014

Kurzfassung

3D-Modelle werden heute in vielen Bereichen wie Multimedia Anwendungen, Robotik oder der Filmindustrie immer wichtiger. Besonders interessant ist dabei die Erstellung eines 3D-Modells aus einer monokularen Bilderserie, da die hierfür nötigen Kameras immer günstiger, kleiner und ausgereifter produziert werden. Geeignetere Kameras werden in immer mehr Geräten wie Smartphones, Tablet-PCs, Autos etc. verbaut, wodurch sich ein großes Potential für die Verwendung dieser Rekonstruktionstechnik ergibt.

Als Grundlage dieser Arbeit dient eine mit einer kalibrierten Kamera aufgenommene Bilderserie. Aus dieser werden 2D-Punktkorrespondenzen, mit den verbreiteten SURF-Features oder den A-KAZE-Features gewonnen. Aufbauend auf den 2D-Punktkorrespondenzen kann aus diesen mit Hilfe verschiedener Algorithmen ein 3D-Modell in Form einer Punktwolke und Kameraposen rekonstruiert werden. Um Fehler in dem entstandenen Modell gering zu halten, wird insbesondere auf den Bündelausgleich zur Fehlerminimierung eingegangen. Anschließend wird das neben dieser Arbeit entstandene Programm zur 3D-Rekonstuktion und Visualisierung des 3D-Modells erläutert. Das implementierte System wird anschließend anhand von Statistiken evaluiert und die hieraus gewonnenen Erkentnisse präsentiert. Abschließend werden die Ergebnisse dieser Arbeit zusammengefasst und ein Ausblick auf mögliche Weiterentwicklungen gegeben.

Abstract

3D-models are getting more important in many areas such as multimedia applications, robotics or film industries. Of particular interest is the creation of 3D-models from a series of monocular images. This is because the cameras that are required for this purpose are becoming cheaper, smaller and more sophisticated at the same time. Increasingly often, suitable cameras are already integrated in devices like smartphones, tablet PCs or cars for example. Hence, there is a great potential for applications of this reconstruction technique.

This thesis is based on the use of a series of images that were taken with a calibrated camera. The first step is to extract point correspondences from this image series making use of the well-known SURF- and A-KAZE-features. Starting from the point correspondences, it is possible to reconstruct a 3D-Modell with different algorithms that consists of a point cloud and camera poses. To reduce errors in the 3D-model, this thesis especially focuses on explaining the bundle adjustment algorithm, which is being used for a non-linear error minimization of a cost function. The thesis also introduces the application for the 3D-reconstruction and the visualization of the results, that was developed in the course of this thesis. The implemented system is evaluated based on statistics and the newly aquired knowledge is presented. The thesis concludes with a summary of its results, and a number of ideas for potential future applications and developments.

Danksagung

Ich möchte mich ganz besonders bei Professor Dietrich Paulus und Frank Neuhaus bedanken, die mir die Möglichkeit gegeben haben meine Bachelorarbeit mit meinem Wunschthema anzufertigen. Mein besonderer Dank gilt aber Frank Neuhaus der mir während der Anfertigung der Bachelorarbeit als Betreuer jederzeit für Fragen zur Verfügung gestanden hat und mir hilfreiche Tipps geben konnte. Auch die Diskussionen mit Ihm waren oft sehr spannend und hilfreich. Weiterhin möchte ich mich bei meine Eltern, meinem Bruder und meinen Freunden bedanken, die mich bei der Arbeit und auf dem Weg durch das Bachelorstudium unterstützt haben.

Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen wurde. Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

Die Vereinbarung der Arbeitsgruppe für Studien- und Abschlussarbeiten habe ich gelesen und anerkannt, insbesondere die Regelung des Nutzungsrechts.

Mit der Einstellung dieser Arbeit in die Bibliothek bin ich einverstanden.	ja ☐	nein ☐
Der Veröffentlichung dieser Arbeit im Internet stimme ich zu.	ja ☐	nein ☐

Koblenz, den 28. September 2014

Inhaltsverzeichnis

Kapitel 1

Einleitung

In der aktuellen Entwicklung der Computertechnik werden dreidimensionale Modelle, im folgenden 3D-Modelle genannt, immer wichtiger. Diese Modelle sind ein reduziertes Abbild der realen Welt, das für die entsprechende Anwendung nur relevante Informationen über die Umgebung enthält. Sie werden in vielen Anwendungsbereichen genutzt. Beispiele hierfür sind die Navigation und Kartierung, das Einblenden von Spezialeffekten in Filmen, das Vermessen von einzelnen Objekten oder die Verknüpfung von computergenerierten Informationen und der Realität in Augmented Reality Anwendungen. Die Modelle sollen dabei je nach Anwendungsfall möglichst präzise, schnell und optisch ansprechend erstellt werden.

Aktive Sensoren wie Laserscanner, Time of Flightkameras oder Techniken wie strukturiertes Licht erfüllen diese Erwartungen sehr gut. Der Nachteil bei diesen Verfahren ist jedoch, dass sie die Umgebung aktiv durch das Aussenden von Licht verändern, um diese zu Vermessen. Das ist bei verschiedenen Anwendungen nicht von Vorteil und kann die Handhabung der Sensoren umständlich machen. Aus monokularen Bildserien, wie sie mit digitalen Film- oder Fotokameras erstellt werden, lassen sich ebenfalls 3D-Modelle rekonstruieren. Der Vorteil dieser Methode ist, dass nicht aktiv in die zu vermessende Umgebung eingegriffen werden muss. Bei diesem Verfahren müssen Bilder einer Szene aus verschiedenen Perspektiven aufgenommen werden. Wobei unter einer Szene ein Ausschnitt aus der realen Welt verstanden wird. Aus den aufgenommenen Bildern lässt sich eine 3D-Punktwolke rekonstruieren, sowie die Position und Ausrichtung, der Kameras bestimmen, mit der die Bilder aufgenommen worden sind. Diese Verfahren sind allerdings rechenaufwendiger, da sie ohne aktive Hilfsmittel arbeiten, die die Vermessung vereinfachen. Weiterhin können die erstellten Modelle nicht vernachlässigbare Ungenauigkeiten aufweisen, die durch eine Optimierung des Modells verbessert werden sollten.

Dass die 3D-Rekonstruktion aus monokularen Bildserien interessante Möglichkeiten bietet, zeigen Agarwal u.a. in [ASS+09]. Sie nutzen die große Sammlung

an Fotos einer Fotocomunity um Rom zu rekonstruieren. Dieses Vorgehen ermöglicht die 3D-Rekonstruktion von Szenen, aus bereits bestehenden digitalen Bildern, die man selbst nicht fotografiert haben muss. Ein anderer Anwendungsfall ist die Selbstlokalisierung von Robotern. Speziell für Flugroboter oder auch *MAVs* (micro air vehicle) bietet die Positionsbestimmung durch Kameras den Vorteil, dass diese viel kompakter und mit geringerem Gewicht als z. B. Laserscanner gebaut werden können.

1.1 Ziel dieser Arbeit

Die 3D-Rekonstruktion aus monokularen Bildserien lässt sich grob in drei Teilaufgaben unterteilen. Dazu gehört die Korrespondenzbildung zwischen Bildpunkten aus zwei Bildern, das initiale Erstellen einer 3D-Rekonstruktion, sowie das nachträgliche Optimieren des entstandenen 3D-Models. Diese Arbeit soll dabei helfen die grundlegenden Methodiken und Vorgehensweisen der drei Abschnitte zu verstehen. Zudem soll sie ein Gefühl für die Problematik der zu lösenden Probleme und das Zusammenspiel der einzelnen Komponenten vermitteln. Dabei sollen bestehende Problematiken und mögliche Problemlösungen beschrieben werden. Weiterhin soll zu dieser Arbeit ein C++-Framework entwickelt werden, das es ermöglicht eine 3D-Modell mit einer digitalen Fotokamera zu rekonstruieren. Wobei sich in dieser Arbeit das 3D-Modell aus einer 3D-Punktwolke und den Posen der Kameras zusammensetzt, mit denen die Bilder für die Rekonstruktion aufgenommen worden sind. Nach der 3D-Rekonstruktion soll dass entstandene Modell visualisiert werden.

Damit das Modell möglichst gut zu erkennen ist, soll die rekonstruierte Punktwolke möglichst dicht und genau sein. Zusätzlich soll bei der Entwicklung des Frameworks Wert darauf gelegt werden, verschiedene Parameter einzustellen, um deren Effekte zu veranschaulichen. Abschließend soll das entstandene Framework anhand von Experimenten getestet und die daraus gewonnenen Ergebnisse präsentiert werden.

1.2 Stand der Wissenschaft

Dieses Kapitel beschreibt den aktuellen Stand der Wissenschaft zweier ausgewählter Themen, die für diese Arbeit wichtig sind. Dabei wird auf die Merkmalsextraktion aus Bildern und deren Beschreibung eingegangen, die die Grundlage für die 3D-Rekonstruktion in dieser Arbeit liefern werden. Als zweites werden aktuelle Optimierungsverfahren vorgestellt, die für den Bündelausgleich wichtig sind.

Dieser wird benötigt, um fehlerhafte Berechnungen im rekonstruierten 3D-Modell zu minimieren.

1.2.1 Merkmalsextraktion und Korrespondenzbildung

Die Wiedererkennung von korrespondierenden Punkten in Bildern spielt in vielen Bereichen wie Objekterkennung, 3D-Rekonstruktion [ASS+09], Roboternavigation [FPS14], Filmindustrie [KTS11] etc. eine wichtige Rolle. Die Entwicklungen beziehen sich hierbei auf das Finden und Beschreiben von markanten Punkten durch einen sogenannten Merkmalsdetektor bzw. Merkmalsdeskriptor. Der Merkmalsdetektor hat die Aufgabe, eine möglichst große Anzahl von markanten Punkten zu finden. Ein markanter Punkt ist dabei ein Punkt, der sich dadurch auszeichnet, dass er in einem Bild mit leicht veränderten Aufnahmebedingungen wieder gefunden wird. Ein solcher markanter Punkt wird dabei von einem Merkmalsdeskriptor beschrieben. Er soll einen Punkt durch seine Umgebung möglichst präzise beschreiben um ihn von anderen unterscheiden zu können. Der Merkmalsdeskriptor ist dabei ein hochdimensionaler Vektor, der Informationen über die Umgebung eines markanten Punktes speichert.

Zwischen zwei Mengen von extrahierten Punkten aus zwei verschiedenen Bildern müssen schließlich Punktkorrespondenzen gebildet werden. Dazu wird anhand der Merkmalsdeskriptoren entschieden, ob eine Korrespondenzbildung zwischen zwei Punkten stattfindet oder nicht. Wichtig bei der Korrespondenzbildung ist, dass möglichst viele richtige Korrespondenzen und möglichst wenig falsche gefunden werden, damit eine 3D-Rekonstruktion stattfinden kann.

Merkmalsdetektion David G. Lowe stellt mit *SIFT* (scale invariant feature transform) einen Merkmalsdetektor und -deskriptor bereit. Beide sind skalierungs- und rotationsinvariant, sowie bedingt robust gegen unterschiedliche Beleuchtungssituationen. Die Skalierungsinvarianz erhält der Deskriptor durch seine Berechnung in einem Skalenraum, der ein Bild in unterschiedlichen Auflösungen und Glättungen enthält. Durch Berechnung der Ausrichtung des markanten Punktes wird schließlich die Rotationsinvarianz erreicht [Low99]. SIFT ist heute immer noch ein sehr verbreiteter Merkmalsdeskriptor und -detektor, auf dessen Vorgehen viele später entwickelte Deskriptoren aufbauen.

Mit *SURF* (Speeded up robust features) wurde ein weiterer Merkmalsdeskriptor und -detektor entwickelt. Das Ziel von SURF ist es sich schneller als SIFT berechnen zulasssen, dabei aber genau so präzise zu sein. Seinen Geschwindigkeitszuwachs erhält SURF durch die Approximierung des für die Glättung verwendeten Gauß-Filters durch Boxfilter. Deren Faltung mit einem Bild lässt sich mit Hilfe von Integralbildern sehr schnell berechnen. Zusätzlich wird die Dimension des berechneten Merkmalsvektors reduziert, der einen markanten Punkt beschreibt. Die

Methode, durch die SURF die Skalierungs- und Rotationsinvarianz erhält, ähnelt ebenfalls den Methoden von SIFT.

Ein sehr aktueller aber noch nicht weit verbreiteter Merkmalsdeskriptor ist *KAZE* bzw. *A-KAZE* (Accelerated KAZE), die beschleunigte Variante von KAZE. KAZE basiert auf den selben Ansätzen wie SIFT und SURF, mit dem Unterschied, dass die Filterung des Skalenraums nicht über einen linearen Gauß-Filter durchgeführt wird sondern über einen nichtlinearen Filter. Dieser soll wichtige Merkmale wie Kanten und Ecken erhalten, die bei einer Gaußfilterung verloren gehen können. Durch die Erhaltung der Details, soll die Leistung des Detektors und Deskriptors verbessert werden. Als Deskriptor wird von KAZE *MSURF* (modified SURF), ein angepasster SURF Deskriptor verwendet [ABD12]. Nachteil der von KAZE verwendeten nichtlinearen Glättung des Bildes ist der hohe Rechenaufwand. Diesen Nachteil versucht A-KAZE durch die Verwendung von *FED* (fast explicit diffusion) für die nicht lineare Filterung zu kompensieren. Weiterhin wird MSURF durch den modified-Local Difference Binary descriptor ausgetauscht. Er ist ein binärer Deskriptor, der einen markanten Punkt durch einen Vektor von binärwertigen Elementen beschreibt.

Korrespondenzbildung Die Korrespondenzbildung zwischen zwei Mengen von markanten Punkten, lässt sich in zwei Teilaufgaben unterteilen. Zum einen muss entschieden werden, ob eine Korrespondenz zwischen zwei markanten Punkten zustande kommt, zum anderen muss es möglich sein, schnell potentiell „ähnliche“ Merkmale in einer Menge von Merkmalen zu finden [Sze10].

Bei der Entscheidung ob die Korrespondenzbildung zwischen zwei markanten Punkten zustande kommmt, haben sich Methoden, die die nächsten Nachbarn des Anfragepunktes betrachten als zuverlässigt erwiesen. Das Nearest Neighbor Distance Ratio Verfahren vergleicht die Distanzen der beiden nächsten Nachbarn zum Anfragepunkt. Nur wenn der Unterschied zwischen diesen beiden Distanzen größer als ein gegebener Schwellwert ist, wird eine Korrespondenz gebildet. Ist der Unterschied zu klein, so ist keine eindeutig Zuordnung möglich, weil beide nächsten Nachbarn sehr nah am Anfragepunkt liegen.

Eine weitere Möglichkeit ist, nur den nächsten Nachbarn zu betrachten. Ist die Distanz zu ihm kleiner als ein gegebener Schwellwert, so wird eine Korrespondenz gebildet. Mit weiteren Nachbarn deren Distanz ebenfalls kleiner als der Schwellwert ist, wird keine Korrespondenz gebildet [Low04, MS05].

Beim Auffinden von möglichen Korrespondenzen ist die naive Variante jedes Merkmal mit jedem anderen Merkmal zu vergleichen. Dieses Vorgehen verursacht bei einer hohen Anzahl an Merkmalen einen inakzeptablen Aufwand. Deswegen werden für eine kürzere Berechnungszeit die nächsten Nachbarn nur approximiert.

Das bedeutet, dass unter den nächsten Nachbarn ein Nachbar sein kann, der bei einer exakten Berechnung in dieser Menge nicht enthalten ist.

Eine Möglichkeit zur Approximation des nächsten Nachbarn ist das Einordnen der Merkmale in Suchbäume. Mit derren Hilfe lässt sich eine Menge an Merkmalen ordnen, um diese schneller durchsuchen zu können. Die Softwarelibrary *FLANN*[1] (fast library for approximate nearest neighbors) ordnet die Deskriptoren beispielsweise als Punkte in einem k-d- oder k-Means-Baum ein, die ein schnelles Durchsuchen der Merkmalsmenge ermöglichen. Zusätzlich optimiert sie verschiedene Parameter anhand der eingegebenen Merkmale, um das Erstellen und Durchsuchen der Suchbäume zu beschleunigen.

Eine andere Methode zum durchsuchen von mehrdimensionalen Suchräumen sind Hashingverfahren. Bei diesen berechnen Hashfunktionen für die Punkte im Suchraum Hashes. Diese Funktionen erfüllen die Eigenschaft, dass für nah bei einander liegende Punkte mit hoher Wahrscheinlichkeit der selbe Hash berechnet wird. Alle Punkte, für die der selbe Hash berechnet wurde, werden nun im selben Bucket abgelegt. Die Hashfunktion bildet in einem günstigen Fall einen Anfragepunkt auf den Bucket ab, in dem seine nächsten Nachbarn liegen. Die Buckets lassen sich nun durch die reduzierte Anzahl an Punkten im Bucket leichter durchsuchen, um den nächsten Nachbarn zu finden [Sze10, AI08].

1.2.2 Bündelausgleich

Der Bündelausgleich ist eine Methode, um die nach der initialen Rekonstruktion verbliebenen Ungenauigkeiten zu minimieren. Die Methode wurde bereits in den 1950er Jahren von Duane Brown im Bereich der Photogrammetrie entwickelt. Es ist die Methode, mit der sich eine Szene aus zweidimensionalen Bildern am genausten rekonstruieren lässt [Sze10, TMHF00]. Eine 3D-Rekonstruktion wird hierbei als genau angenommen, sobald die Summe aller Abstände zwischen den detektierten markanten Punkten aus den Aufnahmebildern und den korrespondierenden auf das Bild projizierten Weltpunkten minimal ist. Der Abstand zwischen einem auf das Bild projizierten Weltpunkt und dessen korrespondierenden detektierten markanten Punkt wird auch als Rückpojektionsfehler bezeichnet.

Zur Minimierung des Rückprojektionsfehlers ist der Levenberg-Marquardt-Algorithmus eine weit verbreitete Methode [ASSS10, KTS11, KM09, LA09, TMHF00]. Er benutzt zur Minimierung eine Kombination aus Gauß-Newton-Verfahren und Gradientenabstieg. Der Levenberg-Marquardt-Algorithmus lässt sich speziell an das Problem des Bündelausgleichs anpassen. Bei den Anpassungen werden bestimmte Eigenschaften des Problems ausgenutzt, um den Algorithmus zu beschleunigen. Sparse Bundle Adjustment nutzt z. B. die dünne Besetztheit von Matrizen

[1] Website: http://www.cs.ubc.ca/research/flann/

aus, wie sie beim Bündelausgleich entstehen, um den Levenberg-Marquardt-Algorithmus zu beschleunigen [TMHF00, LA09].

Neben dem Levenberg-Marquardt-Algorithmus werden noch andere Methoden zur Optimierung verwendet. Ein Beispiel hierfür ist das Conjugate Gradient Bundle Adjustment. Es benutzt konjugierte Gradienten, die verwendet werden um lineare Gleichungssysteme zu lösen. Das Verfahren ersetzt dabei den Levenberg-Marquardt-Algorithmus bei der Minimierung des Rückprojektionsfehlers [BÅ10].

Die Algorithmen iSAM und iSAM2 (incremental smoothing and mapping) lassen sich ebenfalls für die Minimierung des Rückprojektionsfehlers verwenden. Das besondere an ihnen ist, dass man einem bereits optimierten Modell einfach neue Daten hinzufügen kann. Dabei müssen Berechnungen, die nicht vom Hinzufügen der neuen Daten betroffen sind nicht neu durchgeführt werden. iSAM muss dennoch in periodischen Zeitabständen eine komplette Neuberechnung des Modells durchführen, damit das Hinzufügen neuer Daten effizient bleibt. In iSAM2 sind diese Neuberechnungen durch eine Verbesserung beim Hinzufügen neuer Daten nicht mehr notwendig, was den Algorithmus noch einmal beschleunigt [KRD08, KJR+11, KJR+12].

1.3 Aufbau der Arbeit

Dieser Abschnitt stellt den Aufbau der Arbeit der in insgeamt Kapitel 7 Kapitel gegliedert ist vor. Das folgende Kapitel 2 erklärt Punktdetektoren und -deskriptoren sowie eine Möglichkeit Punktkorrespondenzen zwischen diesen herzustellen. Kapitel 3 geht anschließend auf die Kamerageometrie und die theoretischen Grundlagen für eine 3D-Rekonstruktion ausführlich ein. Den Bündelausgleich, der während der Rekonstruktion eines 3D-Modells ebenfalls verwendet wird, erklärt Kapitel 4. Im Anschluss wird das während dieser Arbeit entstandene Programm zur Rekonstruktion eines 3D-Modells in Kapitel 5 vorgestellt. Die aus Experimenten mit dem implementierten System gewonnenen Ergebnisse werden in Kapitel 6 diskutiert. Abschließend wird in Kapitel 7 die Arbeit zusammengefasst und ein Ausblick auf mögliche vertiefende und weiterführende Themen gegeben.

Kapitel 2

Korrespondenzbildung zwischen Bildmerkmalen

Die Korrespondenzbildung zwischen Bildmerkmalen, die denselben Weltpunkt in unterschiedlichen Bildern repräsentieren, ist für viele Anwendungen in der Bildverarbeitung essentiell. Sie spielen bei der Objekterkennung eine wichtige Rolle, werden für das Zusammensetzen von Bildern zu Panoramen benötigt und auch die später vorgestellten Algorithmen zur Rekonstruktion eines 3D-Models funktionieren nur mit der Eingabe von Merkmalskorrespondenzen [Sze10]. Aus diesem Grund bilden sie für die 3D-Rekonstruktion in dieser Arbeit die Grundlage. Um eine genaue Rekonstruktion zu ermöglichen, müssen die Bildmerkmale präzise lokalisiert werden und die Korrespondenzen korrekt sein.

In den vorgestellten Algorithmen werden als Bildmerkmale einzelne Punkte im Bild verwendet, die durch ihre umliegende Bildregion beschrieben werden. Diese Punkte sollen im folgenden als markante Punkte bezeichnet werden. Die Detektion und die Beschreibung der markanten Punkte werden von einem Punktdetektor bzw. einem Punktdeskriptor durchgeführt. Insbesondere letzterer soll dabei möglichst invariant sein gegenüber Rotationen, Translationen und Skalierungen, sowie robust sein gegen unterschiedliche Beleuchtungsbedingungen. All diese Eigenschaften sind jedoch auch für den Detektor wünschenswert, damit immer die gleichen Punkte als „interessant“ erkannt werden [Sze10]. Besonders wichtig ist dies bei der 3D-Rekonstruktion aus monokularen Bilderserien, da hier derselbe Weltpunkt für eine Rekonstruktion aus verschiedenen Perspektiven aufgenommen und wiedergefunden werden muss. Die Position des markanten Punktes wird hierbei von den vorgestellten Algorithmen subpixelgenau berechnet, um die angestrebte Präzision bei der 3D-Rekonstruktion zu erreichen. Nach der Detektion der markanten Punkte findet deren Beschreibung durch den Merkmalsdeskriptor statt. Dieser Deskriptor berechnet einen hochdimensionalen Merkmalsvektor, der den markanten Punkt möglichst unverwechselbar beschreiben soll.

