Terrainrekonstruktion aus Stereodaten von unterschiedlichen Posen

Diplomarbeit
von

Malte Wirkus

FB 3 Mathematik und Informatik
Universität Bremen

Erstgutachter:	Prof. Dr. Udo Frese
Zweitgutachter:	Dr. Bernd Gersdorf

September 2010

Zusammenfassung

In dieser Arbeit wird ein System entwickelt, das ein geometrisches Modell eines Geländes erstellt. Die hierfür notwendigen Daten werden über ein Stereokamerasystem aus unterschiedlichen Perspektiven aufgenommen. Mit den Bildern der Stereokamera werden dichte Punktwolken aus dem Gelände rekonstruiert und in ein Weltmodell überführt. Das Weltmodell ist ein regelmäßiges Dreiecksnetz, das sich dynamisch an die rekonstruierten Punkte anpasst. Es werden hierfür unterschiedliche Methoden untersucht. Bei der Integration der rekonstruierten Punkte in das Weltmodell wird vorausgesetzt, dass die Pose der Kamera in der Welt bekannt ist. Es werden Überlegungen für eine effiziente Implementierung des Verfahrens angestellt und eine Software entwickelt, die diese umsetzt. Das System wird anhand künstlicher Testdaten getestet. Mit ihnen wird auch die Leistungsfähigkeit des Systems untersucht.

Danksagung

Ich möchte Prof. Dr. Udo Frese danken, der mich stets mit gute Ratschlägen bei der Erstellung dieser Arbeit so gut unterstützt hat. Für die Bereitschaft zur Erstellung des zweiten Gutachtens danke ich Dr. Bernd Gersdorf. Für die Implementierung der Cross-Correlation-Überprüfung im GPU-Stereo Matching Algorithmus und einige Hinweise zu Beginn der Arbeit, danke ich Marc Hildebrandt. Außderdem gilt mein Dank meiner Familie und Freunden, die mir während der Arbeit zur Seite standen. Besonders sind Alia Asaad, Alexander Mann, Nadir Sunar, Till von Wenzlawowicz und Daniel Möhlmann zu nennen, die eine große Hilfe bei der Korrektur der Arbeit gewesen sind.

Inhaltsverzeichnis

1. Einleitung

Zunehmend werden Computersysteme eingesetzt, um in komplexen Umgebungen zu agieren. In der Robotik z.B. wächst das Interesse daran, dass sich die Roboter nicht mehr nur in perfekt durchgeplanten und arrangierten, industriellen Umgebungen, sondern in der "realen Welt" zurecht finden sollen. Um dies leisten zu können, müssen sie Informationen über die Welt haben, die von ihnen verarbeitet werden können.

Diese Beschreibung entspricht einer abstrakten Repräsentation des Umfelds (im folgenden auch "Welt" genannt), in dem das Computersystem arbeiten soll. Nun kann diese Darstellung der Welt dem System vom Entwickler mitgegeben werden, es werden also alle notwendigen Informationen von vornherein angegeben. Die "reale Welt" ist in ihrer Vielfalt aber kaum beschreibbar. Sie ist zu detailliert und zu groß. Es ist also wünschenswert, dass der Schritt der Abstraktion der Welt, vom Computersystem selber durchgeführt wird. Hierzu sind Sensoren nötig, aus denen Rohinformationen gewonnen werden, die dann verarbeitet werden, um aus ihnen auf Eigenarten der Welt schließen zu können. Diese Sensoren können zum Beispiel Kameras, Laserscanner oder Ultraschall-Sensoren sein.

Die Sensoren liefern Momentaufnahmen der Welt zum aktuellen Zeitpunkt. Sie sind aber in ihrer Reichweite begrenzt, da zum Beispiel ein Hindernis erst überwunden werden muss, um, mit Kameras, dessen Rückseite betrachten zu können. Wenn aber die Momentaufnahmen von früheren Zeitpunkten nicht in irgendeiner Form gesichert werden, so hat das Computersystem leider wieder die Vorderseite des Hindernis "vergessen". Um nun eine größere, nicht komplett einsichtige Umgebung erforschen zu können, ist es sinnvoll mehrere *lokale* Momentaufnahmen in einem *globalen* Weltmodell zusammenzufassen. Dieses kann dann mit der Zeit durch neue lokale Aufnahmen erweitert und verbessert werden.

In dieser Arbeit wird ein System entwickelt, das geometrische Momentaufnahmen der Welt in einem Weltmodell speichert und sich dann an diese Daten anpasst. Die Daten werden über ein Stereokamerasystem (ein System aus zwei Kameras, deren relative Stellung zueinander bekannt ist) aufgenommen. Aus den zweidimensionalen Bildern werden über Stereo-Matching Punktkorrespondenzen zwischen den beiden zeitgleich aufgenommen Bildern der Stereokamera (im folgenden "Bildpaar" genannt) hergestellt. Hieraus können dann, durch die bekannte relative Stellung der Kameras, 3D-Punkte rekonstruiert werden. Diese werden in ein Weltmodell überführt, das den Schwerpunkt dieser Arbeit darstellt, und in Form eines regelmäßigem Dreiecksnetz realisiert ist. Das Modell soll sich dynamisch an die rekonstruierten Punkte anpassen. Bei der Integration der re-

konstruierten Punkte in das Weltmodell wird vorausgesetzt, dass die Pose der Kamera in der Welt bekannt ist.

Sinnvolle Anwendungen für ein solches System könnten etwa die Analyse der Oberflächenbeschaffenheit unbekannter Gebiete oder die Erstellung einer Karte für autonome Roboter sein. Ein weiteres mögliches Anwedungsgebiet ist die Unterstützung bei der Fernsteuerung von Robotern. Je nach Gelände und Bauart des Roboters ist, für die Person die den Roboter steuert, die Umgebung häufig kaum zu erkennen. Das kann daran liegen, dass der Blickwinkel der Kamera möglicherweise zu klein ist, um alle wichtigen Bereiche einfangen zu können, oder aber, dass durch starke Bewegung des Roboters das Bild zu heftig wackelt. Eine virtuelle Ansicht des Geländes aus einem günstigeren Blickwinkel würde hier bei der Steuerung helfen. Dieses letzte Szenario wird dynamisch von einem Benutzer gesteuert. Dieser möchte den Roboter natürlich relativ zügig durch das Gelände bewegen und nicht nach jedem Schritt darauf warten, dass der Computer die Berechnung abgeschlossen hat. Dieser Zeitfaktor soll bei dieser Arbeit berücksichtig werden.

Abgrenzung Auch wenn das System Stereo-Matching benutzt, so soll dies nicht der Kernpunkt der Arbeit sein. Die Integration von räumlichen Informationen, die aus Bildern aus unterschiedlichen Posen gewonnen werden, und die Anpassung eines Weltmodells an diese, soll hauptsächlich behandelt werden.

Es soll kein *SLAM*-System (Simultaneous localization and mapping) entstehen, wo eine (meist auf relativ wenigen Weltdaten beruhenden) Weltrekonstruktion und gleichzeitig auch eine Lokalisierung in dieser Welt stattfindet. Diese Arbeit beschränkt sich auf die Erstellung eines dichten Weltmodells. Es wird keine Lokalisierung durchgeführt, sondern die Kameraposen werden als bekannt vorausgesetzt. Diese könnten aber durch den Einsatz eines dünnen, Merkmalsbasierten SLAM-Systems gewonnen werden.

Dreidimensionale Objekte sollen nicht originalgetreu nachgebildet werden, auch nicht wenn sich diese auf dem Terrain befinden. Tatsächlich wird bei der Terrainrekonstruktion nicht nach Objekten unterschieden. Sie werden als Teil des Terrains betrachtet. Dadurch wird auch angenommen, dass die von den Kameras wahrgenommene Welt statisch ist. Sie verändert sich also nicht mit der Zeit. Die einzige temporale Veränderung ist die Pose der Stereokamera.

Eine Einordnung des Terrains in Klassen wie etwa *Sand*, *Wiese*, *Steinig* etc. soll nicht stattfinden. Es werden keinerlei Annahmen über andere Attribute des Bodens (etwa Bodenhaftung, Festigkeit etc.) getroffen, als solche, die die Form betreffen.

Schließlich soll auch keine konkrete Anwendung realisiert werden (also etwa Implementierung in einem Robotersystem). Als Test- und Evaluierungsdaten kommen synthetisch erzeugte Bilder zum Einsatz. Die Nutzung von synthetischen Daten erlaubt eine genaue Bewertung der Ergebnisse, da eine quantitative Vergleichsgröße (*Ground Truth* genannt) erstellt werden kann.

Beitrag der Arbeit Das Hauptaugenmerk dieser Arbeit soll die Entwicklung einer geeigneten Weltrepräsenttion sein, die in der Lage ist, sich schnell an eine Punktwolke anzupassen. Eine Software wird programmiert, die den "Echtzeit-Einsatz" dieses Systems simuliert. Abbildung 1.1 zeigt den konzeptionellen Programmablauf und Datenfluss. Als Eingabe dient zu jedem Schritt ein Stereobildpaar sowie eine dazu korrespondierende Pose. In einem System in praktischer Nutzung könnte die Pose etwa aus einem dünnen SLAM-Verfahren gewonnen werden. Aus dem Stereobildpaar wird mittels Stereo-Matching und Punktrekonstruktion eine Punktewolke generiert. Diese wird in ein globales Koordinatensystem überführt und in einem Weltmodell registriert. Dieses Weltmodell soll sich nun bestmöglichst an die Punktwolke anpassen.

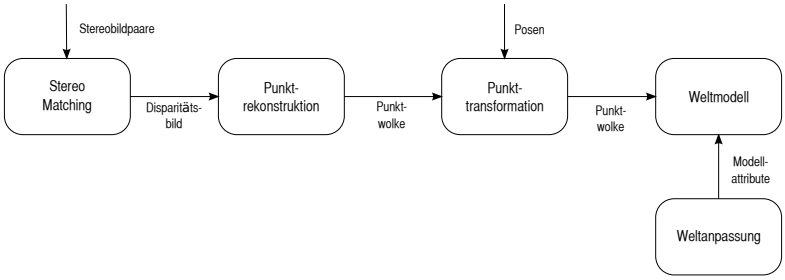


Abb. 1.1.: Schematischer Programmablauf

Das nächste Kapitel soll noch etwas weiter in das Thema einleiten, indem thematisch verwandte Arbeiten vorgestellt werden. Im darauf folgenden Kapitel 3 werden dann die mathematischen und technischen Grundlagen erläutert, die für die Realisierung der Arbeit nötig sind. Kapitel 4 befasst sich intensiv mit dem Entwurf und der Konstruktion des Weltmodells in Form eines regelmäßigen Dreiecksnetzes und den Möglichkeiten dieses an eine Punktwolke anzupassen. Kapitel 5 befasst sich mit dem Entwurf und der Implementierung der Software. In Kapitel 6 wird anhand von Testdaten die Leistungsfähigkeit der Software untersucht und schließlich im Kapitel 7 ein Fazit über die Arbeit gezogen.

2. Stand der Technik

Diese Arbeit beschäftigt sich mit der Rekonstruktion eines Geländes. Bei der Rekonstruktion von Weltdaten wird zwischen *dünnen* (*sparse*) und dichten (*dense*) Verfahren unterschieden. Der Unterschied liegt in der Fülle der Informationen über das Gelände die in die Rekonstruktion einfließen. Werden z.B. nur einzelne Merkmale rekonstruiert, spricht man von dünnen Verfahren. Die resultiert in vereinzelten rekonstruierten Punkten, zwischen denen Bereiche existieren, über die keine Daten vorliegen. In dichten Verfahren sollen möglichst keine Lücken in der Rekonstruktion entstehen. Das Weltmodell, das in dieser Arbeit erzeugt wird, ist ein dichtes.

Im Bereich der dichten Rekonstruktion einer betrachteten Szene hat es in der Vergangenheit viele Fortschritte gegeben. Hierbei werden unterschiedliche Ansätze verfolgt, die im Folgenden kurz vorgestellt werden.

In [Vogiatzis et al., 2008] wurde eine Stereo-Matching-Methode, die auf *Markov Random Fields* (MRF) basiert, generalisiert und auf mehrere Posen ausgeweitet, um Relief-Oberflächen zu rekonstruiert. Als Repräsentationsform wird, statt eines Disparitätsbildes, das an der Kamera ausgerichtet ist, eine Höhenkarte gewählt, die an der Szene ausgerichtet ist.

Eine Höhenkarte wird auch in [Hirschmüller et al., 2005], aus Bildern, die mit einer hochauflösenden Stereokamera erzeugt werden, erstellt. Die Kamera fliegt in relativ großer Distanz über ein Gelände. Der dabei zurückgelegte Bewegungspfad ist nicht genau gradlinig, aber durch den Einsatz eines Lokalisierungssystems bekannt. Die Arbeit beschäftigt sich hauptsächlich mit der Berechnung der Epipolargeometrie zwischen unterschiedlichen Aufnahmen der Kamera, um ein effizientes Stereo-Matching durchführen zu können.

Um Objekte zu rekonstruieren ist in der Regel jedoch eine mächtigere Repräsentationsform nötig. Der in [Turk und Levoy, 1994] verwendete Algorithmus vereint mehrere lokale Entfernungsbilder zu einem gemeinsamen Polygonnetz. Es werden inkrementell aus den einzelnen Entferungsbildern, unabhängig voneinander Polygonnetze erstellt, die unterschiedliche Ansichten eines Objekts widerspiegeln. Diese werden zusammen im Raum angeordnet und miteinander verknüpft, um eine kontinuerliche Oberfläche zu erstellen.

In [Bodenmueller und Hirzinger, 2004] hingegen wird mit nur einem globalen Weltmodell gearbeitet. Das beschriebene Verfahren dient zur Oberflächenrekonstruktion für einen handgeführten Laserscanner. Ein Dreicksnetz wird erzeugt, indem inkrementell 3D-Punkte hinzugefügt werden. Durch den Einsatz eines effizienten Verfahrens zur Mes-

sung von Punktnachbarschaften wird die Dichte der 3D-Punkte limitiert, die Oberflächennormale approximiert und eine lokale Triangulation des Netzes durchgeführt. Die Pose des Scanners wird durch einen externen Manipulator oder ein optisches Tracking-System bestimmt.

Die Vereinigung von Daten, die aus unterschiedlichen Sensoren gewonnen werden, ist ein verwandtes Thema. Zum Beispiel werden mehrere Punktwolken, die ein Roboter über unterschiedliche Sensoren (Tiefenkamera, Laserscanner und Stereokamera) gewinnt, in [Kaustubh Pathak und Schwertfeger, 2007] in einer gemeinsamen, diskreten Belegungskarte vereint.

In [Ohno und Tadokoro, 2005] hingegen werden grobe 3D-Formen verfolgt, die aus Laserscanner-Daten extrahiert werden. Aus ihnen wird die Pose bestimmt und zusammen mit Farbbildern zu einem dichten Modell vereint.

Die Arbeit von [Kagami et al., 2005] ist wieder rein Kamerabasiert. Anhand eines dichten Stereo-Matching-Verfahren werden Punktwolken erstellt. Ein Algorithmus zur Verfolgung von (dünnen) Bildmerkmalen wird genutzt, um die Bewegung eines humanoiden Roboters zu verfolgen und, durch Einsatz des RANSAC Algorithmus zu rekonstruieren. Nun werden die Bildmerkmale und dichten Punktwolken aus dem Stereo-Matching, unter Benutzung der rekonstruierten Bewegung in einer globalen Punktwolke integriert.

In [Lhuillier und Quan, 2005] wird zunächst wieder von wenigen Bildpunkten ausgegangen, die in aufeinanderfolgenden Bildern wiedererkannt werden. Hierbei werden jedoch nur besonders robuste Merkmale verwendet. Ausgehend von diesen *Saatpunkten*, werden in direkter Nachbarschaft weitere potentielle Übereinstimmungen gesucht, wobei lokale Messgrößen zum Einsatz kommen. Die Pixel der so entstanden Disparitätsabbildung werden in kleine quadratische Flächen (*Patches*) umgerechnet, die für die Rekonstruktion genutzt werden.

3. Mathematische und technische Grundlagen

In diesem Kapitel soll der mathematische Hintergrund zur Realisierung der Arbeit geschaffen werden. In Abschnitt 3.1 werden die geometrischen Prinzipien der digitalen Bildgebung erläutert. Aus diesen wird dann, durch Hinzunahme einer zweiten Kamera, in Abschnitt 3.2 der umgekehrte Prozess untersucht. Es soll also vom zweidimiensionalen Bild auf dreidimensionale Weltpunkte geschlossen werden. Es wird sich herausstellen, dass es hierfür nötig ist, zwischen den beiden Kamerabildern Punktkorrespondenzen herzustellen. Verfahren, die dies tun, werden in Abschnitt 3.3 vorgestellt. Schließlich wird in Abschnitt 3.4 noch ein Verfahren zur iterativen Funktionsminimierung vorgestellt, welches für die Anpassung des Weltmodells benutzt wird.

In den Formeln gilt die Konvention, dass Matrizen mit einem normal gedruckten Großbuchstaben bezeichnet werden (M). Vektoren werden mit einem normal gedruckten, kleinen Buchstaben (v) bezeichnet. Punkte werden fett gedruckt, wobei Kleinbuchstaben zweidimensionale und Großbuchstaben dreidimensionale Punkte bezeichnen. Eine Welle über dem Symbol zeigt, dass homogene Koordinaten gemeint sind ($\tilde{\mathbf{x}}$, $\mathbf{X}$).

3.1. Projektive Geometrie

Essenziell für diese Arbeit ist das Verständnis der Zusammenhänge zwischen der betrachteten Szene und ihrer Abbildung, die in Form eines digitalen Bildes vorliegt. Bei der Abbildung einer dreidimensionalen Szene auf eine zweidimensionale Bildfläche spricht man von *Projektion*. In diesem Kapitel sollen die grundlegenden Aspekte der Projektion zunächst anhand des *Lochkameramodells* beschrieben werden. Da es sich bei dem Lochkameramodell um ein idealisiertes Modell handelt, werden im Folgenden noch Verfeinerungen erläutert, um die Projektion mit realen Kameras zu beschreiben.

3.1.1. Kameramodellierung

Das Lochkameramodell Die Lochkamera ist ein idealisiertes Modell zur Beschreibung der grundlegenden Aspekte der Bildgebung von Kameras. Man kann es sich als einen Kasten vorstellen, an dessen Vorderseite sich ein unendlich kleines Loch (die *Lochblende*) befindet. Durch dieses Loch fallen Lichtstrahlen, die von der Szene reflektiert werden. Diese treffen dann auf die Rückwand der Lochkamera, die *Bildebene*, auf der ein Abbild

der Szene entsteht. Eine Illustration dieses Modells ist in Abbildung 3.1 gegeben. Die Lochblende stellt den Ursprung des *Kamerakoordinatensystems* dar, und wird auch das *optische Zentrum* **C** genannt. Die z-Achse, *optische Achse* genannt, steht senkrecht zur Bildebene und schneidet diese im *Bildmittelpunkt* **p**. X- und Y-Achse sind senkrecht zueinander und parallel zur Bildebene. Der Bildmittelpunkt ist der Ursprung der Bildebene, deren Achsen mit U und V bezeichnet sind. Der Abstand der Bildebene zum optischen Zentrum, entlang der optischen, Achse wird *Brennweite* f genannt.

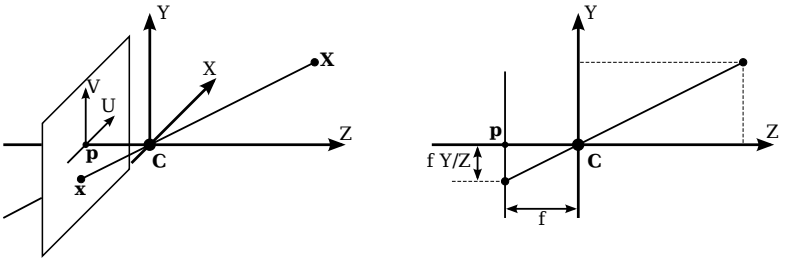


Abb. 3.1.: Das Lochkameramodell. **C** ist das optische Zentrum und der Ursprung des Kamerakoordinatensystems. Der Bildmittelpunkt **p** ist der Schnittpunkt der optischen Achse mit der Bildebene und der Ursprung des Bildkoordinatensystems. Ein Objektpunkt **X** wird auf einen Bildpunkt **x** abgebildet.

Befindet sich jetzt ein Objekt vor der Kamera, wird dessen Abbild erzeugt, indem Lichtstrahlen von einem Objektpunkt $\mathbf{X} = (x, y, z)^{\mathrm{T}}$ durch das optische Zentrum fallen und im Punkt $\mathbf{x} = (u, v)^{\mathrm{T}}$ auf die Bildebene treffen. So entsteht auf der lichtempfindlichen Bildebene ein gedrehtes Bild des Objekts. Hierbei gelten die Relationen $u = -\frac{fx}{z}$ und $v = -\frac{fy}{z}$, was, abgesehen vom Vorzeichen, einer Abbildung vom euklidischen $\mathbb{R}^3$ nach $\mathbb{R}^2$ entspricht [Hartley und Zisserman, 2008, s. 154][Jähne, 2005, s. 204]:

$$(x, y, z)^{\mathrm{T}} \mapsto (fx/z, fy/z)^{\mathrm{T}} \tag{3.1}$$

Projektion mit homogenen Koordinaten Drückt man sowohl Objekt- als auch Bildkoordinaten in homogenen Koordinaten aus, kann man diese Gleichung auch mit

$$\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} fx \\ fy \\ z \end{pmatrix} = \begin{bmatrix} f & & & 0 \\ & f & & 0 \\ & & 1 & 0 \end{bmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \tag{3.2}$$

als Matrixmultiplikation ausdrücken.

Laut Gleichung (3.2) liegt der Ursprung des Bildkoordinatensystems im Bildmittelpunkt. Es ist jedoch üblich, den Bildursprung in der unteren linken Bildecke zu platzieren (üblicherweise wird das um die horizontale Achse gespiegelte Bild betrachtet,

der Ursprung entspricht dann der oberen linken Ecke). Die konstante Verschiebung der Koordinaten wird durch folgende Änderung der Gleichung erreicht:

$$\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} fx + zp_u \\ fy + zp_v \\ z \end{pmatrix} = \begin{bmatrix} f & & p_u & 0 \\ & f & p_v & 0 \\ & & 1 & 0 \end{bmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \tag{3.3}$$

Die Matrix

$$\mathrm{K} = \begin{bmatrix} f & & p_u \\ & f & p_v \\ & & 1 \end{bmatrix} \tag{3.4}$$

wird *Kamerakalibrierungsmatrix* (oder Kameramatrix) genannt.
[Hartley und Zisserman, 2008, s. 155]

Es ist zu beachten, dass die Objektkoordinaten in Gleichung (3.3) im Kamerakoordinatensystem beschrieben werden müssen. Häufig ist es der Fall, dass Objektkoordinaten allerdings in einem statisch in der Szene verankerten *Weltkoordinatensystem* beschrieben sind. Um eine Projektion von Weltkoordinaten auf die Bildebene durchführen zu können, ist ein Zwischenschritt nötig, in dem die Objektkoordinaten in das Kamerakoordinatensystem überführt werden. Dies wird durch eine Rotation R um den Ursprung, sowie eine Translation erreicht. So gelangt man zu

$$\tilde{\mathbf{x}} = \mathrm{KR}\,[\mathrm{I}|-\mathbf{C}]\,\tilde{\mathbf{X}}, \tag{3.5}$$

wobei I die 3×3 Einheitsmatrix ist. $\mathbf{C}$ ist die Position des optischen Zentrums der Kamera. $[\mathrm{I}|-\mathbf{C}]$ bedeutet, dass I um die Spalte $-\mathbf{C}$ erweitert wird.

Der Vollständigkeit halber sei noch ein Faktor erwähnt, der in der Literatur als *skew-Parameter* (Schrägheitsparameter) s beschrieben ist. Mit ihm kann eine Schrägheit der Achsen der Bildebene ausgedrückt werden. Nur in sehr seltenen Sonderfällen ist allerdings $s \neq 0$ [Hartley und Zisserman, 2008, s. 164]. Die Kameramatrix aus Gleichung (3.4) ändert sich unter Berücksichtigung dieses Parameters zu

$$\mathrm{K} = \begin{bmatrix} f & s & p_x \\ & f & p_y \\ & & 1 \end{bmatrix}. \tag{3.6}$$

Digitale Bilder Für die Bildgebung am Computer werden meistens digitale Kameras benutzt. Hier wird kein Fotofilm als Bildebene benutzt, sondern digitale *CCD-Sensoren*. Dies sind, in der Regel matrixförmig angeordnete, lichtempfindliche Bauelemente. Jedes dieser Elemente fängt die Helligkeit für einen diskreten Bildpunkt (*Pixel*) ein, welche

dann digital gespeichert wird [Jähne, 2005, s. 22]. Durch die Größe dieser Bauelemente wird die physische Größe eines Pixels bestimmt. Sie kann in beiden Dimensionen der Bildebene unterschiedlich sein. Der Bildmittelpunkt und die Brennweite können nun mit

$$\mathrm{K} = \begin{bmatrix} \alpha_u & s & u_c \\ & \alpha_v & v_c \\ & & 1 \end{bmatrix} \tag{3.7}$$

in Pixeln ausgedrückt werden, wobei $\alpha_u = f m_u$ und $u_c = m_u p_u$ (analog für v). m_u und m_v sind jeweils die Größe eines Pixels entlang der entsprechenden Achse der Bildebene. [Hartley und Zisserman, 2008, s. 157]

3.1.2. Operationen auf Punkten

Projektion Aus Gleichung (3.5) und (3.7) ergibt sich die *Projektions-Matrix*

$$\mathrm{P} = \mathrm{KR}\left[\mathrm{I}|-\tilde{\mathbf{C}}\right]. \tag{3.8}$$

Mit ihrer Hilfe kann für eine idealisierte CCD-Kamera die Projektion

$$\tilde{\mathbf{x}} = \mathrm{P}\tilde{\mathbf{X}} \tag{3.9}$$

eines Objektpunktes in Weltkoordinaten auf einen Bildpunkt der Bildfläche berechnet werden.

Rückprojektion Der umgedrehte Prozess, also die Rückprojektion eines Punktes auf der Bildebene in seine ursprünglichen Weltkoordinaten, ist nur bedingt möglich. Ein einzelner Bildpunkt kann nicht eindeutig einem Weltpunkt zugeordnet werden, da alle Weltpunkte, die sich auf der Geraden befinden, die sowohl durch das optische Zentrum also auch den Bildpunkt geht, auf denselben Bildpunkt abgebildet werden. Es kann aber diese Gerade bestimmt werden. Hierzu gilt es zwei Punkte, die sich auf der Geraden befinden, zu identifizieren. Als erster Punkt kann einfach das optische Zentrum $\mathbf{C}$ genommen werden. Ein zweiter Punkt der Geraden ergibt sich aus $\mathrm{P}^{+}\tilde{\mathbf{x}}$, wobei P^{+} die Pseudo-Inverse zu P ist. Diese kann mit $\mathrm{P}^{+} = \mathrm{P}^{\mathrm{T}}(\mathrm{P}\mathrm{P}^{\mathrm{T}})^{-1}$ errechnet werden. [Hartley und Zisserman, 2008, s. 590]

Für die Pseudo-Inverse einer Matrix D gilt $\mathrm{D}\mathrm{D}^{+} = \mathrm{I}$. Daraus ergibt sich, dass sich $\mathrm{P}^{+}\mathbf{x}$ auf der gesuchten Gerade befinden, denn $\mathrm{P}(\mathrm{P}^{+}\tilde{\mathbf{x}}) = \mathrm{I}\tilde{\mathbf{x}} = \tilde{\mathbf{x}}$. Es wird also $\mathrm{P}^{+}\tilde{\mathbf{x}}$ auf $\tilde{\mathbf{x}}$ abgebildet, also muss P^{+} folglich auf der gesuchten Geraden liegen.

Die beiden Punkte $\mathrm{P}^{+}\tilde{\mathbf{x}}$ und $\tilde{\mathbf{C}}$ bestimmen die Richtung des Strahls und, mit einem zusätzlichem Parameter λ, lässt er sich mit

$$\tilde{\mathbf{X}}(\lambda) = \mathrm{P}^{+}\tilde{\mathbf{x}} + \lambda\tilde{\mathbf{C}} \tag{3.10}$$

konstruieren. [Hartley und Zisserman, 2008, s. 161f]

3.1.3. Verzerrungsmodellierung

Die Lochkamera ist ein idealisiertes Modell. Kameras im praktischen Gebrauch besitzen ein optisches System aus Sammellinsen, um mehr Licht einzufangen. Die Linsen haben die Aufgabe, parallel einfallende Lichtstrahlen konvergent zu machen und in einem, auf der anderen Seite der Linse liegenden, *Linsenbrennpunkt* zu bündeln. Diese Bündellung bringt aber auch mit sich, dass nur Objektpunkte einer bestimmten Entfernung im Raum scharf auf der Bildebene dargestellt werden. Durch Abblenden der Linse kann dieser Effekt vermindert werden, wobei aber gleichzeitig auch die Menge des einfallenden Lichts reduziert wird. Indem der Abstand der Bildebene zur Linse entlang der optischen Achse modifiziert wird, wird die Linse auf eine bestimmte Entfernung scharfgestellt.[Eichler et al., 1974, s. 155f]

Eine andere Eigenschaft von Linsen ist, dass sie in der Regel eine Verzeichnung (oder Verzerrung) der Abbildung von Objektpunkten auf der Bildebene aufweisen, die mit der Entfernung zum optischen Zentrum wächst. Nun gibt es optische Systeme, die diese durch mehrere hintereinander angebrachte Linsen minimieren. Bei preisgünstigen optischen Systemen jedoch, muss die Verzeichnung für die Bildgebung berücksichtigt werden. [Eichler et al., 1974, s. 153f]

Zur Modellierung der Verzerrung sind zwei Faktoren besonders wichtig. Das sind die *radiale Verzerrung*, die aus der Bauform der Linse resultiert und mit wachsenden Abstand vom optischen Zentrum ebenfalls wächst, und die *tangentiale Verzerrung* die daraus resultiert, dass Linse und CCD-Chip nicht perfekt zueinander ausgerichtet sind. Die Berechnung findet in normalisierten Bildkoordinaten statt, um unabhängig von der Bildgröße zu sein. Ein Bildpunkt $\mathbf{x}$ kann mit

$$\hat{\mathbf{x}} = \frac{\mathbf{x} - \begin{bmatrix} u_c \\ v_c \end{bmatrix}}{\begin{bmatrix} \alpha_u \\ \alpha v \end{bmatrix}} \tag{3.11}$$

normalisiert werden.

Die radiale Verzerrung kann mit Hilfe einer Taylor-Reihe mit dem optischen Zentrum als Entwicklungspunkt angennähert werden. Der Radius $r = \sqrt{\hat{\mathbf{x}}_u^2 + \hat{\mathbf{x}}_v^2}$ misst die Entfernung von diesem Punkt. Für die meisten Kameras genügt eine Reihenentwicklung

von zwei oder drei Termen (k_1, k_2 , k_3). Ein Bildpunkt kann nun mit seiner radialen Verzerrung skaliert werden:

$$\hat{\mathbf{x}}^{(r)} = \hat{\mathbf{x}}(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \tag{3.12}$$

Die tangentiale Verzerrung wird in [Heikkila und Silven, 1997], mit zwei weiteren Parametern p_1 und p_2, durch

$$\hat{\mathbf{x}}^{(t)} = \begin{bmatrix} 2p_1\hat{\mathbf{x}}_u\hat{\mathbf{x}}_v + p_2(r^2 + 2\hat{\mathbf{x}}_u^2) \\ p_1(r^2 + 2\hat{\mathbf{x}}_v^2) + 2p_2\hat{\mathbf{x}}_u\hat{\mathbf{x}}_v \end{bmatrix} \tag{3.13}$$

modelliert.

In der Literatur, die sich intensiver mit optischen Systemen beschäftigt (etwa [Eichler et al., 1974]), werden noch weitere Arten der Verzerrung beschrieben. Die radiale und tangentiale Verzerrung haben jedoch den größten Effekt und werden als ausreichend für die Bildverarbeitung angesehen. Addiert man die beiden Terme aus den Gleichungen (3.12) und (3.13), erhält man ein ausreichend gutes Modell der Verzerrung. Diese, auch *plumb bob*-Modell genannte, Beschreibung der Verzerrung findet ihre Anwendung sowohl in der *Camera Calibration Toolbox for Matlab* [Bouguet, 2010], als auch in der Bildverarbeitungsbibliothek OpenCV[Bradski und Kaehler, 2008, s. 370ff].

Entzerrung von Bildpunkten Der umgekehrte Prozess, die Entzerrung, soll Bildpunkte so verschieben, als sei die Szene mit einer verzerrungsfreien Kamera aufgenommen. Als Resultat werden z.B. gerade Linien aus der Szene auch als gerade Linien im Bild dargestellt. Die Modifikation eines Bildpunktes wird in [Bouguet, 2010] und OpenCV mit folgendem, iterativem Algorithmus approximiert:

$\hat{\mathbf{x}}_0 \leftarrow \hat{\mathbf{x}}$
for $i = 1$ to N **do**
  $r \leftarrow \sqrt{\hat{\mathbf{x}}_u^2 + \hat{\mathbf{x}}_v^2}$
  $k^{(r)} \leftarrow 1 + k_1 r^2 + k_2 r^4 + k_3 r^6$
  $\delta x^{(t)} \leftarrow \begin{bmatrix} 2p_1\hat{\mathbf{x}}_u\hat{\mathbf{x}}_v + p_2(r^2 + 2\hat{\mathbf{x}}_u) \\ p_1(r^2 + 2\hat{\mathbf{x}}_v^2) + 2p_2\hat{\mathbf{x}}_u\hat{\mathbf{x}}_v \end{bmatrix}$
  $\hat{\mathbf{x}} \leftarrow \frac{\hat{\mathbf{x}}_0 - \delta x^{(t)}}{k^{(r)}}$
end for

Da die Verzerrung unabhängig von der Szenengeometrie ist, kann für jeden Bildpunkt so seine entzerrte Position werden. Dies resultiert dann in einer Abbildung für das gesamte Bild und erspart die zeitintensive Entzerrung der Bildpunkte während der Programmlaufzeit.

3.1.4. Kamerakalibrierung

Mit Kamerakalibrierung ist der Prozess gemeint, bei dem die internen geometrischen und optischen Eigenschaften (*innere Parameter*, *intrinsic parameters*), sowie die Platzierung der Kamera im Raum (*externe Parameter*, *extrinsic parameters*), genau bestimmt werden. Für viele Berechnungen des vorherigen Abschnitts war eine Kenntnis der genauen inneren und äußeren Kameraparameter nötig.

Ein verbreitetes Verfahren diese Parameter zu bestimmen, funktioniert unter der Benutzung eines Objekts mit bekannter Größe, mit leicht und präzise optisch identifizierbaren Punkten, deren relative Entfernung voneinander ebenfalls bekannt ist. Hier eignen sich Schachbrettmuster oder Punktmatrizen besonders. Dadurch, dass die geometrischen Eigenschaften des Objekts bekannt sind, kann die relative Pose (R und $\mathbf{C}$ in Gleichung (3.8)) des Objekts zur Kamera aus einem Bild rekonstruiert werden. Die im Bild identifizierten Punkte des Objekts dienen als Messpunkte für eine Fehlerfunktion. Die internen Kameraparameter, also alle Parameter der Kalibrierungsmatrix K (Gleichung (3.7)) und des Verzerrungsmodells (Gleichung (3.12) und (3.13)), werden von der Fehlerfunktion benutzt, um die Projektion des Objekts auf die Bildebene nachzubilden. Diese modellierte Projektion und die Messwerte bilden ein Fehlermaß, nach dem die Kameraparameter optimiert werden können.

In [Heikkila und Silven, 1997] und [Bouguet, 2010] wird dieses Problem als quadratische Fehlerfunktion beschrieben und mit dem Levenberg-Marquardt Algorithmus minimiert. Ist das benutzte Objekt planar, so müssen mehrere unterschiedliche Posen benutzt werden.

3.2. Zwei Kameras

Wie im vorherigem Kapitel beschrieben, kann ein Punkt auf der Bildebene einer Kamera als eine Gerade in den Raum zurückprojiziert werden. Um aber die exakte Position des Weltpunktes beschreiben zu können, ist mindestens eine weitere Ansicht des Punktes nötig. Im Folgenden soll darauf eingegangen werden, was für geometrische Zusammenhänge entstehen, wenn ein zweites Kamerabild hinzugenommen wird. Grundsätzlich ist es dabei egal, ob dieses zweite Kamerabild zeitgleich von einer zweiten Kamera, oder zeitversetzt mit der gleichen Kamera von einer anderen Position aufgenommen wurde (sofern die Szene statisch ist). Ziel dieses Kapitels ist es, die Rekonstruktion einer Szene mit zwei Kameras, die in fester Beziehung zueinander stehen, zu beschreiben. Grundsätzlich gilt es dabei folgende Fragen zu klären:

- Wie ist der geometrische Zusammenhang zwischen der Position eines Bildpunktes $\mathbf{x}$ und dem korrespondieren Punkt $\mathbf{x}'$. Welche Beschränkungen gibt es hier.
- Wie können aus Punktkorrespondenzen von zwei bekannten Kameras Weltpunkte rekonstruiert werden.
- Wie können mit Hilfe von Punktkorrespondenzen die Kameras P und P$'$ rekonstruiert werden (Stereokalibrierung).

3.2.1. Epipolargeometrie

Ein Weltpunkt $\mathbf{X}$ wird in zwei Kameras auf die Bildpunkte $\mathbf{x}$ und $\mathbf{x}'$ abgebildet. Es lässt sich hierbei folgende Beobachtung machen: Die Bildpunkte, Kamerazentren und der Weltpunkt liegen in der Ebene π (siehe Abbildung 3.2). Die Gerade, die die beiden Kamerazentren miteinander verbindet, wird als *Baseline b* bezeichnet. Die Schnittpunkte der Baseline mit den Bildebenen der Kameras sind die Epipole $\mathbf{e}$, $\mathbf{e}'$. Jede Ebene, so auch π, die als Achse die Baseline hat, wird als *Epipolarebene* bezeichnet. Um die Baseline herum entfacht sich also ein Bündel von Epipolarebenen.

Angenommen es gibt einen Punkt $\mathbf{x}$ sowie zwei kalibrierte Kameras mit den Kamerazentren $\mathbf{C}$ und $\mathbf{C}'$. Es soll auf den zu $\mathbf{x}$ korrespondierendem Punkt $\mathbf{x}'$ im anderen Kamerabild geschlossen werden. Durch die Baseline und den Strahl, der durch $\mathbf{x}$ und $\mathbf{C}$ gegeben ist, wird bereits die zu $\mathbf{x}$ gehörige Epipolarebene beschrieben. Es kann mit Sicherheit gesagt werden, dass sich der korrespondierende Bildpunkt auf der Schnittgerade $\mathrm{l}'_{\mathbf{x}}$ der Bildebene der zweiten Kamera mit der Epilpolarebene befindet. $\mathrm{l}'_{\mathbf{x}}$ wird *Epipolarlinie* von $\mathbf{x}$ genannt und kann als Abbildung der Rückprojektion von $\mathbf{x}$ auf die Bildebene der zweiten Kamera gesehen werden. Die Epipolarlinie eines Punktes kann man sich etwa für einen Algorithmus, der nach dem korrespondieren Punkt im Bild der anderen Kamera sucht, zunutze machen. Es ist ein entscheidener Vorteil, dass der Suchraum nun von zwei (gesamte Bildebene) auf eine Dimension, nämlich $\mathrm{l}'_{\mathbf{x}}$, reduziert werden kann. Dieser Zusammenhang ist in Abbildung 3.3 illustriert. Als Anmerkung sei noch erwähnt, dass in dem Fall, dass die beiden Bildebenen der Kameras koplanar

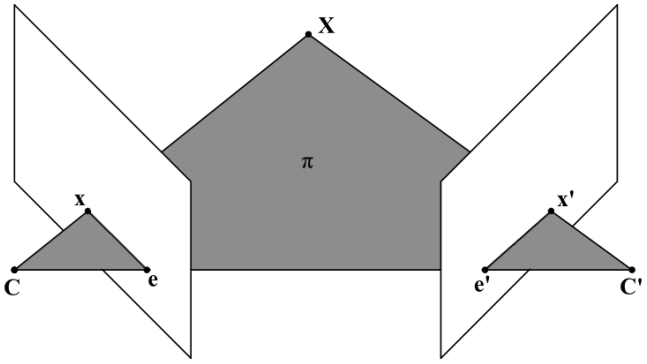


Abb. 3.2.: Zwei korrespondieren Bildpunkte $\mathbf{x}$, $\mathbf{x}'$ eines Bildpaares liegen zusammen mit ihrem Weltpunkt $\mathbf{X}$ und den Kamerazententren $\mathbf{C}$, $\mathbf{C}'$ in der Epipolarebene π. Die Punkte in denen die Baseline die Bildebenen schneidet bilden die Epipole $\mathbf{e}$ und $\mathbf{e}'$. [Hartley und Zisserman, 2008, s. 240]

zueinander angeordnet sind, die Schnittpunkte der Baseline mit den Bildflächen und somit die Epipole, jeweils im unendlichen positioniert sind. Dies resultiert dann in parallel verlaufenden Epipolarlinien. [Hartley und Zisserman, 2008, s. 239ff]

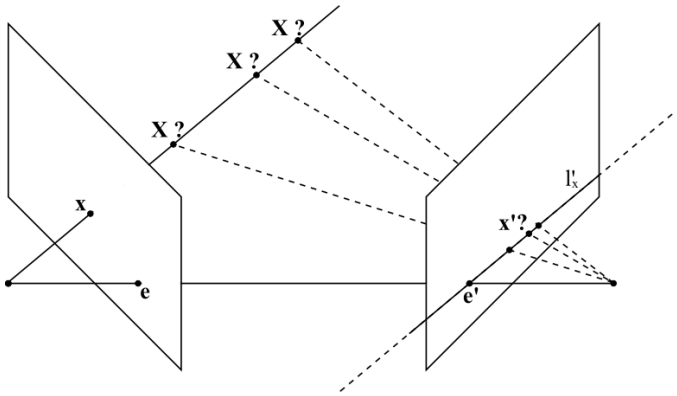


Abb. 3.3.: Die Rückprojektion des Bildpunktes $\mathbf{x}$ zusammen mit der Baseline beschreiben die Epipolarebene von $\mathbf{x}$. Die Schnittgerade dieser Ebene bzw. das Abbild der Rückprojektion von $\mathbf{x}$ beschreibt die Epipolarline $\mathrm{l}'_{\mathbf{x}}$. Auf dieser Linie muss sich der korrespondierende Bildpunkt $\mathbf{x}'$ befinden.

Die Konstruktion der Epipolarline eines Punktes $\mathbf{x}$ kann in zwei Schritten erfolgen:

1. Bestimme einen möglichen Korrespondenzpunkt $\mathbf{x}'$ im zweiten Kamerabild.
2. Verbinde diesen Punkt mit dem Epipolarpunkt $\mathbf{e}'$, um die Epipolarline $\mathrm{l}'_{\mathbf{x}}$ zu erhalten.

Der erste Schritt kann algebraisch durchgeführt werden, indem $\mathbf{x}$ in den Raum zurückprojeziert wird, was, wie in Abschnitt 3.1.2 mit Gleichung (3.10) beschrieben, in der Geraden $\tilde{\mathbf{X}}(\lambda) = \mathrm{P}^{+}\tilde{\mathbf{x}} + \lambda\tilde{\mathbf{C}}$ resultiert. Hieraus kann nun etwa der Punkt mit $\lambda = 0$ bestimmt und auf die Bildfläche der zweiten Kamera projiziert werden: $\tilde{\mathbf{x}}' = \mathrm{P}'\mathrm{P}^{+}\tilde{\mathbf{x}}$. Der Epipol der zweiten Kamera ist das Abbild des Kamerazentrums der ersten Kamera und wird durch

$$\tilde{\mathbf{e}}' = \mathrm{P}'\tilde{\mathbf{C}} \tag{3.14}$$

bestimmt. Die Epipolarline von $\tilde{\mathbf{x}}$ kann nun mit $\mathrm{l}'_{\tilde{\mathbf{x}}} = \tilde{\mathbf{e}}' + \lambda(\tilde{\mathbf{x}}' - \tilde{\mathbf{e}}')$ beschrieben werden.

In [Zhang und Kanade, 1998] und [Hartley und Zisserman, 2008, s. 244] wird noch eine weitere Schreibweise benutzt, in der das Kreuzprodukt mit Hilfe einer *schiefsymmetrische Matrix* (*skew-symmetric*)[Hartley und Zisserman, 2008, s. 581] als Matrixmultiplikation ausgedrückt wird. Hierzu wird $\tilde{\mathbf{e}}'$ in die Form

$$[\tilde{\mathbf{e}}']_{\times} = \begin{bmatrix} 0 & -\tilde{\mathrm{e}}'_3 & \tilde{\mathrm{e}}'_2 \\ \tilde{\mathrm{e}}'_3 & 0 & \tilde{\mathrm{e}}'_1 \\ -\tilde{\mathrm{e}}'_2 & \tilde{\mathrm{e}}'_1 & 0 \end{bmatrix} \tag{3.15}$$

gebracht. Die Epipolarline von $\tilde{\mathbf{x}}$ kann hiermit als $\mathrm{l}'_{\tilde{\mathbf{x}}} = [\tilde{\mathbf{e}}']_{\times}(\mathrm{P}'\mathrm{P}^{+})\tilde{\mathbf{x}} = \mathrm{F}\tilde{\mathbf{x}}$ geschrieben werden. Die 3×3 Matrix F wird als *Fundamentalmatrix*

$$\mathrm{F} = [\tilde{\mathbf{e}}']_{\times}\mathrm{P}'\mathrm{P}^{+} \tag{3.16}$$

bezeichnet. Mit ihr kann eine Abbildung

$$\mathrm{l}'_{\mathrm{x}} = \mathrm{F}\tilde{\mathbf{x}} \tag{3.17}$$

von einem beliebigen Bildpunkt $\mathbf{x}$ auf seine Epipolarline l'_{x} auf der Bildebene der zweiten Kamera erzeugt werden. Auf dieser muss der zu $\mathbf{x}$ korrespondierende Punkt $\mathbf{x}'$ liegen. Sie erfüllt die *Korrespondenzbedingung* (*correspondence condition*)

$$\tilde{\mathbf{x}}'^{\mathrm{T}}\mathrm{F}\tilde{\mathbf{x}} = 0 \tag{3.18}$$

für alle korrespondierenden Punkte $\tilde{\mathbf{x}} \leftrightarrow \tilde{\mathbf{x}}'$. [Zhang und Kanade, 1998]
[Hartley und Zisserman, 2008, s. 541ff]

Falls die Kamerazentren nur durch eine horizontale Translation voneinander verschoben sind, wie es etwa bei rektifizierten Kameras der Fall ist (siehe Absatz *Rektifizierung* im folgenden Unterabschnitt), so ist es hilfreich die Epipole auf den unendlichen Punkt $e = e' = [1, 0, 0]^{\mathrm{T}}$ zu setzen. Es ergibt sich $[\tilde{\mathbf{e}}]_{\times} = [\tilde{\mathbf{e}}']_{\times} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} = \mathrm{F}$ und Gleichung (3.17) bleibt gültig.

3.2.2. Stereokalibrierung

Im vorherigen Abschnitt wurden die geometrischen Beziehungen zwischen zwei Kameras betrachtet. Die Stereokalibrierung ist der Prozess um diese zu identifizieren. Es geht also darum, die Pose der rechten Kamera relativ zu der linken herausfinden, welche durch eine Translation und eine Rotation beschrieben wird.

Aus der Kalibrierung der einzelnen Kameras ergeben die *externen Parameter* den Translationsvektor (t_l bzw. t_r) und die Rotationsmatrix (R_l bzw. R_r), relativ zum betrachteten Kalibrierungsobjekt. Wird dasselbe Objekt für beide Kameras benutzt, lässt sich ein Punkt $\mathbf{X}$ dieses Objekts, relativ zur linken bzw. rechten Kamera, mit

$$\mathbf{X}_l = R_l\mathbf{X} + t_l \tag{3.19}$$

$$\mathbf{X}_r = R_r\mathbf{X} + t_r \tag{3.20}$$

beschreiben. Ein Punkt kann zwischen den Koordinatensystemen der einzelnen Kameras konvertiert werden:

$$\mathbf{X}_l = R^T(\mathbf{X}_r - t) \tag{3.21}$$

Diese Beziehung ist der Schlüssel zur Stereokalibrierung. Mit ihr lässt sich so aus den Kalibrierungen der einzelnen Kameras R und t bestimmen:

$$R = R_r R_l^T \tag{3.22}$$

$$t = t_r - Rt_l \tag{3.23}$$

Durch Rauschen und Rundungsfehler in den Kalibrierungsdaten ergeben sich allerdings leicht unterschiedliche R und t für jeden der Messpunkte. Um nun die optimalen Werte zu finden, werden in [Bradski und Kaehler, 2008, s. 428] die Punkte des Kalibrierungsobjektes von einer Kamera jeweils auf die Bildebene der anderen Kamera projiziert. Der Abstand zwischen den Messdaten und dieser Projektion dient als Fehlermaß für ein iteratives Funktionsminimierungsverfahren, mit dem die besten Parameterwerte für R und t gesucht werden.

Rektifizierung Im Abschnitt 3.3 werden Stereo-Matching-Verfahren gezeigt, die eine spezielle Konfiguration der beiden Kameras vorraussetzen. Um diese Algorithmen effizienter zu gestalten, werden Kameras vorausgesetzt, deren Bildebenen sich gemeinsam in einer Ebene befinden. Außerdem sollen sie in dieser Ebene so ausgerichtet sein, dass die Bildzeilen kolinear liegen. Die Rektifizierung ist der Vorgang eine Transformation der Kameras zu finden, damit diese Bedingungen erfüllt sind. Außerdem werden noch Bildtransformationen erstellt, mit denen die Kamerabilder entsprechend modifiziert und gleichzeitig entzerrt werden (*rectification map*). Das Verfahren nach Bouguet, das dies, ausgehend von der relativen Rotation und Translation zweier Kameras (aus Stereokalibrierung) und dessen Verzerrungskoeffizieten (aus Kamerakalibrierung), macht, ist in [Bouguet, 2010] und OpenCV implementiert.

3.2.3. Strukturrekonstruktion

Grundsätzlich gilt es bei der Strukturrekonstruktion aus einem Stereobildpaar Bildpunkte in 3D-Weltpunkte zu überführen. Hierzu werden zwei korrespondierende Bildpunkte der beiden Kameras, als Strahlen zurück in den Raum projiziert (Gleichung (3.10)). Diese werden miteinander zu geschnitten, der Schnittpunkt ist dann der gesuchte Weltpunkt. Eine Schwierigkeit stellt hier dar, dass die gefundenen Korrespondenzen in der Regel nicht so exakt sind, dass sich die beiden Strahlen tatsächlich im Raum schneiden. Es ist also zuerst ein Zwischenschritt nötig, um die Korrespondenzen so zu korrigieren, dass sie die Epipolarbedingung erfüllen. Anschließend können dann die Strahlen der modifizierten korrespondierenden Punkte geschnitten werden [Hartley und Zisserman, 2008, s. 310ff].

Bei rektifizierten Bilder gilt ein Sonderfall der Strukturrekonstruktion. Hier ist $v' = v$ und es wird ein Disparitätsbild $\mathbf{D}$ mit den Werten $[u, v, d]^{\mathrm{T}}$ erzeugt, wobei d ein eindimensionaler Wert ist. Es ist $d = u' - u$ jeweils der Versatz der korrespondierender Bildpunkte entlang der horizontalen Bildachse und somit entlang der Epipolarline. Die Epipolarbedingung wird also als erfüllt vorausgesetzt. Zur Vereinfachung können die Parameter der beiden Kameras angeglichen werden, so dass $alpha_u = alpha_v = alpha'_u = alpha'_v$ und $u_c = u'_c$, $v_c = v'_c$ sind. Eine Rotation zwischen den beiden Kameras gibt es nicht und die Translation beschränkt sich auf eine Verschiebung entlang der x-Achse um die Baseline b.

In dieser Konstellation kann ein Punkt mit

$$\tilde{\mathbf{X}} = \begin{bmatrix} \frac{(u-u_c)b}{d} \\ \frac{(v-v_c)b}{d} \\ \frac{\alpha b}{d} \end{bmatrix} \tag{3.24}$$

rekonstruiert werden.

In Matrixform kann dies mit einer perspektivischen Projektionsmatrix

$$\mathrm{Q} = \begin{bmatrix} 1 & 0 & 0 & -u_c \\ 0 & 1 & 0 & -v_c \\ 0 & 0 & 0 & \alpha \\ 0 & 0 & 1/b & (u_c - u'_c)/b \end{bmatrix} \tag{3.25}$$

als

$$\tilde{\mathbf{X}} = \mathrm{Q}\tilde{\mathbf{x}} = \begin{bmatrix} u - u_c \\ v - v_c \\ \alpha \\ \frac{d}{b} \end{bmatrix} = \begin{bmatrix} \frac{(u-u_c)b}{d} \\ \frac{(v-v_c)b}{d} \\ \frac{\alpha b}{d} \\ 1 \end{bmatrix} \tag{3.26}$$

ausgedrückt werden. [Bradski und Kaehler, 2008, s. 435]

3.3. Stereo-Matching

Beim Stereo-Matching geht es darum, zwischen zwei Kamerabildern Pixelkorrespondenzen zu finden. Ergebnis dieser Verfahren sind meistens *Disparitätsabbildungen* (disparity maps). Das sind Funktionen $d(x, y)$, die die Pixelkorrespondenzen mit Bezug auf ein Referenzbild beschreiben:

$$x' = x + sd(x, y),\ y' = y, \tag{3.27}$$

wobei $s = \pm 1$ so gewählt wird, dass die Disparitätswerte stets positiv sind.
[Scharstein und Szeliski, 2002]

Es wird zwischen dünnen und dichten Verfahren unterschieden. Bei dünnen Verfahren werden Merkmale aus beiden Kamerabildern extrahiert. Diese werden anschließend miteinander, anhand einer Ähnlichkeitsmessung, verglichen. Unter Berücksichtigung der Epipolarbedingung, werden die Korrespondenzen ermittelt.

Bei dichten Verfahren hingegen soll möglichst für jeden Pixel der Eingabebilder eine Korrespondenz hergestellt werden. Dies resultiert in deutlich mehr geometrischen Informationen über die Szene. Merkmalsbasierte Verfahren sind weniger geeignet, da in der Regel nicht genügend Merkmale gefunden werden können, aus denen möglichst fehlerfrei Korrespondenzen gebildet werden können. In der Regel wird mit Bildern rektifizierter Kameras gearbeitet, da hier die Einschränkung des Suchraums auf nur eine Dimension die Algorithmen vereinfachen und deutlich effizienter gestalten.

Scharstein und Szeliski haben in [Scharstein und Szeliski, 2002] eine Taxonomie für dichte Stereo-Korrespondez-Algorithmen erstellt. Diese soll im Folgenden vorgestellt werden.