Der letzte Schritt ist das Bilden von Merkmalspaaren. Hierbei wird abgefragt, welcher Merkmalsvektor aus einer Menge von Merkmalsvektoren zu einem gegebenen Merkmalsvektor am besten passt und ob eine Paarbildung zustande kommt. Bei der Paarbildung muss sichergestellt werden, dass viele korrekte Zuordnungen und wenige falsche entstehen. Dieser Vorgang ist durch die hohe Anzahl möglicher Kombinationen sehr aufwendig, weshalb hier effiziente Datenstrukturen und Algorithmen verwendet werden müssen.

Im folgenden werden SURF und A-KAZE vorgestellt. Sowohl SURF als auch A-KAZE beschreiben jeweils einen Merkmalsdetektor und -deskriptor für markante Punte. Im Anschluss hieran soll kurz auf die Korrespondenzbildung mit *FLANN* und dem *Nearest-Neighbor-Distance-Ratio*-Verfahren eingegangen werden. FLANN bietet dabei die Möglichkeit eine große Menge an markanten Punkten zu durchsuchen und das Nearest-Neighbor-Distance-Ratio-Verfahren beschreibt, wie entschieden werden kann, ob eine Korrespondenz zwischen zwei markanten Punkten zu Stande kommt.

2.1 SURF: Speeded Up Robust Features

SURF (Speeded Up Robust Features) ist ein Detektor und Deskriptor für markante Punkte. SURF ähnelt SIFT schematisch sehr stark, hat dabei aber den Anspruch, bei gleicher Präzision und Robustheit schneller als SIFT zu sein. Deskriptor und Detektor arbeiten auf Grauwertbildern und sind invariant gegen Rotation, Translation sowie Skalierung. Sie sind weiterhin robust gegen Beleuchtungsänderungen.

Die Berechnung des Detektors und Deskriptors findet in einem Skalenraum statt. Er ist aus mehreren Oktaven aufgebaut, die für die Ausgangsgröße eines Glättungsfilters stehen. Jede Oktave enthält weitere Ebenen, die das Ausgangsbild geglättet enthalten. Dabei wird die Ausgangsgröße des Glättungsfilters in jeder Ebene vergrößert, wodurch das Bild immer stärker geglättet wird. Durch die Detektion und Beschreibung der markanten Punkte im Skalenraum, wird SURF skalierungsinvariant, robust gegen leichte Beleuchtungsveränderungen und robust gegen Rauschen.

Detektor Der *Fast-Hessian* Merkmalsdetektor von SURF basiert auf der Hesse-Matrix $\boldsymbol{H}$. Sie wird für jeden Punkt in den einzelnen Ebenen des Skalenraums angenähert. Somit ist die angenäherte Hesse-Matrix $\boldsymbol{H}_{\text{approx}}(\mathsf{p}^p, s)$ am Punkt p^p für eine Ebene s als

$$\boldsymbol{H}_{\text{approx}}(\mathsf{p}^p, s) = \begin{bmatrix} D_{xx}(\mathsf{p}^p, s) & D_{xy}(\mathsf{p}^p, s) \\ D_{xy}(\mathsf{p}^p, s) & D_{yy}(\mathsf{p}^p, s) \end{bmatrix} \tag{2.1}$$

definiert. $D_{xx}(\mathsf{p}^p, s)$ ist dabei die Näherung der zweiten partiellen Ableitung in x-Richtung und $D_{yy}(\mathsf{p}^p, s)$ bzw. $D_{xy}(\mathsf{p}^p, s)$ in y- bzw. xy-Richtung. Die zweite partielle Ableitung wird nur sehr grob durch Filter angenähert, die Boxfiltern ähneln. Diese Filter sollen im folgenden aus Gründen der Einfachheit Boxfilter genannt werden. Boxfilter berechnen den Wert des Zielpixels aus mehreren quadratisch zusammenhängenden Regionen. Die Pixel innerhalb jeder Region sind alle gleich gewichtet und werden aufsummiert um den Wert der Region zu errechnen. Anschließend wird noch einmal die Summe über alle Regionen gebildet, um den Wert des Zielpixels zu berechnen. Vorteil der Boxfilter ist, dass sie sich sehr schnell mit Integralbildern berechnen lassen. Ein Integralbild $I_\sum$ des Bildes I ist als

$$I_\sum(x, y) = \sum_{i=0}^{y} \sum_{j=0}^{x} I(i, j) \tag{2.2}$$

definiert. Der Vorteil eines Integralbildes ist, dass sich mit niedrigem Rechenaufwand die Summe der Pixel von beliebigen Flächen im Bild ausrechnen lässt. Dies kann besonders bei Boxfiltern vorteilhaft ausgenutzt werden [VJ01]. Eine Veranschaulichung eines Boxfilters zur Berechnung von D_{yy} ist in Abbildung 2.1 zu sehen. Durch die Verwendung von Integralbildern erhält SURF einen Geschwindigkeitszuwachs gegenüber SIFT, denn bei SIFT wird die zweite partielle Ableitung nicht so grob durch Boxfilter angenähert wie bei SURF. Statt dessen nähert SIFT die zweite partielle Ableitung mit einem DoG-Filter an, der wesentlich aufwendiger zu berechnen ist.

Um einen markanten Punkt aufgrund einer Hessematrix $\boldsymbol{H}$ zu identifizieren muss die Determinante der Matrix einen Schwellwert t übersteigen. Da die Hessematrix nur approximiert ist, wird die Berechnung der Determinante von $\boldsymbol{H}_{\text{approx}}$ mit

$$\det\left(\boldsymbol{H}_{\text{approx}}\right) = D_{xx}D_{yy} - (0{,}9D_{xy})^2 \tag{2.3}$$

angenähert.

Der Skalenraum wird nun vom Ausgangsbild aus mit einem 9×9 Filter aufgebaut, dessen Seitenlänge in jeder Ebene der ersten Oktave um 6 erhöht wird. In den darauffolgenden Oktaven wird nicht nur der Filter für das Ausgangsbild der Oktave vergrößert, sondern auch die Vergrößerung des Filters pro Ebene wird verdoppelt. Die genaue Lokalisierung des markanten Merkmals findet im Skalenraum durch eine nicht Maximaunterdrückung mit einem $3 \times 3 \times 3$ Filter statt. Anschließend werden die gefundenen Maxima zwischen den einzelnen Bildebenen und Oktaven im Skalenraum interpoliert, um die finalen markanten Punkte subpixelgenau zu berechnen [BETVG08].

Deskriptor Der Deskriptor von SURF berechnet einen 64-dimensionalen Merkmalsvektor, der für jeden detektierten markanten Punkt im Bild berechnet wird.

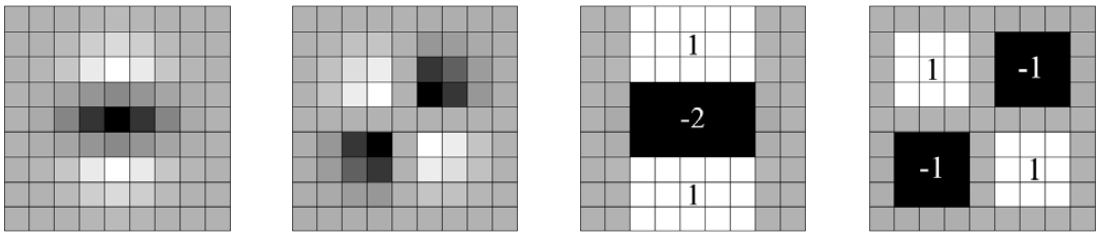


Abbildung 2.1: Die beiden Bilder links Visualisieren einen 9×9 DoG-Filter. Die beiden Bilder rechts zeigen analog die Boxfilter, die die DoG-Filter annähern. Die Gewichtung der einzelnen Pixel ist farblich kodiert, wobei ein Pixel mit dem Wert 0 grau dargestellt wird. Je dunkler ein Pixel ist, desto niedriger ist sein Wert und je heller, desto höher. [BETVG08]

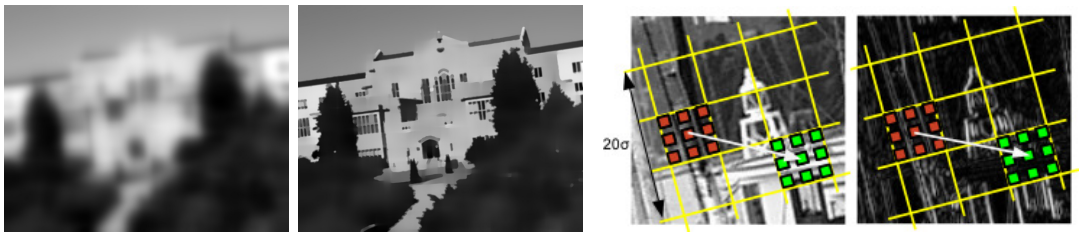
Dafür benutzt der Deskriptor die bereits im Detektor berechneten Integralbilder, mit denen nun geglättete Ableitungen in x- und y-Richtung berechnet werden. Dazu werden Filter benutzt, die einer Haar-Waveletfunktion ähneln und sich ebenfalls als Boxfilter berechnen lassen. Diese werden in einem kreisförmigen Ausschnitt um den markanten Punkt berechnet. Die Größe des Kreises und der Ableitungsfilter steigen ebenfalls mit jeder Oktave und Ebene. In dem kreisförmigen Ausschnitt wird nun in einem Slidingwindow der Größe $\frac{\pi}{3}$ die Ableitungen in x- und y-Richtung des Bildes aufsummiert und daraus ein neuer Vektor berechnet. An der Ausrichtung des Slidingwindow, für das dieser Vektor den größten Betrag aufweist, werden die Berechnungen des Deskriptors ausgerichtet, wodurch die Rotationsinvarianz erreicht wird.

Für die Berechnung des Merkmalsvekor wird nun ein 4×4 Raster um den markanten Punkt gelegt, dessen Seitenlänge sich ebenfalls an der Oktave und der Ebene orientieren. Innerhalb eines Segments des 4×4 Rasters wird an 25 gleichverteilten Bildpunkten p^p wieder eine Haar-Wavelet Filterung $d_x(\mathsf{p}^p)$ und $d_y(\mathsf{p}^p)$ durchgeführt. Diese Werte werden im Merkmalsvektor

$$v = \left(\sum_{i=1}^{25} d_x(\mathsf{p}_i^p), \sum_{i=1}^{25} d_y(\mathsf{p}_i^p), \sum_{i=1}^{25} |d_x(\mathsf{p}_i^p)|, \sum_{i=1}^{25} |d_y(\mathsf{p}_i^p)| \right) \tag{2.4}$$

für ein Segment des 4×4 Rasters aufsummiert. Erstellt man den Merkmalsvektor für jedes Segment des 4×4 Rasters, ergibt sich der $4 \times 4 \times 4 = 64$ dimensionalige Merkmalsvektor.

Eine weitere Version des Deskriptors ist SURF-128, bei dem v ein 8 dimensionaler Merkmalsvektor ist. In dieser Variante werden die Summen $\sum d_x(\mathsf{p}^p)$ und $\sum |d_x(\mathsf{p}^p)|$ separat für $d_y(\mathsf{p}^p) < 0$ und $d_y(\mathsf{p}^p) \geq 0$ ausgerechnet, genau wie für $\sum d_y(\mathsf{p}^p)$ und $\sum |d_y(\mathsf{p}^p)|$, falls $d_x(\mathsf{p}^p) < 0$ bzw. $d_x(\mathsf{p}^p) \geq 0$. SURF-128 führt zu einer besseren Unterscheidbarkeit des markanten Punktes, ist aber durch die hohe



(a) Vergleich FED und Gaußfilter **(b)** Abtastung M-LDB

Abbildung 2.2: Unterabtastung eines 3×3 Grid im Intensitäts- und Gradientenbild, wie sie beim M-LDB Deskriptor durchgeführt wird. Die kleinen Rechtecke in grün und rot sind die unterabgetasteten Punkte der beiden Segmente, die mit einander verglichen werden. Der Pfeil gibt an welches Segement mit welchem verglichen wird. [ANB13]

Dimension bei der Berechnung des Merkmalsvektors und später bei Korrespondenzbildung langsamer.

2.2 A-KAZE: Accelerated-KAZE

Accelerated KAZE (A-KAZE) ist die beschleunigte Variante des KAZE Merkmalsdeskriptors und -detektors. Die wichtigste Neuerung dieser beiden Deskriptoren und Detektoren gegenüber ihren Vorgängern ist, dass sie anstatt der angenäherten Gaußfilter nicht lineare Filter verwenden, um den bereits aus SURF bekannten Skalenraum zu berechnen. Die nicht linearen Filter sollen Rauschen im Bild glätten aber dennoch wichtige Details wie Kanten und Ecken erhalten.

Detektor Für die Detektion von markanten Punkten, wird das Ausgangsbild als erstes mit der schnellen expliziten Diffusion (englisch: fast explicit diffusion oder FED) gefiltert um den Skalenraum aufzubauen. FED ist die nicht lineare Filterung, die Rauschen glättet und Details wie Kanten und Ecken erhält. In Abbildung 2.2a wird ein gaußgefiltertes Bild mit einem FED gefilterten Bild gegenübergestellt. In dem gaußgefilterten Bild ist sehr gut zu erkennen, wie die Kanten des abgebildeten Hauses verschwimmen. Im FED gefilterten Bild hingegen, sind die Kanten noch sehr deutlich zu erkennen. Dennoch sind leicht inhomogene Regionen wie die Fassade des Hauses oder die Bäume im Vordergrund geglättet.

Auf den gefilterten Bildern im Skalenraum wird nun für jeden Pixel die von der Skala abhängige Determinante der Hessematrix berechnet. Die für die Hessematrix benötigten zweiten Ableitungen werden dabei mit einer Verkettung von Scharrfiltern, einer Variante des Sobelfilters, berechnet. Als Schlüsselpunkte im Bild werden nur die Pixel ausgewählt, die das Maxima in einer 3×3 Umgebung

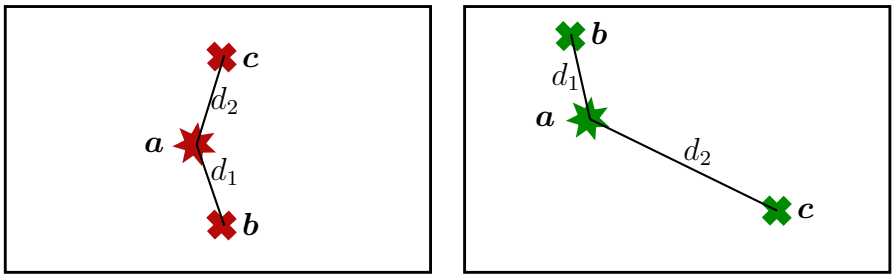
sind und einen Schwellert t übersteigen. Zusätzlich wird der potentiell markante Punkt nur dann als ein markanter Punkt detektiert, wenn er das Maximum seiner beiden Nachbarn in den Ebenen des Skalenraums direkt über und unter ihm ist. Abschließend wird die Position des markanten Punktes subpixelgenau durch eine zweidimensionale Funktion berechnet, wofür die acht Nachbarn des markanten Punktes mit einbezogen werden.

Deskriptor A-KAZE verwendet eine modifizierte Version des LDB (local difference binary) Deskriptors, kurz M-LDB. M-LDB beschreibt einen markanten Punkt durch die Helligkeits- und Gradienteninformationen aus dem Skalenraum. Dazu werden drei quadratische Raster um den markanten Punkt gelegt, die jeweils in $n \times n$ gleiche Unterregionen aufgeteilt werden, mit $n \in [2, 4]$. Innerhalb jedes dieser Raster wird nun die Summe über die Gradienten in x- und y-Richtung sowie über die Intensität des Patches durch eine Unterabtastung approximiert (Abb. 2.2a). Anschließend wird die Differenz der Summen zwischen allen möglichen Paaren der Zellen eines Rasters gebildet. Ist die Differenz größer als 0, so wird der Wert im Deskriptor an dieser Stelle auf 1 gesetzt, andernfalls auf 0.

Die Rotationsinvarianz des Deskriptors wird dabei nach einem ähnlichen Verfahren wie bei SURF erreicht. Es unterscheidet sich lediglich darin, dass für die Berechnung der Ausrichtung des Deskriptors eine unterabgetastete Anzahl an Gradienten verwendet wird [ANB13, YC12].

2.3 Korrespondenzbildung

Bei der Korrespondenzbildung geht es darum, Korrespondenzen zwischen den Mengen der Merkmalsvektoren D_1, D_2 zweier Bilder I_1, I_2 herzustellen. Dazu wird jeder Merkmalsvektor aus einem Bild I_1 mit seiner Position im Bild verknüpft und für die Suche als ein Punkt im Suchraum gespeichert. Um die Korrespondenzen zu den markanten Punkten eines zweiten Bildes D_2 herzustellen, ist es nun möglich jeden markanten Punkt aus I_1 mit denen im Suchraum zu vergleichen. Dieses Vorgehen hat einen quadratischen Aufwand und verursacht bei vielen Anwendungen einen inakzeptablen Rechenaufwand. Um dem entgegenzuwirken lassen sich die Punkte mit der Softwarelibrary FLANN in k-d- bzw. k-Means-Bäumen speichern. Mit ihnen lässt sich der Suchraum effizient durchsuchen und somit die nächsten Nachbarn eines Anfragepunktes effizienter finden. Weiterhin approximiert FLANN die nächsten Nachbarn, wodurch sich die Geschwindigkeit der Suche steigern lässst. Der Suchaufwand für eine Suchanfrage in einem k-d-Baum ist durchschnittlich nur noch $O(\log(n))$ [Moo90]. Nach der Bestimmung der nächsten Nachbarn muss abschließend entschieden werden, ob eine Korrespondenz zwischen dem Anfragepunkt und dem nächsten Nachbarn zustande kommt.



(a) (b)

Abbildung 2.3: Die beiden Bilder Visualisierung das Nearest-Neighbor-Distance-Ratio-Verfahren. Der Anfragepunkt $\boldsymbol{a}$ wird jeweils als Sterne dargestellt und die jeweiligen Nachbarn $\boldsymbol{b}$, $\boldsymbol{c}$ als Kreuz. Mann kann deutlich erkennen, das in Bild 2.3a nicht klar ist, mit welchem der beiden nächsten Nachbarn eine Korrespondenz gebildet werden soll. In Bild 2.3b hingegen, kann eindeutig eine Korrespondenz zwischen $\boldsymbol{a}$ und $\boldsymbol{b}$ festgelegt werden.

2.3.1 Nearest-Neighbor-Distance-Ratio

In vielen Bildern, werden markante Punkte detektiert, die sich nicht eindeutig genug beschreiben lassen. Dieses Problem kann vor allem in Bildern auftreten, in denen sich Strukturen häufig wiederholen. Ein Beispiel hierfür ist eine Fassade, bei der alle Fenster gleich aussehen. Aus diesem Grund, ist nach der Bestimmung des nächsten Nachbarn noch nicht klar, ob die Korrespondenzbildung mit diesem auch zu einer korrekten Korrespondenz führt. Es ist daher möglich, dass sich der zweitnächste Nachbar sehr nah zum nächsten Nachbarn befindet und der korrekte Nachbar ist. Es kann also nicht eindeutig festgestellt werden, welcher Nachbar der korrekte ist.

Das *Nearest-Neighbor-Distance-Ratio*-Verfahren betrachtet sowohl den nächsten Nachbarn $\boldsymbol{b} \in D_2$ als auch den zweitnächsten Nachbar $\boldsymbol{c} \in D_2$ zu einem Anfrage Punkt $\boldsymbol{a} \in D_1$, um zu entscheiden ob eine Korrespondenz mit hoher Wahrscheinlichkeit eindeutig ist. Dazu werden die Distanzen $d_1 = ||\boldsymbol{a} - \boldsymbol{b}||$ und $d_2 = ||\boldsymbol{a} - \boldsymbol{c}||$ betrachtet. Das Verhältnis

$$\text{NNDR} = \frac{||\boldsymbol{a} - \boldsymbol{b}||}{||\boldsymbol{a} - \boldsymbol{c}||} = \frac{d_1}{d_2} \tag{2.5}$$

zwischen d_1 und d_2 ist ein Indikator dafür. Ist die Zahl NNDR klein, so spricht dies dafür, dass d_2 wesentlich größer als d_1 ist. Ist NNDR kleiner als ein gegebener Schwellwert t, so kann zwischen $\boldsymbol{a}$ und $\boldsymbol{b}$ potentiell eine Korrespondenz zustande kommen. Bevor die Korrespondenz gebildet wird, muss das Verfahren noch ein zweites Mal mit den Merkmalsvektoren aus I_1 als Suchmenge angewendet werden. Der nächste Nachbar $\boldsymbol{b}$ ist nun der Abfragepunkt. Dieses Vorgehen ist notwendig,

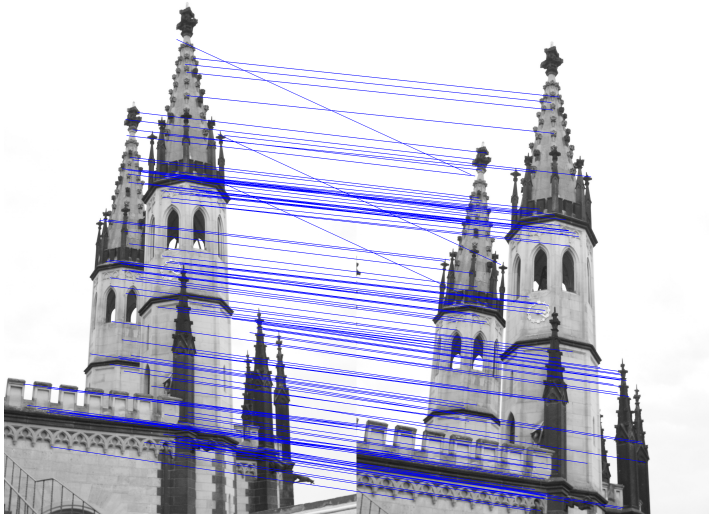


Abbildung 2.4: Korrespondenzbildung zwischen zwei Bildern mit SURF, nach der Korrespondenzsuche mit FLANN und dem Nearest-Neighbor-Distance-Ratio-Verfahren. Die Enden der blauen Linien stellen jeweils einen markanten Punkt dar und die Linie selbst eine Korrespondenz. Im Bild sind 120 Korrespondenzen zu sehen bei $t = 0{,}4$.

da auch der erste Abfragepunkt von nächsten Nachbarn mit ähnlicher Distanz zu $\boldsymbol{a}$ umgeben sein kann. Wenn auch hier NNDR $< t$ gilt, wird eine Korrespondenz gebildet [Sze10].