Typischerweise bestehen Stereo-Matching-Algorithmen aus Teilen der folgenden Schritte:

- Berechnung der Übereinstimmungskosten (matching cost)
- Kostenaggregation
- Disparitätsberechnung
- Disparitätsverfeinerung

Maß für die Berechnung der Übereinstimmungskosten kann der pixelbasierte quadratische oder absolute Grauwert- oder Gradientenunterschied sein. Der gradientenbasierte Ansatz hat den Vorteil, dass dieser nicht anfällig für kamerabedingte Unterschiede in der Bildhelligkeit ist. Derselbe Effekt kann jedoch auch für grauwertbasierte Berechnung erzielt werden, indem in einem Vorverarbeitungsschritt eine Normalisierung der Grauwerte der Eingabebilder durchgeführt wird. Es ist auch möglich nicht einfach die Pixelwerte zu vergleichen, sondern stattdessen das Referenzbild mit einer interpolierenden Funktion des anderen Bildes. Hieraus ergibt sich ein höhere Auflösung des Disparitätsraumes. Durch Errechnung der Korrespondenzkosten für alle Koordinaten- und Disparitätswerte ergibt sich der Disparitätsraum $C(x, y, d)$.

Die Kostenaggregation kommt hauptsächlich bei lokalen und fensterbasierten Verfahren zum Einsatz. Die Kosten werden durch Aufsummieren oder Mittellung der Kosten aus der Kostenberechnung, über einen Bereich im Disparitätsraum gebildet.

Für lokale Verfahren ist es sehr einfach anschließend die Disparitätsberechnung durchzuführen. Es werden die Disparitätswerte gewählt, deren Kosten minimal in der Kostenaggregation sind. Bei globalen Verfahren findet häufig keine Kostenaggregation statt, dafür ist die Disparitätsberechnung komplexer. Ziel ist es eine Disparitätsfunktion d zu finden, die den globalen Fehler

$$E(d) = E_{data}(d) + \lambda E_{smooth}(d) \tag{3.28}$$

minimiert. E_{data} misst die Übereinstimmung der Disparitätsfunktion mit den Eingabebildern und E_{smooth} formuliert eine, dem Algorithmus zu Grunde liegende, Glattheitsbedingung, die mit λ gewichtet wird.

Eine weitere Möglichkeit ist, ein globales Minimum für einzelne Scan-Linien (scanlines) zu finden. Durch die Unterteilung des Problems in voneinander unabhängige Teilprobleme, kann ein deutlicher Geschwindigkeitsgewinn erzielt werden. Allerdings ist es hier schwierig, eine Konsistenz herzustellen, die Scan-Linien-übergreifend ist.

Viele Stereo-Matching-Algorithmen arbeiten im diskreten Pixelraum. Es ist allerdings auch möglich, die Disparität zu verfeinern und etwa als eine kontinuierliche Funktion zu beschreiben, die an die gefundenen Disparitätswerte angepasst wird.

Durch Nachbearbeitung können fehlerhafte Korrespondenzen enternt oder korrigiert werden. Möglichkeiten hierfür sind z.B. ein Vergleich zweier Disparitätsbilder, die aus demselben Stereobildpaar erstellt wurden, oder durch Benutzung eines Median Filters [Scharstein und Szeliski, 2002].

In den folgenden Unterabschnitten werden zwei konkrete Algorithmen zum Stereo-Matching genauer vorgestellt, die in der Implementierung Verwendung finden. Anschließend werden noch weitere Algorithmen kurz vorgestellt, die aber für diese Arbeit nicht geeignet sind. Eine Begründung hierfür wird in der Evaluierung der Stereo Algorithmen in Kapitel 6 gegeben.

3.3.1. Block Matching

Diese Methode arbeitet mit kleinen Fenstern, in denen die absolute Differenzsumme (sum of absolute difference, SAD) der Pixelwerte gebildet wird, um Pixelkorrespondenzen zwischen dem linken und rechten Kamerabild herzustellen. Die Kamerabilder müssen entzerrt und rektifiziert sein. Es können nur in stark texturierten Bereichen Treffer gefunden werden. Eine Verbesserung wird noch durch eine Vor- und Nachverarbeitung der Bilder bzw. der Disparitätswerte erzielt. Der Algorithmus lässt sich in folgende Schritte unterteilen:

- Vorverarbeitung, um Helligkeitsunterschiede in den Bildern auszugleichen.

- Korrespondenzsuche mit verschiebbaren SAD Fenster entlang der horizontalen Bildachse.
- Nachbearbeitung, um fehlerhafte Korrespondenzen zu eliminieren.

Zur Vorverarbeitung wird eine rechteckige Filtermaske mit einstellbarer Größe über das Bild geschoben, wobei der Wert I_c des Pixels, der unter der Mitte der Maske liegt, mit $min[max(I_c - \overline{I}, -I_{cap}), I_{cap}]$ ersetzt wird, wobei $\overline{I}$ der Mittelwert innerhalb der Maske und I_{cap} ein positiver Schwellwert ist.

Beim Matching wird für jeden Pixel ein Fenster in einem angegebenem Disparitätsbereich (definiert durch Werte für die minimale und die maximale Disparität) über das Bild geschoben, das den SAD-Wert errechnet. Für jede Disparität und jeden Pixel wird der SAD-Wert gespeichert. Mathematisch ausgedrückt sieht die Berechnung des SAD-Wertes wie folgt aus:

$$SAD_{u,v} = \sum_{i=u-R_H}^{u+R_H} \sum_{j=v-R_V}^{v+R_V} \|Left_{i,j} - Right_{i-d,j}\|, \tag{3.29}$$

wobei R_H und R_V den Radius des Fensters in horizontaler und vertikaler Richtung angeben. Es wird anschließend die Disparität mit der kleinsten Differenz zur Definition der Korrespondenz gewählt.

In der Nachbearbeitung geht es darum, falsche Korrespondenzen herauszufiltern. Hierzu kommen drei Methoden zum Einsatz:

- Es gilt die Annahme, dass eine korrekte Zuordnung von korrespondierenden Bildpunkten lokal einen deutlich besseren Matching-Wert hat als umliegende Bildpunkte. Es werden die Matches herausgefiltert, bei denen die benachbarten Disparitäten ebenfalls hohe Matchingwerte liefern. Gesteuert wird dies über einen einstellbaren Schwellwert.
- Bereiche, die zu geringe Texturen aufweisen, so dass zufälliges Bildrauschen sich stark auf das Matching auswirken könnte, werden mit einem Schwellwert herausgefiltert.
- Fensterbasierte Verfahren haben Schwierigkeiten bei Objekten in der Szene, da innerhalb eines Fensters dann Vorder- und Hintergrund gleichzeitig vertreten sind. Als Resultat davon entstehen lokale Bereiche mit großen und kleinen Disparitäten. Diese Randbereiche werden ebenfalls mit einem Fenster detektiert, wobei ein Bereich für erlaubte Disparitätsabweichungen angegeben werden kann.

[Bradski und Kaehler, 2008, s. 439ff]

3.3.2. GPU-Stereo Matcher

Dieser Algorithmus ist dem im vorherigen Unterabschnitt beschriebenen vom mathematischen Prinzip her sehr ähnlich. Die Besonderheit ist jedoch, dass dieser auf dem 3D-Grafikprozessor, kurz *GPU* implementiert ist und ein anderes Verfahren zur Nachverarbeitung benutzt wird.

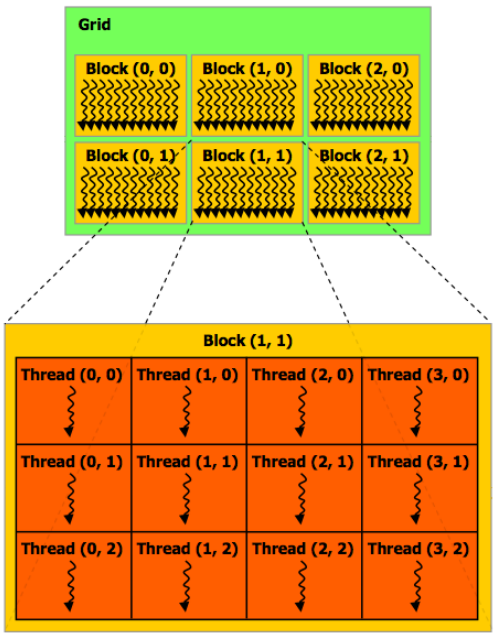


Abb. 3.4.: Threadhierachie für die Ausführung von Kernels in CUDA. [NVIDIA, 2010]

Der Algorithmus und große Teile des Codes stammen aus dem *OpenVIDIA*-Projekt[1]. Er kann zusammen mit der Dokumentation aus deren Online-Repository[2] heruntergeladen werden.

Im Gegensatz zum OpenCV-Algorithmus findet hier keine Vorverarbeitung statt. Als Nachverarbeitung wurde der Algorithmus von [Hildebrandt, 2010] um eine Cross Correlation-Überprüfung ergänzt. Das Matching selbst ist ebenfalls ein Block Matching-Verfahren. Hier wird jedoch nicht die absolute Differenzsumme gebildet, sondern die

[1] OpenVIDIA: `http://openvidia.sourceforge.net/index.php/OpenVIDIA`
[2] OpenVIDIA Stereo Repository: `http://sourceforge.net/projects/openvidia/files/CUDAStereoCamera/`

quadratische, deren Berechnung auf der GPU dieselbe Zeit beansprucht wie die absolute Summe. [Stam, 2008]

$$SSD_{u,v} = \sum_{i=x-R_H}^{x+R_H} \sum_{j=y-R_V}^{y+R_V} (Left_{i,j} - Right_{i-d,j})^2 \tag{3.30}$$

Eine Implementierung auf der GPU macht jedoch nur Sinn, wenn sich der Algorithmus gut parallelisieren lässt. Es soll nun der Aufbau des Algorithmus erläutert werden.

CUDA setzt eine Aufteilung des Problems in Blöcke von Threads vorraus, wie es in Abbildung 3.4 illustriert ist. Das Bild wird in Blöcke geteilt, in denen jede Pixelzeile einem Thread entspricht. Innerhalb der Threads wird die Summe der quadratischen Differenz zwischen dem linken und dem rechten Bild aufaddiert. Der benutzte Pixel im rechten Bild wird hierbei stets um die momentane Disparität aus einem definierten Disparitätsbereich versetzt. Sind alle Threads des Blocks mit der Berechnung in ihrer Spalte fertig, werden die SSD-Werte der benachbarten Spalten in Fenstergröße aufaddiert.

Nachdem die SSD-Werte für die erste Pixelreihe des Thread-Blocks auf diese Weise berechnet wurden, können für die folgenden Pixelreihen die bereits berechneten Werte wiederbenutzt werden. Bei der folgenden Pixelreihe ist das SSD-Fenster um eine Zeile nach unten verschoben. Es genügt also die erste Pixelreihe des 'alten' SSD-Fensters zu subtrahieren und den der 'neuen' Pixelreihe zu addieren. Dies bringt besonders bei großen Blöcken einen Geschwindigkeitsgewinn.

Dieser Prozess wird für jede Disparität in der definierten Disparitätsspanne (minimale Disparität $\leq d \leq$ maximale Disparität) wiederholt. Zwischen diesen Werten müssen nicht unbedingt ganzzahlige Disparitätsschritte gemacht werden. Wird eine Schrittweite < 1 gewählt, so werden die Grauwerte zwischen den Pixeln interpoliert.

An jedem Disparitätsschritt wird der SSD-Wert der aktuellen Disparität mit einem globalen (Thread-übergreifenden) Speicher in Bildgröße verglichen. Sollte der aktuelle Wert kleiner sein, wird dieser in den globalen Speicher geschrieben. Der Disparitätswert mit derzeit minimalem SSD-Wert für jeden Pixel, wird ebenso in einen globalen Speicher geschrieben [Stam, 2008].

Um fehlerhafte Matches zu entfernen, wird der Algorithmus ein zweites Mal ausgeführt. Diesmal allerdings mit dem rechten als Referenzbild. Die Disparitätswerte müssen dementsprechend invertiert iteriert werden. Ist dies abgeschlossen, werden die Disparitäten für jeden Pixel aus dem ersten und dem zweiten Durchlauf miteinander verglichen. Ist der Unterschied größer als ein Schwellwert, wird die Disparität für diesen Pixel gelöscht.

3.3.3. Weitere Algorithmen

Zusätzlich zu den beiden oben beschriebenen Algorithmen wurden weitere Algorithmen untersucht:

- Ein in OpenCV enthaltener, GraphCuts-basierter Algorithmus, der in [Kolmogorov und Zabih, 2002] beschrieben ist. Dieser globale Algorithmus ist sehr präzise, allerdings bei großen Bildern und großem Disparitätsbereich extrem langsam.
- Ein Dynamic Programming-basierter Algorithmus, der ebenfalls in OpenCV enthalten ist. Es handelt sich um eine leicht abgewandelte Version des Algorithmus, der in [Birchfield und Tomasi, 1999] vorgestellt wurde. Hier werden zunächst Korrespondenzen für einzelne Bildzeilen unabhängig voneinander gesucht. Dies resultiert in vertikal sehr inhomogenen Ergebnissen. Durch eine Nachbearbeitung werden diese noch verbessert, dennoch sind die resultierenden Disparitätsbilder noch mit Schlieren und Blöcken durchzogen.
- Den *StereoMatcher* aus [Scharstein und Szeliski, 2002], der eine Art Baukasten zum Zusammenstellen von Stereo-Algorithmen darstellt.
- Einen auf SURF-Features [Bay et al., 2008] basierten Algorithmus. Aus dem rechten und linken Bild werden SURF-Features extrahiert. Korrespondenzen zwischen den Merkmalen werden mit der *k-nächste Nachbarn* Suche der FLANN-Bibliothek (Fast Library for Approximate Nearest Neighbors)[3] hergestellt. Schließlich werden die gefundenen Matches noch einmal nach Erfüllung der Epipolarbedingung gefiltert. Dieses Verfahren liefert keine dichten Disparitätsbilder.

3.4. Minimierung nicht-linearer Funktionen mit einer Variable

Später, in Kapitel 4.2, werden Methoden vorgestellt, mit denen das Weltmodell an die Punktwolke angepasst werden soll. Grundsätzlich handelt es sich um Funktionen, die unterschiedliche Maße nutzen um den geometrischen Abstand des Netzes von der Punktwolke zu messen. Es gilt dann, diesen Abstand zu minimieren. Für einige dieser Funktionen gibt es keine analytische Lösung, so dass ein iteratives Verfahren zur Funktionsminimierung benutzt wird.

Dem eingesetzten Verfahren liegt die Überlegung zu Grunde, dass ausreichend glatte Funktionen in der Nähe ihres Minimums eine annähernd parabolische Form haben. Wie in Abbildung 3.5 illustriert, wird durch einen Bereich, der von zwei Grenzwerten (1) und (3) eingeschlossen ist, eine Parabel durch einen dritten Punkt (2) gelegt. (2) ist die bisherige Schätzung für den minimalen Funktionsparameter. Am Minimum der Parabel wird die Funktion (durchgezogene Linie) ausgewertet, was den Punkt (4) ergibt. Eine neue Parabel wird durch die Punkte (1), (4) und (2) gelegt. Das Minimum dieser Parabel, der Punkt (5), ist bereits sehr nahe am Minimum der Funktion.

Nun lässt sich aber nicht jede Funktion ausreichend gut durch eine Parabel approximieren. Der Algorithmus soll aber in jedem Fall ein Minimum liefern. Um dies zu gewährleisten, verwendet die eingesetzte Methode nach Brent [Press et al., 2007, s. 496ff]

[3] FLANN: `http://www.cs.ubc.ca/~mariusm/index.php/FLANN/FLANN`

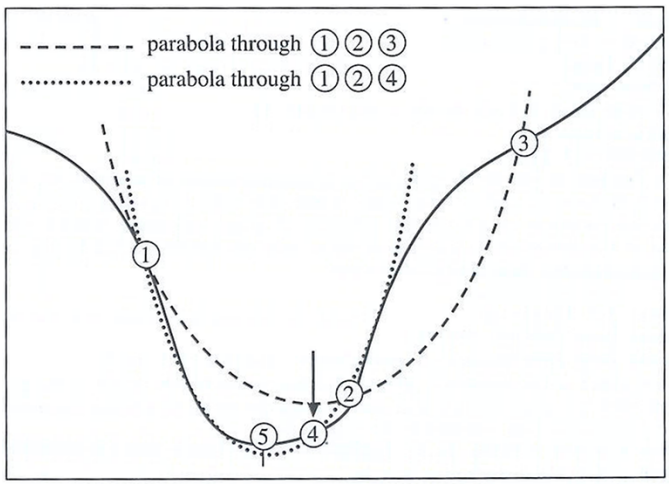


Abb. 3.5.: Inverse parabolische Interpolation, um eine Funktion zu minimieren. Durch einen Bereich, eingeschlossen durch (1) und (3), wird mit einem dritten Punkt (2) ein Parabel (gestrichelte Linie) gelegt. Das Minimum dieser definiert ein neues Intervall $(1), (4), (2)$ durch das erneut eine Parabel (gepunktete Linie) gelegt wird. Deren Minimum führt bereits nach wenigen Schritten zu einem Punkt nahe dem Minimum der Funktion (durchgezogene Linie).[Press et al., 2007, s. 497]

zwei unterschiedliche Ansätze, die je nach Funktionsform gewählt werden. Das sind die oben genannte inverse parabolische Interpolation und außerdem eine langsame, dafür aber sichere Methode wie z.B. *Golden Section Search* [Press et al., 2007, s. 492ff]. Hierzu muss die zu minimierende Funktion analysiert werden um zu entscheiden, nach welchem Verfahren vorgegangen werden soll. Außerdem muss sichergestellt werden, dass die Minimierungsmethode nahe des Funktionsminimums gut mit Rundungsfehlern umgeht und dass, beim Umschalten zwischen den Verfahren, unnötige Rechenoperationen vermieden werden. Das Verfahren nach Brent weist diese Merkmale auf, so dass es sich als sicheres, trotzdem aber im günstigen Fall sehr schnelles Verfahren zur iterativen Minimierung von einparametrigen Funktionen eignet. [Press et al., 2007, s. 496]

4. Weltmodellierung und Anpassung

Die Welt soll aus unterschiedlichen Posen betrachtet und aus den daraus gewonnenen Informationen rekonstruiert werden. Hierzu genügt es nicht jeweils die lokal gewonnenen Geländeinformationen der einzelnen Posen (aus Stereo-Matching) unabhängig voneinander zu betrachten, sondern sie werden zusammen in ein globales Weltmodell integriert.

Als Repräsentationsform kommt ein regelmäßiges Dreiecksnetz zum Einsatz, welches aus einer Menge von Ankern (oder Knoten) und Facetten (oder Kacheln) besteht. Je drei Anker beschreiben die Eckpunkte einer dreiecksförmigen Facette. Die Anordnung der Ankerpunkte und deren Zuordnung zu den Facetten beschreiben somit die Form des Modells. Es gelten hierbei folgende Einschränkungen für das regelmäßige Dreiecksnetz (vergleiche Abbildung 4.1):

- Im Ursprungszustand liegen alle Ankerpunkte in einer Ebene, der *Weltmodellebene*. Die Vektoren u und v repräsentieren die Achsen, die diese aufspannen, und der Punkt **O** deren Ursprung.
- Die Verteilung der Anker entlang jeder Achse ist regelmäßig. So ist der Abstand zwischen den Ankern entlang der Modellachsen ebenfalls regelmäßig.
- Je drei benachbarte Anker bilden eine Facette.
- Nur die Höhe der Ankerpunkte ist variabel. Ihre (u, v)-Koordinaten bleiben konstant.

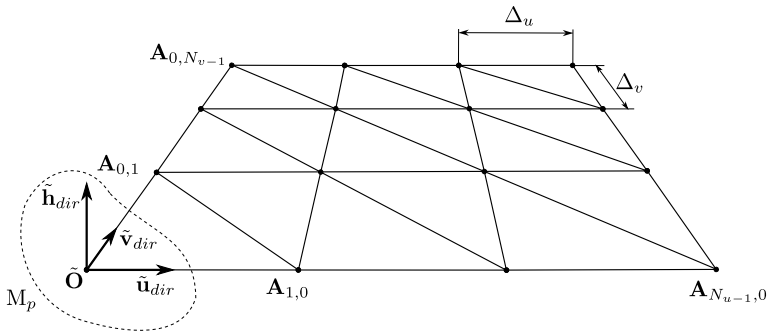


Abb. 4.1.: Die Vektoren u und v spannen die Weltmodellebene auf. Deren Pose wird durch M_p beschrieben. Die Anker sind regelmäßig im durch Δ_u und Δ_v beschriebenen Abstand über das Netz verteilt.

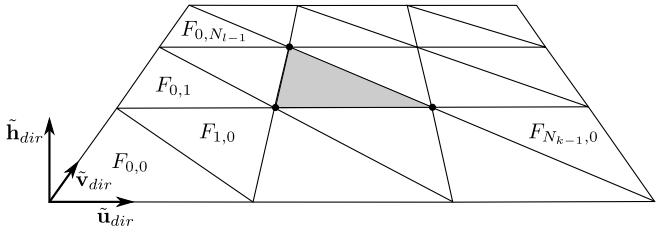


Abb. 4.2.: Eine Facette besteht aus je drei Ankern. In Richtung der u-Achse werden zwischen zwei Ankern jeweils zwei Facetten gezählt. In v-Richtung wird nur eine gezählt.

Das regelmäßige Dreiecksnetz ist in seiner Ausdrucksstärke beschränkt. Es können keine beliebigen 3D-Strukturen dargestellt werden, da es auf einen Höhenwert pro Punkt der Weltebene limitiert ist. Zwischen den Ankerpunkten wird die Höhe des Netzes linear interpoliert. Diese Art der Repräsentation wird in der Literatur häufig als 2,5 Dimensional bezeichnet. Das ist zwar eine starke Einschränkung, die aber wegen zwei interessanten Eigenschaften in Kauf genommen wird. Das regelmäßige Dreiecksnetz erlaubt eine sehr effiziente, analytische Herangehensweise bei der Anpassung an eine Punktwolke (siehe Abschnitt 4.2.1). Dies ist auf die lokale Auswirkung eines Punktes auf das Modell zurückzuführen. Hierdurch ist es wiederum möglich, dass sich Punkte über einfache Indizierung schnell im Modell registrieren lassen (siehe Abschnitt 4.1.1).

In den folgenden Abschnitten soll nun zunächst die Konstruktion eines solchen regelmäßigen Dreiecksnetz, die Registrierung und Konvertierung von Weltpunkten, sowie die Modifikation des Netzes nach der Konstruktion gezeigt werden. Anschließend werden unterschiedliche Arten zum Anpassen des Netzes an eine Punktwolke erarbeitet. Am Ende wird dann noch eine Bedingung die besagt, dass das Gelände insgesamt einen homogenen Eindruck machen soll, vorgestellt und in die Geländeanpassung integriert.

4.1. Erstellung des Dreiecksnetzes

Zur Erstellung werden als Parameter der Weltmodellursprung **O**, die Achsenvektoren u und v, deren Länge auch die Größe des Weltmodells angeben, sowie die Anzahl der Anker entlang der Achsen N_u und N_v, benötigt. Die Pose der Weltmodellebene im Raum wird aus den Einheitsvektoren der Modellachsen u_{dir} und v_{dir}, dem hierzu orthogonal stehendem Vektor $\mathrm{h}_{dir} = \mathrm{u}_{dir} \times \mathrm{v}_{dir}$ sowie dem Ursprungspunkt O, jeweils in homogenen Koordinaten, konstruiert:

$$\mathrm{M}_P = \left[\begin{array}{cccc} \tilde{\mathrm{u}}_{dir} & \tilde{\mathrm{v}}_{dir} & \tilde{\mathrm{h}}_{dir} & \tilde{\mathbf{O}} \end{array} \right]. \tag{4.1}$$

Der Abstand zwischen den Ankerpunkten ergibt sich aus

$$\Delta_u = \frac{\|\mathrm{u}\|}{N_u - 1} \tag{4.2}$$

$$\Delta_v = \frac{\|\mathrm{v}\|}{N_v - 1}. \tag{4.3}$$

Abbildung 4.1 illustiert die Verteilung der Anker.

Die Positionen der Ankerpunkte in der Weltmodellebene werden mit

$${}^{w}\mathbf{A}^{0}_{i,j} = \mathbf{O} + (i\Delta_u)\mathrm{u}_{dir} + (j\Delta_v)\mathrm{v}_{dir}, \quad \begin{matrix} 0 \leq i < N_u \\ 0 \leq j < N_v \end{matrix} \tag{4.4}$$

in Weltkoordinaten und mit

$${}^{m}\mathbf{A}_{i,j} = \begin{bmatrix} i\Delta_u \\ j\Delta_v \\ 0 \\ 1 \end{bmatrix} \quad \begin{matrix} 0 \leq i < N_u \\ 0 \leq j < N_v \end{matrix} \tag{4.5}$$

in Weltmodellkoordinaten, beschrieben.

Schließlich werden die Facetten erstellt. In je einer Reihe von Ankern werden in u-Richtung $N_k = 2(N_u - 1)$ und in v-Richtung $N_l = N_v - 1$ Facetten erstellt (vgl. Abbildung 4.2). Nun müssen die Facetten noch mit je drei Ankern verbunden werden.

4.1.1. Operationen

Konvertierung von Punkten Punkte können mit

$${}^{w}\tilde{\mathbf{X}} = \mathrm{M_P}\ {}^{m}\tilde{\mathbf{X}} \tag{4.6}$$

von Weltmodellkoordinaten in Weltkoordinaten konvertiert werden. Im umgekehrter Richtung kann die Konvertierung entsprechend mit

$${}^{m}\tilde{\mathbf{X}} = \mathrm{M_P^{-1}}\ {}^{w}\tilde{\mathbf{X}} \tag{4.7}$$

erfolgen.

Registrierung von Punkten In Abschnitt 4.2 wird gezeigt werden, dass ein Punkt ${}^{m}\mathbf{X} = [u, v, h]^{\mathrm{T}}$ sich nur lokal direkt auf das Netz auswirkt, nämlich auf die Facette über, unter oder in der der Punkt liegt. Bei der Registrierung eines Punktes im Netz

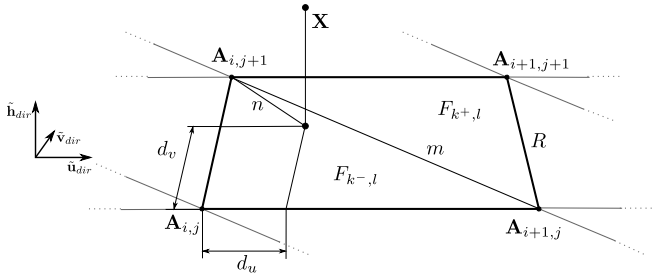


Abb. 4.3.: Ein Punkt $\mathbf{X}$ kann über seine u- und v-Koordinaten direkt einer Facette im regelmäßigem Dreiecksnetz zugeordnet werden. Er muss in einer der beiden benachbarten Facetten (F_{k^-}, F_{k^+}, l) liegen, die durch die Anker $\mathbf{A}_{i,j}, \mathbf{A}_{i+1,j}, \mathbf{A}_{i,j+1}$ und $\mathbf{A}_{i+1,j+1}$ gebildet werden. Die beiden Facetten bilden das durch die Gerade m geschnittene Rechteck R. Die relativen Distanzen innerhalb des Rechtecks d_u und d_v, normiert zur Rechtecksbreite und -höhe, werden in Gleichung (4.11) benutzt, um zwischen den beiden Facetten zu entscheiden. Hierbei werden die Steigungen der Geraden m und n verglichen.

wird ermittelt, welche Facette $F_{k,l}$ dies ist. Der Punkt wird dieser dann zugeordnet, wobei die Regelmäßigkeit des Netzes hier hilfreich ist.