Mikolajczyk und Schmid haben in [MS05] gezeigt, dass sich mit dem Nearest-Neighbor-Distance-Ratio-Verfahren eine hohe Anzahl korrekter Korrespondenzen bei einer gleichzeitig niedrigen Anzahl von falschen Korrespondenzen erzielen lassen. Dennoch ist die Verwendung des Verfahrens keine Garantie dafür, dass ausschließlich korrekte Korrespondenzen gebildet werden. Durch die Approximation der nächsten Nachbarn, Ungenauigkeiten bei den Deskriptoren oder Ausreißer kann es immer noch zu Fehlern kommen.

Kapitel 3

Grundlagen der 3D-Rekonstruktion

Dieses Kapitel beschreibt grundlegend die Rekonstruktion von Kameraposen und 3D-Weltpunkten aus Punktkorrespondenzen zwischen 2D-Punkten. Die Rekonstruierten 3D-Weltpunkte und Kameraposen ergeben zusammen ein 3D-Modell, wie es in Abbildung 3.1 gezeigt wird. Es ist ein vereinfachtes Abbild der Realität, die mit einer Kamera aufgenommen wurde. Die Realität wird anschließend im Computer vereinfacht als 3D-Modell rekonstruiert und visualisiert.

Um den Vorgang der 3D-Rekonstruktion zu verstehen, soll als erstes das Lochkameramodell und die perspektivische Projektion in Kapitel 3.1 erklärt werden. Sie beschreiben, wie ein Weltpunkt auf den Sensor der Kamera abgebildet wird. Alle weiteren Algorithmen, die in diesem Kapitel beschrieben werden, bauen auf dem Verständnis des Kameramodells auf und benötigen dieses bei der Erstellung einer 3D-Rekonstruktion, wie sie in Abbildung 3.1 gezeigt wird.

In den folgenden Absätzen sollen die weiteren Grundlagen für eine 3D-Rekonstruktion und deren Zusammenspiel zum Erstellen eines 3D-Modells kurz erläutert werden, bevor in den einzelnen Unterkapiteln im Detail auf diese eingegangen wird. Im Kapitel 3.2 wird die *Epipolargeometrie* und die *Fundamentalmatrix* ausführlich erklärt. Sie beschreiben die geometrische und projektive Beziehungen zwischen zwei Kameras. Auf Grundlage der Fundamentalmatrix lässt sich die relative Pose zweier Kameras zueinander, wie in Kapitel 3.3 erläutert, bis auf einen Skalierungsfaktor rekonstruieren. Aus dieser Pose lässt sich ein 3D-Modell initialisieren, indem die absolute Pose der ersten Kamera im Ursprung des Weltkoordinatensystems festgelegt wird. Die absolute Pose der zweiten Kamera lässt sich jetzt aus der relativen Pose zur ersten Kamera errechnen.

Mit den beiden Kameraposen und den Punktkorrespondenzen zwischen den beiden zugehörigen Bildern kann ein *Triangulierungsalgorithmus* pro Punktkorrespondenz einen 3D-Weltpunkt, wie in Kapitel 3.5 erklärt, rekonstruieren. Die rekonstruierten 3D-Weltpunkte werden dann dem 3D-Modell hinzu gefügt und ergeben zusammen eine 3D-Punktewolke. Mit den rekonstruierten 3D-Weltpunkten,

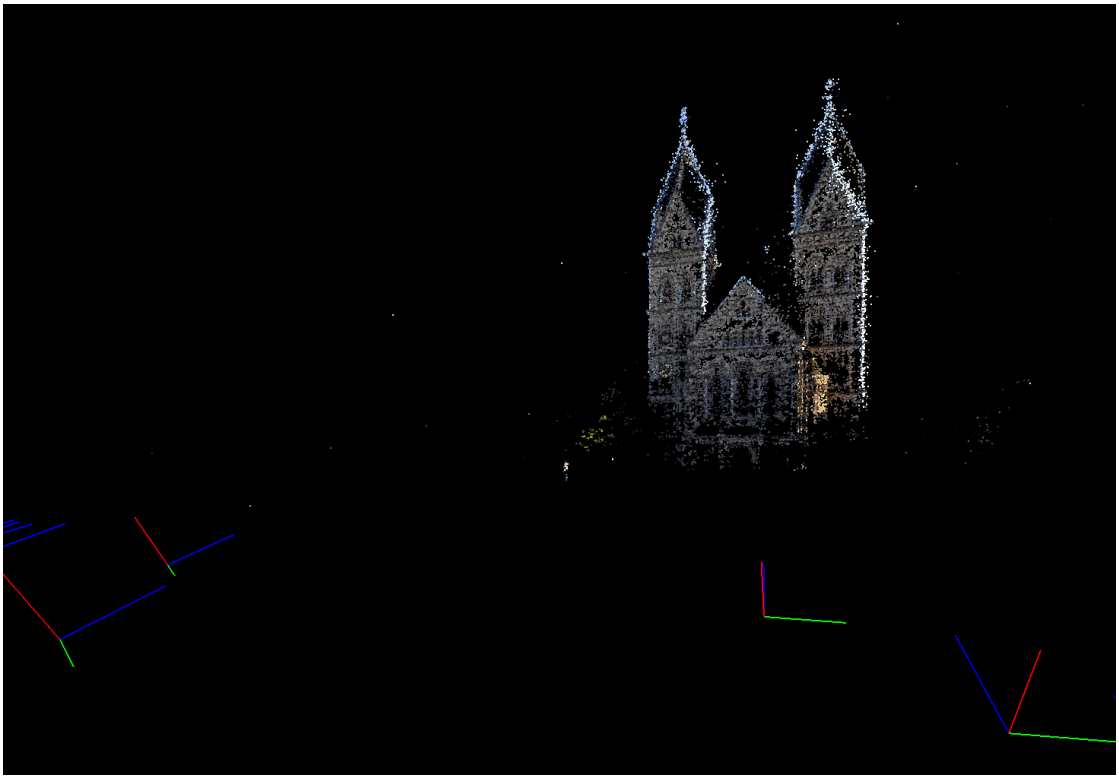


Abbildung 3.1: Die Abbildung zeigt eine Rekonstruktion der Herz-Jesu-Kirche in Koblenz. Das Modell besteht aus ca. 66000 Punkten aus 12 Bildern. Die Positionen von denen aus die Bilder aufgenommen wurden werden im Bild durch Koordinatensysteme visualisiert.

deren korrespondierenden 2D-Bildpunkten aus einem dritten Bild und dem Kameramodell der zugehörigen Kamera, lässt sich durch Lösen des *Perspective-n-Point*-Problems eine weitere Kamerapose wie in Kapitel 3.4 bestimmen. Durch das lösen des Perspective-n-Point-Problems wird die Skalierung der bereits vorhandenen Rekonstruktion übernommen, was bei der Kameraposenbestimmung aus der Fundamentalmatrix nicht der Fall sein muss. Da weitere 2D-2D Punktkorrespondenzen zwischen dem Bild der neu rekonstruierten Kamera und den bereits rekonstruierten Bildern liegen, lassen sich weitere 3D-Weltpunkte mit dem Triangulierungsalgorithmus der Punktwolke hinzufügen. Diese letzten beiden Schritte können für jedes Bildpaar wiederholt werden, um das 3D-Modell zu erweitern.

Bei jeder Rekonstruktion von 3D-Weltpunkten oder einer Kamera, entstehen Fehler in den Berechnungen für die Rekonstruktion. In Bild 3.2a werden diese durch unklare Strukturen und durch rekonstruierte Wände, die nicht exakt übereinander liegen sichtbar. Gründe hierfür sind vor allem Ungenauigkeiten bei der Lokalisation der markanten Punkte, aber auch falsche Punktkorrespondenzen oder

(a) (b)

Abbildung 3.2: Bild 3.2a zeigt die Rekonstruktion der Herz-Jesu-Kirche ohne Optimierung des Modells. In Bild 3.2b ist die rekonstruierte Kirche nach einer Optimierung zu sehen. Die Konturen der Kirche sind hier viel deutlicher zu erkennen als in Bild 3.2a.

das geschätzte Kameramodell. Diese Fehler akkumulieren sich mit jeder weiteren Rekonstruktion im Modell auf und können so groß werden, dass folgende Rekonstruktionen nicht mehr möglich beziehungsweise unbrauchbar sind. Um dies zu vermeiden, lässt sich das rekonstruierte Modell in einem Zwischenschritt, nach jeder Rekonstruktion der 3D-Weltpunkte verbessern. Dazu werden die Fehler im Modell minimiert. Bild 3.2b zeigt eine solche Rekonstruktion, bei der die Fehler in Zwischenschritten minimiert wurden. Die Details der Kirche sind hier viel deutlicher zu erkennen und die rekonstruierten Wände liegen in einer Ebene.

Das Verringern der Fehler, kann nach jedem Bild geschehen aber auch erst nachdem einige Bilder rekonstruiert wurden. Nachdem die Initialisierung des Modells abgeschlossen ist, kann noch mal eine abschließende Optimierung des Modells durchgeführt werden. Das genaue Vorgehen zur Fehlerminimierung wird in Kapitel 4 erläutert.

3.1 Das Kameramodell

Der Strahlengang einer Kamera wird durch ein geometrisches Kameramodell angenähert. Durch ein solches Modell lässt sich berechnen, wie ein Weltpunkt aus der Umwelt auf den Kamerasensor abgebildet wird. Dabei wird durch das Kameramodell die Realität angenähert aber nie vollständig beschrieben. Das Kameramodell wird durch Kameraparameter beschrieben, die das Modell auf unterschiedliche Kameras anpassen können. Sie lassen sich durch eine Kalibrierung der verwendeten Kamera ermitteln. Die Kalibrierung ist notwendig, weil sich jede Kamera

- durch ihren Aufbau,
- Unterschiede bei der Fertigung,
- wechselbare Objektive,
- Objektiveinstellungen und
- ihre Position unterscheidet.

Die Modellparameter lassen sich zusätzlich in intrinsische und extrinsiche Parameter unterteilen. Die extrinsischen Parameter geben die Position und Ausrichtung der Kamera in einem Weltkoordinatensystem w an und die intrinsischen Parameter beschreiben den Strahlengang der aus der Umwelt einfallenden Lichtstrahlen im Inneren der Kamera. Mit einem Kameramodell lässt sich nun errechnen, wie ein dreidimensionaler Weltpunkt

$$\mathsf{p}^w = (x, y, z)^{\mathrm{T}} \tag{3.1}$$

auf den Sensor einer Kamera projiziert wird, wodurch man diesem Punkt einen Pixel

$$\mathsf{p}^p = (x, y)^{\mathrm{T}} \tag{3.2}$$

im Bild zuordnen kann.

Um p^w auf einen Pixel zu projizieren, muss dieser zunächst vom Weltkoordinatensystem w in das Kamerakoordinatensystem c transformiert werden. Der Ursprung von c liegt dabei im Brennpunkt der Kamera und wird optisches Zentrum $\boldsymbol{O}_c$ genannt. Die x-, y- und z-Achsen des Kamerakoordinatensystems sind wie in Abbildung 3.3 ausgerichtet. Bildlich gesprochen zeigt die z-Achse aus dem Objektiv, die y-Achse nach oben und die x-Achse nach rechts, wenn man in Blickrichtung der Kamera schaut.

Die Umrechnung des Weltpunktes in das Kamerakoordinatensystem erfolgt durch die extrinsischen Kameraparameter. Sie setzen sich aus einer Translation $\boldsymbol{t}$ und einer Rotationsmatrix $\boldsymbol{R}$ zusammen. Die extrinsischen Kameraparameter transformieren den Weltpunkt p^w aus dem Weltkoordinatensystem in das Kamerakoordinatensystem, sodass p^w zu

$$\mathsf{p}^c = \left[\boldsymbol{R} \mid \boldsymbol{t}\right] \mathsf{p}^w \tag{3.3}$$

wird.

Der Punkt p^c kann nun, in das 2D-Bildkoordinatensystem i durch eine perspektivische Projektion projiziert werden. Dabei muss zusätzlich die Brennweite des Objektives F berücksichtigt werden.

$$\mathsf{p}^i = \begin{pmatrix} \frac{Fx}{z} \\ \frac{Fy}{z} \\ \frac{z}{z} \end{pmatrix} = \begin{pmatrix} \frac{Fx}{z} \\ \frac{Fy}{z} \\ 1 \end{pmatrix} \tag{3.4}$$

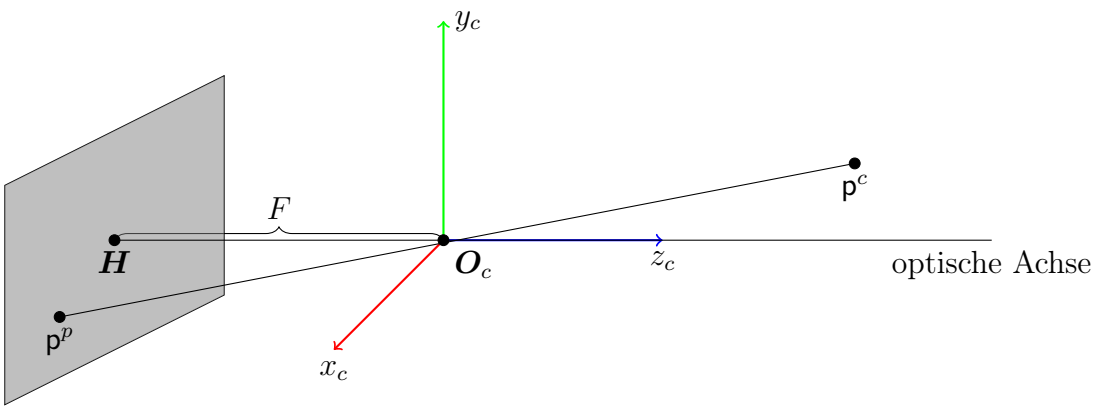


Abbildung 3.3: Projektion eines Punktes p^c auf den Punkt p^p auf dem Kamerasensor. Die optische Achse geht vom Bildhauptpunkt $\boldsymbol{H}$ durch das Kamerazentrum $\boldsymbol{O}_c$ entlang der z-Achse des Kamerakoordinatensystems. Die Brennweite F ist der Abstand zwischen dem Kamerazentrum $\boldsymbol{O}_c$ und dem Bildhauptpunkt $\boldsymbol{H}$.

Bei der Projektion wird p^c durch seine z-Komponente geteilt. Dies hat zu Folge, dass die Distanzinformation verloren geht, weshalb kein Rückschluss mehr auf die Entfernung des Punktes zur Kamera möglich ist.

Diese Projektion kann auch durch Vektoren im projektiven Raum $\mathbb{P}^3$ beschrieben. Hierbei findet eine Überführung vom dreidimensionalen projektiven Raum $\mathbb{P}^3$ in den zweidimensionalen projektiven Raum $\mathbb{P}^2$ statt. In diesem Fall wird p^c zu $\tilde{\mathsf{p}}^c = (x, y, z, 1)$ und p^i zu $\tilde{\mathsf{p}}^i = (x, y, 1)$. Die Überführung vom $\mathbb{P}^3$ in den $\mathbb{P}^2$ kann nun durch eine 3×4 Matrix, wie in Formel (3.5) geschrieben werden:

$$\tilde{\mathsf{p}}^i = \begin{pmatrix} F & 0 & 0 & 0 \\ 0 & F & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tilde{\mathsf{p}}^c \tag{3.5}$$

Der Vektor $\tilde{\mathsf{p}}^i$ wird dann normalisiert und in das das Pixelkoordinatensystem p überführt, das seinen Ursprung in der linken oberen Ecke des Bildes hat. Die Abszissenachse zeigt im Bildkoordinatensystem nach rechts und wird mit u bezeichnet, die Ordinatenachse wird mit v bezeichnet und zeigt nach unten.

Die Kalibrierungsmatrix

$$\boldsymbol{K} = \begin{pmatrix} d_x & 0 & c_x \\ 0 & d_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \tag{3.6}$$

transformiert einen Pixel von Bildkoordinaten in Pixelkoordinaten:

$$\tilde{\mathsf{p}}^p = \boldsymbol{K}\tilde{\mathsf{p}}^i \tag{3.7}$$

Dabei muss die Größe der Pixel, deren Seitenlängen nicht gleich sind und die Position des Bildhauptpunktes $\boldsymbol{H}$ berücksichtigt werden. Die Seitenlänge eines Pixels in u-Richtung wird mit d_x und in v-Richtung mit d_y bezeichnet. Weiterhin wird die Position des Bildhauptpunktes $\boldsymbol{H}$ in Pixelkoordinaten mit $\boldsymbol{H} = (c_x, c_y)$ bezeichnet. Seine Position ist die, an der die optische Achse den Bildsensor durchstößt.

Bei dem vorgestellten Kameramodell wurden bisher noch nicht radiale Verzerrungen berücksichtigt, die vor allem bei der Verwendung von Objektiven mit einer kurzen Brennweite und schlechter Qualität entstehen können. Sie sind in den Randbereichen eines Bildes besonders stark und müssen deshalb für eine präzise Modellierung der Kamera berücksichtigt werden. Die radialen Verzerrungen lassen sich nach der perspektivischen Projektion auf die Bildebene und vor der Anwendung der Kalibrierungsmatrix modellieren. Hierzu wird üblicherweise ein Polynom vierten Gerades verwendet. Die Verzerrung ist vom euklidischen Abstand

$$r = \sqrt{(x^i)^2 + (y^i)^2} \tag{3.8}$$

eines Punktes $\mathsf{p}^i = (x^i, y^i)$ zum Bildhauptpunkt und dem Aufbau des Objektivs ansich abhängig. Die Verzerrung wird durch durch zwei Verzerrungskoeffizienten κ_1 und κ_2 im Zusammenhang mit r modelliert. Die Gleichung

$$\mathsf{p}^d = \quad (1 + \kappa_1 \cdot r^2 + \kappa_2 \cdot r^4)\mathsf{p}^i \tag{3.9}$$

transformiert somit die Position des unverzerrtes Punktes p^i in die Position des verzerrten Punktes p^d. Nach der Verzerrung lässt sich $\mathsf{p}^d \in \mathbb{R}^2$ wieder um seine homogene Koordinate erweitern, um ihn mit der Kalibrierungsmatrix $\boldsymbol{K}$ zu multiplizieren. Hieraus resultiert der korrekt verzerrte Punkt p^p im Pixelkoordinatensystem [Sze10, TV98, RH00].

Zusammengefasst bedeutet dies, dass ein Weltpunkt p^w wie in Formel (3.3) mit den extrinsischen Kameraprametern in das Kamerakoordinatensystem transformiert wird, woraus sich p^c ergibt. Anschließend wird p^c unter Berücksichtigung der Brennweite in das ideale Bildkoordinatensystem wie in Formel (3.5) projiziert, sodass p^i entsteht. Jetzt wird die Verzerrung durch Gleichung (3.9) angewendet. Der nun entstandene Punkt p^d kann jetzt mit der Kalibrierungsmatrix $\boldsymbol{K}$ multipliziert werden, um den finalen Pixel p^p zu erhalten.

3.2 Epipolargeometrie

Die Epipolargeometrie beschreibt die relative Position zweier Kameras zueinander und die Projektion von Weltpunkten auf deren Sensoren. Wie in Abbildung 3.4 visualisiert, wird der Weltpunkt p^w von der Kamera Cam_1 als p_1^p und von Cam_2 als p_2^p aufgenommen. Verbindet man die Kamerazentren der beiden Kameras mit

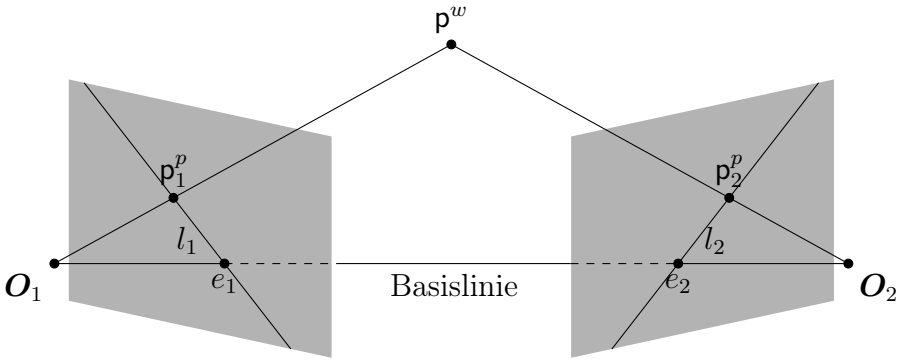


Abbildung 3.4: Im Bild ist eine Visualisierung der Epipolargeometrie zu sehen. $\boldsymbol{O}_1$ und $\boldsymbol{O}_2$ bezeichnen die optischen Zentren der beiden Kameras. Der Weltpunkt p^w wird als die beiden Pixel p_1^p und p_2^p auf die Kamerasensoren projiziert. Hierbei ist zu beachten, dass der Sensor für eine einfachere Visualisierung vor dem optischen Zentrum liegt. l_1 und l_2 bezeichnet die beiden Epipolarlinien, die Bildpunkte p_1^p und p_2^p mit den beiden Epipolen e_1 und e_2 verbindet.

einer Linie, erhält man die sogenannte Basislinie. Die Schnittpunkte der Basislinie mit den Bildebenen der beiden Kameras nennt man die Epipole. Sie werden mit e_1 für Cam_1 und e_2 für Cam_2 bezeichnet.

Projiziert man einen Weltpunkt p^w auf den Kamerasensor von Cam_1 erhält man den Pixel p_1^p. Durch ihn und p^w verläuft ein Strahl, auf dem auch das Kamerazentrum von Cam_1 liegt. Spannt man mit diesem Strahl und der Basislinie eine Ebene auf, erhält man die Epipolarebene. In dieser Ebene liegen immer der zu p_1^p korrespondierende Punkt p_2^p, die beiden Epipole e_1 und e_2, sowie der Weltpunkt p^w.

Die beiden Geraden, die jeweils durch die beiden Epipole und die beiden Bildpunkte laufen, heißen Epipolarlinien l_1 und l_2. Sie sind die beiden Schnittlinien der Bildebenen mit der Epipolarebene. Folglich wird jeder Strahl, der durch das Kamerazentrum der ersten Kamera und einem Punkt auf l_1 geht, auf l_2 projiziert. Dieser Sachverhalt gilt umgekehrt für die zweite Kamera. Für die Suche von korrespondierenden Bildpunkten bedeutet das, dass sich ein korrespondierender Punkt nur entlang der Epipolarlinie im zweiten Bild befinden kann.

Der durch die Epipolargeometrie beschriebene Zusammenhang wird mathematisch durch die sogenannte Fundamentalmatrix beschrieben. Die Fundamentalmatrix $\boldsymbol{F}$ ist eine 3×3 Matrix, die bis auf einen Skalierungsfaktor eindeutig definiert ist [RH00]. Sie berechnet sich wie folgt:

$$\boldsymbol{F} = \boldsymbol{K_1}^{-\mathrm{T}}[\boldsymbol{t}]_\times \boldsymbol{R}\boldsymbol{K_2}^{-1} \tag{3.10}$$

Dabei sind $\boldsymbol{K_1}$ und $\boldsymbol{K_2}$ die beiden Kalibrierungsmatrizen der Kameras Cam_1 und Cam_2. $\boldsymbol{R}$ ist die Rotationsmatrix und $[\boldsymbol{t}]_\times$ die Kreuzproduktmatrix des Translationsvektors $\boldsymbol{t}$. $\boldsymbol{R}$ und $\boldsymbol{t}$ beschreiben dabei die relative Position von Cam_1 zu Cam_2 im $\mathbb{R}^3$.

Multipliziert man einen Punkt $\tilde{\mathsf{p}}_2^p$ von rechts mit der Fundamentalmatrix, wird dieser auf die korrespondierende Epipolarlinie

$$l_1 = \boldsymbol{F}\tilde{\mathsf{p}}_2^p \tag{3.11}$$

in Bild I_1 abgebildet. Der Vektor l_1 ist dabei ein Zeilenvektor, der die Parameter für eine homogene Liniengleichung in Pixelkoordinaten enthält. Für die Berechnung von l_2 in I_2 über einen Punkt $\tilde{\mathsf{p}}_1^p$ gilt weiterhin:

$$l_2 = \boldsymbol{F}^{\mathrm{T}}\tilde{\mathsf{p}}_1^p \tag{3.12}$$

Eine andere Möglichkeit die Fundamentalmatrix $\boldsymbol{F}$ zu errechnen ohne die extrinsischen Kameraparameter und die Kameramatrizen der beiden Kameras kennen zu müssen, ist der 8-Punkt-Algorithmus. Er berechnet die Fundamentalmatrix anhand von acht oder mehr Punktkorrespondenzen, die sich wie in Kapitel 2 beschrieben, gewinnen lassen. Die einzelnen Punkte müssen vor der Berechnung der Fundamentalmatrix allerdings entzerrt werden, da die Verzerrungen in dieser nicht modelliert sind.

Die Epipolargeometrie kann weiterhin verwendet werden, um zu überprüfen ob zwischen zwei Bildpunkten eine korrekte oder eine falsche Korrespondenz gebildet wurde. Dies kann mit der Epipolarbedingung

$$\begin{aligned} \tilde{\mathsf{p}}_1^{p\mathrm{T}}\boldsymbol{F}\tilde{\mathsf{p}}_2^p &= 0 \\ \tilde{\mathsf{p}}_1^{p\mathrm{T}}l_1 &= 0 \end{aligned} \tag{3.13}$$

überprüft werden. Sie sagt aus, dass ein zu $\tilde{\mathsf{p}}_1^p$ korrespondierender Punkt $\tilde{\mathsf{p}}_2^p$ auf der Epipolarlinie von $\tilde{\mathsf{p}}_1^p$ liegen muss. Punktkorrespondenzen, die bei der Korrespondenzbildung falsch gebildet wurden, können durch diese geometrische Bedingung nachträglich gelöscht werden. Allerdings kann es auch hier passieren, dass nicht alle falschen Punktkorrespondenzen erkannt werden. Das kann deswegen passieren, weil alle Punkte, die auf der Epipolarlinie von $\tilde{\mathsf{p}}_2^p$ liegen ebenfalls auf die Epipolarlinie von $\tilde{\mathsf{p}}_1^p$ abgebildet werden. Diese Probleme treten besonders leicht bei sich wiederholenden Strukturen im Bild, bei denen die Wiederholungsrichtung parallel zur Basislinie des Kamerapaars verläuft auf [Dec12].

3.3 Bestimmung von Kameraposen aus der Fundamentalmatrix

Wie im vorherigen Kapitel beschrieben, lässt sich die Fundamentalmatrix ohne Wissen über Kameraposen aus Punktkorrespondenzen errechnen. Dennoch ist die relative Pose zweier Kameras zueinander in ihr bis auf einen Skalierungsfaktor kodiert.

Das Ziel ist nun, diese Pose aus der Fundamentalmatrix zu extrahieren, um die Kameraposen in das Modell einzufügen. Für die Extraktion der relativen Kamerapose zweier Kameras zueinander werden die Fundamentalmatrix $\boldsymbol{F}$ und die beiden Kalibrierungsmatrizen $\boldsymbol{K_1}$ und $\boldsymbol{K_2}$ benötigt. Mit diesen Informationen lässt sich die sogenannte essentielle Matrix $\boldsymbol{E}$ berechnen.

$$\begin{aligned} \boldsymbol{E} &= \boldsymbol{K_1}^{\mathrm{T}} \boldsymbol{F} \boldsymbol{K_2} \\ &= \boldsymbol{K_1}^{\mathrm{T}} \boldsymbol{K_1}^{-\mathrm{T}} [\boldsymbol{t}]_\times \boldsymbol{R} \boldsymbol{K_2}^{-1} \boldsymbol{K_2} \\ &= [\boldsymbol{t}]_\times \boldsymbol{R} \end{aligned} \tag{3.14}$$

Mit ihr lassen sich mittels der Singulärwertzerlegung, auf die hier nicht näher eingegangen werden soll, vier mögliche Kameraposen in Form einer Rotationsmatrix $\boldsymbol{R}$ und einem Translationsvektor $\boldsymbol{t}$ berechnen. Von diesen vier möglichen Posen ist allerdings nur eine richtig. Die Entscheidung welche der vier Kameraposen richtig ist, wird mittels Testen, ob rekonstruierte 3D-Weltpunkte vor beiden Kameras liegen, entschieden. Wie sich 3D-Weltpunkte für diesen Test rekonstruieren lassen, wird in Kapitel 3.5 beschrieben.

Mit dem vorgestellten Algorithmus lässt sich so die relative Kamerapose zweier Kameras zueinander bis auf einen Skalierungsfaktor rekonstruieren. Mit der relativen Kamerapose ist es nun möglich ein 3D-Modell zu initialisieren. Dazu wird angenommen, dass eine der beiden Kameras mit ihrem Koordinatensystem an dem Weltkoordinatensystem ausgerichtet ist. ihre absolute Pose ist somit durch den Ursprung des Weltkoordinatensystems definiert. Hierdurch kann die relative Pose der zweiten Kamera zur ersten als eine absolute Pose interpretiert werden [RH00].

Diese Methode eigent sich allerdings nur dafür ein 3D-Modell zu initialisieren, da die Rekonstruktion jedes weiteren Kamerapaars eine andere Skalierung hätte. Daher wird in Kapitel 3.4 das Perspective-n-Point-Problem beschrieben, das die erkannten Merkmale in einem neu hinzukommenden Kamerabild mit den vorhandenen im Modell abgleicht und durch diese Korrespondenzen eine Kamerapose rekonstruiert, die in ihrer Skalierung zu dem bestehenden Modell passt.

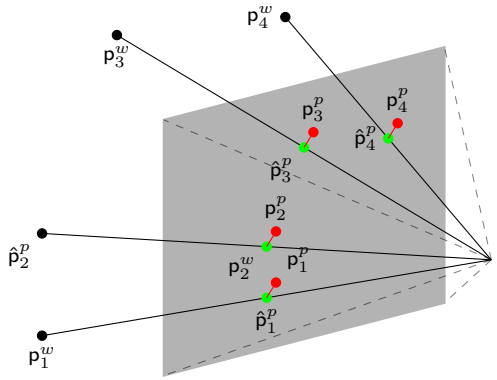


Abbildung 3.5: Im Bild ist eine Visualisierung des Perspective-n-Point-Problem zu sehen. Die bekannten Weltpunkte p_j^w werden als grüne Punkte $\hat{\mathsf{p}}_j^p$ auf die Bildebene projiziert. Die roten Punkte visualisieren die entsprechenden observierten Punkte p_j^p. Zwischen den korrespondierenden Punkten ergibt sich der Rückprojektionsfehler, der als rote Linie visualisiert ist.

3.4 Das Perspective-n-Point-Problem

Durch das Lösen des *Perspective-n-Point-Problems* lässt sich die absolute Pose einer Kamera bestimmen. Um das Problem zu lösen, werden Punktkorrespondenzen zwischen entzerrten 2D-Bild- und 3D-Weltpunkten sowie die Kalibrierungsmatrix der Kamera, deren Pose zu bestimmen ist, benötigt.

Anhand dieser Eingaben lässt sich die Position und Ausrichtung einer Kamera bestimmen. Dabei wird die Skalierung, die durch die Weltpunkte gegeben ist, beibehalten. Hierin unterscheidet sich diese Methode zur Extraktion von Kameraposen aus der Fundamentalmatrix, da man bei ihr keinen Skalierungsfaktor festlegen kann.

Um das Perspective-n-Point-Problem zu lösen, gibt es verschiedene Ansätze. Einer davon ist, die Pose der Kamera grob zu schätzen und diese anschließend zu verfeinern. Dazu wird der quadrierte Rückprojektionsfehler

$$\begin{aligned} d^2 &= |\mathsf{p}^p - \boldsymbol{P}\mathsf{p}^w|^2 \\ &= |\mathsf{p}^p - \boldsymbol{K}(\boldsymbol{R}\mathsf{p}^w + \boldsymbol{t})|^2 \\ &= |\mathsf{p}^p - \hat{\mathsf{p}}^p|^2 \end{aligned} \tag{3.15}$$

aller Punktkorrespondenzen minimiert. Der Punkt p^p ist der im Bild observierte Punkt und der Punkt p^w der zu p^p korrespondierende Weltpunkt im Modell. Der Weltpunkt p^w wird mit der geschätzten Kameramatrix $\boldsymbol{P}$ in Pixelkoordinaten umgerechnet, sodass sich $\hat{\mathsf{p}}^p$ ergibt. Da $\boldsymbol{P}$ nur geschätzt ist, entspricht $\hat{\mathsf{p}}^p$ nicht genau p^p, weshalb ein Abstand d zwischen den beiden Punkten entsteht.

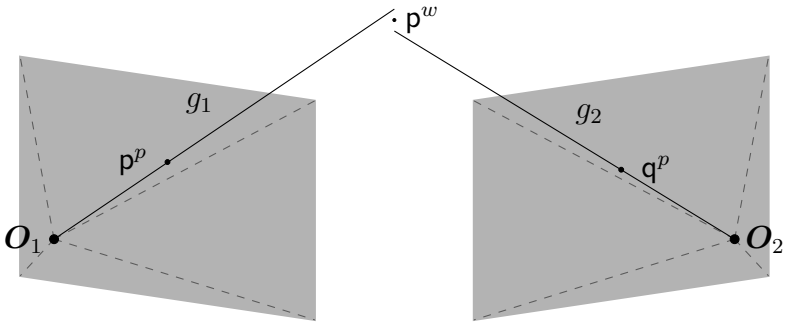


Abbildung 3.6: In der Abbildung ist die Bestimmung des Weltpunktes p^w zu sehen. Durch die optischen Zentren $\boldsymbol{O}_1$, $\boldsymbol{O}_2$ und die beiden korrespondierenden Pixel p^p, q^p wird jeweils eine Gerade g_1, g_2 gelegt. g_1 und g_2 schneiden sich aufgrund von Ungenauigkeiten im Modell nicht direkt in einem Weltpunkt p^w, deswegen kann p^w nur approximiert werden.

Um die korrekte Pose der Kamera zu berechnen, muss der aufsummierte Rückprojektionsfehler über alle observierten Bildpunkte p_j^p und alle modellierten Bildpunkte $\hat{\mathsf{p}}_j^p$ minimiert werden. Die Berechnung lautet somit

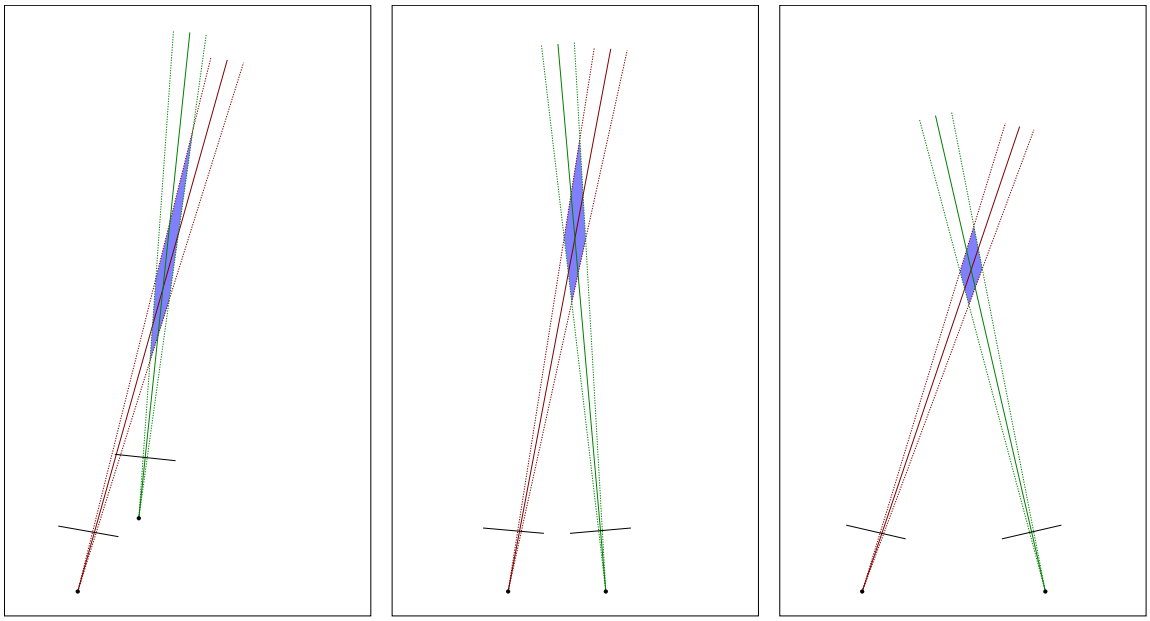
$$\underset{\boldsymbol{R},\boldsymbol{t}}{\operatorname{argmin}} \sum_{j=1}^{n} |\mathsf{p}_j^p - \boldsymbol{K}(\boldsymbol{R}\mathsf{p}_j^w + \boldsymbol{t})|^2 \tag{3.16}$$

Hierzu wird die Kamerapose so lange angepasst, bis die Summe der Rückprojektionsfehler einen Schwellwert unterschreitet oder eine gegebene Anzahl an Iterationen erreicht wurde. Abbildung 3.4 veranschaulicht das Problem. Hier werden die Weltpunkte auf die in grün dargestellten Bildpunkte projiziert, die einen Versatz zu den in rot dargestellten observierten Punkten haben. Zwischen den Punktpaaren ist der Rückprojektionsfehler als rote Linie eingezeichnet.

Für die Anpassung der Pose wird meist der Levenberg-Marquardt-Algorithmus oder das Gauß-Newton Verfahren verwendet. Der Levenberg-Marquardt-Algorithmus wird in Kapitel 4.3 vorgestellt [LHM00].

3.5 Triangulation von 3D-Weltpunkten

Bei der Triangulation von 3D-Weltpunkten geht es darum, aus zwei korrespondierenden entzerrten Pixel p^p und q^p, aufgenommen von zwei Kameras Cam_1 und Cam_2, den gemeinsamen 3D-Weltpunkt p^w zu errechnen. Neben den zwei korrespondierenden, entzerrten Punkten sind für die Berechnung des Weltpunktes die



(a) Vorwärtsbewegung der zweiten Kamera

(b) Schmale Basislinie

(c) Breite Basislinie

Abbildung 3.7: Die Visualisierungen zeigen zwei skizzierte Kameras mit optischem Zentrum und Bildebene. Die gestrichelten Linien begrenzen jeweils den Bereich, den ein Strahl von seinem Ideal durch Messungenauigkeiten abweichen kann. Die blau hinterlegte Fläche kennzeichnet den Bereich, in dem der 3D-Weltpunkt durch Ungenauigkeiten trianguliert werden könnte. Er ist in Abbildung 3.7a und 3.7b wesentlich größer als in Abbildung 3.7c.

extrinsischen als auch die intrinsischen Kameraparameter der beiden Kameras gegeben.

Aufgrund der gegebenen Parameter lässt sich eine Gerade durch das optische Zentrum von Cam_1 und den entsprechenden Pixel p^p legen und dies genauso für Cam_2 und q^p wiederholen. Die Gerade für p^p soll hierbei mit g_1 und für q^p mit g_2 bezeichnet werden. Entlang dieser beiden Geraden wurde der unbekannte Weltpunkt p^w auf die beiden Bildpunkte p^p und q^p projiziert. Also muss p^w der Schnittpunkt von g_1 und g_2 sein.

Durch Ungenauigkeiten der Kameraparameter oder der Position der markanten Punkte stehen die beiden Geraden g_1 und g_2 in der Praxis, wie in Abbildung 3.6 gezeigt windschief zueinander. Aus diesem Grund lässt sich kein exakter Schnittpunkt berechnen. Deswegen wird die Position des Weltpunkts p^w, wie in Abbildung 3.6 dargestellt, zwischen den beiden Geraden approximiert.

(a) **(b)**

Abbildung 3.8: Bild 3.8a zeigt eines der beiden Ausgangsbilder, die für die Rekonstruktion des Modells, das in Bild 3.8b zu sehen ist verwendet wurde. In der rot eingekreisten Stelle ist ein deutlicher Versatz in der Bremsschwelle zu sehen. Der Versatz ist Aufgrund der geringen Basisline zwischen den beiden Kameras unten rechts im Bild entstanden.

Weiterhin hat die relative Pose der beiden Kameras zueinander, einen Einfluss auf die Genauigkeit der Positionsbestimmung des 3D-Weltpunktes. Sind die beiden Geraden, die sich im Weltpunkt schneiden, fast parallel, lässt sich die genaue Position von p^w nur schwer bestimmen. Ein Grund hierfür kann eine Bewegung der zweiten Kamera nach vorne in Richtung des Blickfelds der ersten Kamera wie in Abbildung 3.7a sein oder eine im Verhältnis zur Entfernung des zu triangulierenden Punktes relativ kleine Basislinie, wie in Abbildung 3.7b. In beiden fällen Verlaufen die Geraden der beiden Punkte fast parallel, was eine Bestimmung des Schnittpunkts schwierig macht. Der Grund für die Schwierigkeit der Bestimmung ist, dass der Bereich in dem der tatsächliche Schnittpunkt liegen kann durch Messungenauigkeiten sehr groß ist. Denn kleine Ungenauigkeiten im Modell haben durch die Parallelität der beiden Strahlen eine große Auswirkung auf deren Schnittpunkt. Ein relativ großer Winkel zwischen den beiden Geraden vereinfacht also die Bestimmung des Schnittpunktes [RH00].

Das aus Bild 3.8a rekonstruierte 3D-Modell aus Abbildung 3.8b zeigt das Problem. Die Sichtstrahlen innerhalb des roten Kreises sind nahezu parallel, wodurch die Bremsschwelle nicht wie auf dem Ausgangsbild rekonstruiert werden kann. Durch Ungenauigkeiten in der Lokalisierung der korrespondierenden Bildpunkte, werden die Positionen der Weltpunkte falsch berechnet. Sie können sehr weit hinter oder vor ihrer korrekten Position liegen.

Für eine gute Rekonstruktion sollte sich die zweite Kamera seitlich der Blickrichtung der ersten Kamera befinden und beide Kameras sollten sich etwas in

Richtung des zu vermessenden Objektes drehen, so wie es in Abbildung 3.7c schematisch dargestellt ist. Hierdurch ist der Winkel zwischen den beiden Geraden relativ groß und der Schnittpunkt kann exakter bestimmt werden.

Kapitel 4

Bündelausgleich

Im vorherigen Kapitel 3 wurden das Kameramodell und verschiedene Methoden zum Rekonstruieren eines 3D-Models aus Bildern vorgestellt. Bei all diesen Verfahren wurde eine Rekonstruktion von Punkten oder einer Kamerapose mit maximal zwei Bildern durchgeführt, wobei Fehler wie in Abbildung 3.2a entstehen können. Diese akkumulieren sich beim iterativen Hinzufügen von neuen Kameras oder Weltpunkten zum Model auf, so dass mit jedem Anfügen weiterer Rekonstruktionen immer gröbere Fehler entstehen. Gründe für diese Fehler können ungenau lokalisierte markante Punkte, eine zu ungenaue Approximation der Kamera durch das Kameramodell, falsche Punktkorrespondenzen oder ungenaue Lösungen der Algorithmen sein. Weiterhin wurde für die Rekonstruktion immer nur ein kleiner Teil der zur Verfügung stehenden Informationen betrachtet, da jeder Weltpunkt aus nur einer Punktkorrespondenz rekonstruiert worden ist. Es ist aber häufig so, dass ein Punkt in mehreren Bildern detektiert werden kann. Diese zusätzlichen Detektionen sollen für eine genauere Rekonstruktion des Modells verwendet werden. Der Bündelausgleich nutzt diese Detektionen sowie deren geometrischen Abhängigkeiten untereinander, um die Rekonstruktion zu verbessern.

Der Name des Algorithmus kommt von den Lichtstrahlen, die von jedem 3D-Weltpunkt ausgestrahlt werden und gebündelt in den Kamerazentren zusammenlaufen. Diese passt er durch die Veränderung der Weltpunkte sowie der Kameraparameter an, um den Fehler im Modell zu minimieren [TMHF00]. Als Startwert benötigt der Algorithmus ein initiales 3D-Model. Dieses wird unter Betrachtung aller rekonstruierten Weltpunkte, den observierten markanten Punkten und allen Kameras so angepasst, dass das Modell möglichst gut zu den observierten markanten Punkten passt.