Die Reihe der Kachel l ergibt sich einfach aus

$$l = \lfloor \frac{v}{\Delta v} \rfloor. \tag{4.8}$$

Die gesuchte Index k innerhalb einer Reihe kann zunächst auf $k^- = 2\lfloor \frac{u}{\Delta u} \rfloor$ und $k^+ = 2\lceil \frac{u}{\Delta u} \rceil$ eingegrenzt werden. Die beiden Kacheln $F_{k^-,l}$ und $F_{k^+,l}$ bilden ein Rechteck R. Eine Diagonale m, zwischen den beiden Ankern, die sich die beiden Kacheln teilen, schneidet dieses (Abbildung 4.3). Der zur Rechtecksbreite normierte Abstand, relativ zum linken Rand von R ist

$$d'_u = \frac{u}{\Delta u} - \lfloor \frac{u}{\Delta u} \rfloor. \tag{4.9}$$

Analog dazu ist

$$d'_v = \frac{v}{\Delta v} - \lfloor \frac{v}{\Delta v} \rfloor. \tag{4.10}$$

Es wird die Gerade n gebildet, die vom Anker $\mathbf{A}_{i,j+1}$ durch dem Punkt geht, in dem $\mathbf{X}$ die Weltmodellebene in h-Richtung schneidet. Bei deren Steigung werden die folgenden Fälle unterschieden:

$$\frac{d'_u}{1-d'_v} \begin{cases} < 1: & \text{Punkt ist in } F_{k^-,l} \\ = 1: & \text{Punkt ist auf } m. \text{ Zählt zu } F_{k^+,l} \\ > 1: & \text{Punkt ist in } F_{k^+,l} \\ = 0: & \text{Rand } (\mathbf{A}_{i,j+1} - \mathbf{A}_{i,j}) \text{ zählt zu } F_{k^-,l} \\ \text{Undef.}: & \text{Rand } (\mathbf{A}_{i+1,j+1} - \mathbf{A}_{i+1,j}) \text{ zählt zu } F_{k^+,l} \end{cases} \tag{4.11}$$

Es gelten folgenden Beziehungen:

$$d'_u + d'_v < 1 \rightarrow \frac{d'_u}{1-d'_v} < 1$$
$$d'_u + d'_v = 1 \rightarrow \frac{d'_u}{1-d'_v} = 1$$
$$d'_u + d'_v > 1 \rightarrow \frac{d'_u}{1-d'_v} > 1.$$

Statt Gleichung (4.11) kann also auch

$$k = 2\lfloor \frac{u}{\Delta u} \rfloor + d'_u + d'_v \tag{4.12}$$

benutzt werden.

Modifikation des Netzes Ist das Netz erstellt, ist die einzige Möglichkeit zur Veränderung die Variation der Höhe h der Anker. Dies bringt, in Weltmodellkoordinaten, lediglich eine Änderung seiner h-Komponente mit sich. In Weltkoordinaten errechnet sich die neue Position durch

$$^w\mathbf{A} = {}^m\mathbf{A}^0 + \mathrm{h}_{dir} \cdot h. \tag{4.13}$$

Eine Modifikation der Höhe eines Ankers bedeutet ebenfalls eine Modifikation der Normalen der anliegenden Kacheln. Die Normale einer einzelnen Kachel errechnet sich aus dem Kreuzprodukt zweier Vektoren, gespannt aus ihren Ankern. Beim Bilden der Normalen, ist auf die richtige Orientierung zu achten. Haben alle Anker die Höhe Null, so entsprechen die Normalen aller Kacheln h_{dir}.

4.2. Anpassung des Weltmodells

Im vorherigen Abschnitt wurde erläutert wie das Weltmodell erstellt und verändert werden kann. Nun soll darauf eingegangen werden, welche Veränderungen der Ankerhöhen vorgenommen werden müssen, um das Modell bestmöglich an eine Punktwolke anzupassen.

Das Weltmodell kann als eine partiell lineare Funktion, mit den Ankerhöhen als Parametetervektor $\mathrm{p} = [h_{0,0}, \cdots, h_{N_u-1,N_v-1}]$, betrachtet werden. Zu einem Koordinatenpaar

(u, v) gibt diese die Höhe des Models an dieser Stelle zurück. Grundsätzlich geht es darum, eine Funktion f zu definieren, die den Abstand zwischen dem Modell und allen Punkten $\mathbf{X}_c$ eines Punktvektors $\mathrm{q} = [\mathbf{X}_0, \cdots, \mathbf{X}_{N_\mathbf{X}-1}]$ misst. Es soll nun eine Parametereinstellung $\hat{\mathrm{p}}$ gefunden werden, die die Summe der Abstände minimiert. Hier kommen unterschiedliche Fehlermessungen zum Einsatz, die teilweise unterschiedliche Herangehensweisen zur Optimierung voraussetzen. Namentlich sind dies die Messung des Fehlers in vertikaler Richtung zur Weltmodellebene (im Folgenden *z-Richtung* genannt) oder orthogonal zur Ebene der Kachel.

Weiter wird zwischen quadratischer und absoluter Abstandsmessung unterschieden. Mathematisch ausgedrückt gilt es demnach, die folgenden Probleme zu lösen:

$$\arg\min_{\mathrm{p}} e_{z^2} = \sum_{c=0}^{N_\mathbf{X}-1} \|f_z(\mathrm{p}; [u_c, v_c]) - h_c\|^2 \tag{4.14}$$

$$\arg\min_{\mathrm{p}} e_{o^2} = \sum_{c=0}^{N_\mathbf{X}-1} \|f_o(\mathrm{p}; \mathbf{X}_c) - \mathbf{X}_c\|^2 \tag{4.15}$$

$$\arg\min_{\mathrm{p}} e_{\|z\|} = \sum_{c=0}^{N_\mathbf{X}-1} \|f_z(\mathrm{p}; [u_c, v_c]) - h_c\| \tag{4.16}$$

$$\arg\min_{\mathrm{p}} e_{\|o\|} = \sum_{c=0}^{N_\mathbf{X}-1} \|f_o(\mathrm{p}; \mathbf{X}_c) - \mathbf{X}_c\| \tag{4.17}$$

Es stellt sich die quadratische Abstandmessung in z-Richtung als besonders interessant heraus, da hier bei variierendem p, der Abstand zu den Punkten linear wächst. Dies ermöglicht eine analytische Lösung, während bei den anderen Verfahren auf iterative Methoden zurückgegriffen wird.

Im Folgenden werden die oben genannten Methoden diskutiert. Anschließend wird noch ein weiteres Fehlermaß eingeführt, das die Form des Weltmodells beschränkt. Diesem Maß liegt der Vermutung zu Grunde, dass das Gelände insgesamt einen eher glatten Eindruck macht und keine impulsartigen Sprünge aufweist. Schließlich wird noch eine heuristische Anpassungsmethode gezeigt, die zwar keine optimalen Ergebnisse liefert, aber sehr schnell und unabhängig von der Punktanzahl arbeitet.

4.2.1. Abstand in Z-Richtung

Der quadratische Abstand in z-Richtung ist ein, in der Literatur gerne benutztes Maß, zur Approximation linearer Modelle an eine Reihe von Messgrößen. Denn soll dieser minimiert werden, so lässt sich das Problem als lineares Gleichungssystem der Form

$$\mathrm{e} = \|\mathrm{Ax} - \mathrm{b}\|^2 \tag{4.18}$$

formulieren, wobei b ein Vektor mit verrauschten Messdaten ist. Ax liefert, unter Benutzung des Parametervektors x, die zu b passenden Modellwerte. Die Differenz der Modellwerte und der Messdaten ergibt dann den Fehlervektor e. Gleichungen dieser Art haben die Eigenschaft, dass das Problem

$$\arg\min_{\mathrm{x}} e = \|\mathrm{Ax} - \mathrm{b}\|^2, \tag{4.19}$$

also dass der optimale Parametervektor $\hat{\mathrm{x}}$, geometrisch, algebraisch oder analytisch bestimmt werden kann. [Strang, 2003, s. 217]

Die angesprochene z-Richtung entspricht in Weltmodellkoordinaten der Richtung der h-Achse. Alle Berechnungen werden in Weltmodellkoordinaten durchgeführt. In diesem Abschnitt werden z- und h- Achse synonym benutzt.

Zur Übersicht wird hier zunächst die Problemdefinition aus Gleichung (4.14) wiederholt:

$$\arg\min_{\mathrm{p}} \sum_{c=0}^{N_{\mathbf{X}}-1} \|f_z(\mathrm{p}; [u_c, v_c]) - h_c\|^2$$

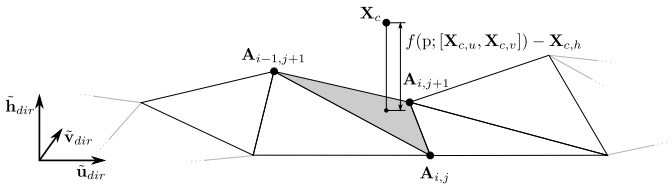


Abb. 4.4.: Von einem Punkt $\mathbf{X}_c$ wird der Abstand zum Weltmodell mit der Funktion f_z, in Richtung der z-Achse gemessen. Die Werte der Abstandsmessung sind lediglich von den drei Ankern der Facette, der der Punkt zugeordnet ist, abhängig.

In Abbildung 4.4 wird die Abstandsmessung für einen Punkt anhand einer Reihe von Facetten illustriert. Es wird ausschließlich die h-Komponente in Weltmodellkoordinaten (bzw. z-Komponente) gemessen. Die Abstandmessung geschied also senkrecht zur Weltmodellebene. Zwischen den Ankern ist das Weltmodell linear. Hierdurch ergibt sich, dass lediglich die Anker der Kachel, die die (u, v)-Koordinaten von $\mathbf{X}_c$ einschließen, für die Berechnung von Belang sind. Alle anderen Anker haben keinen Einfluß.

Modelliert werden kann dies durch

$$f_z(\mathrm{p}; [u, v]) = \sum_{i=0}^{N_{\mathrm{p}}-1} \mathrm{p}_i \phi_i([u, v]), \tag{4.20}$$

eine lineare, von p abhängige Funktion, die zwischen den Ankern interpoliert. Hierbei ist ϕ_i die Basisfunktion, für die Gewichtung der Ankerhöhe p_i. Die Gewichtung ist abhängig von einem (u, v)-Koordinatenpaar. Die Basisfunktion hat folgende Eigenschaften:

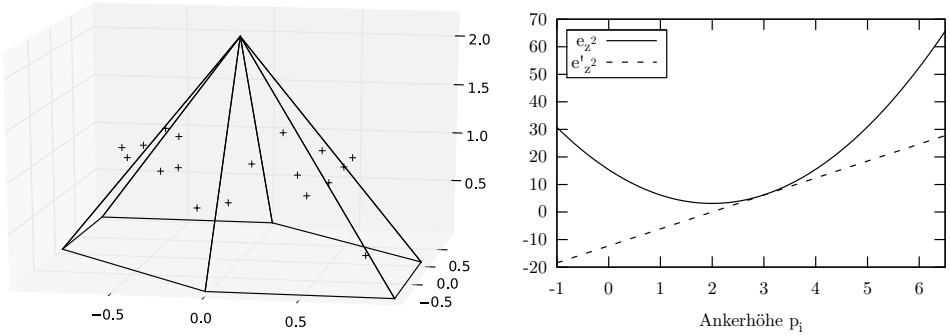
- Liegt (u, v) innerhalb der Kachel F, so ist für alle Anker der Kachel $\phi(u, v) > 0$.
- Die Summe der Gewichtungen dieser Anker ist 1. Hierbei wächst $\phi(u, v)$, je näher (u, v) den Koordinaten eines der Anker ist.
- Entspricht (u, v) den Koordinaten eines Ankers, so ist für diesen Anker $\phi(u, v) = 1$.
- Liegt (u, v) außerhalb der Facette F, so ist für alle Anker dieser Facette $\phi(u, v) = 0$.

Gleichung (4.20) ausgewertet, errechnet nun die Höhe des Weltmodells an der Stelle u_c, v_c):

$$f_z(\mathrm{p}; [\mathbf{U}_c, v_c]) = 0 + f_a h_{i,j} + 0 + f_b h_{i-1,j+1} + f_c h_{i-1,j+1} + 0, \tag{4.21}$$

wobei f_a,f_b und f_c jeweils die Gewichtungen aus $\phi(u_c, v_c)$ für die jeweiligen Anker sind.

In Abbildung 4.5 wird die Entwicklung des Fehlers eines kleinen Netzes gezeigt, wenn die Höhe eines Ankers variiert wird. Man kann erkennen, dass sich der Fehler mit sich ändernder Ankerhöhe linear verändert.



(a) (b)

Abb. 4.5.: Ein kleines Netz bestehend aus 7 Ankern und 6 Facetten, mit 3 Punkten pro Facette (a). (b) zeigt einen Plot des quadratischen Fehlers in z-Richtung dieses Netzes, über die Höhe des Ankers in der Netzmitte. Die gestrichelte Linie zeigt die Ableitung des Fehlers zur Ankerhöhe.

Sei D eine Matrix der Größe $N_{\mathbf{X}} \times N_{\mathrm{p}}$ mit

$$\mathrm{D}_{c,i} = \phi_i([u_c, v_c]), \tag{4.22}$$

so kann die Problemdefinition (4.14) unter Benutzung von Gleichung (4.20) als

$$\min_{\mathrm{p}} e_{z^2} = \|\mathrm{Dp} - \mathrm{h}\|^2 \tag{4.23}$$

ausgedrückt werden. h ist hier ein Vektor der h-Komponenten aller Messpunkte aus $\mathrm{q} = [\mathbf{X}_0, \cdots, \mathbf{X}_{N_{\mathbf{X}}-1}]$. Diese Darstellung entspricht nun genau der aus Gleichung (4.19). Es handelt sich also um dieselbe Problemklasse.

Sukzessive, analytische Lösung Die Gleichungen (4.20) und (4.21) zeigen eine lokale Beziehung zwischen Weltmodell und Punkten. Ein Punkt erzeugt einen Wert für den globalen Fehler e_{z^2}. Dieser errechnet sich jedoch nur in Abhängigkeit von den drei Ankern *einer* Facette. Eine sinnvolle Modifikation dieser drei Anker minimiert also einerseits den lokalen Fehler dieser Facette, wirkt sich aber gleichwertig auch auf den globalen Fehler aus. Gleichung (4.23) zeigt, dass das Problem als lineares *Least-Squares*-Problem zu verstehen ist. Dies führt zu folgenden Erkenntnissen:

- Eine analytische Lösung ist möglich.
- Die Minimierung des Fehlers kann in lokale Teilprobleme zerlegt werden.
- Punkte werden, wie bereits in Abschnitt 4.1.1 beschrieben, direkt einer Facette zugeordnet. Dies erspart bei der Anpassung das Auswerten der ϕ_i-Funktionen.

Im Folgenden wird eine analytische Lösung des Problems vorgestellt, bei der sukzessiv der Fehler einzelner Anker, d.h. der Fehler der Summe, der an diesem Anker anliegenden Facetten, minimiert wird.

Der Abstand eines Punktes $\mathbf{X} = [x, y, z]$ zu einer Ebene in z-Richtung kann mit

$$d = Ax + By - z + D \tag{4.24}$$

errechnet werden. Für eine Punktwolke $\mathrm{q} = [\mathbf{X}_0, \cdots, \mathbf{X}_{N_\mathbf{X}}]$, die ausschließlich den Facetten F_j (mit $0 \leq j < N$) des Ankers $\mathbf{A}$ zugeordnet ist, wird die Summe der quadratischen Abstände der Punkte mit

$$e_{z^2}(\mathbf{A}_z) = \sum_{j=0}^{N} \sum_{c=0}^{N_{\mathbf{X}j}} (A_j x_c + B_j y_c + D_j - z_c)^2 \tag{4.25}$$

errechnet. $\mathbf{A}$ soll nun in seiner Höhe so verändert werden, dass $e_z(\mathbf{A}_z)$ minimiert wird. Für Gleichung (4.25) bedeutet das, dass für jede Facette A_j, B_j und D_j gefunden werden müssen, so dass der Abstand minimal wird. Dabei dürfen die Bedingungen an das regelmäßige Dreiecksnetz nicht verletzt wird.

Für jede Facette gilt, dass ihre drei Ankerpunkte $\mathbf{A}_{j0}$, $\mathbf{A}_{j1}$ und $\mathbf{A}_{j2}$ in der durch A_j, B_j und D_j beschriebenen Ebene liegen müssen. Diese Anker in Gleichung (4.24) eingesetzt gibt:

$$\begin{aligned}
\mathbf{A}_{j0x} A_j + \mathbf{A}_{j0y} B_j - \mathbf{A}_{j0z} + D_j &= 0 \\
\mathbf{A}_{j1x} A_j + \mathbf{A}_{j1y} B_j - \mathbf{A}_{j1z} + D_j &= 0 \\
\mathbf{A}_{j2x} A_j + \mathbf{A}_{j2y} B_j - \mathbf{A}_{j2z} + D_j &= 0,
\end{aligned}$$

was aufgelöst nach A_j, B_j und D_j in

$$A_j = -\frac{(\mathbf{A}_{j1y} - \mathbf{A}_{j0y})\ \mathbf{A}_{j2z} + (\mathbf{A}_{j0z} - \mathbf{A}_{j1z})\ \mathbf{A}_{j2y} + \mathbf{A}_{j0y}\,\mathbf{A}_{j1z} - \mathbf{A}_{j0z}\,\mathbf{A}_{j1y}}{(\mathbf{A}_{j1x} - \mathbf{A}_{j0x})\ \mathbf{A}_{j2y} + (\mathbf{A}_{j0y} - \mathbf{A}_{j1y})\ \mathbf{A}_{j2x} + \mathbf{A}_{j0x}\,\mathbf{A}_{j1y} - \mathbf{A}_{j0y}\,\mathbf{A}_{j1x}} \tag{4.26}$$

$$B_j = \frac{(\mathbf{A}_{j1x} - \mathbf{A}_{j0x})\ \mathbf{A}_{j2z} + (\mathbf{A}_{j0z} - \mathbf{A}_{j1z})\ \mathbf{A}_{j2x} + \mathbf{A}_{j0x}\,\mathbf{A}_{j1z} - \mathbf{A}_{j0z}\,\mathbf{A}_{j1x}}{(\mathbf{A}_{j1x} - \mathbf{A}_{j0x})\ \mathbf{A}_{j2y} + (\mathbf{A}_{j0y} - \mathbf{A}_{j1y})\ \mathbf{A}_{j2x} + \mathbf{A}_{j0x}\,\mathbf{A}_{j1y} - \mathbf{A}_{j0y}\,\mathbf{A}_{j1x}} \tag{4.27}$$

$$\begin{aligned} D_j = &\frac{(\mathbf{A}_{j0x}\,\mathbf{A}_{j1y} - \mathbf{A}_{j0y}\,\mathbf{A}_{j1x})\ \mathbf{A}_{j2z} + (\mathbf{A}_{j0z}\,\mathbf{A}_{j1x} - \mathbf{A}_{j0x}\,\mathbf{A}_{j1z})\ \mathbf{A}_{j2y}}{(\mathbf{A}_{j1x} - \mathbf{A}_{j0x})\ \mathbf{A}_{j2y} + (\mathbf{A}_{j0y} - \mathbf{A}_{j1y})\ \mathbf{A}_{j2x} + \mathbf{A}_{j0x}\,\mathbf{A}_{j1y} - \mathbf{A}_{j0y}\,\mathbf{A}_{j1x}} \\ &+\frac{(\mathbf{A}_{j0y}\,\mathbf{A}_{j1z} - \mathbf{A}_{j0z}\,\mathbf{A}_{j1y})\ \mathbf{A}_{2x}}{(\mathbf{A}_{j1x} - \mathbf{A}_{j0x})\ \mathbf{A}_{j2y} + (\mathbf{A}_{j0y} - \mathbf{A}_{j1y})\ \mathbf{A}_{j2x} + \mathbf{A}_{j0x}\,\mathbf{A}_{j1y} - \mathbf{A}_{j0y}\,\mathbf{A}_{j1x}} \end{aligned} \tag{4.28}$$

resultiert. Angenommen $\mathbf{A}_0$ ist der zu modifizierende Anker. Die komponentenweise Ableitung von Gleichung (4.25) nach seiner Höhe $\mathbf{A}_{0z}$ führt nun zu

$$e'_{z^2}(\mathbf{A}_z) = \sum_{j=0}^{N_{F\mathbf{A}}} \sum_{c=0}^{N_{\mathbf{X}j}} (A'_j x_c + B'_j y_c + D'_j)(A_j x_c + B_j y_c + D_j - z_c), \tag{4.29}$$

mit

$$\begin{aligned} A'_j &= \frac{(\mathbf{A}_{j2y} - \mathbf{A}_{j1y})\ x_c}{\mathbf{A}_{j0x}\ (\mathbf{A}_{j2y} - \mathbf{A}_{j1y}) - \mathbf{A}_{j1x}\,\mathbf{A}_{j2y} + \mathbf{A}_{j1y}\,\mathbf{A}_{j2x} + \mathbf{A}_{j0y}\ (\mathbf{A}_{j1x} - \mathbf{A}_{j2x})} \\ B'_j &= \frac{(\mathbf{A}_{j1x} - \mathbf{A}_{j2x})\ y_c}{\mathbf{A}_{j0x}\ (\mathbf{A}_{j2y} - \mathbf{A}_{j1y}) - \mathbf{A}_{j1x}\,\mathbf{A}_{j2y} + \mathbf{A}_{j1y}\,\mathbf{A}_{j2x} + \mathbf{A}_{j0y}\ (\mathbf{A}_{j1x} - \mathbf{A}_{j2x})} \\ D'_j &= \frac{\mathbf{A}_{j1y}\,\mathbf{A}_{j2x} - \mathbf{A}_{j1x}\,\mathbf{A}_{j2y}}{\mathbf{A}_{j0x}\ (\mathbf{A}_{j2y} - \mathbf{A}_{j1y}) - \mathbf{A}_{j1x}\,\mathbf{A}_{j2y} + \mathbf{A}_{j1y}\,\mathbf{A}_{j2x} + \mathbf{A}_{j0y}\ (\mathbf{A}_{j1x} - \mathbf{A}_{j2x})}. \end{aligned}$$

Da sich $e'_{z^2}(\mathbf{A})$ linear zur Ankerhöhe verhält, lässt sicht die optimale Höhe analytisch durch

$$\begin{aligned} d_0 &= \sum e'_{z^2}(0) \\ d_1 &= \sum e'_{z^2}(1) \\ \Delta d &= d_1 - d_0 \\ A_{0z} &= d_0/\Delta d \end{aligned} \tag{4.30}$$

ermitteln. Dies wird nun für jeden Anker wiederholt. Nach mehreren Iterationen konvergiert der globale Fehler zu seinem Minimum.

4.2.2. Orthogonaler Abstand

Eine andere Möglichkeit den Abstand eines Punktes zur Ebene zu messen stellt die Messung in Richtung der Ebenennormale, also orthogonal zur Ebene, dar. Abbildung

4.6 illustiert dieses Prinzip anhand einer Reihe von Facetten. Diese Messmethode scheint intuitiver, birgt aber Probleme in sich.

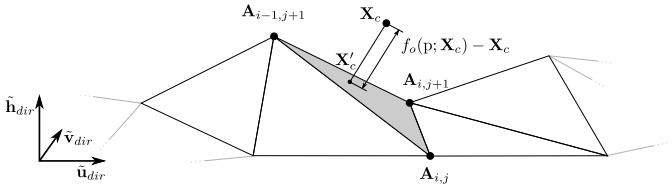


Abb. 4.6.: Von einem Punkt $\mathbf{X}_c$ wird der Abstand zu einem Punkt $\mathbf{X}'_c$ auf dem Weltmodell gemessen, der mit der Funktion f_o ermittelt wurde. Der Vektor $(\mathbf{X}_c - \mathbf{X}'_c)$ entspricht der Richtung der Facettennormale. Diese ist durch die drei Anker der Facette definiert.

Zur Übersicht noch einmal die Problemdefinition aus Gleichung (4.31):

$$\arg\min_{\mathrm{p}} e_{o^2} = \sum_{c=0}^{N_{\mathbf{X}}-1} \|f_o(\mathrm{p};\mathbf{X}_c) - \mathbf{X}_c\|^2 \tag{4.31}$$

Hier gilt es also eine Funktion f_o zu finden, die abhängig von der Parametrisierung des Weltmodells und des Punkts $\mathbf{X}_c$, den Punkt $\mathbf{X}'_c$ zurückgibt. Dieser ist der Schnittpunkt des Strahls, der von $\mathbf{X}_c$ orthogonal auf die Weltebene trifft. Es wird der Vektor $(\mathbf{X}'_c - \mathbf{X}_c)$ ermittelt, dessen Länge der gesuchte Abstand ist.

Dieses Problem ähnelt der Suche nach der orthogonalen Regressionsebene einer Punktwolke. Jedoch gilt es hier, nicht eine Ebene mit unendlicher Ausdehung zu finden, sondern eine Vielzahl räumlich berenzter Dreiecke. Abbildung 4.7 illustriert zwei Sonderfälle die hier auftreten können. Da die Facetten räumlich begrenzt sind, passiert es in (a), dass kein Strahl, konstruiert werden kann, der, in Richtung der Facettennormale, den Punkt schneidet. In (b) hingegen, wird die Entfernung eines einzelnen Punktes zu mehreren Facetten gleichzeitig gemessen. Es gibt also keine eindeutige Zuordnung.

Für diese Sonderfälle wird das Problem umdefiniert. Es soll nun nicht mehr der orthogonale Abstand zur Weltebene gemessen werden, sondern es wird wieder jeder Punkt, über seine (u, v)-Koordinaten, eindeutig einer Facette zugeordnet. Der orthogonale Abstand des Punktes zur Ebene, in der die Facette liegt, ist das Fehlermaß. Die Zuordnung der Facette ist in Abbildung 4.7 als gestrichelte Linie, und die Ebene der Facette für den Fall (a) ist in grau eingezeichnet. Ein wichtiger Punkt dieser Herangehensweise ist, dass hierdurch wieder die Auswirkung eines Punktes auf einen lokalen Bereich des Netzes sichergestellt wird.

Das Problem kann mit diesen Änderungen mathematisch als

$$\arg\min_{\mathrm{p}} e_{o^2} = \sum_{i=0}^{N_F-1} \sum_{c=0}^{N_{\mathbf{X}_i}-1} \|\mathrm{X}_{ic} \cdot \phi_i(\mathrm{p}) + \psi_i(\mathrm{p})\|^2 \tag{4.32}$$

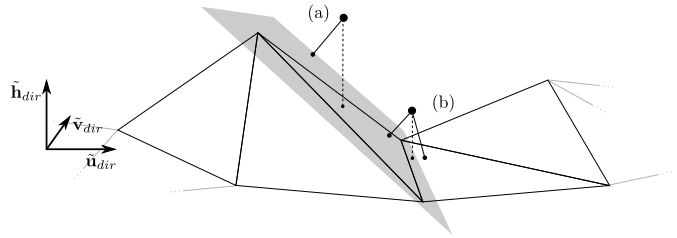
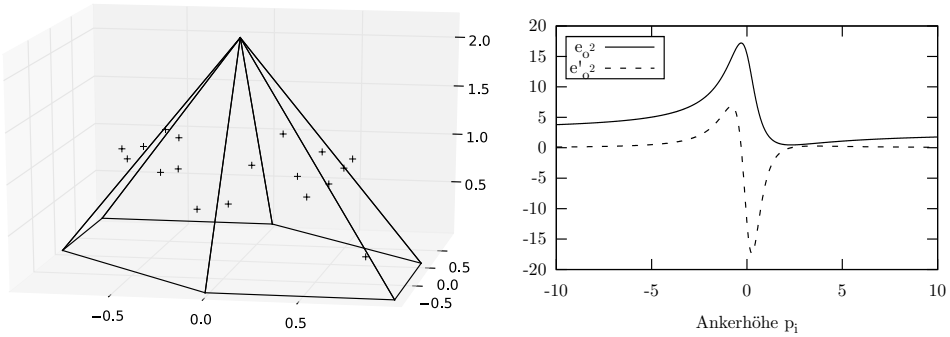


Abb. 4.7.: Bei der orthogonalen Abstandsmessung gibt es keine eindeutige Zuordnung der Punkte zu Punkten im Weltmodell. In (a) gibt es keinen Schnittpunkt von Facette und Punkt in Richtung der Facettennormale. In (b) gibt es mehrere. Abhilfe schafft es, Punkte über ihre (u, v)-Koordinaten eindeutig einer Facette zuzuordnen (gestrichelte Linien) und den Abstand zur Ebene (grau) zu messen, in der die Facette liegt.

geschrieben werden. Die Matrix X_i ist eine $N_{\mathbf{X}_i} \times 3$ Matrix mit den der Facette zugeordneten Punkten als Zeilenvektoren. $\phi_i(\mathrm{p})$ ist eine Funktion, die die Normale der Facette mit Einheitslänge errechnet. $\psi_i(\mathrm{p})$ errechnet die Verschiebung der Ebene der Facette zum Nullpunkt. Der Vektor p enthält die variablen Ankerhöhen.