Im nächsten Kapitel 4.1 soll genauer darauf eingegangen werden, welches Problem der Bündelausgleich mathematisch zu lösen versucht. Im Anschluss daran, wird in Kapitel 4.2 erklärt, wie die im Bündelausgleich verwendete Fehler- und Verlustfunktion Einfluss auf die Fehlerminimierung im Modell nimmt. Abschließend

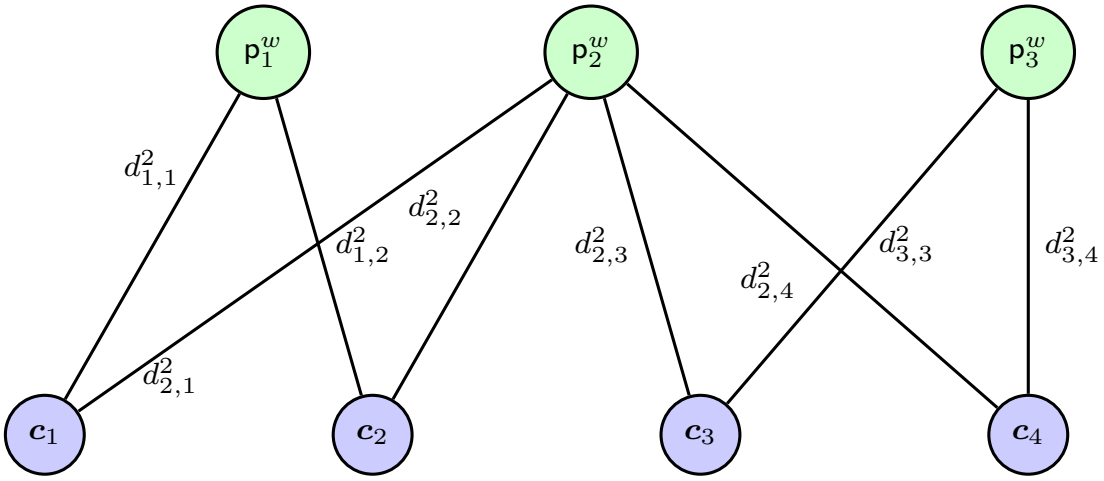


Abbildung 4.1: Schematische Darstellung der Verbindungen zwischen modellierten Weltpunkten und Kameras. Die grünen Knoten stellen die rekonstruierten Weltpunkte dar und die blauen die Kameras durch ihre Kameraparameter. Die Kanten stellen dar, welcher Weltpunkt von welcher Kamera gesehen wird. An ihnen sind die Kosten im trivialen Fall, als quadratischer Rückprojektionsfehler $d_{j,k}^2$ eingetragen.

geht Kapitel 6.3 auf den expliziten Schleifenschluss ein, der erst durch die Verwendung des Bündelausgleich für eine genauere Rekonstruktion verwendet werden kann.

4.1 Problemdefinition

Der Bündelausgleich hat das Ziel, eine Konfiguration von Weltpunkten, Kameraposen und Kameraparametern zu finden, die möglichst gut zu den observierten Punktkorrespondenzen passt, also möglichst konsistent ist. Um das Ziel zu erreichen, wird dem Algorithmus eine Menge an Weltpunkten W, die Menge der Kameraparameter der einzelnen Kameras C, sowie die Menge der observierten markanten Bildpunkte I übergeben. Jedem Weltpunkt $\mathsf{p}_j^w \in W$ sind die markanten Bildpunkte $\mathsf{p}_{j,k}^p \in I$ zugeordnet. Der Bildpunkt $\mathsf{p}_{j,k}^p$ steht dabei für den j-ten Bildpunkt im k-ten Bild. Es ist also bekannt, welchem Weltpunkt welche Bildpunkte zugeordnet sind. Hierdurch können für die Positionsbestimmung der Weltpunkte und die Bestimmung der Kameraparameter nun die Informationen der Punktkorrespondenzen aus dem gesamten Modell einfließen. In Abbildung 4.1 ist die Verbindung der einzelnen Daten schematisch dargestellt. Die Observationen werden dabei durch die Kanten zwischen den grün dargestellten Weltpunkten und den blau dargestellten Kameras visualisiert.

Um die Güte des Modells zu quantifizieren, wird eine Kostenfunktion aufgestellt, die berechnet wie gut die geschätzten Parameter aus dem Modell zu den in den Bildern observierten markanten Punkten passen. Sind die berechneten Kosten hoch, passen die aktuellen Modellparameter nur schlecht zu den Observationen aus den Bildern. Um die Kosten zu senken passt der Algorithmus nun die Kameraparameter und die Positionen der rekonstruierten Weltpunkte an. Wie die Anpassung durchgeführt wird, wird dabei durch Ableiten der Kostenfunktion berechnet. Durch mehrmaliges Anpassen der einzelnen Parameter soll so eine Stelle im Parameterraum angenähert werden, an der die Kosten minimal sind. Wichtig hierbei ist, dass die Positionen der markanten Punkte und die Punktkorrespondenzen nicht verändert werden.

Wie gut die Position eines Weltpunktes, die Kameraparameter und die Observation des Weltpunktes im Bild zueinander passen, geben deren Kosten an. Sie werden durch die Kostenfunktion $f(\boldsymbol{c}_k, \mathsf{p}^p_{j,k}, \mathsf{p}^w_j)$ berechnet, die neben dem observierten Punkt $\mathsf{p}^p_{j,k}$ und seinem korrespondierenden rekonstruierten 3D-Weltpunkt p^w_j, die Kameraparameter $\boldsymbol{c}_k$ zu dieser Korrespondenz übergeben bekommt. Sie werden benötigt, um p^w_j mit dem Kameramodell aus Kapitel 3.1 auf den Punkt $\hat{\mathsf{p}}^p_{j,k}$ zu projizieren.

Im naiven Fall berechnet f den quadrierten Rückprojektionsfehler

$$\begin{aligned} f(\boldsymbol{c}_k, \mathsf{p}^p_{j,k}, \mathsf{p}^w_j) &= |\mathsf{p}^p_{j,k} - \hat{\mathsf{p}}^p_{j,k}|^2 \\ &= d^2_{j,k} \qquad , \end{aligned} \tag{4.1}$$

der sich aus der quadrierten geometrischen Distanz d^2 zwischen einem observierten Punkt $\mathsf{p}^p_{j,k}$ und dem im Modell auf einen Kamerasensor projizierten Weltpunkt $\hat{\mathsf{p}}^p_{j,k}$ ergibt. In vielen Fällen existieren im Modell allerdings sehr ungenaue und falsche Korrespondenzen, die einen ungewöhnlich hohen quadrierten Rückprojektionsfehler verursachen. Sie werden üblicherweise wie in Kapitel 4.2 erklärt durch eine Verlustfunktion die auf die eigentliche Kostenfunktion angewendet wird gemindert.

Die Kosten zwischen einem 3D-Weltpunkt und einem seiner korrespondierenden Observationen, werden von der Kostenfunktion für jede Korrespondenz zwischen einem Weltpunkt und einem observierten Punkt berechnet. In Abbildung 4.1 sind sie an den Kanten zwischen den blau dargestellten Kameras und den grün dargestellten Weltpunkten eingetragen. Je kleiner die Summe über alle Kosten sind, desto besser passt das rekonstuierte Modell zu den observierten markanten Punkten. Daher werden die Kosten über alle rekonstruierten Kameras und Weltpunkte berechnet, aufsummiert und minimiert:

$$\underset{C,W}{\operatorname{argmin}} \sum_{j=1}^{|W|} \sum_{k=1}^{|C|} f(\boldsymbol{c}_k, \mathsf{p}^p_{j,k}, \mathsf{p}^w_j) \tag{4.2}$$

Eine besondere Herausfoderung des Bündelausgleichs ist die Größe des Problems. Das in Kapitel 3.1 vorgestellte Kameramodell wird durch 12 Kameraparameter, die aus $\boldsymbol{R}$ und $\boldsymbol{t}$ für die extrinsischen Kameraparameter, sowie aus F_x, F_y, κ_1, κ_2, c_x und c_y für die intrinsischen Kameraparameter besteht, beschrieben. Die Weltpunkte im Modell haben insgesamt 3-Parameter für die x-, y- und z-Koordinate. Somit ergibt sich bei einem Modell, dass aus n Bildern mit m Weltpunkten rekonstruiert wurde, eine Anzahl an zu optimierenden Parametern von $n \times 12 + m \times 3$.

4.2 Kostenfunktion

Die Kostenfunktion modelliert beim Bündelausgleich das Kameramodell mathematisch und führt somit die in Kapitel 3.1 beschriebene Projektion eines Weltpunktes auf den Kamerasensor durch. Hiermit lässt sich der bereits in Kapitel 4.1 beschriebene Rückprojektionsfehler errechnen, der darüber Aufschluss gibt, wie gut das Modell zu den observierten markanten Punkten der Eingabebilder passt.

Die Eingaben der Kostenfunktion sind davon abhängig, welche Parameter im Modell optimiert werden sollen und welche als gegeben angenommen werden. Üblich ist es, dass sowohl die Kameraparameter als auch die rekonstruierten Weltpunkte angepasst werden. Es ist aber nicht notwendig alle Kameraparameter zu optimieren. Wird die Rekonstruktion mit Bildern durchgeführt, die mit einer einzigen zuvor kalibrierten Kamera aufgenommen wurden, so können die intrinsischen Kameraparameter als gegeben angenommen werden. Der Grund hierfür ist, dass sie bei einer reinen Poseänderung der Kamera gleich bleiben. In diesem Szenario passt der Optimierungsalgorithmus nur die extrinsischen Kameraparameter und die Position der Weltpunkte an. Prinzipiell ist es aber möglich, alle Kameraparameter für unterschiedliche Kameras in einem Modell anzupassen.

Ein Problem der Kostenfunktion sind eventuelle Ausreißerkorrespondenzen, die einen ungewöhnlich hohen Rückprojektionsfehler und somit hohe Kosten verursachen. Durch die hohen Kosten haben sie einen besonders starken Einfluss auf den Optimierungsalgorithmus und halten ihn dadurch davon ab, ein korrektes Minimum der aufsummierten Kostenfunktion zu finden. Damit der Einfluss der Ausreißer auf das Modell nicht zu groß ist, werden sogenannte Verlustfunktionen verwendet. Sie reduzieren die Kosten der Korrespondenzen, die über einem Schwellwert liegen und ordnen diese neu zu, indem sie auf den quadrierten Rückprojektionsfehler angewendet werden und so die kosten für f berechnen. Die einfachste Verlustfunktion ist die triviale Verlustfunktion:

$$g_{\text{Trivial}}(d^2) = d^2 \tag{4.3}$$

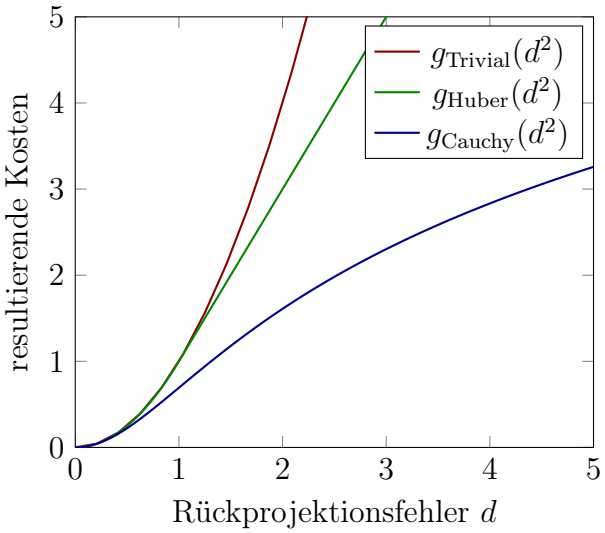


Abbildung 4.2: Plot der Trivialen-, der Huber- und der Cauchy-Verlustfunktion. Es ist deutlich die Wirkung der Verlustfunktionen auf den quadrierten Rückprojektionsfehler zu sehen. Bei der Anwendung der trivialen Verlustfunktion steigen die Kosten viel stärker an, als bei der Huber- und Cauchy-Verlustfunktion. Auch zwischen der Huber- und Cauchy-Verlustfunktion ist noch mal ein deutlicher Unterschied in der Gewichtung von d^2 zu sehen.

Sie lässt den quadrierten Rückprojektionsfehler d^2, wie in Abbildung 4.2 zu sehen unverändert und ist somit auch nicht robust gegen Ausreißer. Im Gegenteil, durch die Quadrierung der Rückprojektionsfehler werden Ausreißer wesentlich stärker gewichtet als Inlier. Die Huber-Verlustfunktion

$$g_{\text{Huber}}(d^2) = \begin{cases} d^2 & d^2 \leq 1 \\ 2|d| - 1 & d^2 > 1 \end{cases} \tag{4.4}$$

lässt den quadrierten Rückprojektionsfehler d^2 nur unter dem Schwellwert 1 unverändert. Über dem Schwellwert 1 wird d^2 durch den Term $g_{\text{Huber}}(d^2) = 2|d| - 1$ geringer gewichtet. Eine weitere Verlustfunktion ist die Cauchy-Verlustfunktion:

$$g_{\text{Cauchy}}(d^2) = \log(1 + d^2) \tag{4.5}$$

Sie verringert d^2 nicht erst ab einem Schwellwert sondern direkt und wirkt viel stärker als die Huber-Verlustfunktion auf diese ein. Der Unterschied zwischen diesen beiden Verlustfunktionen wird durch deren Plot in Abbildung 4.2 noch einmal besonders deutlich.

Die Funktion, die die optimalen Modellparameter findet, würde unter der Verwendung der Kostenfunktion

$$f(\boldsymbol{c}_k, \mathsf{p}^p_{j,k}, \mathsf{p}^w_j) = g_{\text{Huber}}(|\mathsf{p}^p_{j,k} - \hat{\mathsf{p}}^p_{j,k}|^2) \tag{4.6}$$

wie folgt aussehen:

$$\begin{aligned} &\underset{C,W}{\operatorname{argmin}} \sum_{j=1}^{|W|} \sum_{k=1}^{|C|} f(\boldsymbol{c}_k, \mathsf{p}^p_{j,k}, \mathsf{p}^w_j) \\ =&\underset{C,W}{\operatorname{argmin}} \sum_{j=1}^{|W|} \sum_{k=1}^{|C|} g_{\text{Huber}}(|\mathsf{p}^p_{j,k} - \hat{\mathsf{p}}^p_{j,k}|^2) \end{aligned} \tag{4.7}$$

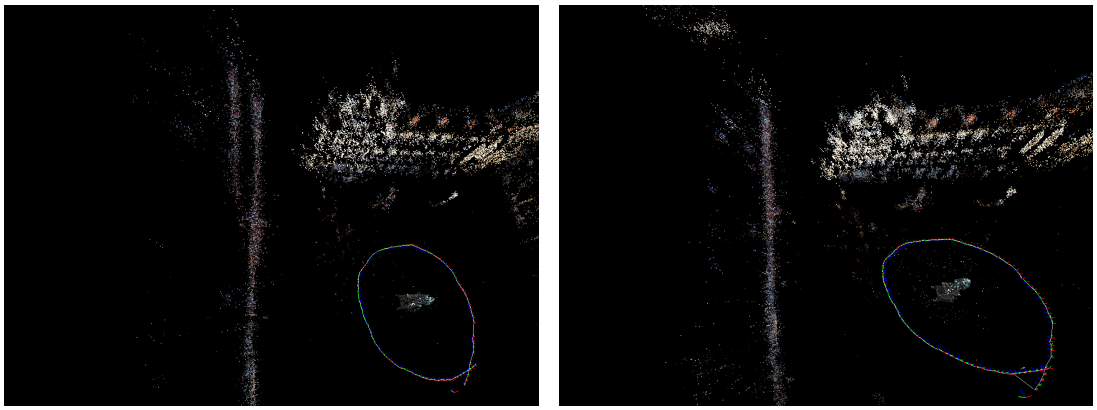
Neben den hier vorgestellten Verlustfunktionen gibt es noch viele weitere. Diese Arbeit beschränkt sich allerdings auf diese beiden.

4.3 Optimierung

Für die Minimierung des Rückprojektionsfehlers beim Bündelausgleich lassen sich verschiedene Optimierungsverfahren verwenden, von denen einige in Kapitel 1.2.2 vorgestellt wurden. Bei all diesen Verfahren wird nach den Parametern gesucht, an denen die Kostenfunktion ein Minimum hat. Der Levenberg-Marquardt-Algorithmus ist der beim Bündelausgleich am weitesten verbreitete und der in dieser Arbeit verwendete Algorithmus. Er soll in diesem Kapitel kurz vorgestellt werden.

Der Levenberg-Marquardt-Algorithmus ist ein Optimierungsverfahren zum Lösen nicht linearer Ausgleichsprobleme, wie dem hier beschriebene Bündelausgleich. Er versucht die Eingabeparameter für eine Kostenfunktion zu finden, denen die Kostenfunktion die niedrigsten Kosten zuordnet. Der Algorithmus wendet zur Minimierung der Kosten das Gauß-Newton-Verfahren sowie den Gradientenabstieg iterativ an, um die optimalen Parameter zu finden. Beide werden vom Algorithmus für die nächste Annäherung an das Minimum gewichtet. Der Gradientenabstieg hat auf die nächste Näherung dann starken Einfluss, wenn sich die Eingabeparameter noch weit entfernt von einem Optimum befinden und der Gauss-Newton-Algorithmus dann, wenn sich die Eingabeparameter nah an einem lokalen Minimum befinden.

Der Levenberg-Marquardt-Algorithmus lässt sich speziell auf den Bündelausgleich anpassen, um die Optimierung zu beschleunigen. Dabei wird die spezielle Struktur des Bündelausgleichproblems ausgenutzt. Stellt man die Abhängigkeiten zwischen den Elementen in einem rekonstruieren Modell einer typischen Szene tabellarisch dar, so stellt man fest, dass ein Element nur wenige Abhängigkeiten zu den anderen Elementen im Modell besitzt. Mit dieser Struktur kann man bestimmte Vorteile bei Matrix Berechnungen ausnutzen um sie zu beschleunigen.



(a) **(b)**

Abbildung 4.3: In Abbildung 4.3a ist eine Rekonstruktion aus dem Datensatz Jesuitenplatz ohne Schleifenschluss zu sehen. Man kann deutlich den Versatz Hauswand im linken Teil des Bildes erkennen. Bild 4.3b zeigt die selbe Szene mit nur einem Schleifenschluss unten rechts im Bild. Der Versatz der Hauswand im oberen linken Teil des Bildes wurde ausgeglichen.

Diese Matrizen werden als dünn besetzt bezeichnet und die entsprechenden Bündelausgleichsverfahren als dünn besetzter Bündelausgleich (englisch: sparse bundle adjustment) [RH00].

4.4 Expliziter Schleifenschluss

Der Bündelausgleich verringert den Fehler im Modell stark, dennoch entsteht häufig vor allem bei großen Modellen ein sogenannter Drift, dessen Ursache die Akkumulation kleiner Fehler über mehrere Bilder hinweg ist. Er wird vor allem dann sichtbar, wenn das Ende der Kameratrajektorie wieder auf den Anfang der Kameratrajektorie stößt, so dass die Trajektorie eine Schleife bildet. An diesen Stellen hat das Modell im Falle eines Drifts einen Versatz, der auch als Bruch im Modell bezeichnet wird [Dec12]. Verbindet man die beiden Enden des Modells, indem man die Korrespondenzen zwischen den Bildern an der Bruchstelle herstellt, kommt ein expliziter *Schleifenschluss* zustande. An dieser Stelle ist der Rückprojektionsfehler zunächst besonders hoch, da die rekonstruierten Kameraposen und die rekonstruierten Weltpunkte beider Enden durch den Versatz nicht zueinander passen. Optimiert man das Modell nun noch mal mit einem Bündelausgleich, so wird der Drift im Modell ausgeglichen, sodass dieser mit dem Bruch im Modell verschwindet. Dabei wird das komplette Modell angepasst, um den Fehler an der Bruchstelle auszugleichen. Abbildung 4.3a zeigt eine Rekonstruktion aus dem Datensatz Je-

suitenplatz ohne Schleifenschluss. Hier ist in der oberen linken Ecke deutlich zu erkennen, dass die selbe Hauswand zweimal mit einem Versatz rekonstruiert wurde. Durch den Schleifenschluss im Modell, das in Abbildung 4.3b zu sehen ist, ist der Versatz verschwunden. Der Schleifenschluss ist durch eine zusätzliche Verbindung zwischen der ersten rekonstruierten Kamera und einer der letzten Kameras zustande gekommen. Im Bild wird dies durch eine kleine weiße Linie zwischen den beiden Kameras dargestellt.

An welchen Stellen ein Schleifenschluss zustande kommt kann im Programm automatisch entschieden werden. Eine Möglichkeit ist die Pose einer neu hinzugefügten Kamera mit den bereits vorhanden Kameraposen zu vergleichen und nach der Überprüfung bestimmter Kriterien einen Schleifenschluss herzustellen. Kriterien die ein potentielles Bildpaar für einen Schleifenschluss erfüllen muss könnten eine gemeinsame Blickrichtung und eine geeignete Basislinie zwischen den beiden Kameras sein. Im für diese Arbeit implementierten Programm muss ein Schleifenschluss allerdings manuell durch eintragen von Bildpaaren hergestellt werden.

Kapitel 5

Vorstellung des implementierten Systems

Neben diesem Dokument ist begleitend ein Softwareframework entwickelt worden, das eine praktische Umsetzung der 3D-Rekonstruktion aus einer monokularen Bilderserie realisiert. Ziel war es, ein Programm zu implementieren, das die Rekonstruktion eines 3D-Modells und dessen Visualisierung ermöglicht. Dabei ist eine Reihenfolge der zu rekonstruierenden Bilder und eine Kamerakalbrierung durch den Anwender vorgegeben. Weiterhin ist an das zu entwickelnde Programm die Anforderung gestellt worden, dass es ein Modell robust rekonstruiert und dabei eine dichte Punktwolke mit möglichst vielen 3D-Weltpunkten erzeugt. Um die Wirkung der einzelnen Parameter und Algorithmen auf die Rekonstruktion in einer Visualisierung oder durch Statistiken zu demonstrieren, sollte es möglich sein, die Programmparameter einfach zu ändern und die Algorithmen auszutauschen.

Dieses Kapitel erklärt das implementierte Programm und ist in drei Unterkapitel aufgeteilt. In Unterkapitel 5.1 soll der praktische Programmablauf erklärt werden, der sich zum theoretischen Ablauf aus Kapitel 3 in einige für die Praxis wichtigen Details unterscheidet. Im Anschluss hieran soll in Kapitel 5.2 ein grober Überblick über die Benutzeroberfläche und deren bereitgestellten Funktionen gegeben werden. Abschließend beschreibt Kapitel 5.3 einige Details der Implemntierung und stellt die verwendeten Softwarebibliotheken vor.

5.1 Programmablauf

In diesem Kapitel wird auf die praktische Implementiertierung des Programmablaufs für eine 3D-Rekonstruktion eingegangen. Wie in Abbildung 5.1 zu sehen ist, wird an das Programm eine vom Benutzer festgelegte Liste an Bildpaaren sowie eine XML-Datei mit intrinsischen Kameraparametern übergeben. An die Liste

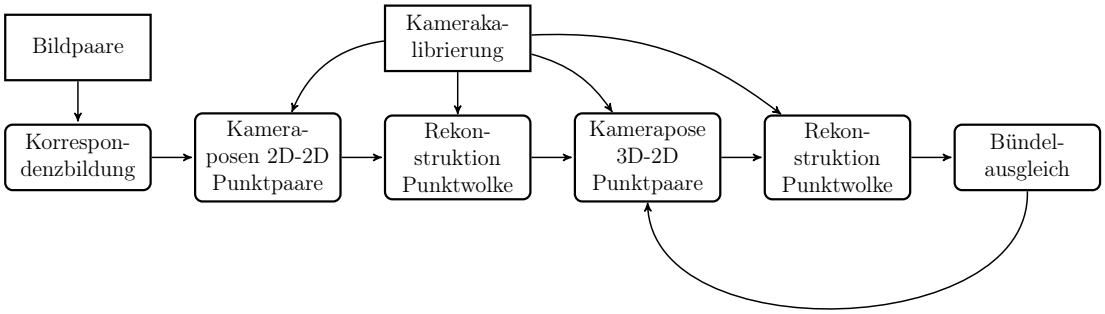


Abbildung 5.1: Visualisierung des Programmablaufs. Die Eingabedaten werden durch die Rechtecke mit spitzen Ecken und die Algorithmen durch die Rechtecke mit runden Ecken visualisiert. Zwischen den einzelnen Algorithmen und Eingabedaten findet ein Datenfluss statt, der durch die Pfeile visualisiert ist.

mit Bildpaaren wird die Anforderung gestellt, dass ab dem zweiten Bildpaar jedes Bildpaar ein Bild enthalten muss, das bereits eingelesen wurde. Die Reihenfolge ist deswegen wichtig, weil sich an den bereits eingelesenen Bildern die Positionierung neuer Kameras und Weltpunkte orientiert. Die Liste der Bildpaare enthält die Pfade zu den für die Rekonstruktion zu verwendenden Bilder, die alle eine Kamera mit den selben Einstellungen der Optik aufgenommen hat. Diese Kamera wird durch die intrinsischen Kameraparameter aus der Kalibrierdatei beschrieben.

Zwischen den einzelnen Bildpaaren lassen sich nun Merkmalskorrespondenzen mit den vorgestellten Algorithmen aus Kapitel 2 erstellen. Jeder Korrespondenz und den dazugehörigen markanten Punkten wird eine ID zugeordnet. Wird in einem Bildpaar zwischen einem Punkt, der bereits eine ID hat und einem Punkt der noch keine ID hat, eine Korrespondenz gebildet, so wird dem Punkt ohne ID die bereits vorhandene zugewiesen. Hierdurch ergeben sich Gruppen an Merkmalskorrespondenzen, die die Observationen eines einzelnen Weltpunktes darstellen. Im Falle eines Schleifenschlusses kann es vorkommen, dass eine Korrespondenz zwischen zwei markanten Punkten gebildet wird, die zwei unterschiedliche IDs haben. In diesem Fall wird die kleinere ID ausgewählt und alle Vorkommen der größeren ID durch die kleinere ersetzt.

Aus den gebildeten Punktkorrespondenzen des ersten Bildes lässt sich zur Initialisierung des 3D-Modells, wie in Kapitel 3.3 beschrieben, die Fundamentalmatrix berechnen. Anschließend können aus dieser die Posen der Kameras, mit denen das zugehörige Bildpaar aufgenommen wurde, rekonstruiert werden. Durch falsche oder schlechte Punktkorrespondenzen ist allerdings eine fehlerhafte Berechnung der Fundamentalmatrix möglich. Deswegen wird mit einem *RANSAC*-Algorithmus eine bestmöglich passende Fundamentalmatrix geschätzt, bei der möglichst viele Punktpaare die Epipolarbedingung erfüllen. Die Ausgabe des

RANSAC-Algorithmus ist neben der geschätzten Fundamentalmatrix eine Liste an Ausreißerkorrespondenzen. Diese Korrespondenzen können nun aus der Menge aller Korrespondenzen entfernt werden. Hiernach trianguliert der Tringulationsalgorithmus 3D-Weltpunkte anhand der verbliebenen Punktkorrespondenzen. Im nächsten Schritt entfernt das Programm die Ausreißerkorrespondenzen zwischen dem nächsten Bildpaar mit Hilfe einer neu errechneten Fundamentalmatrix. Da eines der beiden Bilder des neuen Bildpaars schon verarbeitet worden ist, können Korrespondenzen zwischen 2D-Bildpunkten und 3D-Weltpunkten hergestellt werden. Diese 2D-3D-Punktkorrespondenzen dienen dem Algorithmus zum Lösen des Perspective-n-Point-Problems als Eingabe, um die Pose der dritten Kamera zu rekonstruieren. Der Triangulationsalgorithmus errechnet nun für die noch nicht triangulierten Punktkorrespondenzen des aktuellen Bildpaars weitere 3D-Weltpunkte. Im darauffolgenden Schritt wird der Bündelausgleich durchgeführt, um bereits entstandene Fehler im 3D-Modell zu minimieren. Alle folgenden Bildpaare werden beginnend mit der Lösung des Perspektive-n-Point-Problems nachdem selben Schema dem Modell hinzugefügt. Nachdem alle Bildpaare verarbeitet worden sind, kann abschließend ein ausführlicher Bündelausgleich durchgeführt werden, um die Genauigkeit des Modells zu maximieren.

Das im Programm verwendete Kameramodell unterscheidet sich leicht von dem Kameramodell das in Kapitel 3.1 vorgestellt wurde. Es ist das von der Bildverarbeitungsbibliothek OpenCV verwendete Kameramodell[1]. Es wurde gewählt, da es die Verzerrungen eines Objektivs durch drei Parameter mehr beschreibt, als das in Kapitel 3.1 vorgestellte Modell. Durch die Verwendung dieses Kameramodells wurde sich erhofft, dass die Rekonstruktion des 3D-Modells noch genauer durchgeführt werden kann.

5.2 Benutzeroberfläche und Programmfunktionen

Die Benutzeroberfläche des entwickelten Programms wurde mit Qt in der Programmiersprache C++ umgesetzt. Sie dient zur Visualisierung und Einstellung der Programmparameter. Die Programmoberfläche ist horizontal in zwei Teile aufgeteilt. Im rechten Teil lassen sich Einstellungen vornehmen und Statistiken anzeigen. Der linke Teil dient zur Anzeige der Ergebnisse der Korrespondenzbildung oder zur Visualisierung des rekonstruierten 3D-Modells, wie es in Abbildung 5.2 zu sehen ist. Die Oberfläche bietet insgesamt drei Ansichten, eine zur Korrespondenzbildung, eine zum Initialisieren des 3D-Modells und eine für die abschließende Bündelausgleichung. Die einzelnen Ansichten können durch den Reiter an Markierung (1) umgestellt werden. Im Einstellungsbereich, der mit der Markierung (3) gekenn-

[1] Website: http://docs.opencv.org/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html

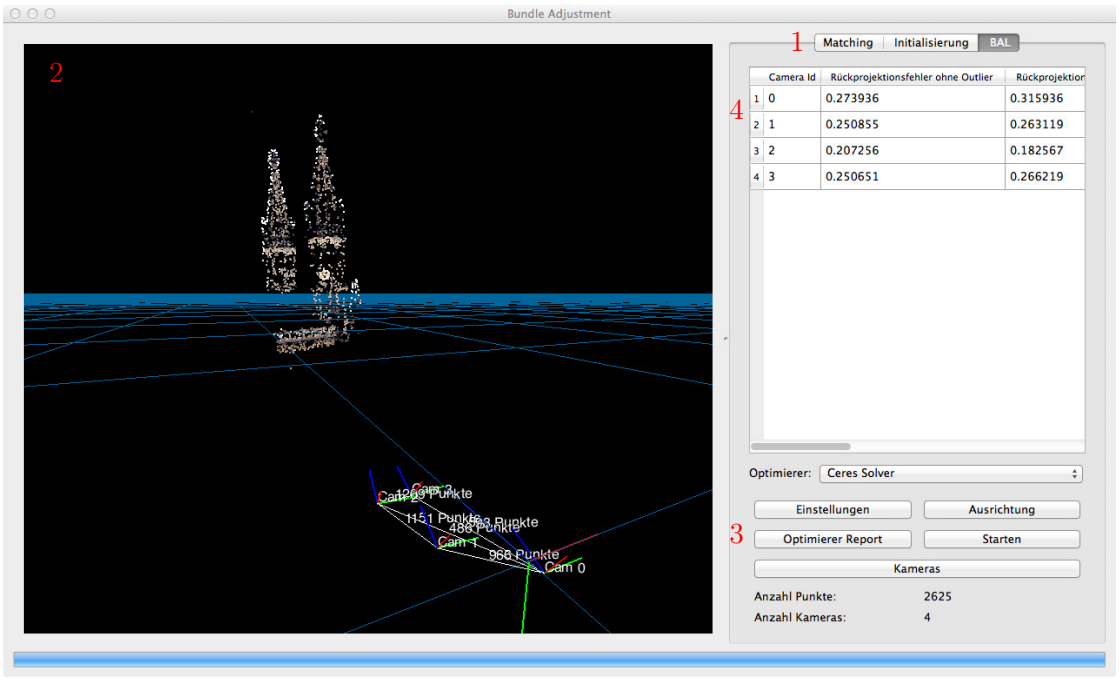


Abbildung 5.2: Screenshot der Benutzeroberflächer des implementierten Rekonstruktionsprogramms. Die einzelnen Zahlen im Bild markieren die Leiste zum Wechseln der Ansicht (1), den Bereich zur Visualisierung des 3D-Modells und von Punktkorrespondenzen (2), den Bereich zum Öffnen von Einstellungen (3) und den Bereich zum Anzeigen von Statistiken (4).

zeichnet ist, lassen sich Dialoge für Einstellungen oder zum Laden von Dateien aufrufen. Im Informationsbereich, durch Markierung (4) gekennzeichnet, werden Informationen zu den Ergebnissen der Algorithmen in Textform angezeigt.

Der Bereich zur Korrespondenzbildung lässt sich über den Reiter „Matching“ aufrufen. In der Ansicht lassen sich im Einstellungsbereich die Bildpaare für die Korrespondenzbildung hinzufügen oder durch eine Bildpaardatei im- bzw. exportieren. Zudem wird hier der NNDR-Schwellwert als auch der verwendete Merkmalsdeskrtiptor eingestellt. Im Visualisierungsbreich zeigt das Programm die Bildpaare und die Punktkorrespondenzen wie in Abbildung 2.4 an.

Die Initialisierungs Ansicht zeigt das Programm nach der Auswahl des gleichnamigen Reiters an. Im Visualisierungsbereich kann das initialisierte 3D-Modell mit einem virtuellen Rundflug betrachtet werden. Die Steuerung erfolg dabei über die Tasten W, A, S und D sowie mit der Maus bei gedrückter linker Maustaste. Der Informationsbereich zeigt in dieser Ansicht Statistiken, wie Anzahl der Inlierkorrespondenzen, der druchschnittliche Rückprojektionsfehler etc. zur Beurteilung des initialisierten Modells an. Der Einstellungsbereich bietet hier die Möglichkeit, über

ein Untermenü, Einstellungen an den Algorithmen zur Initialisierung vorzunehmen oder die Ausrichtung des 3D-Modells zu ändern. Über den Button „Kameras“ lässt sich ein Popup zur Anzeige von Statistiken und Parametern der einzelnen Kameras aufrufen. Es zeigt einige Kameraparameter an und durch einen Rechtsklick auf eine Kamera kann die Sichtpyramide der entsprechenden Kamera angezeigt werden. Über den Button „Kalibrierungsdatei laden“ lässt sich eine Datei mit intrinsischen Kameraparametern der Kamera, mit denen die Eingabebilder erstellt wurden, laden.

Die Ansicht für den Bündelausgleich ist in Abbildung 5.2 zu sehen und über den Reiter BAL erreichbar. Sie unterscheidet sich nur leicht von der Ansicht zur Initialisierung des 3D-Modells. Über den Button „Einstellungen“ können Einstellungen, wie die verwendete Verlustfunktion oder die Anzahl der Iterationen für den Levenberg-Marquardt-Algorithmus vorgenommen werden. Der Button „Optimierer“ Report öffnet ein Fenster, in dem sich Informationen zu den einzelnen Iterationen des Levenberg-Marquardt-Algorithmus nach einem Bündelausgleich anzeigen lassen.

Die Menüleiste des Programms bietet über das Menü „Ansicht“ einige Einstellungen für die Visualisierung des 3D-Modells an. Hier kann unter anderem bestimmt werden, welche Informationen zu den einzelnen Kamerapaaren angezeigt werden oder ob z. B. nur die Punktwolke sichtbar sein soll. Über das Menü „Tools“ kann ein Tool zur Kamerakalibrierung geöffnet werden. Mit ihm nimmt der Benutzer die Kalibrierung der Kamera für die Aufnahme der monokularen Bilderserie vor. Um eine Kamera zu kalibrieren, muss ein schachbrettartiges Kalibriermuster verwendet und die Maße der einzelnen Quadrate, so wie die Anzahl derer inneren Ecken eingegeben werden.

5.3 Implementierung

Dieses Kapitel stellt kurz die verwendeten Libraries sowie deren Verwendung im implementierten Framework vor. Das komplette Framework ist in der Programmiersprache C++ unter Verwendung des C++11 Standards implementiert. Die Benutzeroberfläche ist, wie in Abschnitt 5.2 erwähnt, mit Qt 4.8[2] erstellt worden. Für die Datenstrukturen und Algorithmen der Bildverarbeitung wurde OpenCV[3] verwendet. Diese Bibliothek bietet die verwendeten Algorithmen für SURF, das Berechnen der Fundamentalmatrix und zum Lösen des Perspective-n-Point-Problems, sowie die FLANN Schnittstelle zur Korrespondenzbildung an. Den A-KAZE Algorithmus stellte OpenCV nicht bereit. Deswegen wurde die Implementierung der

[2] Website: http://www.qt-project.org
[3] Website: http://www.opencv.org

Entwickler[4], die ebenfalls mit OpenCV implementiert ist, verwendet. Der verwendete Levenberg-Marquardt-Algorithmus stammt aus der Bibliothek Ceres-Solver[5]. Sie wurde speziell für nicht lineare Optimierungsprobleme entwickelt und ist auf große Bündelausgleichsprobleme ausgelegt. Für lineare Algebra Berechnungen, wie sie bei der Bestimmung der Kamerapose aus der essentiellen Matrix oder bei der Projektion eines Weltpunktes auf den Kamerasensor benötigt werden, wird die Bibliothek Eigen[6] verwendet.

[4]Website: http://www.robesafe.com/personal/pablo.alcantarilla/kaze.html
[5]Website: http://ceres-solver.org
[6]Website: http://eigen.tuxfamily.org

Kapitel 6

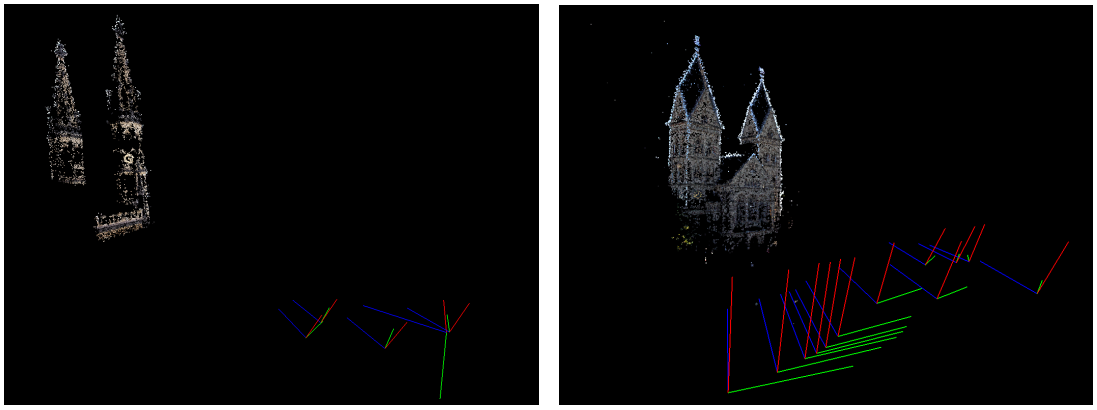
Experimente und Ergebnisse

Dieses Kapitel untersucht die im Programm verwendete Vorgehensweise zum inkrementellen Erstellen eines 3D-Models und zeigt Vor- und Nachteile dieser auf. Für die Experimente wurden drei Bilddatensätze verwendet:

- Burg Stolzenfels (4 Bilder)
- Herz-Jesu-Kirche (32 Bilder)
- Jesuitenplatz (125 Bilder)

Beispielrekonstruktionen der Datensätze sind in den Abbildungen 6.1a für Burg Stolzenfels, 6.1b für die Herz-Jesu-Kirche und 6.2 für den Jesuitenplatz zu sehen. Die Datensätze befinden sich auf der beigelegten DVD zu dieser Arbeit. Die in den Datensätzen enthaltenen Bilder sind alle mit einer Spiegelreflexkamera, die über ein 30mm Objektiv verfügt, in einer Auflösung von 6 Megapixeln aufgenommen worden. Zu den Bilddatensätzen existieren unterschiedliche Bildpaardateien, an denen sich Unterschiede in der Rekonstruktion durch Schleifenschluss oder die Anzahl der verwendeten Bilder, verdeutlichen lassen. Für die Erhebung der Daten wurde mit SURF und A-KAZE bei unterschiedlichen NNDR-Schwellwerten Merkmalskorrespondenzen gebildet. Initialisiert wurden die ausgewerteten Modelle jeweils mit unterschiedlichen Einstellungen, abhängig von Test und Größe des zu initialisierenden Modells. Anschließend ist auf dem initialisierten Modell mehrfach mit unterschiedlich vielen Iterationen ein Bündelausgleich ausgeführt worden. Am Ende jedes Bündelausgleichs wurden unterschiedliche Messungen am 3D-Modell vorgenommen.

Die größten Modelle, die während der Tests rekonstruiert werden konnten enthielten bis zu 650.000 Punkte und 120 Kameras. Anhand der großen Zahlen ist zu erkennen, dass das Prinzip des implementierten Algorithmus funktioniert. Eines der größten Modelle ist in Abbildung 6.2 zu sehen.



(a) (b)

Abbildung 6.1: Abbildung 6.1a zeigt eine Rekonstruktion aus dem Datensatz Burg Stolzenfels mit 8223 Punkten, rekonstruiert aus 4 Bildern. Das 3D-Modell der Herz-Jesu-Kirche ist in Abbildung 6.1b zu sehen. Es enthält 65112 Punkte und ist aus 12 Bildern rekonstruiert worden.

Datensatz	**Bildpaare**	**Kameras**	**Punkte**
Stolzenfels	3	4	8082
Stolzenfels Schleifenschluss	5	4	8630
Herz-Jesu	11	12	15394
Herz-Jesu Schleifenschluss	16	12	15805
Jesuitenplatz	119	120	129014
Jesuitenplatz Schleifenschluss	124	120	129503

Tabelle 6.1: Die Tabelle zeigt Informationen zu den Rekonstruierten Modellen, die für die Statistiken aus den Abbildungen 6.4 und 6.3 verwendet wurden.

Der Rest dieses Kapitels ist wie folgt aufgebaut. In Kapitel 6.1 wird untersucht wie sich der Rückprojektionsfehler bei unterschiedlich vielen Iterationen Bündelausgleich verhält. Weiterhin wird die für einen Bündelausgleich benötigte Zeit gemessen, um eine Aussage darüber machen zu können, wie viele Iterationen sinnvoll sind.

6.1 Wirkung des Bündelausgleich auf den Rückprojektionsfehler

In diesem Kapitel soll die Minimierung des Rückprojektionsfehlers durch den Bündelausgleich untersucht werden. Dabei wird der Fragestellung nachgegangen, wie

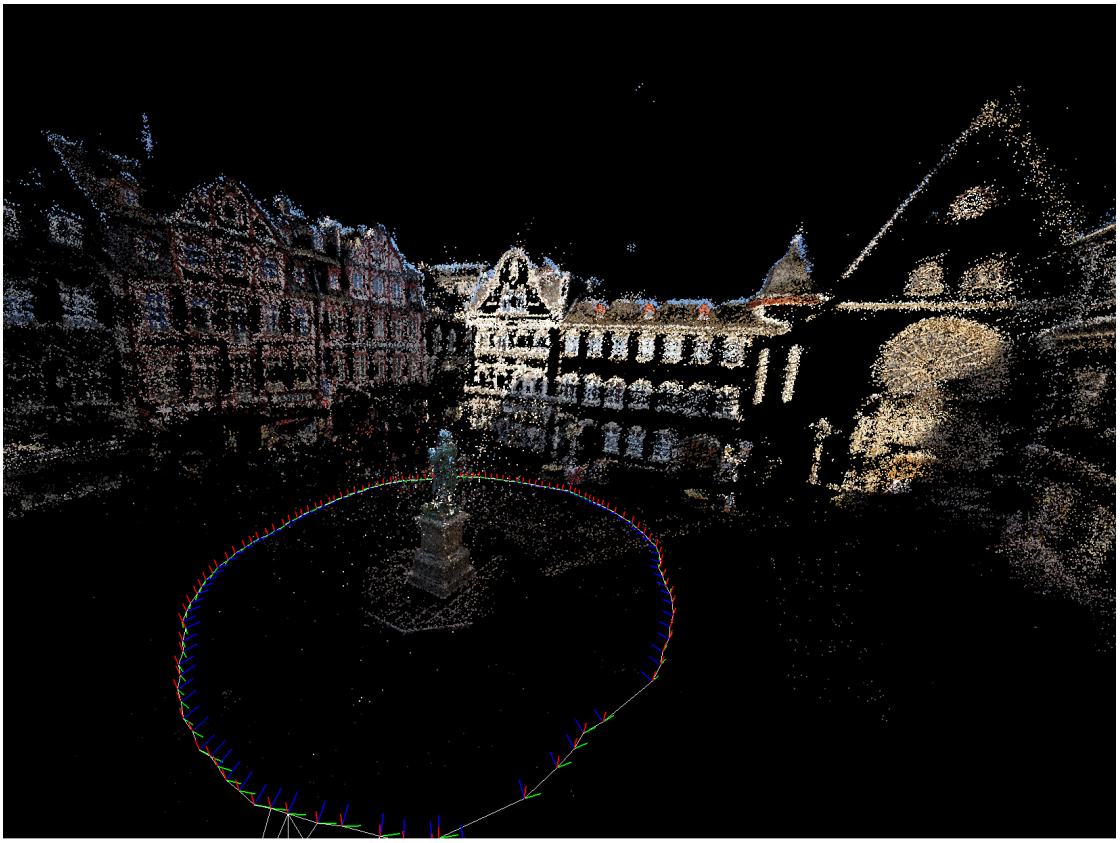


Abbildung 6.2: Das Bild zeigt eine Rekonstruktion des Jesuitenplatz mit 635.152 rekonstruierten Punkten und 120 rekonstruiertenKameras. Durch die hohe Anzahl an Punkten und einen geringen Ausreißerschwellwert bei der Epipolarbedingung sind in den Gebäuden sehr gut Details, wie die Struktur der Fensterrose im rechten Teil des Bildes, zu erkennen.

sich der Rückprojektionsfehler bei einer bestimmten Anzahl an Punkten und Kameras im Modell verringert und wie viel Zeit dafür benötigt wird. Für alle in diesem Kapitel aufgenommenen Datensätze wird die Huber-Verlustfunktion verwendet. Informationen über die Anzahl der für die Statistiken verwendeten Punkte und Kameras im Modell können Tabelle 6.1 entnommen werden.