(a) (b)

Abb. 4.8.: Ein kleines Netz bestehend aus 7 Ankern und 6 Facetten mit 3 Punkten pro Facette (a). (b) zeigt einen Plot des quadratischen Fehlers in orthogonaler Richtung dieses Netzes, über die Höhe des Ankers in der Netzmitte. Die gestrichelte Linie zeigt die Ableitung des Fehlers zur Ankerhöhe.

Ein kleines Netz und der Plot dieser Funktion in diesem Netz, bei vertikaler Verschiebung der Höhe des mittleren Ankers , wird in Abbildung 4.8 gezeigt. Man kann erkennen, dass eine Veränderung der Ankerhöhe sich nicht linear auf den Fehler auswirkt. Die, gestrichelt eingezeichnete, Ableitung des Fehlers zur Ankerhöhe konnte nicht zur Ankerhöhe aufgelöst werden. Von daher wird zur Lösung des Problems das iterative

Funktionsminimierungsverfahren aus Abschnitt 3.4 eingesetzt. Dabei wird, wie auch im vorherigem Abschnitt beschrieben, jeder Anker einzeln optimiert.

4.2.3. Absolute Abstandsmessung

Die quadratische Abstandsmessung ist praktisch. Sie hat stets positive Werte, so dass sich Abstände unterschiedlicher Punkte nicht gegenseitig auslöschen. Auch werden größere Entfernungen stärker bestraft als kleinere, was wünschenswert sein kann. Im Folgenden soll nun aber erst ein Schwachpunkt dieser Messmethode aufgezeigt werden. Sie ist anfällig gegenüber Ausreißern, also starken, aber relativ seltenen Messfehlern. Die absolute Abstandsmessung wird anschliessend vorgestellt. Sie ist robust gegenüber solchen Ausreißern, hat allerdings den Nachteil, dass eine optimale Parameterkonfiguration hier nur über iterative Verfahren ermittelt werden kann.

Betrachten wir ein Weltmodell, das ausschliesslich aus einer Ebene besteht (also nicht wie sonst in dieser Arbeit aus einem regelmäßigem Dreiecksnetz). Dieses soll ebenfalls an eine Punktwolke angepasst werden, wobei in z-Richtung gemessen wird. Das Problem lautet also:

$$\arg\min_{A,B,D} e_{z^2} = \sum_{i=0}^{N-1} \|Ax_i + By_i + D - z_i\|^2 \tag{4.33}$$

Die Ableitung der Fehlerfunktion nullgesetzt und zu D aufgelöst führt zu

$$D = \frac{\sum_i z_i - A\sum_i x_i - B\sum_i y_i}{N} = \overline{z}_a - A\overline{x}_a - \overline{y}_a, \tag{4.34}$$

wobei $\overline{\mathbf{X}}_a = [\overline{x}_a, \overline{y}_a, \overline{z}_a]$ das arithmetische Mittel der Punktwolke ist. Die Ebene mit minimalem, quadratischem Abstand zur Punktwolke geht also durch diesen Punkt. Eine Eigenschaft des arithmetischen Mittels ist, dass ein Messpunkt, der stark von den anderen abweicht, den Wert direkt in seine Richtung beeinflusst.

Bei einer Ebene, die anstatt durch das arithmetische Mittel der Messpunkte, durch den Median geht, erhält man ein Robustheit gegenüber einem solchen Ausreißer. Die stark abweichenden Entfernungsdifferenz eines Ausreißers hat keine Auswirkung auf den Fehlerwert. Dass der Median gerade den Fehler bei absoluter Abstandsmessung minimiert, sieht man, indem man die Funktion

$$e = \sum_i \|z_i - \overline{z}_m\| \tag{4.35}$$

zu z_M ableitet und nullsetzt:

$$0 = \sum_i \frac{z_i - \overline{z}_m}{\|z_i - \overline{z}_m\|} \tag{4.36}$$

Diese Gleichung kann nur erfüllt werden, also e ein Minimum haben, wenn es genau gleich viele Messwerte z_i gibt, die größer als $\overline{z}_m$ sind, wie es Messwerte gibt, die kleiner sind. Also nur, wenn $\overline{z}_m$ dem Median der Messdaten entspricht.

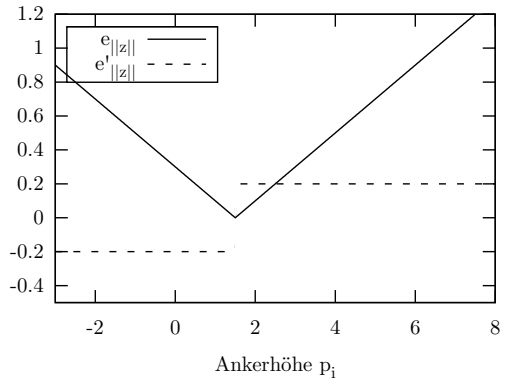


Abb. 4.9.: Plot des absoluten Abstands eines Punkts zum Weltmodell über die Höhe eines Ankers. Im Minimum der Fehlerfunktion ist die gestrichelt gezeichnete Ableitung nicht definiert.

Abbildung 4.9 zeigt einen Plot der Fehlerfunktion

$$e_{\|z\|}(\mathbf{A}_z) = \sum_{j=0}^{N_{F\mathbf{A}}} \sum_{c=0}^{N_{\mathbf{X}j}} \|A_j x_c + B_j y_c + D_j - z_c\| \tag{4.37}$$

und dessen Ableitung zur Ankerhöhe. Es wurde die Entfernung zu nur einem Punkt, in einem kleinen Netz mit variierender Ankerhöhe, gemessen. Wie dort zu erkennen ist, ist die Ableitung im Minimum der Funktion nicht definiert. Um dennoch eine Lösung für die Parameter der Funktion zu bekommen, wird das in Kapitel 3.4 beschriebene Verfahren benutzt. Dies gilt ebenso für die absolute orthogonale Fehlermessung.

4.2.4. Glattheitsbedingung

Neben den reinen geometrischen Informationen, die in Form der Punktwolke vorliegen, soll noch eine Glattheitsbedingung bei der Terrainrekonstruktion berücksichtig werden. Motiviert ist dies, um fehlerhafte oder fehlende Punktdaten in Teilen des Weltmodells auszugleichen. Der Glattheitsbedingung liegt die Vermutung zu Grunde, dass ein Gelände in sich einen eher homogenen, glatten Eindruck macht, und keine sprunghaften und impulsartigen Änderungen in der Oberflächenbeschaffenheit aufweist. Solche Sprünge sollen geglättet werden. Außerdem soll in Bereichen mit fehlenden Punktinformationen interpoliert werden.

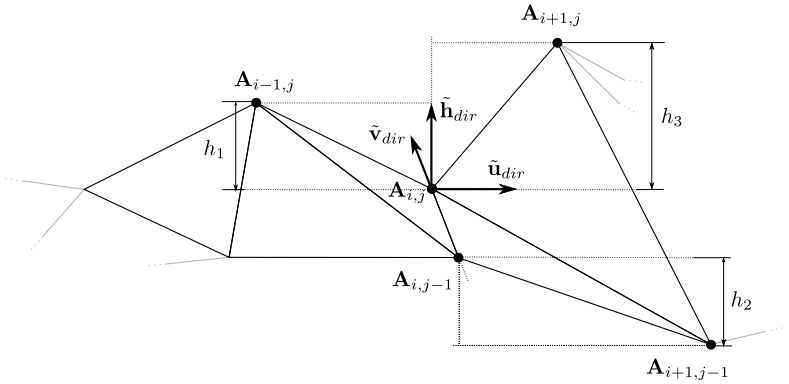


Abb. 4.10.: Messung der Glattheit des Terrains an einem Anker $\mathbf{A}_{i,j}$. Der Anker hat die vier Nachbarn $\mathbf{A}_{i,j-1}$, $\mathbf{A}_{i-1,j}$, $\mathbf{A}_{i+1,j-1}$ und $\mathbf{A}_{i+1,j}$. $\mathbf{A}_{i,j}$ und $\mathbf{A}_{i,j-1}$ haben dieselbe Höhe. Die Differenz der Höhen von $\mathbf{A}_{i-1,j}$ und $\mathbf{A}_{i,j}$ bzw. $\mathbf{A}_{i+1,j-1}$ und $\mathbf{A}_{i,j}$ unterscheidet sich nur durch ihr Vorzeichen, haben aber denselben Betrag. $\mathbf{A}_{i+1,j}$ ist deutlich Höher als $\mathbf{A}_{i,j}$.

Um dies zu realisieren wird ein Fehlermaß für die Glattheit des Geländes eingeführt. Dieses wird errechnet, indem Höhenunterschiede benachbarter Anker gemessen werden. Je inhomogener diese ausfallen, desto stärker wächst der Fehler. Es wird, wie bei der Punktabstandsmessung, zwischen absoluter und quadratischer Fehlermessung unterschieden. Folgende Terme messen nun, je nach gewünschner Messmethode, die Glattheit des Geländes:

$$e_{s^2} = \sum_{i=0}^{i<N_A} \left\| \sum_{i,j \text{ benachbart}} \mathbf{A}_{jh} - \mathbf{A}_{ih} \right\|^2 \tag{4.38}$$

$$e_{\|s\|} = \sum_{i=0}^{i<N_A} \left\| \sum_{i,j \text{ benachbart}} \mathbf{A}_{jh} - \mathbf{A}_{ih} \right\| \tag{4.39}$$

Ein gleichmäßiger Anstieg des Geländes wird durch Verwendung dieser Terme nicht bestraft, da sich die Differenzen der Ankerhöhen gegenseitig aufheben können. In Abbildung 4.10 wird dies verdeutlicht. Die Höhenunterschiede h_1 und h_2 haben denselben Betrag aber unterschiedliche Vorzeichen. $\mathbf{A}_{i,j-1}$ und $\mathbf{A}_{i,j}$ sind gleich hoch. Es ergibt sich also $(0 + h_1 + h_2 + h_3)^2 = h_3^2$ als Glattheitsfehler am Anker $\mathbf{A}_{i,j}$. Gleiches gilt für die absolute Fehlermessung, die in diesem Beispiel in $\|h_3\|$ resultiert.

Um die Glattheitsbedingung bei der Geländeanpassung zu berücksichtigen, wird der Glattheitsfehler mit einem Faktor α gewichtet und zu der Fehlermessung addiert. Zum Beispiel wird Gleichung (4.25) also zu

$$e = e_z + \alpha_{smooth} \tag{4.40}$$

ergänzt, wobei α die stärke der Glättung des Geländes steuert.

Der Glattheitsfehler steigt ebenfalls linear mit dem geometrischen Fehler für Gleichung (4.40). Für ihre analytische Lösung (vgl. Gleichung (4.41)) ergibt sich dann

$$\begin{aligned} d_0 &= \sum e'_{z^2}(0) + \alpha\delta e_s(a) \\ d_1 &= \sum e'_{z^2}(1) + \alpha\delta e_s(a) \\ \Delta d &= d_1 - d_0 \\ A_{0z} &= d_0/\Delta d. \end{aligned} \tag{4.41}$$

4.2.5. SimpleFit - Parameterinitialisierung

In den vorherigen Abschnitten wurde festgestellt, dass, für die Anpassung des Weltmodells nach quadratischer orthogonaler Abstandsmessung und dem absolutem Fehlermaß, das iterative Funktionsminimierungsverfahren aus Abschnitt 3.4 benutzt werden muss. Dieses braucht als Initialisierung ein Fenster, in dem der optimale Parameterwert liegt, sowie eine initiale Schätzung des optimalen Parameterwerts. Es gilt also, für jeden Anker drei Parameterwerte (h_{min}, h_0, h_{max}) zu finden. h_{min} und h_{max} stellen die obere und untere Grenze des Suchfensters dar, und h_0 ist der geschätzte Wert für den optimalen Parameterwert.

Folgende drei Eigenschaften sind wichtig für eine solche initiale Parameterschätzung:

- Das Fenster, das gefunden wird, muss das Funktionsminimum enthalten.
- Der initiale Parameterwert sollte nahe bei diesem Minimum liegen.
- Das Verfahren sollte schnell sein.

Ein möglicher Ansatz wäre, zur Initialisierung zunächst eine Anpassung nach dem quadratischen Fehler in z-Richtung durchzuführen. Hier soll nun aber ein etwas schnellerer heuristischer Algorithmus vorgestellt werden. Er lässt sich grob in folgende drei Phasen unterteilen:

1. Gute Schätzung der optimalen Ankerhöhe errechnen.
2. Glattheitsbedingung integrieren.
3. Obere und untere Grenze definieren.

Die folgenden Absätze beschreiben jede dieser Phasen.

Ankerhöhe schätzen anhand von Stichproben Für jede Kachel eines Ankers wird ein zufällige Stichprobe von N Punkten gewählt. Hierdurch bleibt die Laufzeit des Algorithmus konstant gegenüber der Anzahl der Punkte im Weltmodell.

Für jeden dieser Punkte wird nun die Höhe des Ankers bestimmt, bei der der Punkt in der Kachel liegen würde (siehe Abbildung 4.11).

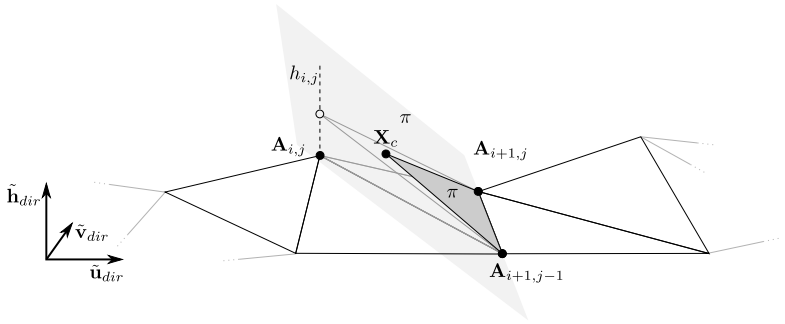


Abb. 4.11.: Es wird die Höhe $h_{i,j}$ des Ankers $\mathbf{A}_{i,j}$ gesucht, bei der der Punkt $\mathbf{X}_c$, zusammen mit allen drei Ankern der Facette, in einer Ebene liegt.

Sei $\mathbf{X}_j$ ein Punkt aus der Stichprobe. Er ist der Facette mit den Ankern $\mathbf{A}_{i,j}$, $\mathbf{A}_{i+1,j-1}$, $\mathbf{A}_{i+1,j}$ zugeordnet. $\mathbf{A}_{i,j}$ sei der zu variierende Anker. Aus den beiden festen Ankern und $\mathbf{X}_c$ wird eine Ebene π konstruiert. Mit der Normalenform kann festgestellt werden, ob der variable Anker in dieser Ebene liegt. Das ist der Fall wenn folgende Gleichung gilt:

$$(\mathbf{A}_{i,j} - \mathbf{X}_c) \cdot \mathrm{n} = 0 \tag{4.42}$$

Hierbei ist n ein Vektor der senkrecht zu π liegt.

Es soll nun eine Höhe h' für $\mathbf{A}_{i,j}$ gefunden werden, so dass der Anker ebenfalls in der Ebene liegt. Um diese Höhe zu ermitteln, wird Gleichung (4.42) nach der Höhe des variablen Ankers aufgelöst:

$$z_{i,j} = \frac{-x_{i,j}n_x + x_c - y_{i,j}n_y + y_c n_y + z_c n_z}{n_z} \tag{4.43}$$

Die optimale Höhe des variablen Ankers wird für jeden Punkt der Stichprobe einzeln ermittelt. Die Schätzung der optimalen Ankerhöhe für alle Punkte ergibt sich aus dem Median $\overline{h}_m$ dieser Werte. Der Median wird gewählt, um Robustheit gegenüber Ausreißern in der Stichprobe zu erlangen (vergleiche hierzu Abschnitt 4.2.3).

Integrierung der Glattheitsbedingung Um die Glattheitsbedingung zu integrieren, wird ein Least Squares Problem formuliert, das, unter Berücksichtigung der Glattheitsbedingung aus Gleichung (4.38), die Abweichung der Ankerhöhe von $\overline{h}_m$ minimiert:

$$\arg\min_h e = \sum_{c=0}^{M-1} (h - \overline{h}_m)^2 + \alpha e_{s^2} = M(h - \overline{h}_m)^2 + \alpha e_{s^2} \tag{4.44}$$

M ist die Anzahl der Punkte, die einer Kachel des variablen Ankers zugeordnet sind. Die Bedeutung der Summe in dieser Gleichung ist die Annahme, dass für jeden dieser M Punkte die optimale Ankerhöhe eigentlich $\overline{h}_m$ wäre. Der Glattheitsterm αe_{s^2} ist abhängig von der Ankerhöhe h.

Gleichung (4.44) wird nach h abgeleitet und nullgesetzt. Dies Aufgelöst nach h ergibt

$$h_0 = h = \frac{N_a \alpha S_h + M\overline{h}_m}{N_a^2 \alpha + M}. \tag{4.45}$$

N_a ist die Anzahl der benachbarten Anker, S_h die Summe ihrer Höhen. M ist die Anzahl aller Punkte, die den Facetten des variablen Ankers zugeordnet sind. Der Wert h_0 ist nun der, heuristisch bestimmte, initiale Wert für die iterative Funktionsminimierung.

Durch die zufällige Auswahl der Stichproben konvergiert dieses Verfahren generell nicht. In der Praxis wird sich in Kapitel 6 allerdings zeigen, dass dieses Verfahren allein bereits sehr gute (und schnelle) Ergebnisse liefert.

Bestimmung der oberen und unteren Grenze Die Grenzen sollen so initialisiert werden, dass sichergestellt ist, dass sie auch wirklich das Minimum der Funktion einschliessen. So kann das iterative Funktionsminimierungsverfahren, auch bei schlecht geschätztem h_0, die optimale Parameterstellung finden.

Um dies zu erreichen wird die Tatsache ausgenutzt, dass zwei Terme in die Berechnung des Fehlers eingehen. Das sind der Term zur Messung des geometrischen, und der zur Messung des Glattheitsfehler. Für die geschätzte Ankerhöhe h_0 wird der lokale Fehler e_{h_0} errechnet. Nun werden die beiden Ankerhöhen gesucht, bei denen allein der Glattheitsfehler schon so groß ist, wie der gesamte Fehler bei der geschätzten Ankerhöhe. Da der Glattheitsfehler bei variabler Ankerhöhe monoton steigt, ist es ausgeschlossen, dass es außerhalb dieses Fensters Ankerhöhen gibt, die einen kleineren Gesamtfehlerwert liefern.

Der lokale Glattheitsfehler für einen Anker $\mathbf{A}$ mit der Höhe h errechnet sich aus

$$e_s(\mathbf{A}) = \left\| \sum_{i=1}^{N_a} h_i - h \right\|^2, \tag{4.46}$$

wobei h_i jeweils die Höhe eines benachbarten Ankers, und N_a die Anzahl aller benachbarten Anker ist. Gesucht ist nun der Wert h, bei dem $e_s = e_{h_0}$ ist. Diesen Wert in Gleichung (4.46) eingesetzt und zu h aufgelöst ergibt

$$h^a = \frac{\alpha S_h - \sqrt{\alpha e_{h_0}}}{N_a \alpha},\ h^b = \frac{\alpha S_h + \sqrt{\alpha e_{h_0}}}{N_a \alpha}. \tag{4.47}$$

S_h ist die Summe der Höhen, der mit $\mathbf{A}$ benachbarten Anker. Die Werte h^a und h^b sind nun die obere bzw. untere Grenze für den Algorithmus zur Funktionsminimierung. Es ergeben sich also insgesamt $(\min(h^a, h^b), h_0, \max(h^a, h^b))$ für dessen Initialisierung.

5. Implementierung

Die in den Kapiteln 3 und 4 beschriebenen Methoden wurden in einem C++ -Programm entwickelt, um eine Terrainrekonstruktion aus Stereodaten von unterschiedlichen Kameraperspektiven zu realisieren. Die beiden Hauptprobleme, Stereo-Matching und Geländemodellierung, wurden jeweils in die Bibliotheken *libStereoVision* und *libPolygonnetz* ausgelagert, für die auch Testprogramme geschrieben wurden. Außerdem wurde ein Programm *Presentation* entwickelt, das diese Bibliotheken benutzt, um das Verfahren für den Einsatz in einem realen System zu simulieren. Dies drückt sich darin aus, dass der Datensatz an Stereo- und Posendaten, der als Eingabe dient, als stetiger Datenstrom aufgefasst wird. Dieser muss von dem System, unabhängig von der Laufzeit der Algorithmen, verarbeitet werden. Hieran kann überprüft werden, ob das Verfahren für den "Echtzeit"-Einsatz geeignet ist.

Für die Realisierung dieses Systems wurde auf frei verfügbare Programmbibliotheken zurückgegriffen:

- OpenCV: Bibliothek für Bildverarbeitung. Es werden Stereo-Matching Algorithmen aus dieser Bibliothek benutzt. Außerdem werden einige Datentypen und Algorithmen von OpenCV für algebraische Berechnungen und zur persistenten Datenspeicherung genutzt.
- OpenGL: Wird benutzt, um das Geländemodell zu rendern und Informationen über die Geländerekonstruktion zu visualisieren.
- GSL: Bibliothek mit diversen numerischen Algorithmen. Das iterative Funktionminimierungsverfahren für die Geländeanpassung wird aus dieser Bibliothek benutzt.
- CML: Bibliothek für lineare Algebra. Für algebraische Berechungen bei der Konstruktion, Anpassung und Visualisierung des Weltmodells.
- Qt: GUI-Programmierung und Threading.

Abbildung 5.1 zeigt ein Diagramm der im Rahmen der Diplomarbeit implementierten Klassen (eckige Kästen). Es wird illustriert wie diese zu Bibliotheken (gestrichelte, abgerundete Kästen) zusammengefasst sind und welche Programme (abgerundete Kästen) sie benutzen. Die Bibliotheken werden zusammen mit ihren jeweiligen Testprogrammen in den folgenden Abschnitten genauer erklärt. In Abschnitt 5.3 wird das Programm *Presentation* vorgestellt, die Geländerekonstruktion unter Benutzung der erzeugten Bibliotheken implementiert. In dem folgenden Kapitel *Evaluierung* wird dann das implementierte Verfahren anhand von Testdaten auf seine Leistungsfähigkeit untersucht.

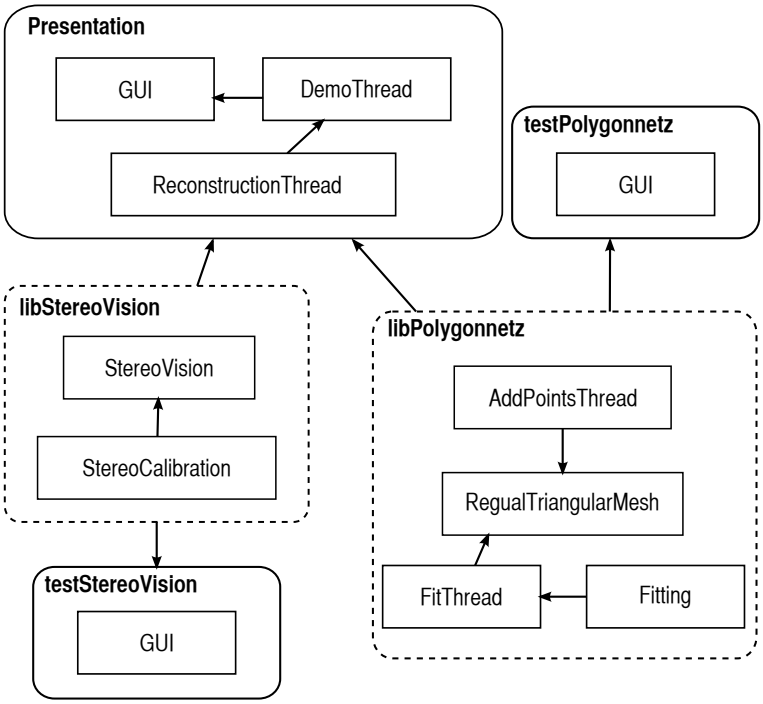


Abb. 5.1.: Diagramm, das die implementierten Komponenten und deren Abhängigkeit untereinander darstellt. Unterschiedliche Programmteile (eckige Kästen) werden in Bibliotheken (gestrichelte, abgerundete Kästen) oder Programmen (abgerundete Kästen) zusammengefasst. Die Pfeile symbolisieren, dass die jeweilige Komponente (Start des Pfeils) von der verbundenen Komponente benutzt wird (Pfeilspitze).

5.1. libStereoVision

Die Bibliothek *libStereoVision* wird benutzt, um aus Eingabebildern (jeweils Stereobildpaare) Punktwolken zu generieren. Hierzu sind im Wesentlichen die Verfahren aus den Abschnitten 3.1-3.3 implementiert. Den Großteil der dort beschriebenen Funktionen stellt die Klasse *StereoCalibration* bereit. So bietet sie die folgenden Möglichkeiten:

- Eine (Stereo) Kamerakalibrierung kann geladen werden. Dies ist essentiell, da die folgenden Funktionen von der Kamerakalibrierung abhängen.
- Es können Punkte auf die Bildebenen der einzelnen Kameras projiziert werden.
- Eine Rückprojektion von Bildpunkten als Strahl im Raum ist möglich.
- Bildpunkte können ver- und entzerrt werden.
- Für eine Stereokalibrierung wird die Fundamentalmatrix errechnet.

- Es können die Epipole und Epipolarlinien des Stereokamerasystems angefordert werden.
- Die Kameras können rektifiziert werden. Anschließend können Stereobildpaar ebenfalls rektifiziert werden.
- Gegeben von Punktkorrespondenzen im Stereobildpaar können 3D-Weltpunkte rekonstruiert werden.

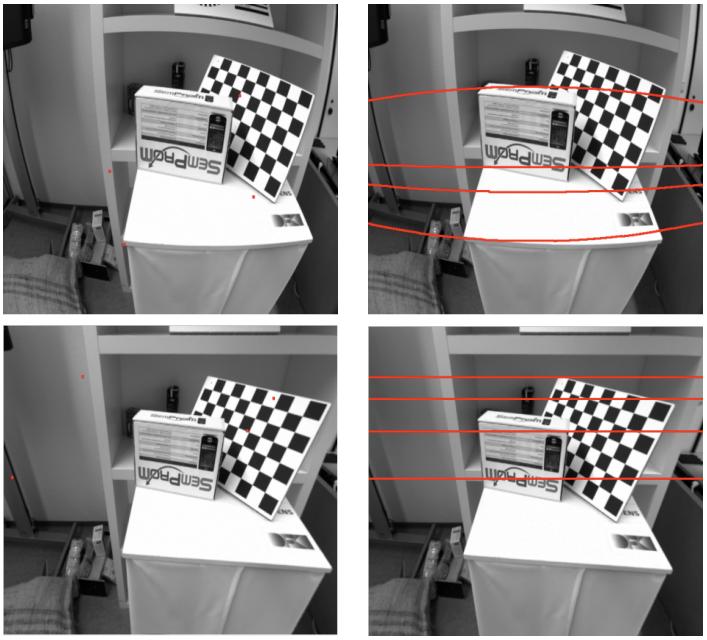


Abb. 5.2.: Ein Eingabebildpaar (obere Zeile) mit eingezeichneten Epipolarlinen zu vier Punkten im linken Kamerabild. Die untere Zeile zeigt die rektifizierten Versionen der Eingabebilder und zu vier anderen Bildpunkten die Epipolarlinien. Die Kamerabilder stammen aus [DFKI, 2010].

Abbildung 5.2 zeigt für das Stereobildpaar (obere Zeile) die rektifizierten Versionen der Bilder und hat jeweils zu vier Punkte im linken Kamerabild, die Epipolarlinien im rechten Kamerabild eingezeichnet.

Die Klasse *StereoVision* stellt eine Schnittstelle zu *StereoCalibration* dar und erlaubt zudem die in Abschnitt 3.3.1-3.3.3 beschrieben Stereo-Matching-Verfahren auf ein Stereobildpaar anzuwenden. Wurde eine Kamerakalibrierung geladen, werden die Eingabebilder rektifiziert und, wenn gewünscht, skaliert, um die Bearbeitungsgeschwindigkeit beim Matching zu erhöhen. Für die daraus resultierenden Disparitätsbilder kann eine Punktwolke angefordert werden. Der Daten- und Kontrollfluss zwischen einem Programm, das *libStereoVision* benutzt und der Bibliothek, wird in Abbildung 5.3 gezeigt. Befehle werden hier als gestrichelte und Daten als durchgezogene Pfeile dargestellt.

5.1.1. testStereoVision

Das Programm *testStereoVision* dient zum Testen der Bibliothek *libStereoVision*. Über den Button *Load Config* kann eine Konfigurationsdatei für die Kamerakalibrierung geladen werden. Der Button *Init Rectification* und dient dazu, die Rektifizierung der Stereokamera zu initialisieren. Über das Eingabefeld *Scale* kann ein Skalierungsfaktor angegeben werden, mit dem die Größe der Eingabebilder vor dem Matching verändert wird.

Mit der Checkbox *Show Rect.* können rektifizierte Versionen der Eingabebilder angezeigt werden. Die weiteren Checkboxen in der Gruppe *Input Display* dienen dazu, auf die angezeigten Eingabebilder, die über die Buttons *Load Left Image* und *Load Right Image* von der Festplatte geladen werden können, Epipolarlinen bzw. die Epipole einzuzeichnen (wobei letztere häufig außerhalb der Bilder liegen). *Draw Disparity* legt ein regelmäßiges Muster von vertikalen Linien im Abstand von 10 Pixeln über die Bilder, das einen dabei unterstützen soll, für bestimmte Punkte die Disparität manuell zu schätzen.

In der Gruppe *Matching* gibt die Checkbox *Rectify Input Images* an, ob die Eingabebilder vor dem Matching zu rektifiziert werden sollen. *LoG prefilter* ermöglicht vor dem Matching die Bilder durch einen *Laplacian of Gaussian* Filter vorab zu bearbeiten. Über die Combobox kann der gewünschte Stereo-Algorithmus gewählt werden und ein Klick auf den Button *Match* startet das Matching mit den gewählten Einstellungen.

In der Gruppe *Output Display* kann das gewünschte Ausgabeformat gewählt werden (es ist allerdings momentan nur *Greyscale* implementiert) und mit dem Eingabefeld *Scale Grey values* kann ein Faktor zur Skalierung der Grauwerte des Disparitätsbild eingestellt werden. Hiermit kann die Anzeigequalität des Disparitätsbildes verbessert werden.

Mit dem Button *To Point Cloud* kann das aktuelle Disparitätsbild in eine Punktwolke konvertiert, und auf der Festplatte gespeichert werden. Abbildung 5.4 zeigt einen Screenshot des Programms.

Über die Eingabefelder der Reiter *Mdlb, BM, GC* und *GPU* lassen sich die Parameter der entsprechenden Stereoalgorithmen einstellen (siehe Abbildung 5.5).

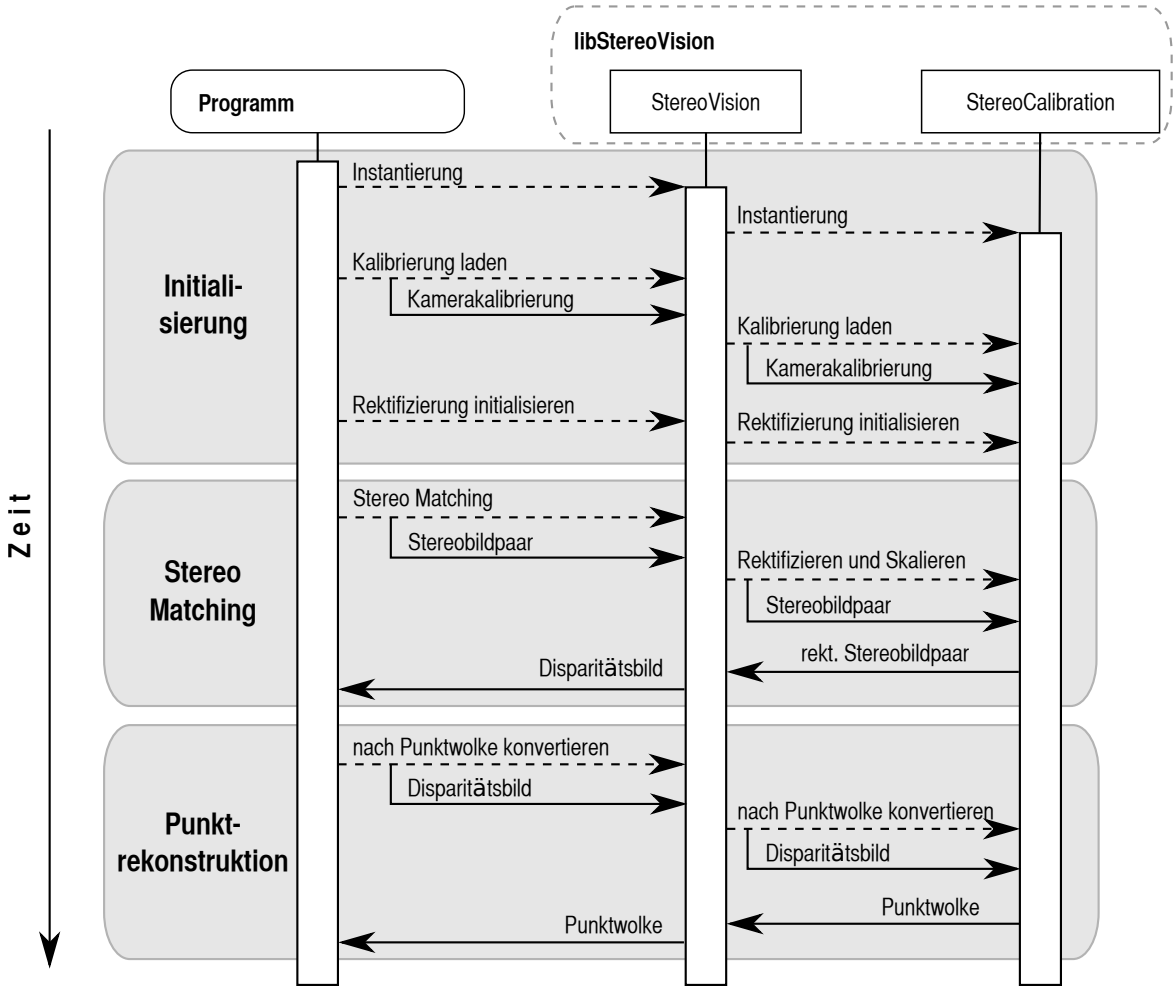


Abb. 5.3.: Der Kontrollfluss für ein Programm, das die Bibliothek *libStereoVision* benutzt, um eine Punktwolke aus einem Stereobildpaar zu erzeugen. Kontrollbefehle werden als gestrichelte und Daten als durchgezogene Pfeile dargestellt.

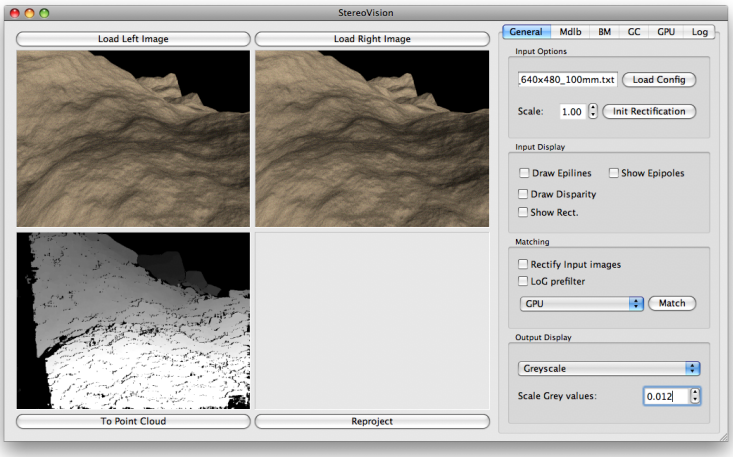


Abb. 5.4.: Screenshot des Testprogramms für die Bibliothek *libStereoVision*

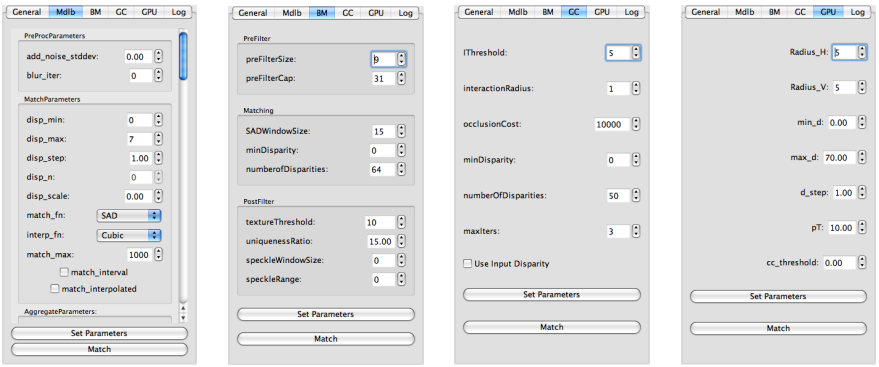


Abb. 5.5.: Parametereinstellungen für die Stereoalgorithmen von *libStereoVision* lassen sich über das Testprogramm vornehmen.

5.2. libPolygonnetz

Die Bibliothek *libPolygonnetz* stellt mit der Klasse *RegularTriangularMesh* die nötigen Funktionen bereit, um ein regelmäßiges Dreiecksnetz wie in Abschnitt 4.1 zu erzeugen und zu modifizieren. Die in Abschnitt 4.2 beschriebenen Verfahren um das Netz an eine Punktwolke anzupassen, sind in *Fitting* enthalten.

Einzelne Punkte dem Netz hinzuzufügen ist ein schneller Prozess. Kommen aber gleichzeitig sehr viele Punkte herein, wie es beim Stereomatching, je nach Dichte der Korrespondenzen und Auflösung der Ausgangsbilder, der Fall sein kann, dauert dies doch einige Zeit. Damit der Programmablauf in dieser Zeit nicht komplett blockiert wird, wurde die Abarbeitung der Punktwolke in einen Thread ausgelagert, der in der Klasse *AddPointsThread* implementiert ist. Es wird kontinuierlich ein Puffer von Punkten, der als Liste gespeichert ist, abgearbeitet, wobei die Punkte einzeln dem Netz hinzugefügt und aus dem Puffer gelöscht werden. Hierbei wird nach dem *First in First Out*-Prinzip (FIFO) vorgegangen. Ein Mutex verhindert den Überlauf des Puffers, indem die Abarbeitungsroutine so lange das weitere Hinzufügen von Punkten sperrt, bis die Anzahl der Punkte im Puffer unter einem Schwellwert liegt.

Das Anpassen des Weltmodells ist ein iterativer, relativ zeitaufwändiger Vorgang. Um den Programmablauf nicht zu unterbrechen, ist die Anpassung in der Klasse *FitThread*, als stetig im Hintergrund laufender Thread implementiert. *RegularTriangulatMesh* stellt eine Schnittstelle zur Steuerung der Parameter für die Anpassung bereit. So können die gewünschte Anpassungsstrategie, sowie der Glättungsfaktor konfiguriert werden.

Der Kontroll- und Datenfluss zwischen den Threads, für einen Anwendungsfall, in dem ein Programm *libPolygonnetz* benutzt, um sequentiell Punktwolken zum Weltmodell hinzuzufügen und dieses im Hintergrund anpassen zu lassen, ist in Abbildung 5.6 dargestellt. Die Kommunikation des Programms findet über ein *RegularTriangularMesh*-Objekt statt, das Instanzen der Thread-Klassen erstellt. Der Anpassungsprozess wird gestartet und nach und nach sollen weitere Punktwolken dem Netz hinzugefügt werden. Diese werden vom *AddPointsThread*-Objekt abgearbeitet.

Kommen die Punkte zeitversetzt aus unterschiedlichen Posen herein, so ist jeweils nur ein Teilbereich des Netzes von Veränderungen betroffen und muss neu angepasst werden. Dies wird von dem *FitThread* berücksichtig, indem für jeden Anker folgende Kriterien überprüft werden:

- Eine Facette, die an dem Anker hängt, hat einen Punkt hinzugefügt bekommen.
- Ein Nachbaranker wurde in der Höhe verändert.

Treffen diese Kriterien nicht zu, so kann der Anker bei der Geländeanpassung übersprungen werden. Dies zeigt Abbildung 5.7, wo Screenshots aus dem Programm *Presentation*, das in Abschnitt 5.3 vorgestellt wird, dargestellt sind. Die anzupassenden Anker sind rot dargestellt, diejenigen, die übersprungen werden grün. Das Programm *Presentation* benutzt *libPolygonnetz* und wurde mit Testdaten ausgeführt, die in Kapitel 6

vorgestellt werden. Das Video *Videos/Show_updated-1200_frames.avi*[1] auf der beiliegenden DVD vermittelt einen noch besseren Eindruck als die Screenshots.

Über den Zustand des Netzes wird stets eine Reihe an statistischen Informationen erstellt:

- Die mittlere Ankerhöhe des Netzes und für jeden Anker seine relative Höhe hierzu.
- Das Verhältnis von Punktanzahl zur Anker- und Facettenanzahl.
- Die mittlere Belegung der Facetten mit Punkten. Hieraus wird die relative Belegung der Facetten und Anker errechnet.
- Die mittlere Höhe, Varianz und Standardabweichung der Punkte innerhalb einer Facette.
- Die maximale Abweichung von der mittleren Höhe der Punkte innerhalb einer Facette.
- Die Standardabweichung der Punkte zu dem Netz.

Das Netz kann auf unterschiedliche Arten visualisiert werden, wobei auch diese statistischen Informationen berücksichtigt werden können. Bei der Evaluierung in Kapitel 6, werden die statistischen Daten dargestellt und ausgewertet.

[1] Online unter `http://vimeo.com/14633023` (Stand 2.9.2010)

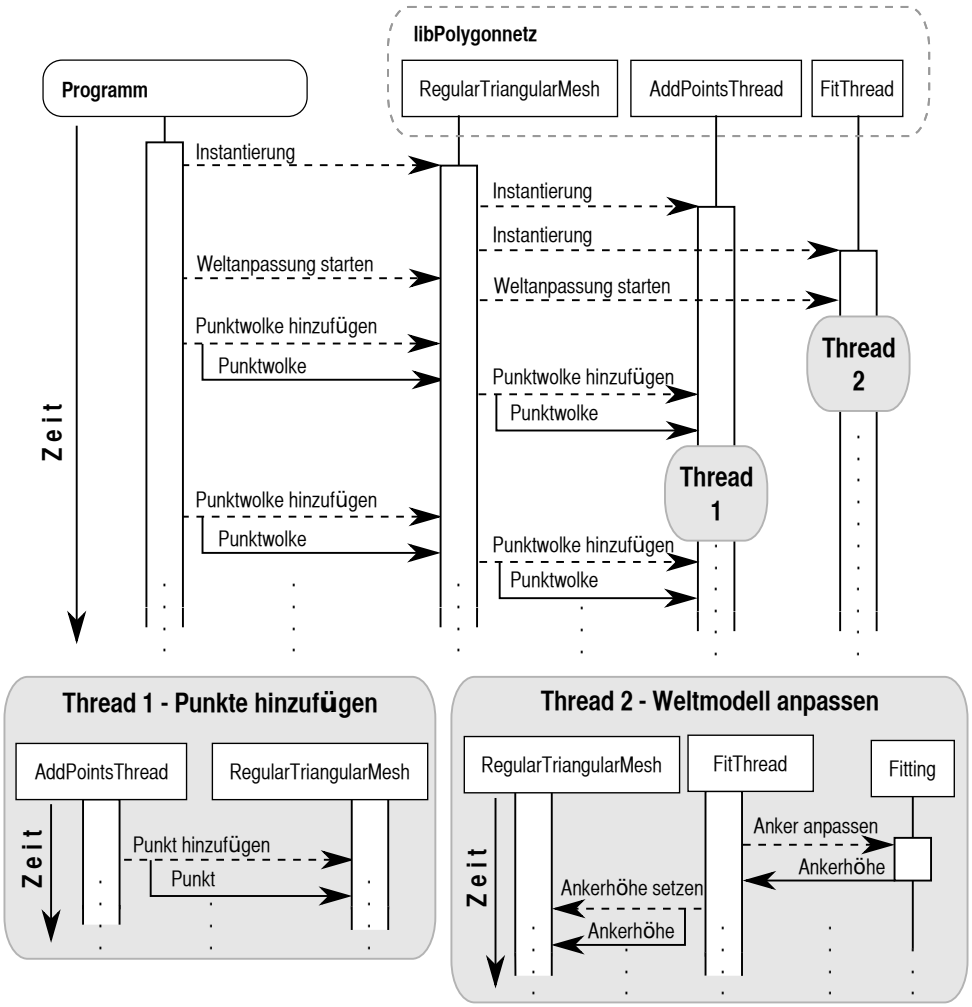


Abb. 5.6.: Daten- und Kontrollfluss für ein Anwendungsszenario von *libPolygonnetz*, in dem ein Programm die Bibliothek benutzt, um ein Weltmodell an sequentiell erzeugte Punktwolken anzupassen.

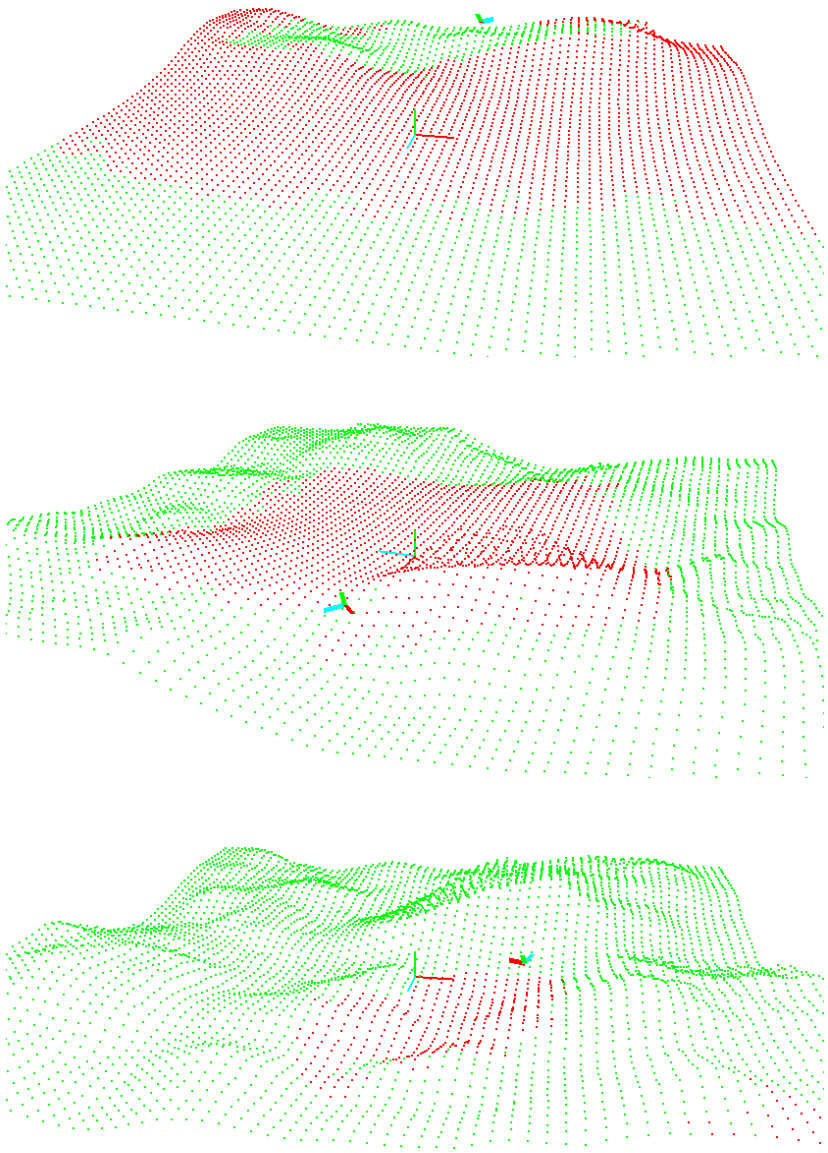


Abb. 5.7.: Es wird berücksichtigt, welche Anker momentan angepasst werden müssen. Diese werden in Rot dargestellt, die Anker, die ausgelassen werden können, in Grün. Die Bilder stammen aus einem Durchlauf durch ein virtuelles Terrain mit dem Programm *Presentation* zu unterschiedlichen Zeitpunkten. Sind bereits große Teile des Geländes bekannt, konvergiert das Gelände, bei neu hereinkommenden Punkten, bereits nach wenigen Iterationen. So muss stets nur ein kleiner Teil des Geländes neu angepasst werden.

5.2.1. Testprogramm - testPolygonnetz

Zum Testen der Funktionalität des Netzes wurde das Programm *testPolygonnetz* als Qt-Applikation geschrieben. Über eine Benutzeroberfläche kann eine Instanz der Klasse *RegularTriangularMesh* erzeugt und dessen Parameter kontrolliert werden. Punktwolken können geladen und die Geländeanpassung kann konfiguriert und durchgeführt werden. Eine Visualisierung des Netzes ist in OpenGL realisiert, wobei durch das Laden einer Kamerakalibrierung beim Rendern eine reale Kamera simuliert wird. Abbildung 5.8 zeigt einen Screenshot von *testPolygonnetz*. Es wurde eine Punktwolke in Form einer zweidimensionalen Sinuskurve geladen und ein Weltmodell an diese angepasst.

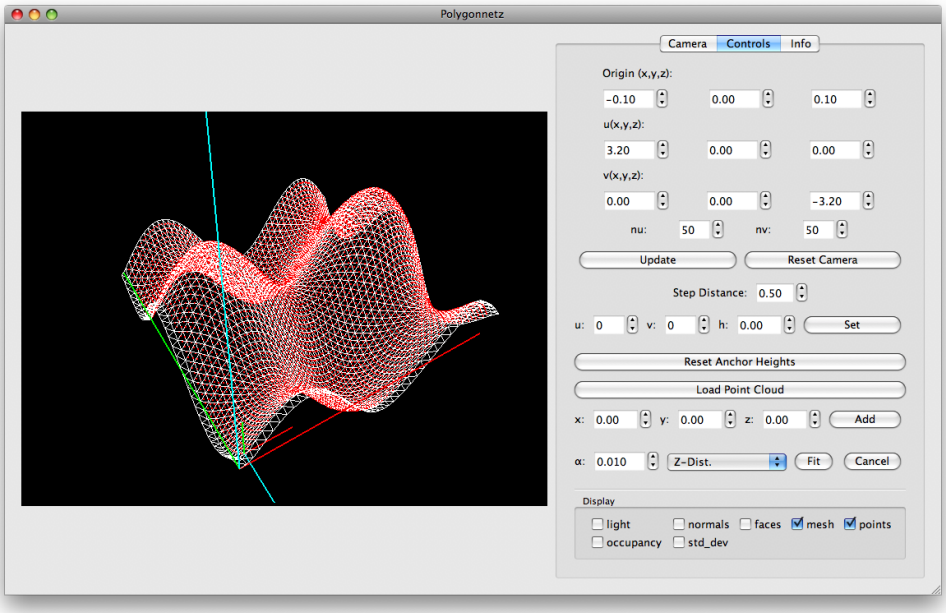


Abb. 5.8.: Screenshot des Testprogrammes für die Bibliothek *libPolygonnetz*

5.3. Presentation

In dem Programm *Presentation* werden die Bibliotheken *libStereoVision* und *libPolygonnetz* benutzt, um aus einer Stereobildsequenz mit korrespondierende Posen ein Weltmodell zu erzeugen. Hierbei werden die Eingabedaten sequentiell abgearbeitet, was in folgenden Schritten geschied:

- Eingabedaten lesen. Das sind Bilddaten mit korrespondierenden Posen.
- Stereo-Matching auf den Bilddaten durchführen.
- Punktrekonstruktion aus dem Disparitätsbild.
- Punkte transformieren um Pose zu berücksichtigen.
- Punkte im Weltmodell registrieren.
- Weltmodell an Punkte anpassen.

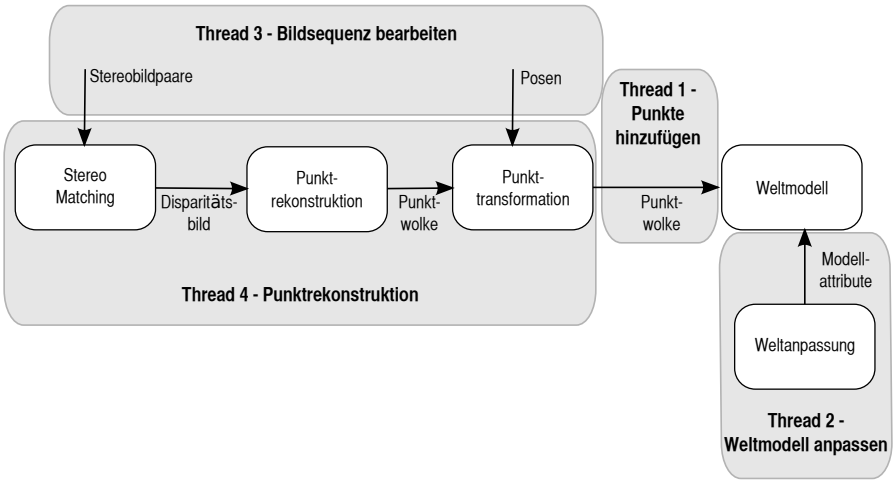


Abb. 5.9.: Unterschiedliche Aktivitäten für die Geländerekonstruktion sind in Threads ausgelagert.

Das Programm ist so entworfen, dass nicht erst die gesamte Bildsequenz verarbeitet wird, bevor die Welmodellanpassung begonnen wird, sondern dass diese Prozesse zeitgleich passieren. Es ist also zu jedem Zeitpunkt in der Bildfolge bereits ein Weltmodell vorhanden, das den derzeitigen Kenntnisstand über die Welt widerspiegelt. Diese Art der Implementierung soll die "Echtzeitfähigkeit" des Systems testen. Hierzu wurden Teile des Programmablaufs in Threads ausgelagert, damit zeitraubende Aktivitäten nicht den Programmablauf behindern. Abbildung 5.9 zeigt den Programmablauf und welche Aktivitäten von Threads bearbeitet werden. Dabei werden die Threads 1 und 2 aus der Abbildung 5.6, sowie 3 und 4 aus Abbildung 5.11 verwendet.