Zunächst soll auf die Minimierung des Rückprojektionsfehlers ohne Ausreißer Korrespondenzen eingegangen werden. Dazu wird der Bezug zwischen der Anzahl an Iterationen im Bündelausgleich und dem Rückprojektionsfehler erklärt. Wie in Abbildung 6.3 zu sehen ist, fällt der Rückprojektionsfehler während der ersten Iterationen am stärksten und flacht dann ab, bis er sich auf einem lokalen Minimum der Kostenfunktion einpendelt. An diesem Minimum schwankt der quadrierte Rückprojektionsfehler ohne Ausreißer nur noch minimal. Das heißt, er kann auch

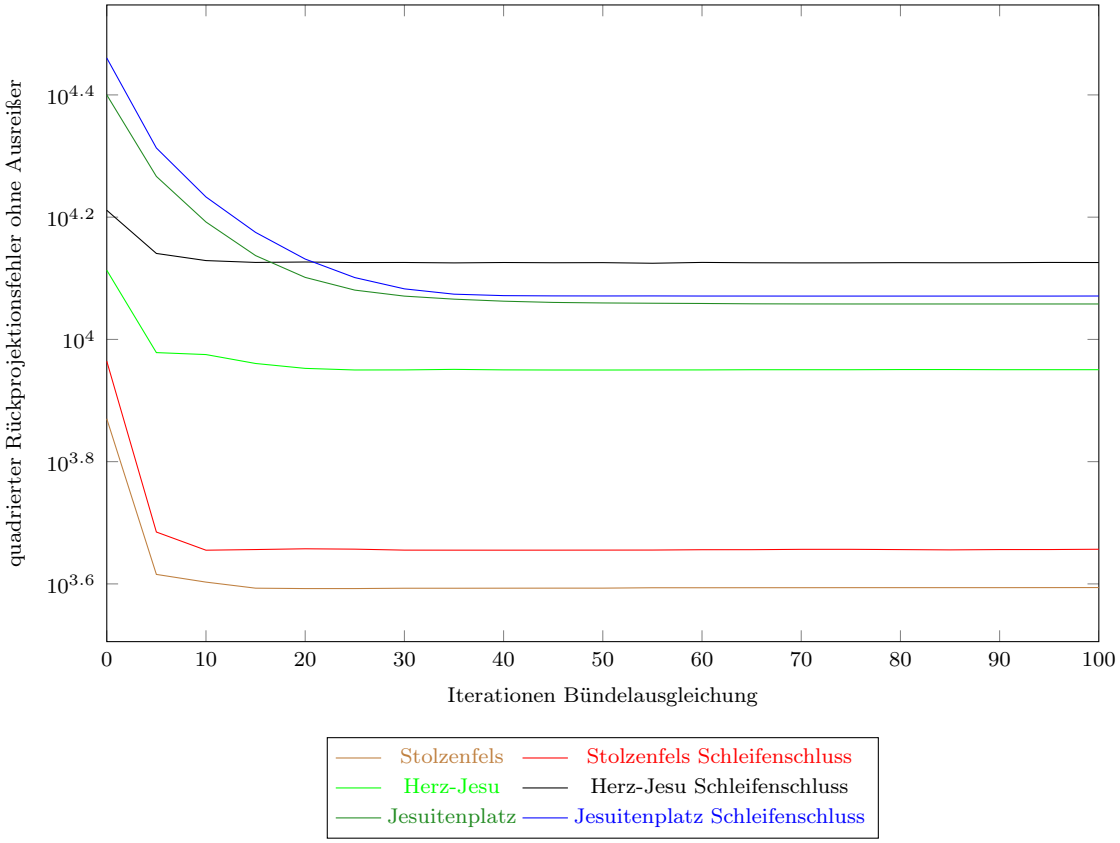


Abbildung 6.3: Das Diagramm stellt die Anzahl an Iterationen des Levenberg-Marquardt-Algorithmus der Minimierung des quadrierten Rückprojektionsfehlers gegenüber. Für die Berechnung des quadrierten Rückprojektionsfehlers wurden keine Ausreißerkorrespondenzen berücksichtigt, die einen Rückprojektionsfehler größer als fünf Pixel haben. Die Anzahl an Punkten, Kameras und verwendete Bildpaare können Tabelle 6.1 entnommen werden.

etwas steigen, da die Optimierung der Ausreißerkorrespondenzen Einfluss auf diesen haben kann, auch wenn sie nicht berücksichtigt werden. Hierbei ist zu beachten, dass bei der Minimierung der quadrierte Rueckprojektionsfehler durch die Huber-Verlustfunktion abgeschwächt wurde und somit der Wert der Kostenfunktion nicht mehr direkt geometrisch zu interpretieren ist.

Weiterhin fällt auf, dass der Rückprojektionsfehler bei Modellen mit einer großen Anzahl an Punkten und Kameras langsamer abflacht. Bei kleinen Modellen, wie aus dem Datensatz Burg-Stolzenfels, fällt dieser viel stärker. Das heißt, der Aufwand für die Minimierung des Rückprojektionsfehlers in großen Modellen ist durch die große Anzahl an Elementen im Modell und die dadurch erhöhte Komplexität des Optimierungsproblems viel höher. Folglich steigt auch die benötigt Zeit

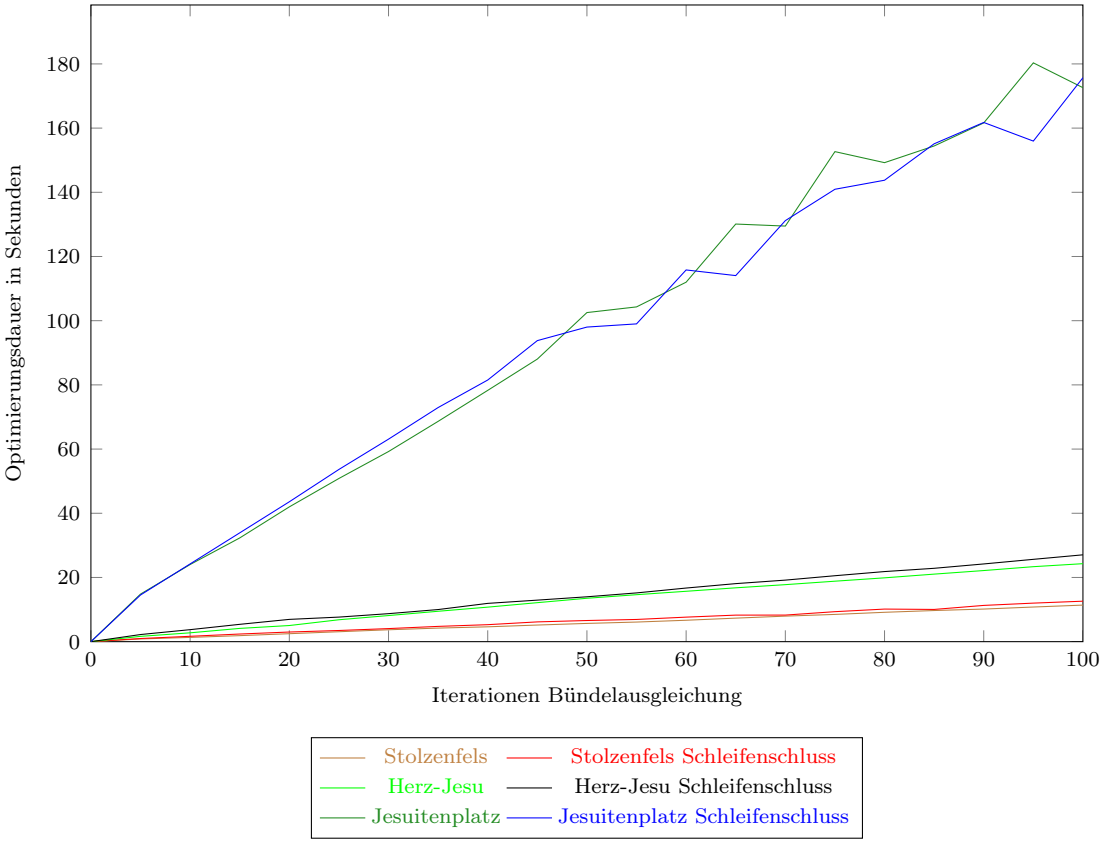


Abbildung 6.4: Das Diagramm stellt die Anzahl an Iterationen des Levenberg-Marquardt-Algorithmus der benötigten Zeit pro Iteration gegenüber. Anzahl an Punkten, Kameras und verwendeten Bildpaare können Tabelle 6.1 entnommen werden.

für eine gewisse Anzahl an Iterationen des Bündelausgleichs. Dies ist in Abbildung 6.4 zu sehen. Hier ist gut zu erkennen, dass die für die Optimierung benötigte Zeit nahezu linear mit der Anzahl der Iterationen steigt. Zudem steigt die benötigte Zeit bei den Datensätzen mit mehr Elementen im Modell wesentlich schneller an, als bei denen mit wenig Elementen.

Zusammengefasst bedeuten die Beobachtungen, dass für ein großes Modell mit vielen Elementen prinzipiell mehr Iterationen für den Bündelausgleich durchgeführt und mehr Zeit verwendet werden muss. Ab einer Gewissen Anzahl an Iterationen lohnt es sich allerdings nicht mehr weitere Iterationen durchzuführen, da sich der Rückprojektionsfehler nur noch minimal verringert.

6.2 Untersuchung der Verlustfunktionen

In diesem Kapitel soll die Wirkung unterschiedlicher Verlustfunktionen untersucht werden. Um die Unterschiede aufzuzeigen, wurde ein initiales Modell erstellt, das in den Zwischenschritten mit 10 Iterationen unter Verwendung der Huber-Verlustfunktion optimiert wurde. Die initialisierten Modelle sind anschließend mit unterschiedlichen Verlustfunktionen ein weiteres Mal optimiert worden. Um das Verhalten des quadrierten Rückprojektionsfehlers zu beobachten, wurde die Summe des quadrierten Rückprojektionsfehlers $\sum d^2$ im Modell mit und ohne Einbeziehung der Ausreißerkorrespondenzen aufgezeichnet. Zusätzlich ist die Anzahl Gesamtkorrespondenzen und Inlierkorrespondenzen im 3D-Modell gemessen worden. Tabelle 6.2 stellt die gewonnenen Ergebnisse anhand der Modelle Herz-Jesu-Kirche und Jesuitenplatz gegenüber.

Bei der Untersuchung der Verlustfunktionen fällt auf, dass es wichtig ist, überhaupt eine Verlustfunktion zu verwenden. Ohne die Verwendung einer Verlustfunktion ist es häufig vorgekommen, dass das Modell nicht korrekt rekonstruiert wurde. Stattdessen war das Modell nach der Optimierung stark verfälscht, wodurch z. B. Wände einen Versatz zeigten. Auch in den geführten Statistiken ist deutlich erkennbar, dass die Anzahl der Inlierkorrespondenzen sinkt und der quadrierte Rückprojektionsfehler ohne Ausreißer steigt, was auf eine Verschlechterung des Modells hindeutet. Die Verschlechterung ist dadurch zu erklären, dass bei der initialen Rekonstruktion zwischenzeitlich unter Verwendung der Huber-Verlustfunktion optimiert wurde und dies einen postiven Einfluss hatte. Bei der anschließenden Optimierung ohne Verlustfunktion wurden die einzelnen Kostenterme neu gewichtet. Ausreißerkorrespondenzen mit einem hohen Rückprojektionsfehler haben durch das quadratische Wachstum der Kostenfunktion einen viel größeren Anteil an den Gesamtkosten als Inlier mit einem geringeren Rückprojektionsfehler. Dadurch können die Kosten im Modell stärker gesenkt werden, wenn der Rückprojektionsfehler von Ausreißerkorrespondenzen veringert, dafür aber der Rückprojektionsfehler von Inlierkorrespondenzen etwas erhöht wird. Die Kosten durch Verringern des Rückprojektionsfehlers der Ausreißer sinken somit schneller, als die Kosten der Inlier durch Erhöhen des Rückprojektionsfehlers wachsen.

Die Optimierung des Modells mit einer Verlustfunktion hat das Modell visuell verbessert. Durch die nachträgliche Optimierung des Modells mit der Huber- oder Cauchy-Verlustfunktion ist in beiden Fällen der Rückprojektionsfehler ohne Ausreißerkorrespondenzen weiter gesunken und der Anteil der Inlierkorrespondenzen gestiegen. Auffällig ist, dass unter Verwendung der Cauchy-Verlustfunktion der Anteil der Inlierkorrespondenzen tendentiell niedriger ist, als bei der Verwendung der Huber-Verlustfunktion. Bei allen Versuchen bis auf den Versuch mit dem Modell Jesuitenplatz ist die Anzahl der Inlier bei Verwendung der Cauchy-Verlustfunktion niedriger, als bei der Verwendung der Huber-Verlustfunktion. Dies ist dadurch zu

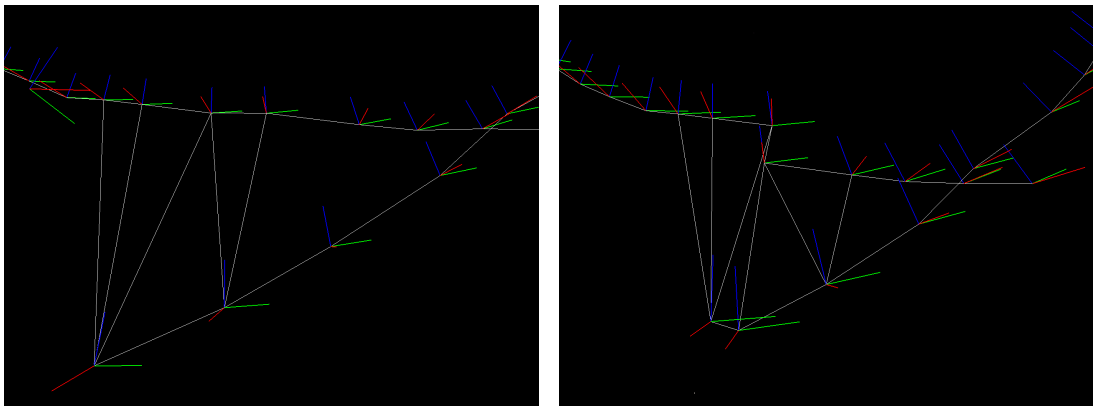
Datensatz	Herz-Jesu-Kirche			Jesuitenplatz		
Verlustfunktion	Trivial	Cauchy	Huber	Trivial	Cauchy	Huber
$\sum d^2$ ohne Ausreißer nach Initialisierung	8170,25			88318,6		
$\sum d^2$ mit Ausreißer nach Initialisierung	$6{,}5 \cdot 10^6$			$1{,}88 \cdot 10^8$		
$\sum d^2$ ohne Ausreißer nach Bündelausgleich	33423,40	6129,26	6871,50	332449,0	45432,3	48935,6
$\sum d^2$ mit Ausreißer nach Bündelausgleich	$2{,}12 \cdot 10^6$	$7{,}18 \cdot 10^6$	$4{,}22 \cdot 10^6$	$1{,}20 \cdot 10^8$	$2{,}33 \cdot 10^8$	$1{,}80 \cdot 10^8$
Inlierkorrespondenzen nach Bündelausgleich	28138	39240	39299	318281	403005	402983
Inlierkorrespondenzen nach Initialisierung	36308			399913		
Gesamtanzahl Korrespondenzen	39652			406215		

Tabelle 6.2: Die Tabelle zeigt die Wirkung der drei Verlustfunktionen beim Bündelausgleich anhand der 3D-Modelle Herz-Jesu-Kirche und Jesuitenplatz. Die gemessenen Daten wurden jeweils vor und nach dem Bündelausgleich aufgezeichnet. Gemessen wurde die Summe des quadrierten Rückprojektionsfehlers $\sum d^2$ im 3D-Modell, mit und ohne Einbeziehung der Ausreißerkorrespondenzen sowie die Anzahl der Inlierkorrespondenzen.

erklären, dass die Ausreißerkorrespondenzen die noch zu Inlierkorrespondenzen werden könnten, durch ihre geringe Gewichtung nicht stark in die Kostenfunktion einfließen. Dementsprechend werden sie nicht angepasst, um zu Inlierkorrespondenzen zu werden. Dies erkennt man auch an der Höhe des totalen quadratischen Rückprojektionsfehlers. Er ist nach einer Optimierung mit der Cauchy-Verlustfunktion viel höher als nach der Optimierung mit der Huber-Verlustfunktion. Die Kosten der Ausreißerkorrespondenzen werden durch die Cauchy-Verlustfunktion so stark gemindert, dass sie kaum Auswirkungen auf die Berechnung des lokalen Minimums haben.

6.3 Schleifenschluss

In diesem Abschnitt soll kurz auf die gesammelten Erfahrungen mit dem implementierten Schleifenschluss eingegangen werden. Der Schleifenschluss ist an verschieden großen Datensätzen getestet worden. Bei den Datensätzen Herz-Jesu-Kirche und Burg-Stolzenfels war der explizite Schleifenschluss allerdings sehr klein, da die Datensätze nicht viele Bilder enthalten. Ein deutlicher Drift konnte nur im Datensatz Jesuitenplatz erkannt werden. Dieser Datensatz hat genügend Bilder und die Kameratrajektorie ist groß genug, um einen Drift im Modell festzustellen.



(a) **(b)**

Abbildung 6.5: In den beiden Bildern sind rekonstruierte Kameras die durch Koordinatensysteme visualisiert sind zu sehen. Die weißen Linien zwischen den Kameras visualisieren die Kamerapaare, zwischen denen Korrespondenzen gebildet wurden. Abbildung 6.5a zeigt einen erfolgreichen Schleifenschluss. In Abbildung 6.5b ist die Kameratrajektorie durch einen großen Versatz zerstört.

Ein Drift im Datensatz Jesuitenplatz konnte immer dann festgestellt werden, wenn relativ wenige Punktkorrespondenzen aus den Bildern extrahiert worden sind. Auch nach einer ausführlichen Minimierung des Rückprojektionsfehlers ist der Bruch im Modell nicht mehr ausgeglichen worden. Allerdings genügt der Schleifenschluss mit nur einem Kamerapaar, um die Bruchstelle mit einem abschließenden Bündelausgleich zu schließen. Bei Modellen ohne Schleifenschluss, die viele Punktkorrespondenzen enthalten, ist der Drift im Modell so gering, dass dieser nicht erkennbar ist. Damit kann gefolgert werden, dass die große Anzahl an Punktkorrespondenzen einen positiven Einfluss auf die Präzision der 3D-Rekonstruktion hat. Grund hierfür ist, dass die Positionen der Kameras durch mehr Messungen besser geschätzt werden können.

Bei einigen Schleifenschlüssen ist es vorgekommen, dass das Modell an der Bruchstelle zerstört wurde. Die Vermutung ist, dass die Kameras beim Schleifenschluss eine problematische Pose zueinander eingenommen haben, wodurch viele Sichtstrahlen parallel zueinander verlaufen. Die hierbei rekonstruierten Weltpunkte wurden dann falsch rekonstruiert, so dass das Modell eine zu schlechte Initialisierung für den darauf folgenden Bündelausgleich war. Bei weiteren Tests mit Bildpaardateien, bei denen eine problematische Stellung zwischen Kamerapaaren vermieden wurde, ist das Problem nicht mehr aufgetreten. Abbildung 6.5 zeigt die Rekonstruierten Kameraposen eines zerstörten Modelles und den korrekt geschlossenen Schleifenschluss im Vergleich dazu. In beiden Bildern ist der Schleifenschluss durch voreinander liegende Kameras zustande gekommen, was problematisch sein

Datensatz	Stolzenfels		Herz-Jesu-Kirche		Jesuitenplatz	
Schleifenschluss	Ja	Nein	Ja	Nein	Ja	Nein
2 Observationen	69,26%	77,26%	54,43%	61,26%	84,88%	85,10%
3 Observationen	20,16%	16,53%	19,76%	20,10%	11,66%	11,6%
4 Observationen	10,58%	6,21%	10,10%	8,00%	2,65%	2,6%
5 Observationen	-	-	7,20%	5,29%	0,57%	0,53%
6 Observationen	-	-	5,91%	4,21%	0,17%	0,14%
7 Observationen	-	-	1,27%	0,68%	0,04%	0,03%
Observationen ≥ 8	-	-	1,33%	0,45%	0,03%	0,0%
3D-Weltpunkte	8746	8082	65603	68652	129503	129014

Tabelle 6.3: Die Tabelle zeigt die prozentuale Verteilung an 3D-Weltpunkten mit einer bestimmten Anzahl an Observationen in 3D-Modellen mit und ohne Schleifenschluss. Die Prozentangaben beziehen sich dabei auf die Gesamtanzahl an 3D-Weltpunkten in Modellen. Es ist zu erkennen, dass der Anteil an 3D-Weltpunkten mit mehr als zwei Observationen in Modellen mit Schleifenschlüssen immer etwas höher ist als in Modellen ohne Schleifenschluss.

kann. In Abbildung 6.5a konnte dennoch ein erfolgreicher Schleifenschluss zustande kommen. In Abbildung 6.5b hingegen wurde durch eine ungünstige Wahl der Kamerapaare das Modell zerstört.

Weiterhin wurde untersucht, wie sich der Rückprojektionsfehler durch einen Schleifenschluss verändert. Hier war unklar, ob dieser durch die Auflösung des Bruchs im Modell sinken oder steigen würde. Es wurde vermutet, dass der durchschnittliche Rückprojektionsfehler durch die Nichtlinearität des Problems nach dem Schleifenschluss sinken könnte. Hierzu wurde der durchschnittliche Rückprojektionsfehler ohne Ausreißer, nach 100 Iterationen Bündelausgleich für Modelle aus dem gleichen Datensatz, mit und ohne Schleifenschluss verglichen. Hierbei muss erwähnt werden, dass in den rekonstruierten Modellen mit Schleifenschluss mehr Punktkorrespondenzen vorhanden sind, als in denen ohne Schleifenschluss. Grund hierfür ist, dass beim Schleifenschluss zusätzliche Korrespondenzen gebildet werden. Das Testergebnis zeigt, dass der durchschnittlichen Rückprojektionsfehler in den Rekonstruktionen mit Schleifenschluss leicht angestiegen ist. Auch in Abbidlung 6.3 ist der totale quadrierte Rückprojektionsfehler ohne Aureißer geringer als bei Modellen mit Schleifenschluss.