Über eine grafische Benutzeroberfläche (GUI) können die Eingabedaten, in Form von zwei Textdateien mit Pfaden zu Bilddaten in chronologischer Reihenfolge, sowie korre-

spondierenden Posen angegeben werden. Die Eingabedaten werden als kontinuierlicher Datenstrom verarbeitet, dessen Abarbeitungsgeschwindigkeit konfiguriert werden kann. Ebenso können das Stereo-Matching und das Weltmodell konfiguriert werden. Außerdem kann ein Faktor angegeben werden, um den die Größe der Punktwolke aus jedem Schritt reduziert wird. Permanent wird das derzeitige linke Bild aus den Eingabendaten, sowie zwei Versionen des Weltmodells angezeigt. Bei der einen Version (unten rechts) ist die Pose der virtuellen Kamera gleich der Pose aus den Eingabedaten. Die Farben der Kacheln ergeben sich als Mittelwert der Farben ihrer Punkte. Bei der anderen Ansicht (oben) handelt es sich um eine Vogelperspektive, wobei die virtuelle Kamera stets um den Mittelpunkt des Weltmodels rotiert. Abbildung 5.10 zeigt das Programm während eines Durchlaufs mit künstlichen Testdaten (diese werden im folgenden Kapitel beschrieben). Auf der beiliegenden DVD ist ein Video des Durchlaufs unter *Videos/BM-SqrZ-7000.avi*[2] zu finden. Dieses stellt die Arbeitsweise der Software besser dar, als es mit Screenshots möglich ist.

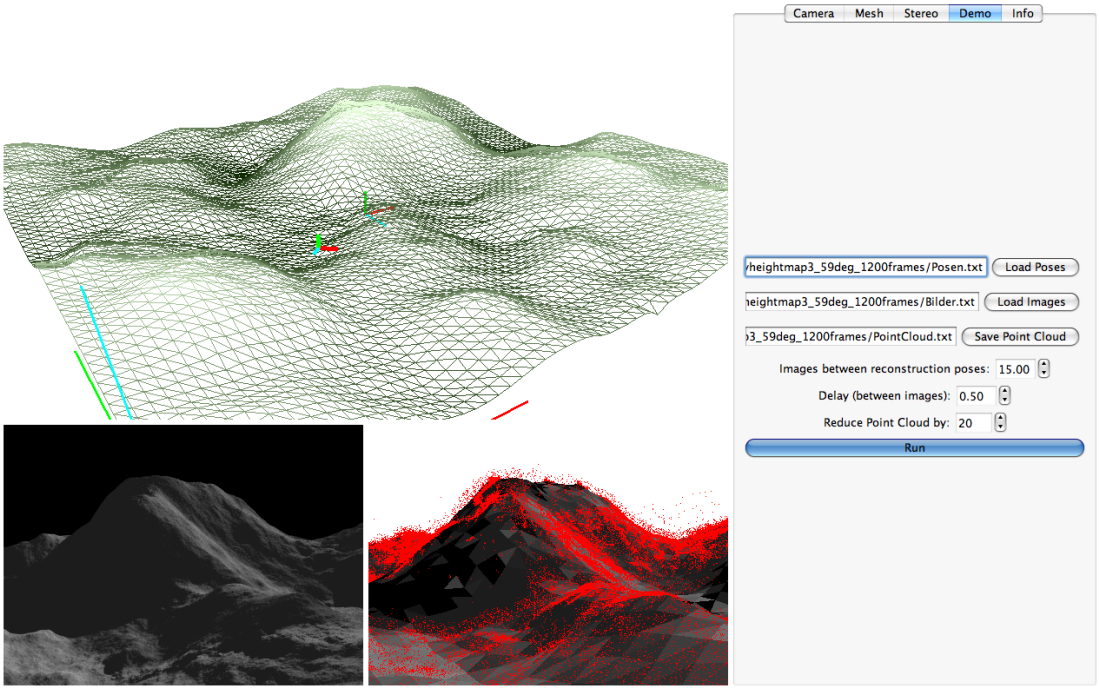


Abb. 5.10.: Das Programm *Presentation* während der Ausführung mit künstlichen Testdaten. Oben wird eine Vogelperspektive auf das Weltmodell angezeigt, links unten das derzeitige Eingabebild, rechts unten das Weltmodell in der derzeitigen Pose, mit Einfärbung der Kacheln. Die Punktwolke wird in dieser Ansicht rot dargestellt.

[2]Online unter `http://vimeo.com/14633200` (Stand 2.9.2010)

In Abbildung 5.11 ist der Daten- und Kontrollfluß für einen typischen Ablauf des Programms aufgeführt. Zunächst werden über die Benutzeroberfläche die nötigen Konfigurationen vorgenommen. Anschließend wird die Geländerekonstruktion gestartet. Zum besseren Verständnis des Diagramms sei noch darauf hingewiesen, dass die Threads 1 und 2 kontinuierlich laufen, und erst terminieren, wenn sie den entsprechenden Befehl (etwa von der GUI) bekommen. Thread 3 hingegen terminiert sobald alle Bilder abgearbeitet wurden. Thread 4 terminiert nachdem eine Punktwolke fertig erstellt wurde. Während Thread 4 arbeitet werden weiterhin von Thread 3 weitere Bilder und Posen aus den Eingabedaten gelesen und in der GUI dargestellt. Für diese Posen wird keine Punktrekonstruktion durchgeführt.

Ist die Konfiguration vom Benutzer abgeschlossen, werden als erstes der *FittingThread* des Netzes und der *DemoThread* gestartet, der das Lesen der Eingabedaten steuert. Aus den übergebenen Dateien werden ein Bildpaar und die dazugehörige Pose gelesen. Das Laden der neuen Bilddaten wird der GUI signalisiert, damit diese die neuen Bilder darstellen kann. Für das geladenen Bildpaar wird ein *ReconstructionThread* gestartet. Dieser führt mit *libStereoVision* ein Stereo-Matching auf den Eingabenbildern aus. Das daraus resultierende Disparitätsbild wird in eine Punktwolke umgerechnet, deren Koordinaten von Kamera- in Weltkoordinaten transformiert werden. Ist die Transformation abgeschlossen, wird dies dem *DemoThread* signalisiert. Dieser übergibt die Punktwolke nun dem *AddPointsThread*, der sie dem Netz hinzufügt.

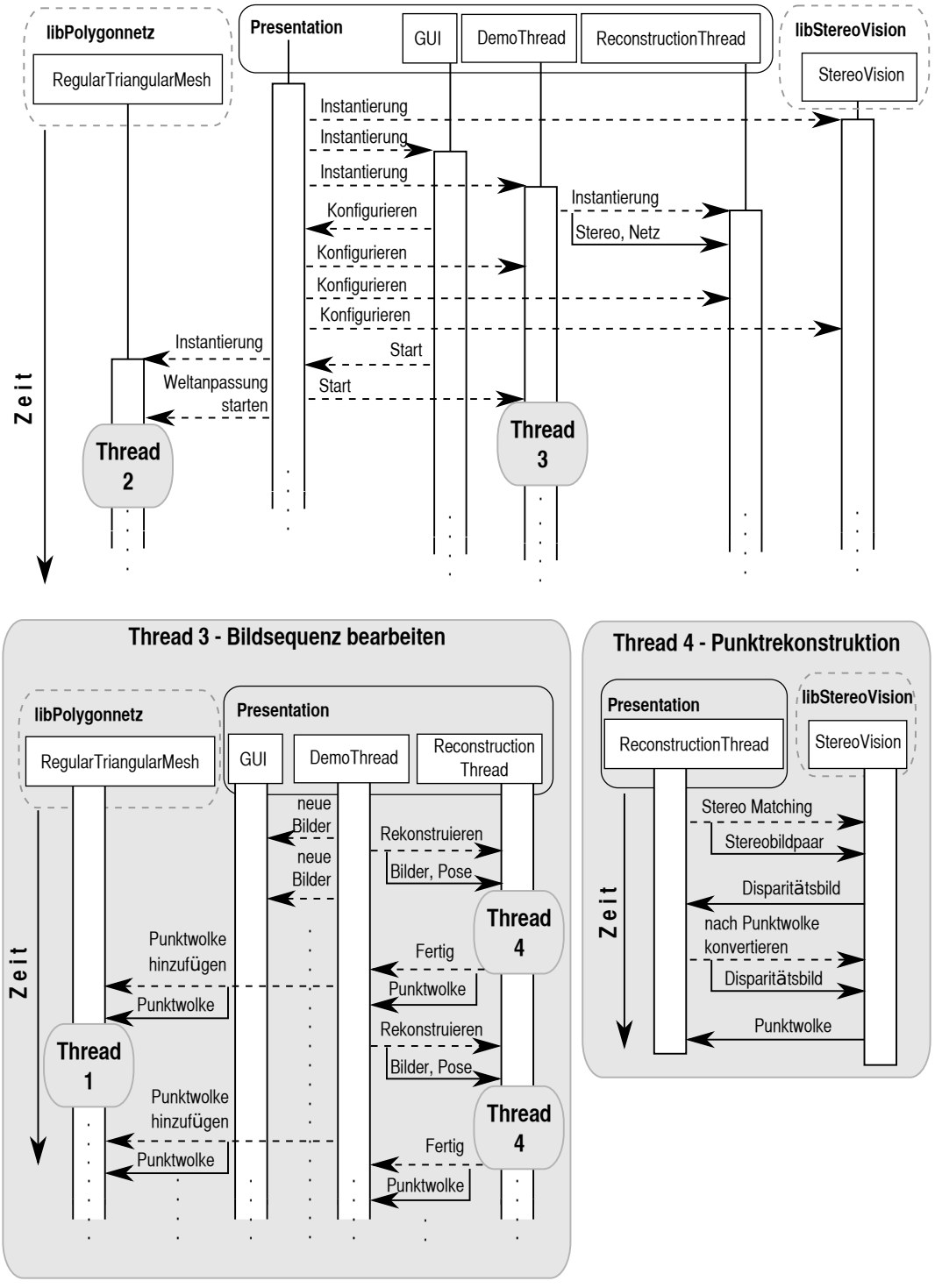


Abb. 5.11.: Daten- und Kontrollfluss des Programms *Presentation*. Es werden die Bibliotheken *libPolygonnetz* und *libStereoVision* benutzt.

6. Evaluation

In den folgenden Unterabschnitten wird zunächst beschrieben, wie künstliche Testdaten erzeugt wurden. Anhand dieser Testdaten wurde die Genauigkeit und Geschwindigkeit der Rekonstruktion der unterschiedlichen Anpassungsverfahren untersucht. Die Ergebnisse dieser Untersuchung werden anschließend vorgestellt. Die Testdaten bestehen aus einem Satz an computergenerierten Stereobildern, die ein künstliches Gelände darstellen. Zu jedem dieser Stereobildpaare wurde die Pose innerhalb des künstlichen Geländes gespeichert. Das Programm *Presentation* wurde mit diesen Daten ausgeführt und die dabei entstandene Punktwolke gespeichert. Die Genauigkeit der Geländerekonstruktion wird anhand dieser Punktwolke evaluiert. Anschließend wird noch ein Experiment zur Echtzeitfähigkeit des Systems durchgeführt, dessen Ergebnisse am Ende dieses Kapitels gezeigt werden. Zunächst wird allerdings die Erstellung der künstlichen Testdaten beschrieben und die Stereo-Matching Algorithmen kritisch betrachtet, die in der Implementierung verfügbar sind.

Da die Lokalisierung im Raum nicht Teil dieser Arbeit ist, werden korrekte Posen an das Programm übergeben. Die Punktwolke, die zur Anpassung des Geländes verwendet wird, weist hingegen Fehler auf, die durch das Stereo-Matching entstehen. Im praktischen Einsatz würde natürlich auch die Information der Pose verrauscht sein. Hier soll aber das in dieser Arbeit entwickelte System isoliert untersucht werden.

6.1. Erstellung von Testdaten

Eine Höhenkarte ist eine zweidimensionale Abbildung einer dreidimensionalen Oberfläche. In einem Graustufenbild entspricht jeder Pixel einer Position auf der Oberfläche und dessen Intesitätswert der Höhe an dieser Position. Geomorph[1] ist ein Programm mit dem Höhenkarten von Geländeoberflächen erstellt werden können. Es wurde mit Hilfe einiger Zeichen- und Effektwerkzeuge dieses Programms eine Höhenkarte mit hoher Auflösung und Farbtiefe erstellt. Abbildung 6.1 zeigt diese.

Für *POV-Ray*[2] (Persistence of Vision Raytraycer), wurde ein Skript geschrieben, das aus der Höhenarte ein virtuelles Gelände erzeugt, durch das eine Kamerafahrt entlang eines vorgegebenen Pfads simuliert wird (siehe Abbildung 6.2). Bei jedem Schritt wird je ein Stereobildpaar gerendert (Abbildung 6.3) und zusammen mit der aktuellen Pose

[1] Geomorph Website: `http://geomorph.sourceforge.net/intro_en.html`
[2] POV-Ray Website: `http://www.povray.org`

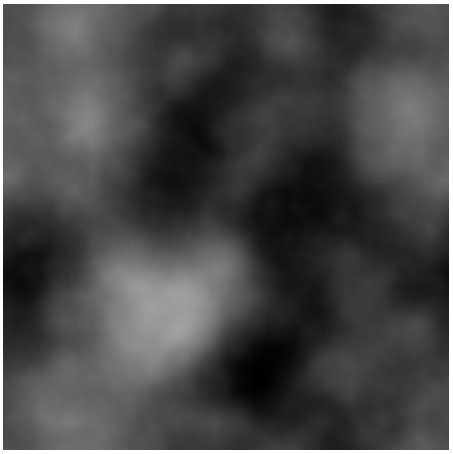


Abb. 6.1.: Höhenkarte Ground Truth

gespeichert, so dass sich eine Bildsequenz von der Kamerafahrt ergibt. Diese Daten wurden als Test- und Evaluierungsdaten genutzt.

Die Abmessungen des künstlichen Geländes sind 8x8 Meter, mit einer maximalen Höhe von 1.592 Meter. Die virtuelle Stereokamera hat die folgenden Eigenschaften:

Bildgröße	Pixelgröße	Brennweite	Öffnungswinkel	Baseline
640x480 Pixel	0.0099 mm	5.5 mm / 555.55 Pixel	H: 59.884° V: 46.729°	10 cm

Die Kamera schwebt stets 0.5 m über dem Grund und blickt in einem Winkel von 20° nach unten.

6.2. Evaluation

Die Evaluation der Geländerekonstruktion basiert auf einer Punktwolke, die mit dem Programm *Presentation* erstellt wurde. Das Programm benutzt Stereo-Matching um aus unterschiedlichen Posen Punktwolken von lokalen Ansichten des Geländes zu generieren. Diese werden dann dem Weltmodell hinzugefügt. Aus dem Weltmodell wurde die gesamte Punktwolke aller Posen extrahiert und für die Evaluierung benutzt. Bevor die Evaluierung der Geländerekonstruktion durchgeführt wird, werden zuvor noch die Stereo-Matching Algoirithmen untersucht. Dadurch sollen bestimmten Eigenarten der Evaluierungsdaten identifiziert und verstanden werden. Außerdem werden die Stereo-Matching Algorithmen auf ihre Geschindigkeit hin untersucht.

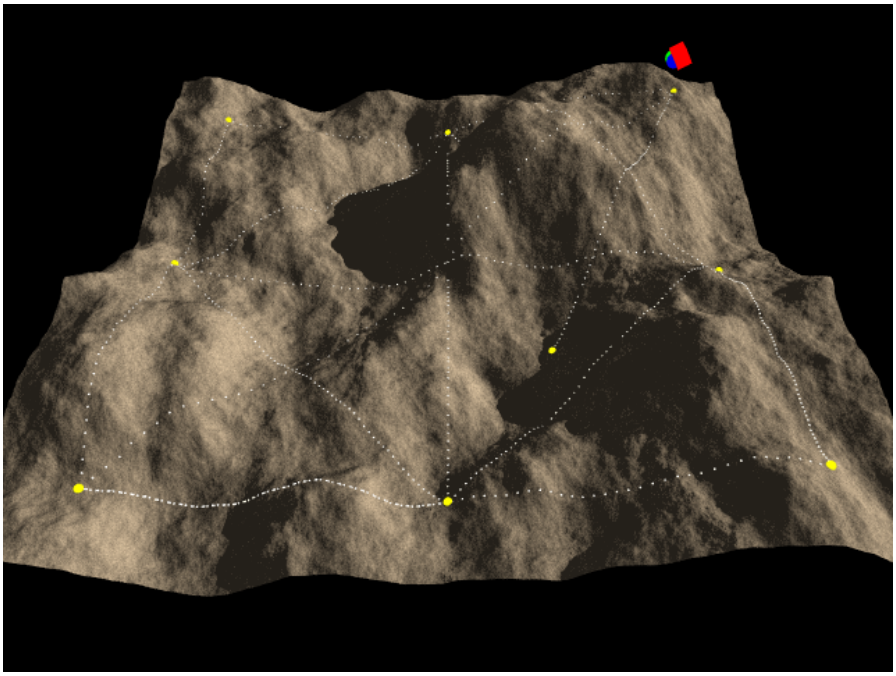


Abb. 6.2.: Das künstliche Gelände, das als Grundlage für die Rekonstruktion dient, aus der Vogelperspektive. Der Pfad, entlang dem eine virtuelle Kamerafahrt simuliert wird, um die Testdaten zu erzeugen, ist gepunktet in weiß eingezeichnet. Die gelben Punkte sind Ankerpunkte, anhand derer der Pfad erstellt wird. Der rote Kasten mit den Halbkugeln ist die virtuelle Stereokamera an ihrer derzeitigen Position.

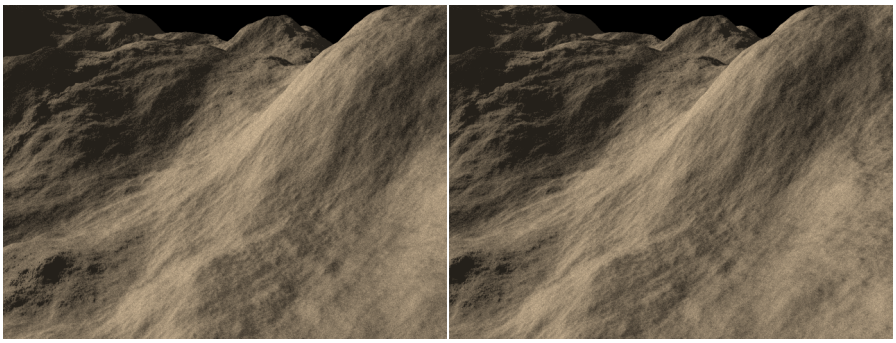


Abb. 6.3.: Ein Stereobildpaar aus der Bildsequenz einer virtuellen Kamerafahrt durch ein künstliches Gelände. Das rechte und linke Bild unterscheiden sich durch eine geringfügig verschobene Perspektive.

	BM	GPU	Birch	GC
Bildpaar 1	0.435 s	1.865 s	1.025 s	211.052 s
Bildpaar 2	0.317 s	1.843 s	1.087 s	230.850 s

Tab. 6.1.: Vergleich der Geschwindigkeit unterschiedlicher Stereo-Matching Algorithmen

6.2.1. Stereo-Matching - Punktrekonstruktion

Beim Stereo-Matching werden Pixelkorrespondenzen gesucht und diese auf einen einwertigen Disparitätswert abgebildet. Für diese Arbeit sind folgenden Faktoren wichtig:

- Geschwindigkeit: Wie schnell ist der Algorithmus in der Lage, Ergebnisse zu liefern?
- Fehleranfälligkeit: Wie wahrscheinlich ist es, dass gefundene Korrespondenzen fehlerhaft sind?
- Rekonstruktionsgenauigkeit: Mit welcher Genauigkeit lassen sind Punkte aus gefundenen Disparitäten rekonstruieren?

Die Algorithmen *BM* (Block Matching aus OpenCV), *GPU* (Block Matching auf GPU), *Birch* (Dynamic-Programming-basiert) und *GC* (Graph Cuts basiert) sollen in diesem Abschnitt auf diese Merkmale hin untersucht werden. Für eine Beschreibung der Algorithmen wird auf Abschnitt 3.3 verwiesen.

Geschwindigkeit

Um die Geschwindigkeit zu messen, wurden zwei Bildpaare aus den Testdaten (Abschnitt 6.1) in die Software *testStereo* geladen. Hier wurde ein Software-Timer eingebaut, der die Zeit misst, die der Algorithmus benötigt, um eine Distaritätsbild zu erstellen. Tabelle 6.1 zeigt die Ergebnisse. Es wurde ein Apple MacBook mit folgender Hardware-Austattung benutzt:

- Prozessor: Intel Core 2 Dou mit 2.4 GHz
- 2 GB Arbeitsspeicher
- Grafikkarte: NVIDIA GeForce 9400M, 256 MB VRAM
- SATA Festplatte

Die benutzten Bilder hatten die Auflösung 640x480 Pixel mit einem Farbkanal. Der Disparitätsbereich war von 0 bis 192 für *BM*, *GPU* und *Birch* gesetzt. Für *GC* von 0 bis 99. Für *GPU* war der Disparitätsschritt auf einen Pixel gesetzt.

In voller Auflösung und dem gewählten recht großen Disparitätsbereich leistet keiner der benutzten Algorithmen “Echtzeit”-Performance im Sinne von Videogeschwindigkeit, also schneller als 20 Bilder pro Sekunde. Ein deutlicher Geschwindigkeitsgewinn lässt sich erreichen, indem Auflösung und Disparitätbereichs verkleinert werden.

Rekonstruktionsgenauigkeit

Die Disparitätswerte liegen in der Regel in einem diskreten, ganzzahligen Wertebereich. Bei der Rekonstruktion von 3D-Punkten aus diesen diskreten Disparitätswerten sinkt die Genauigkeit eines rekonstruierten Punktes stark mit sinkender Disparität. Abbildung 6.4 zeigt ein Diagramm, das zu Disparitätswerten zwischen 1 und 30 die Entfernung des Punktes von der Kamera (Kameraparameter wie in Tabelle 6.1) entlang der optischen Achse abbildet.

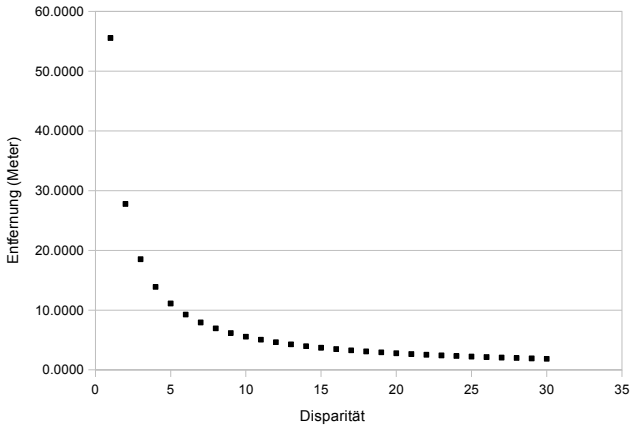


Abb. 6.4.: Mit sinkendem Disparitätswert steigt die Entfernung zur Kamera stark.

Der Disparitätsschritt von 1 bis 2 überbrückt 27.78 m. Ab einer Disparität von über 75 liegt die Entfernung zwischen den Disparitätsschritten unter einem Zentimeter. Der minimale Disparitätswert der in den 8x8 m großem Terrain zu erwarten ist, ist etwa 5. Hier entspricht die Entfernung zur Kamera ca. 11 Metern, der maximalen, diagonalen Entfernung im Terrain. Im schlechtesten Fall kommt es also, selbst bei einem korrekten Matching, in der z-Komponente zu einer Abweichung von 1.25 m.

Fehleranfälligkeit

Eine andere Fehlerquelle sind falsch gefundene Korrespondenzen. Die Stereo-Algorithmen *GPU* und *BM* haben allerdings Parameter mit denen, im Anschluß an das Matching Fehler gefunden und recht gut herausgefiltert werden können. Der Graph Cuts und der auf Dynamic-Programming basierende Algorithmus benutzten eine Glattheitsbedingung, mit deren Hilfe ein möglichst homogenes Disparitätsbild erzeugt werden soll. So werden auch in Bereichen, in denen kein erfolgreiches Matching durchgeführt werden konnte, zwischen bekannten Werten interpoliert. Im schlechten Fall führt dies zu stellenweise dichten Fehlinformationen. Dieses Verhalten erweist sich als Problematisch für den Anwendugszweck in dieser Arbeit. Eine Interpolation bei fehlenden Daten soll von

der Glattheitsbedingung in der Anpassung des Weltmodells durchgeführt werden. Durch die hohe Anzahl an Fehlinformationen würde dies verhindert werden.

In Abbildung 6.5 werden zu einem Bildpaar aus den Testdaten die Disparitätsbilder der einzelnen Algorithmen gezeigt. Je heller ein Bildpunkt in diesen Bildern ist, desto höher ist die Disparität, und desto näher ist der Weltpunkt an der Kamera. Die Eingabebilder zeigen ein Gelände aus einer Perspektive, die eine recht weite Sicht aufweist. Es gibt aber viele Schatten, die in Regionen ohne identifizierbare Merkmale im Bild führen. Hier kann man sehen, dass *BM* und *GPU* in diesen Regionen weitestgehend auch keine Annahmen über die Struktur treffen. Diese Bereiche bleiben leer. *Birch* und *GC* hingegen füllen große Teile dieser Bereiche anhand von Annahmen, die aus den umliegenden Bereichen getroffen werden. Diese Annahmen sind zum Teil falsch, wie z.B. im unteren Bildausschnitt. Auch der Hintergrund wird mit einem konstanten Disparitätswert aufgefüllt, was zu falsch rekonstruierten Punkten führt. Verglichen mit *BM* hat *GPU* mehr falsche Korrespondenzen gefunden. Diese sind jedoch auf relativ kleine Bereiche im Bild beschränkt.

Ein weiteres Bildpaar und dazu erstellte Disparitätsbilder sind in Abbildung 6.6 dargestellt. Hier ist die Sicht von einem recht nahe gelegenem Hügel versperrt. Alle Algorithmen schneiden hier recht gut ab, nur bei *BM* ist zu erkennen, dass deutlich weniger Korrespondezen hergestellt werden konnten. Die Algorithmen *Birch* und *GC* füllen auch hier wieder den Hintergrund mit einem konstanten Wert auf.