Ein weiterer Vorteil des Schleifenschluss ist, dass sich die Anzahl an Observationen für einzelne 3D-Weltpunkte erhöht. Diese zusätzlichen Observationen können für eine präzisere Rekonstruktion des 3D-Modells genutzt werden. Sie kommen dadurch zustande, dass an einer Stelle eines Schleifenschlusses ein Bereich einer Szene von mindestens drei Kameras aufgenommen wird. Tabelle 6.3 stellt den Anteil an Weltpunkten mit einer bestimmten Anzahl an Observationen in einem 3D-Modell gegenüber. Der Anteil an 3D-Weltpunkten mit mehr als zwei Observationen ist

in den Modellen mit Schleifenschluss immer etwas höher als in Modellen ohne Schleifenschluss. Im Modell Jesuitenplatz ist der Unterschied zwischen einer Rekonstruktion mit und ohne Schleifenschluss nicht sehr groß. Das liegt daran, dass nur ein kleiner Anteil an Bildpaaren für einen Schleifenschluss in diesem Modell existiert.

Kapitel 7

Zusammenfassung und Ausblick

In diesem letzten Kapitel werden die in dieser Arbeit vorgestellten Methoden für 3D-Rekonstruktionen aus monokularen Bilderserien noch einmal zusammengefasst. Im Anschluss hieran wird ein Ausblick gegeben, wie das gewonnene Wissen weiter vertieft und darauf weiter aufgebaut werden kann.

7.1 Zusammenfassung

Das Ziel der Arbeit war, die Methoden, die zur 3D-Rekonstruktion benötigt werden, grundlegend zu erklären. Ergänzend sollte hierzu eine robuste Software entwickelt werden, an der sich die vorgestellten Methoden testen lassen und mit der es möglich ist, eine dichte Punktwolke zu rekonstruieren. Dazu wurde zu Beginn dieses Dokuments ein einleitender Überblick über die großen Möglichkeiten sowie die Vorteile der 3D-Rekonstruktion aus einer monokularen Bilderserie gegeben. Im Anschluss hieran wurde der aktuelle Stand der Wissenschaft zu den einzelnen Teilthemen, die für eine 3D-Rekonstruktion verstanden werden müssen, vorgestellt. Hierzu gehört die Korrespondenzbildung zwischen markanten Merkmalen aus verschiedenen Bildern, die Berechnung der dreidimensionalen Struktur sowie die Kamerabewegung und das Optimiern des entsandenen 3D-Modells. Hierbei ist schnell klar geworden, dass es weit aus mehr interessante Möglichkeiten gibt, als in dieser Arbeit vorgestellt werden können. Deswegen wurde sich in den folgenden Kapiteln größtenteils auf die Erklärung gut erprobter und weit verbreiteter Methoden beschränkt.

Im darauffolgenden Kapitel wurde die Korrepondenzbildung zwischen markanten Punkten erklärt. Hierzu wurde der weit verbreitete SURF-Algorithmus und der Neuere A-KAZE Algorithmus zum Detektieren und Beschreiben von markanten Punkten vorgestellt. Zwischen den Merkmalen zweier Bilder sind im Anschluss mit

Hilfe von FLANN Korrespondenzen hergestellt worden. Sie dienen als Eingabe für alle folgenden Algorithmen, die Kameraposen und 3D-Weltpunkte rekonstruieren.

Um die Algorithmen zur 3D-Rekonstruktion zu verstehen, musste zuerst die Funktionsweise einer Kamera und die Beziehungen zwischen mehreren Kameras verstanden werden. Aus diesem Grund wurde das Lochkameramodell und die Epipolargeometrie erklärt. Hierauf aufbauend wurde die Bestimmung von Kameraposen aus 2D-2D- und 2D-3D-Punktkorrespondenzen sowie die Triangulierung von 3D-Weltpunkten hergeleitet. Mit ihnen lässt sich iterativ ein 3D-Modell initialisieren. Da sich beim iterativen Aufbau eines Models Fehler akkumulieren können, müssen diese ausgeglichen werden. Dazu wurde nach der Initialisierung des 3D-Modells eine globale Bündelausgleichung angewendet und diese ausführlich erklärt.

Nach der Erläuterung der theoretischen Grundlagen ist die implementierte Software vorgestellt worden. Dazu wurde auf den Programmablauf und dessen Unterschiede zur Theorie eingegangen. Im Anschluss hieran wurde die Benutzeroberfläche und die bei der Implementierung verwendeten Bibliotheken vorgestellt.

Das vorletzte Kapitel stellt dann die Ergebnisse der durchgeführten Experimente mit dem implementierten System vor. Hierbei konnten Statistiken aus drei verschiedenen Datensätzen mit unterschiedlichen Parametern ausgewertet werden. Durch eine hohe Anzahl an verwendeten Fotos und die entsprechend hohe Anzahl an 3D-Weltpunkten konnte dargelegt werden, dass das Vorgehen bei der Implementierung richtig war. Zudem wurden die Wirkung des Bündelausgleichs und die beim Bündelausgleich verwendeten Verlustfunktionen untersucht. Bei der Untersuchung des implementierten Schleifenschluss zur Erhöhung der Modellgenauigkeit stellte sich heraus, dass dieser vor allem optisch die Genauigkeit im Modell erhöht. Allerdings bietet er, bei einer besonderen Anordnung von rekonstruierten Kameras, die Schwachstelle, dass Weltpunkte falsch trianguliert werden, was das Modell möglicherweise zerstört.

7.2 Ausblick

Ausblickend kann gesagt werden, dass die entwickelte Software noch viel Potential für weitere Tests und Verbesserungen bietet. Diese hängen sicherlich auch von den unterschiedlichen Einsatzgebieten ab.

Zunächst wäre das Erkennen und Entfernen weiterer Ausreißer- und ungenauen Korrespondenzen wünschenswert, damit keine Weltpunkte mehr trianguliert werden, die potentiell das Modell zerstören können. Die Entfernung solcher Korrespondenzen würde weiterhin den Vorteil mit sich bringen, dass die Präzision sowie die Geschwindigkeit, mit der das Modell rekonstruiert wird, steigen. Das Problem, dass vor allem bei geringer Basislinie Weltpunkte falsch trianguliert werden, würde sich auch durch eine Triangulierung mit einer projektiven Faktorisierung lösen

lassen. Hierbei kommen mehrere Kameras zum Einsatz, um Punkte zu triangulieren. Um diesen Ansatz zu implementieren, sind allerdings größere Änderungen am Programmablauf vorzunehmen.

Weiterhin wäre eine Automatisierung zur Entscheidung, zwischen welchen Fotos Korrespondenzen gebildet werden, denkbar. Hierzu könnten Metadaten wie GPS-Informationen aus den Bildern genutzt werden. Eine andere Möglichkeit ist die Verwendung eines auf dem Erscheinungsbild des Fotos basierende Ortserkennung. Bei diesen Verfahren wird durch den Vergleich zweier Fotos geprüft, wie hoch die Wahrscheinlichkeit ist, ob man an einer bestimmten Stelle schon einmal war.

Eine andere interessante Vertiefung wäre den Levenberg-Marquardt-Algorithmus durch ein anderes Optimierungsverfahren ausztauschen und zu prüfen, ob hierdurch schneller Ergebnisse erzielt werden können. Hier wäre eines der in Kapitel 1.2.2 vorgestellten Verfahren wie ISAM2 oder konjugierte Gradienten vorstellbar.

Um die Modelle visuell anspruchsvoller darzustellen, ist die Erstellung eines Drahtgitternetz aus der erstellten Punktwolke sinnvoll. Über dieses Drahtgitternetz könnten dann Ausschnitte aus den Ursprungsbildern gelegt werden.

Anhang A

Nomenklatur

$\boldsymbol{I}$	Einheitsmatrix
$\boldsymbol{E}$	Essentielle Matrix
$\boldsymbol{K}$	Kalibrierungsmatrix
$\boldsymbol{F}$	Fundamentalmatrix
$\boldsymbol{P}$	Kameramatrix
w	Weltkoordinatensystem
c	Kamerakoordinatensystem
i	Bildkoordinatensystem
p	Pixelkoordinatensystem
I	Ein Bild
I_1	Bild 1
I_2	Bild 2
t	Ein Schwellwert
p^w	Dreidimensionaler Punkt im Weltkoordinatensystem
p^p	Zweidimensionaler Bildpunkt im Pixelkoordinatensystem von Bild 1
$\hat{\mathsf{p}}^p$	Projizierter zweidimensionaler Bildpunkt im Pixelkoordinatensystem von Bild 1
p^i	Zweidimensionaler Bildpunkt im Bildkoordinatensystem von Bild 1
p^c	Dreidimensionaler Punkt im Kamerakoordinatensystem von Bild 1
$\tilde{\mathsf{p}}^p$	Homogener zweidimensionaler Bildpunkt im Pixelkoordinatensystem von Bild 1
$\tilde{\mathsf{p}}^i$	Homogener zweidimensionaler Bildpunkt im Bildkoordinatensystem von Bild 1
$\tilde{\mathsf{p}}^c$	Homogener dreidimensionaler Punkt im Kamerakoordinatensystem von Bild 1
q^p	Zweidimensionaler Bildpunkt im Pixelkoordinatensystem von Bild 2
q^i	Zweidimensionaler Bildpunkt im Bildkoordinatensystem von Bild 2
q^c	Dreidimensionaler Punkt im Kamerakoordinatensystem von Bild 2

$\tilde{\mathsf{q}}^p$	Homogener zweidimensionaler Bildpunkt im Pixelkoordinatensystem von Bild 2
$\tilde{\mathsf{q}}^i$	Homogener zweidimensionaler Bildpunkt im Bildkoordinatensystem von Bild 2
$\tilde{\mathsf{q}}^c$	Homogener dreidimensionaler Punkt im Kamerakoordinatensystem von Bild 2
$\boldsymbol{H}$	Hessematrix
$\boldsymbol{H}_{\text{approx}}$	Approximierte Hessematrix
$I_{\sum}$	Ein Integralbild
D	Menge der Merkmalsvektoren
NNDR	Nearest Neighbor Distance Ratio
σ	Eine Oktave im Skalenraum
s	Eine Ebene im Skalenraum
v	Ein Merkmalsvektor
D_{xx}	Näherung der zweiten partiellen Ableitung in x-Richtung
D_{yy}	Näherung der zweiten partiellen Ableitung in y-Richtung
D_{xy}	Näherung der zweiten partiellen Ableitung in xy-Richtung
d_x	Eine Haarwaveletfilterung in x-Richtung
d_y	Eine Haarwaveletfilterung in y-Richtung
$\boldsymbol{a}$	Ein Punkt im Merkmalsraum
$\boldsymbol{b}$	Ein Punkt im Merkmalsraum
$\boldsymbol{c}$	Ein Punkt im Merkmalsraum
d_1	Distanz nächster Nachbar
d_2	Distanz zweit nächster Nachbar
$\boldsymbol{H}$	Bildhauptpunkt
$\boldsymbol{O}_c$	Optisches Zentrum
F	Brennweite
F_x	Brennweite mit Pixelgröße in x-Richtung verrechnet
F_y	Brennweite mit Pixelgröße in y-Richtung verrechnet
c_x	Position des Hauptpunktes in x-Richtung in Pixelkoordinaten
c_y	Position des Hauptpunktes in y-Richtung in Pixelkoordinaten
d_x	Horizontale Pixelgröße
d_y	Vertikale Pixelgröße
p^d	Zweidimensionaler verzerrter Bildpunkt im Bildkoordinatensystem
κ_1	Verzerrungskoeffizien 1
κ_2	Verzerrungskoeffizien 2
$\boldsymbol{R}$	Eine Rotationsmatrix
$\boldsymbol{t}$	Eine Translation
Cam	Bezeichner für eine Kamera
Cam_1	Bezeichner für Kamera 1
Cam_2	Bezeichner für Kamera 2

e_1	Epipol im Bild 1
e_2	Epipol im Bild 2
l_1	Epipolarlinie für Kamera 1
l_2	Epipolarlinie für Kamera 2
$\boldsymbol{K_1}$	Kalibrierungsmatrix Kamera 1
$\boldsymbol{K_2}$	Kalibrierungsmatrix Kamera 2
d	Rückprojektionsfehler
d^2	Quadrierter Rückprojektionsfehler
g	Eine Gerade
W	Menge der Weltpunkte
C	Menge der Vektoren der Kameraparameter
I	Menge der Bildpunkte
f	Kostenfunktion
g_{Trivial}	Triviale Verlustfunktion
g_{Huber}	Huber Verlustfunktion
g_{Cauchy}	Cauchy Verlustfunktion
$\boldsymbol{c}$	Ein Vektor aus der Menge der Kameraparameter

Anhang B

Inhalt der DVD

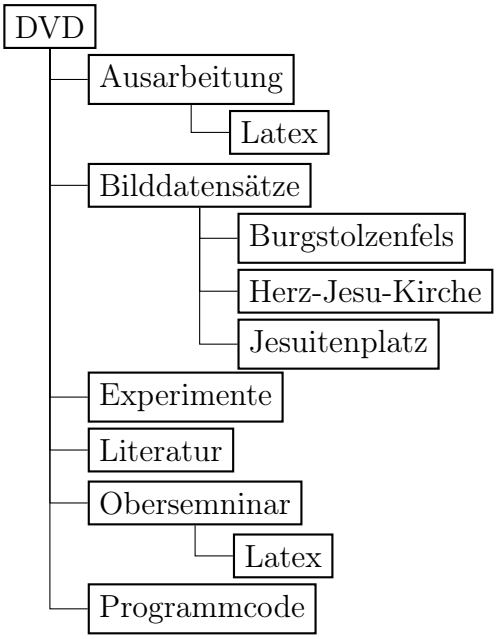


Abbildung B.1: Die Abbildung zeigt die Verzeichnisstruktur der beigelegten DVD.

Der Verzeichnisbaum der beigelegten DVD ist wie in Abbildung B.1 aufgebaut. Der Ordner Ausarbeitung enthält dieses Dokument als PDF-Datei und die Latexquelldateien, aus denen das PDF generiert worden ist. Im Ordner Bilddatensätze sind die Bilddatensätze, die für die Experimente verwendet worden sind und die entsprechenden Paar- bzw. Kalibrierdateien enthalten. Die bei den Experimenten entstandenen Statistiken sind als Excel-Dateien im Ordner Experimente gespeichert. Sämtliche Literatur, die digital als PDF für diese Arbeit verwendet worden ist, liegt im Ordner Literatur. Im Ordner Oberseminar sind die Folien,

die für die Präsentation beim Oberseminar der AG Aktives Sehen verwendet worden sind zu finden. Deren Latex-Quelldateien befinden sich im Unterordner Latex. Die Quelldateien des zu dieser Arbeit entwickelten Programms sind im Ordner Programmcode zu finden. Um den Programmcode zu kompilieren müssen die Programmbibliotheken und Programme

- CMake
- OpenCV
- Ceres-Solver
- Eigen3
- A-KAZE
- Qt 4.8

installiert sein. Danach sollte sich der Quellcode ohne größere Anpassungen durch das Ausführen von CMake und Make kompilieren lassen.

Anhang C

Tabellenverzeichnis

Anhang D

Abbildungsverzeichnis

Anhang E

Literaturverzeichnis

[ABD12] Alcantarilla, P. F. ; Bartoli, A. ; Davison, A. J.: KAZE Features. In: *Eur. Conf. on Computer Vision (ECCV)*, 2012

[AI08] Andoni, Alexandr ; Indyk, Piotr: Near-optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions. In: *Commun. ACM* 51 (2008), Januar, Nr. 1, 117–122. http://dx.doi.org/10.1145/1327452.1327494. – DOI 10.1145/1327452.1327494. – ISSN 0001–0782

[ANB13] Alcantarilla, P. F. ; Nuevo, J. ; Bartoli, A.: Fast Explicit Diffusion for Accelerated Features in Nonlinear Scale Spaces. In: *British Machine Vision Conf. (BMVC)*, 2013

[ASS+09] Agarwal, Sameer ; Snavely, Noah ; Simon, Ian ; Seitz, Steven M. ; Szeliski, Richard: Building Rome in a Day. (2009), September. http://research.microsoft.com/apps/pubs/default.aspx?id=101029

[ASSS10] Agarwal, Sameer ; Snavely, Noah ; Seitz, Steven M. ; Szeliski, Richard: Bundle Adjustment in the Large. In: *Proceedings of the 11th European Conference on Computer Vision: Part II*. Berlin, Heidelberg : Springer-Verlag, 2010 (ECCV'10). – ISBN 3–642–15551–0, 978–3–642–15551–2, 29–42

[BÅ10] Byröd, Martin ; Åström, Kalle: Conjugate Gradient Bundle Adjustment. In: *Proceedings of the 11th European Conference on Computer Vision: Part II*. Berlin, Heidelberg : Springer-Verlag, 2010 (ECCV'10). – ISBN 3–642–15551–0, 978–3–642–15551–2, 114–127

[BETVG08] Bay, Herbert ; Ess, Andreas ; Tuytelaars, Tinne ; Van Gool, Luc: Speeded-Up Robust Features (SURF). In: *Comput. Vis. Image Underst.* 110 (2008), Juni, Nr. 3, 346–359. http://dx.doi.org/10.1016/j.cviu.2007.09.014. – DOI 10.1016/j.cviu.2007.09.014. – ISSN 1077–3142

[Dec12] Decker, Peter: *Modellbasierte Kameraposebestimmung aus Bildern.* Der Andere Verlag, 2012. – ISBN 978–3–86247– 307–6

[FPS14] Forster, C. ; Pizzoli, M. ; Scaramuzza, D.: SVO: Fast Semi-Direct Monocular Visual Odometry. In: *Proc. IEEE Intl. Conf. on Robotics and Automation. 2014.*, 2014

[KJR+11] Kaess, M. ; Johannsson, H. ; Roberts, R. ; Ila, V. ; Leonard, J.J. ; Dellaert, F.: iSAM2: Incremental Smoothing and Mapping with Fluid Relinearization and Incremental Variable Reordering. In: *IEEE Intl. Conf. on Robotics and Automation, ICRA*. Shanghai, China, May 2011, S. 3281–3288

[KJR+12] Kaess, Michael ; Johannsson, Hordur ; Roberts, Richard ; Ila, Viorela ; Leonard, John ; Dellaert, Frank: iSAM2: Incremental Smoothing and Mapping Using the Bayes Tree. In: *International Journal of Robotics Research (IJRR)* (2012). http://frank.dellaert.com/pub/Kaess12ijrr.pdf. – To appear

[KM09] Klein, Georg ; Murray, David: Parallel Tracking and Mapping on a Camera Phone. In: *Proc. Eigth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'09)*. Orlando, October 2009

[KRD08] Kaess, Michael ; Ranganathan, Ananth ; Dellaert, Frank: iSAM: Incremental Smoothing and Mapping. In: *Transactions on Robotics (TRO)* (2008). http://frank.dellaert.com/pub/Kaess08tro.pdf

[KTS11] Kurz, Christian ; Thormählen, Thorsten ; Seidel, Hans-Peter: Bundle Adjustment for Stereoscopic 3D. In: *Proceedings of the 5th International Conference on Computer Vision/Computer Graphics Collaboration Techniques*. Berlin, Heidelberg : Springer-Verlag, 2011 (MIRAGE'11). – ISBN 978–3–642–24135–2, 1–12

[LA09] Lourakis, Manolis I. A. ; Argyros, Antonis A.: SBA: a software package for generic sparse bundle adjustment. In: *ACM Transactions on Mathematical Software* (2009), S. 1–30

[LHM00] Lu, Chien-Ping ; Hager, Gregory D. ; Mjolsness, Eric: Fast and Globally Convergent Pose Estimation from Video Images. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 22 (2000), Juni, Nr. 6, 610–622. http://dx.doi.org/10.1109/34.862199. – DOI 10.1109/34.862199. – ISSN 0162–8828

[Low99] Lowe, David G.: Object Recognition from Local Scale-Invariant Features. In: *Proceedings of the International Conference on Computer Vision-Volume 2 - Volume 2*. Washington, DC, USA : IEEE Computer Society, 1999 (ICCV '99). – ISBN 0–7695–0164–8, 1150–

[Low04] Lowe, David G.: Distinctive Image Features from Scale-Invariant Keypoints. In: *Int. J. Comput. Vision* 60 (2004), November, Nr. 2, 91–110. http://dx.doi.org/10.1023/B:VISI.0000029664.99615.94. – DOI 10.1023/B:VISI.0000029664.99615.94. – ISSN 0920–5691

[Moo90] Moore, Andrew W.: Efficient memory-based learning for robot control / University of Cambridge, Computer Laboratory. Version: November 1990. http://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-209.pdf. 1990 (UCAM-CL-TR-209). – Forschungsbericht

[MS05] Mikolajczyk, Krystian ; Schmid, Cordelia: A performance evaluation of local descriptors. In: *IEEE Transactions on Pattern Analysis & Machine Intelligence* 27 (2005), Nr. 10, 1615–1630. http://lear.inrialpes.fr/pubs/2005/MS05

[RH00] Richard Hartley, Andrew Z.: *Multiple View Geometry in Computer Vision, 2nd Edition*. Camebridge University Press, 2000

[Sze10] Szeliski, Richard: *Computer Vision: Algorithms and Applications*. 1st. New York, NY, USA : Springer-Verlag New York, Inc., 2010. – ISBN 1848829345, 9781848829343

[TMHF00] Triggs, Bill ; McLauchlan, Philip F. ; Hartley, Richard I. ; Fitzgibbon, Andrew W.: Bundle Adjustment - A Modern Synthesis. In: *Proceedings of the International Workshop on Vision Algorithms: Theory and Practice*. London, UK, UK : Springer-Verlag, 2000 (ICCV '99). – ISBN 3–540–67973–1, 298–372

[TV98] Trucco, Emanuele ; Verri, Alessandro: *Introductory Techniques for 3-D Computer Vision*. Upper Saddle River, NJ, USA : Prentice Hall PTR, 1998. – ISBN 0132611082

[VJ01] Viola, P. ; Jones, M.: Rapid object detection using a boosted cascade of simple features. In: *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on* 1 (2001), S. I–511–I–518 vol.1. http://dx.doi.org/10.1109/CVPR.2001.990517. – DOI 10.1109/CVPR.2001.990517. – ISSN 1063–6919

[YC12] Yang, Xin ; Cheng, Kwang-Ting (.: LDB: An Ultra-Fast Feature for Scalable Augmened Reality on Mobile Devices. In: *International Symposium on Mixed and Augmented Reality* (2012), November, S. 49 – 57