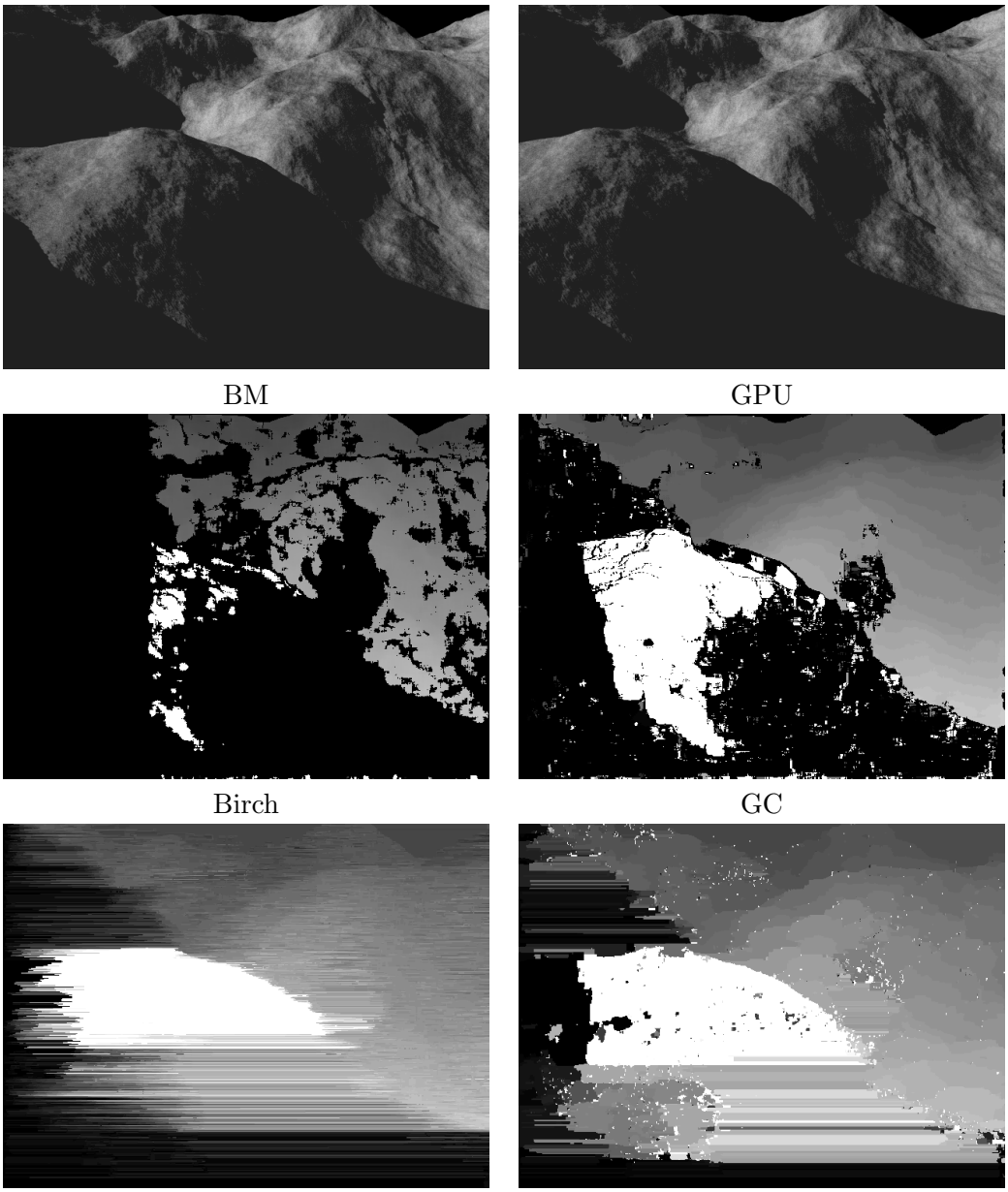


Abb. 6.5.: Disparitätsbilder unterschiedlicher Stereo-Matching-Algorithmen im Vergleich. Die oberste Zeile zeigt die Eingabebilder.

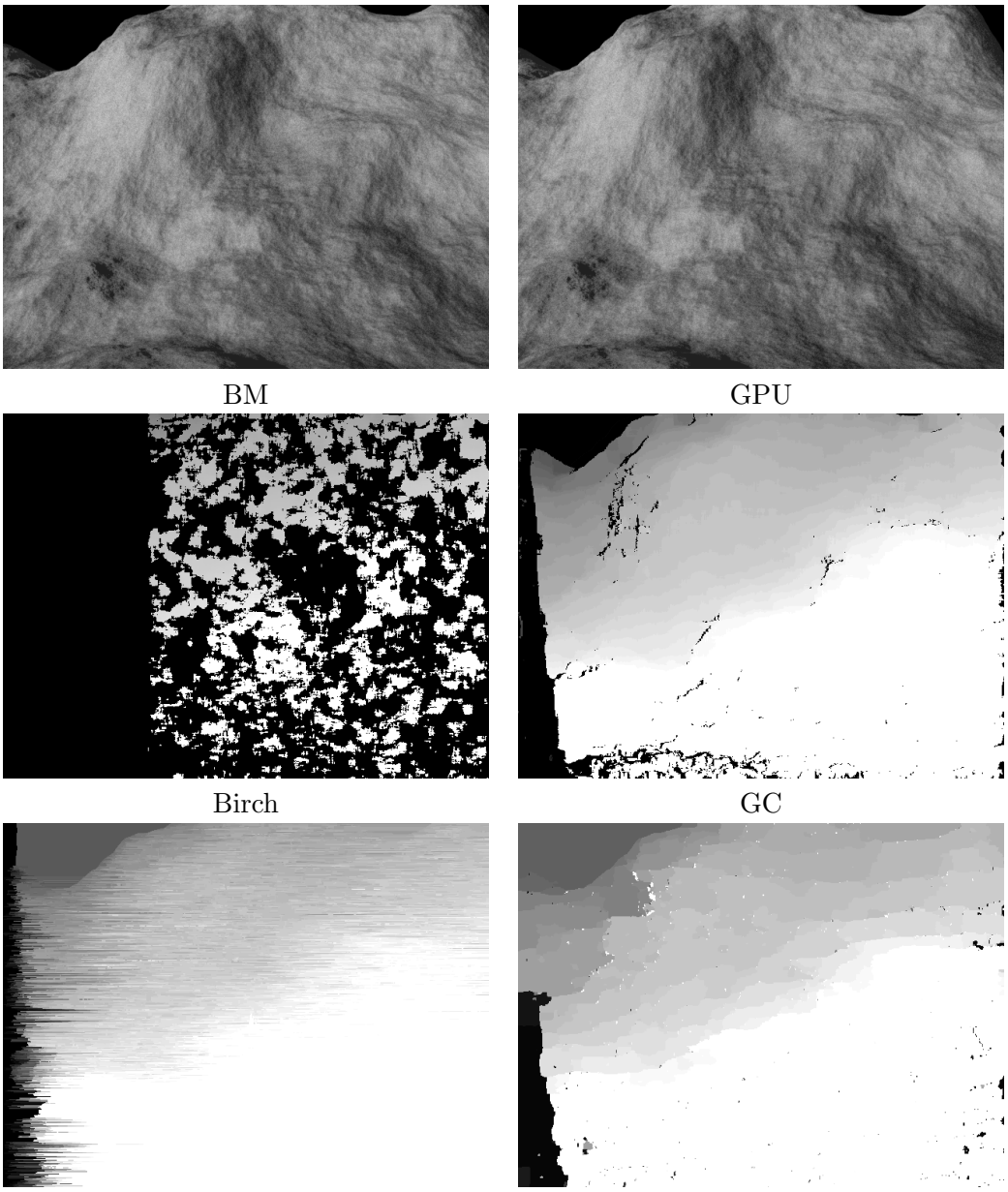


Abb. 6.6.: Disparitätsbilder unterschiedlicher Stereo-Matching-Algorithmen im Vergleich. Die oberste Zeile zeigt die Eingabebilder.

6.2.2. Terrainrekonstruktion

Im vorherigem Unterabschnitt wurde das Stero-Matching untersucht. Es wurden mögliche Urasachen für Fehler in der Punktwolke identifiziert. In diesem Unterabschnitt soll nun die Geländerekonstruktion aufgrund solcher verrauschter Daten untersucht werden. Als Grundlage der Evaluierung dient eine Punktwolke, die mit dem Programm *Presentation* erzeugt wurden. Ein Weltmodell wurde daran angepasst und dabei die Geschwindigkeit mit Software-Timern gemessen. Anschließend wurde die geometrische Genauigkeit der Rekonstruktion, anhand eines Vergleiches mit der Höhenkarte, die zur Erzeugung der Testdaten diente, gemessen.

Als Testdaten wurde eine Bildsequenz von 6537 Bildpaaren mit zugehörigen Posen in *Presentation* erstellt. Jedes 30. Bildpaar wurde für die Rekonstruktion mit *GPU* genutzt und die resultierenden Punktwolken jeweils um den Faktor 25 verkleinert. D.h. es wurde nur jeder 25te gültige Pixel aus den Disparitätsbildern in die endgültige Punktwolke übernommen. Dies resultierte schließlich in 2561672 Punkten. Ein Video von der Erstellung der Punktwolke ist auf der DVD unter *Videos/Erstellung_der_Testdaten.avi* zu finden. Die verwendeten Parameter für *GPU* waren:

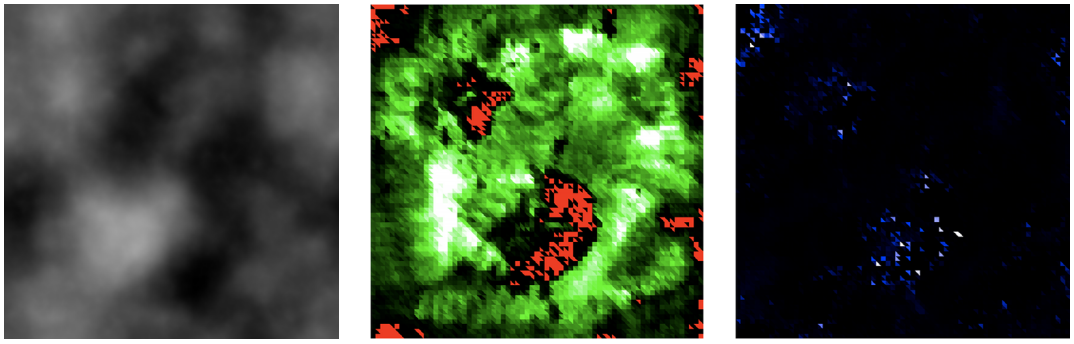
- Minimale Disparität: 0
- Maximalie Disparität: 192
- Disparitätsschrittweite: 0.5
- Fenstergröße: 10
- Cross Correlation Schwellwert: 1.5

Im folgenden Unterabschnitt wird die erstellte Punktwolke analysiert. Anschließend wird die Auswirkung des Glättungsparameters α untersucht. Mit einem guten Wert für α werden in den beiden darauf folgenden Unterabschnitten, Geschwindigkeit und Rekonstruktionsgenauigkeit der unterschiedlichen Methoden zur Weltmodellanpassung verglichen. Schließlich wird ein Experiment beschrieben, in dem der Echtzeiteinsatz des Systems überprüft wird.

Analyse der Punkwolke

In einem Netz mit den Ausmaßen von 8x8 Meter mit 70x70 Ankern ergeben ergibt sich im Mittel etwa 270 Punkte pro Facette und 523 pro Anker. Innerhalb der Facetten weisen die Punkte eine mittlere Standardabweichung von etwa 3,7 cm auf. Die höchste gemessene Standardabweichung innerhalb einer Facette liegt bei 2,14 m. Abbildung 6.8(a) zeigt die relative Belegung der einzelnen Facetten mit Punkten. (b) zeigt die Standardabweichung der Punkte der einzelnen Facetten. Zum Vergleich ist die Höhenkarte, die den Testdaten zu Grunde liegt, erneut mit abgebildet.

Stark unterbelegte Regionen können entweder daher kommen, dass diese Bereiche während der Kamerafahrt durch das künstliche Gelände nicht betrachtet wurden, oder



(a) (b)

Abb. 6.7.: (a) relative Belegung der Kacheln mit Punkten. Rot bedeutet gar keine Punkte. Schwarz: unterbelegt, Grün: Normal belegt, Weiß: Überbelegt. (b): mittlere Standardabweichung zwischen den Punkten einer Kachel. Je heller, desto höher die Standardabweichung.

aber dass eine Punktrekonstruktion hier nicht durchgeführt wurde konnte. Beide Fälle treten bei den Testdaten auf. Von den äußersten Ecken des Geländes wurden tatsächlich keine Bilder erzeugt. Die großen Bereiche in der Geländemitte, in denen keine Punkte vorhanden sind, sind hingegen Schattenseiten von Hügeln, an denen keine Strukturen zu erkennen sind. An diesen Stellen hat das Stereo-Matching fehlgeschlagen. Man kann eine Korrelation zwischen unterbelegten Regionen und steigender Standardabweichung der Punkte erkennen. Die starken Ausschläge in der Standardabweichung sind Rauschen, das zufällig, eigentlich leeren Facetten zugeordnet wurde. Bei stark belegten Kacheln fällt das Rauschen dagegen kaum ins Gewicht.

Auswirkung der Glättung

Ausgehend von den Daten, die im letzten Absatz beschrieben wurden, wurde zunächst die Auswirkung des α-Werts untersucht. Der Fitting-Algorithmus für den quadratischen Fehler in z-Richtung wurde mit unterschiedlichen Werten für α (0.2, 3, 30 und 99) ausgeführt. Bei Konvergenz wurde aus dem Weltmodell eine Höhenkarte erzeugt, die dann mit der Ground Truth verglichen wurde. Eine grafische Darstellung der Ergebnisse ist in Abbildung 6.8 zu sehen. Die linke Spalte zeigt die erzeugte Höhenkarte aus der Rekonstruktion. Die mittlere Spalte zeigt die pixelweise Differenz zu der Ground-Truth. Weiß bedeutet hier keine Differenz. Der Farbbereich Rot bis Gelb zeigt an, dass das rekonstruierte Gelände an dieser Stelle zu niedrig ist. Der Farbbereich Blau bis Türkis zeigt an, dass das rekonstruierte Gelände an dieser Stelle zu hoch ist. Eine quantitative Auswertung ist in Tabelle 6.2 für die Werte $\alpha = 3$ und $\alpha = 30$ gegeben.

Bei kleinerer Gewichtung der Glättungsfunktion ist die Optimierung störungsanfälliger für fehlerhafte oder fehlende Punkte. Dies resultiert darin, dass in diesen Regionen hohe

	z-Abstand $\alpha = 3$			z-Abstand $\alpha = 30$		
	Max. Dif.	Mtl. Dif.	Mtl. Std.ab.	Max. Dif.	Mtl. Dif.	Mtl. Std.ab.
g	17963	489.98	497.69	6288	742.382	594.719
m	0.685	0.0186	0.019	0.2398	0.0283	0.0226

Tab. 6.2.: Vergleich einer aus dem Weltmodell rekonstruierten Höhenkarte mit der Ground Truth. Zum intuitiveren Verständnis wurden die verglichenen Grauwerte (g) in Meter (m) umgerechnet. Diese Daten beziehen sich auf die Bilder in Abbildung 6.8.

Fehlerwerte auftreten können, die bei stärkerer Glättung relativiert werden würden. So reduziert sich die maximale Abweichung bei $\alpha = 30$ um fast einen halben Meter, im Vergleich zu $\alpha = 3$. Jedoch gehen bei stärkerer Glättung auch Details verloren, die möglicherweise korrekt rekonstruiert wurden. Vergleicht man etwa die zweite und dritte (von oben) Höhenkarten in Abbildung 6.8 miteinander fällt auf, dass die mit dem geringeren α-Wert eine feinere Struktur aufweist.

Der Wert $\alpha = 3$ hat in dieser Testreihe mit einer Mittleren Differenz von umgerechnet 0.0186m besonders gut abgeschnitten. Also wurde dieser Wert für den weiter unten beschriebenen Vergleich der unterschiedlichen Anpassungsalgorithmen übernommen.

Laufzeit und Konvergenz

Um die Geschwindigkeit der Geländeanpassung zu messen, wurde mit Software-Timern gemessen, wie lange jeder der Anpassungs-Algorithmen benötigt, um einmal alle 4900 Anker an die etwa 2.5 Millionen Punkte große Punktwolke anzupassen. Außerdem wurde gemessen, wie viele Iterationen und wie viel Zeit jeweils benötigt wurde, bis das Modell konvergiert. Als Konvergenzkriterium wurde gewählt, dass kein Anker in einer Iteration mehr als 1 mm bewegt wurde. Die Ergebnisse sind in Tabelle 6.3 aufgelistet. Es werden für die Algorithmen fogende Abkürzungen verwendet:

- Sq. z: Basierend auf quadratischem Abstand in z-Richtung
- Sq. o: Basierend auf quadratischem Abstand in orthogonaler Richtung
- Abs. z: Basierend auf absolutem Abstand in z-Richtung
- Abs. o: Basierend auf absolutem Abstand in orthogonaler Richtung

Der mit Abstand schnellste Algorithmus is *SimpleFit*. Der Grund dafür ist, dass nicht alle Punkte für diesen Algorithmus berücksichtigt werden, sondern lediglich eine Stichprobe, die aus den Punkten zufällig gewählt wird. Von den restlichen Algorithmen setzt sich *Sq. z* besonders ab. Das war zu erwarten, da hier eine analytische Lösung verwendet wird, während bei den verbleibenden iterativ vorgegangen wird.

	Zeit 1 Iter.	pro Anker	Iter.	Zeit gesamt
Sq. z	4.1 s	0.837 ms	50	28.969 s
Sq. o	21.355 s	4.358 ms	52	180.953 s
SimpleFit	0.613 s	0.125 ms	konvergiert nicht	
Abs. z	45.550 s	9.296 ms	89	256.067 s
Abs. o	46.624 s	9.515 ms	62	310.598 s

Tab. 6.3.: Zeit für eine Iteration der Geländeanpassung über alle Anker.

	Sq. z $\alpha = 3$			Sq. o $\alpha = 3$		
	Max. Dif.	Mtl. Dif.	Mtl. Std.ab.	Max. Dif.	Mtl. Dif.	Mtl. Std.ab.
g	17963	489.98	497.69	20756	497.259	524.935
m	0.6852	0.0186	0.0189	0.7918	0.0189	0.0199
	SimpleFit $\alpha = 3$			Abs. z $\alpha = 3$		
	Max. Dif.	Mtl. Dif.	Mtl. Std.ab.	Max. Dif.	Mtl. Dif.	Mtl. Std.ab.
g	20953	482.929	442.511	7366	400.862	439.554
m	0.799	0.0184	0.0169	0.280991	0.0152	0.0167

	Abs. o $\alpha = 3$		
	Max. Dif.	Mtl. Dif.	Mtl. Std.ab.
g	5970	388.49	409.293
m	0.2277	0.0148	0.0156

Tab. 6.4.: Vergleich einer aus dem Weltmodell rekonstruierten Höhenkarte mit der Ground Truth. Zum intuitiveren Verständnis wurden die verglichenen Grauwerte (g) in Meter (m) umgerechnet. Diese Daten beziehen sich auf die Bilder in Abbildung 6.8

Rekonstruktionsgenauigkeit

Um die Rekonstruktionsgenauigkeit der einzelnen Algorithmen zu testen, wird vorgegangen wie schon bei der Ermittlung von α. Für jeden Algorithmus wird das Weltmodell an die Punktwolke angepasst. Aus dem resultierenden Weltmodell wird eine Höhenkarte erzeugt und diese mit der Ground Truth verglichen. Die Ergebnisse sind in Tabelle 6.4 aufgelistet.

Der mittlere Fehler ist bei allen Algorithmen zwischen 1 und 2 Zentimetern. Die robusten Algorithmen *Abs. z* und *Abs. o* schneiden etwas besser ab als die nicht robusten Versionen der Fehlermaße. Von dieser Robustheit profitiert auch *SimpleFit*. Dieser Algorithmus konvergiert allerdings im Gegensatz zu den anderen nicht. Das Ergebnis hängt also maßgeblich von der momentan gewählten Stichprobe aus der Punktwolke ab. Darunter leidet auch der optische Eindruck der Geländeanpassung. So wirkt das Gelände

unruhig (vergleiche Video auf DVD, unter *Videos/BM-SimpleFit-7000_frames.avi*[3]), da sich stets alle Anker leicht bewegen. Bei den anderen Algorithmen konvergieren sie zum Minimum und verbleiben dann in dieser Position.

[3]Online unter `http://vimeo.com/14634360` (Stand: 02.09.2010)

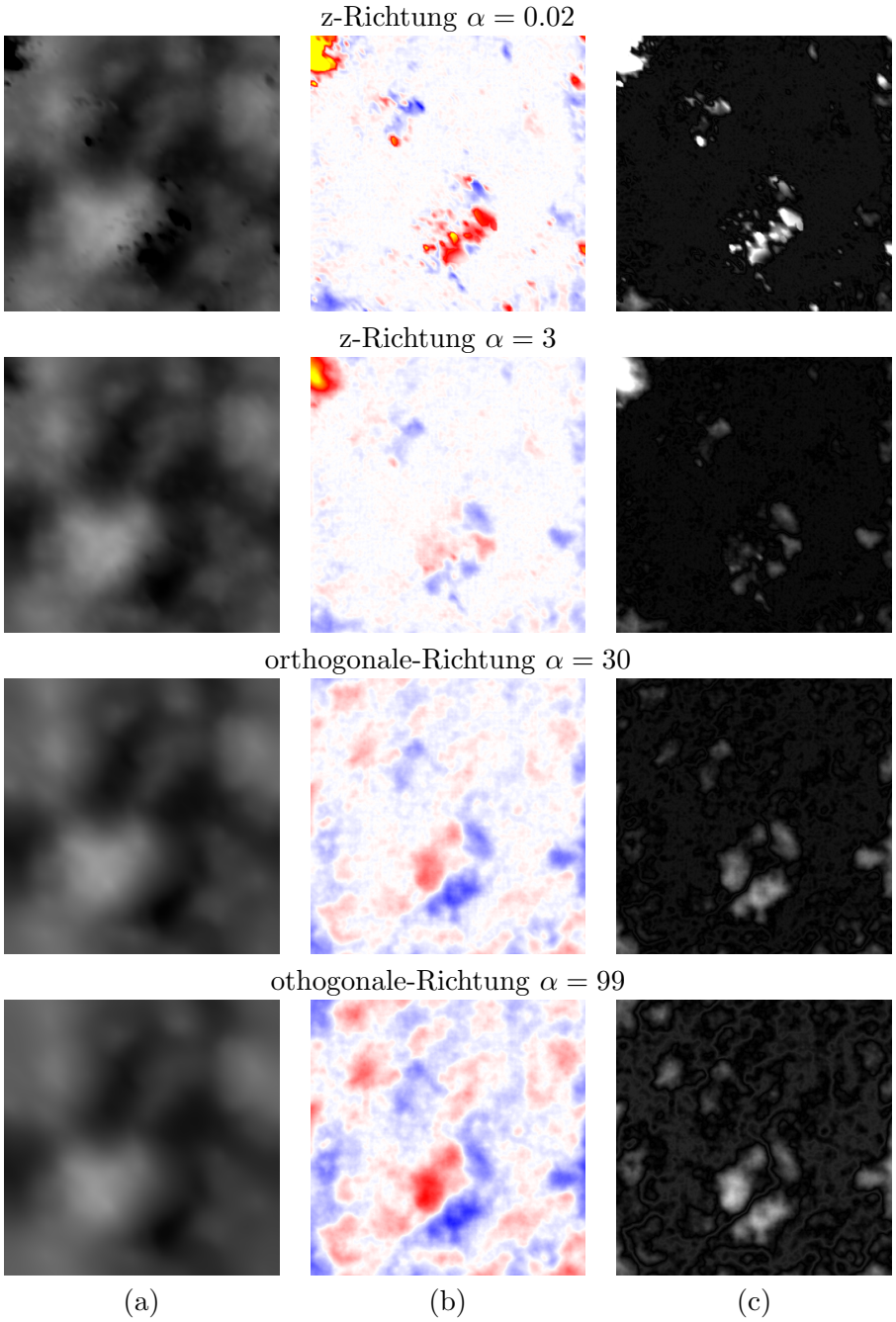


Abb. 6.8.: Vergleich einer aus dem Weltmodell rekonstruierten Höhenkarte (Spalte (a)) mit der Höhenkarte, mit derjenigen, aus der die Testdaten erzeugt wurden. Die Differenz, gemessen in Grauwerten, wird in Spalte (b) gezeigt. Rot bis Gelb zeigt, dass das Weltmodell an dieser Stelle zu tief, Blau, dass es zu hoch ist. Spalte (c) zeigt die Standardabweichung in Graustufen. Eine numerische Auswertung der Bilder ist in Tabelle 6.2 gegeben.

Echtzeiteinsatz

In den vorherigen Absätzen wurden stets isolierte Aspekte der Geländerekonstruktion gemessen. So wurde festgestellt, dass *BM* der schnellste Stereo-Algorithmus ist, aber nicht immer so dichte Distparitätsbilder erzeugt wie die anderen Algorithmen. Dafür werden aber auch die meisten fehlerhaften Korrespondenzen herausgefiltert. Im Gegensatz dazu trifft der zweitschnellste Algorithmus, *Birch*, annahmen über Bildregionen, über die er keine direkten Informationen gewinnen kann. Dies kann zu stellenweise dichten, falsch rekonstruierten Punkten führen, was unerwünscht ist. Es wurden unterschiedliche Werten für den Glättungsfaktor untersucht, um eine gute Parametrisierung der Anpassungsalgorithmen zu finden. Die Algorithmen wurden auf ihre Laufzeit und Rekonstruktionsgenauigkeit untersucht. Hier stellte sich heraus, dass der schnellste Algorithmus der heuristische *SimpleFit* ist. Gefolgt von *Sq. z* (quadratischer Abstand in z-Richtung) und dann, recht weit abgeschlagen, denjenigen die eine iterative Funkitonsminimierung für jeden Anker vornehmen. Allderings zeigte sich, dass die iterativen Algorithmen *Abs. o* (absoluter Abstand in orthogonaler Richtung) gefolgt von *Abs. z* (absoluter Abstand in z-Richtung) die beste Rekonstruktionsgenauigkeit liefern.

Für den Echtzeiteinsatz scheint sich bei den Stereo-Algorithmen am besten *BM* zu eigenen. Für die Geländeanpassung empfiehlt sich von der Laufzeit her *SimpleFit* und *Sq. z*. Ein Testlauf in der Software *Presentation*, die alle Komponenten der Rekonstruktion in teilweise parallel laufenden Threads vereint, soll nun prüfen, in wie weit eine Echtzeit-Nutzung des Systems möglich ist.

Die Software wurde wieder mit Timern in den unterschiedlichen Komponenten ausgestattet. Es wurden folgenden Einstellungen gewählt:

- Weltmodell: 70x70 Anker
- Punktekonstruktion: *BM*, Bilder 640x480 Pixel, Punktwolkenreduzierung 50
- Weltanpassung: *Sq. z*, $\alpha = 3.0$

Der Testdatensatz, also die 6537 Stereobildpaare und korrespondierenden Posen, wurden, mit einer durchschnittlichen Geschwindigkeit von knapp 10 Bildern pro Sekunde, in *Presentation* abgearbeitet. Bei dieser Geschwindigkeit konnte von 670 Posen eine Punktrekonstruktion durchgeführt werden, was im Durchschnitt etwa jedem zehnten Bild entspricht. Pro Pose, in der eine Punktrekonstruktion durchgeführt wurde, wurde im Schnitt eine Punktwolke von 1698 Punkten erzeugt. Eine Iteration der Weltanpassung dauerte im Schnitt 62.5 ms. Pro Iteration wurden noch Statistiken errechnet, was im Schnitt 143 ms dauerte. Insgesamt wurde jeder Anker 3314 Mal angepasst.

Die im Schnitt deutlich schnellere Anpassung des Weltmodells gegenüber den Daten aus Tabelle 6.3 ist dadurch begründet, dass stets nur ein kleiner Teil der Anker neu angepasst werden muss (vgl. Abbildung 5.7). Die Daten aus Tabelle 6.3 beziehen sich auf den Fall in dem, alle Anker des Netzes angepasst werden müssen.

Aus technischen Gründen konnten von genau diesem Experiment keine Screenshots gemacht werden. Von einem vergleichbaren Durchlauf, mit geringerer Bildrate, zeigt Abbildung 6.9 zwei Screenshots. Einer stammt aus der Anfangs- und der andere aus der Endphase. Es ist jedoch ein mit Kamera gefilmtes Video des Experiments, mit voller Bildrate auf der beigelegten DVD unter *Videos/BM-SqrZ-7000-Echtzeit.mov* zu finden. In dem Video sieht man, dass für die recht langsame Bewegung der Kamera durch das virtuelle Gelände die erzielte Geschwindigkeit ausreichend ist. In den ca. 10 Frames zwischen zwei Posen, in denen Stereo-Matching durchgeführt wird, ändert sich das überblickte Terrain nicht wesentlich. Die Geländeanpassung nähert sich bei neuen Daten in vorher unbeobachteten Regionen recht schnell an diese an. Die parallele Implementierung der einzelnen Module erlaubt eine konstante, visuelle Repräsentation des bisher erforschten Geländes aus beliebigen Kameraperspektiven.

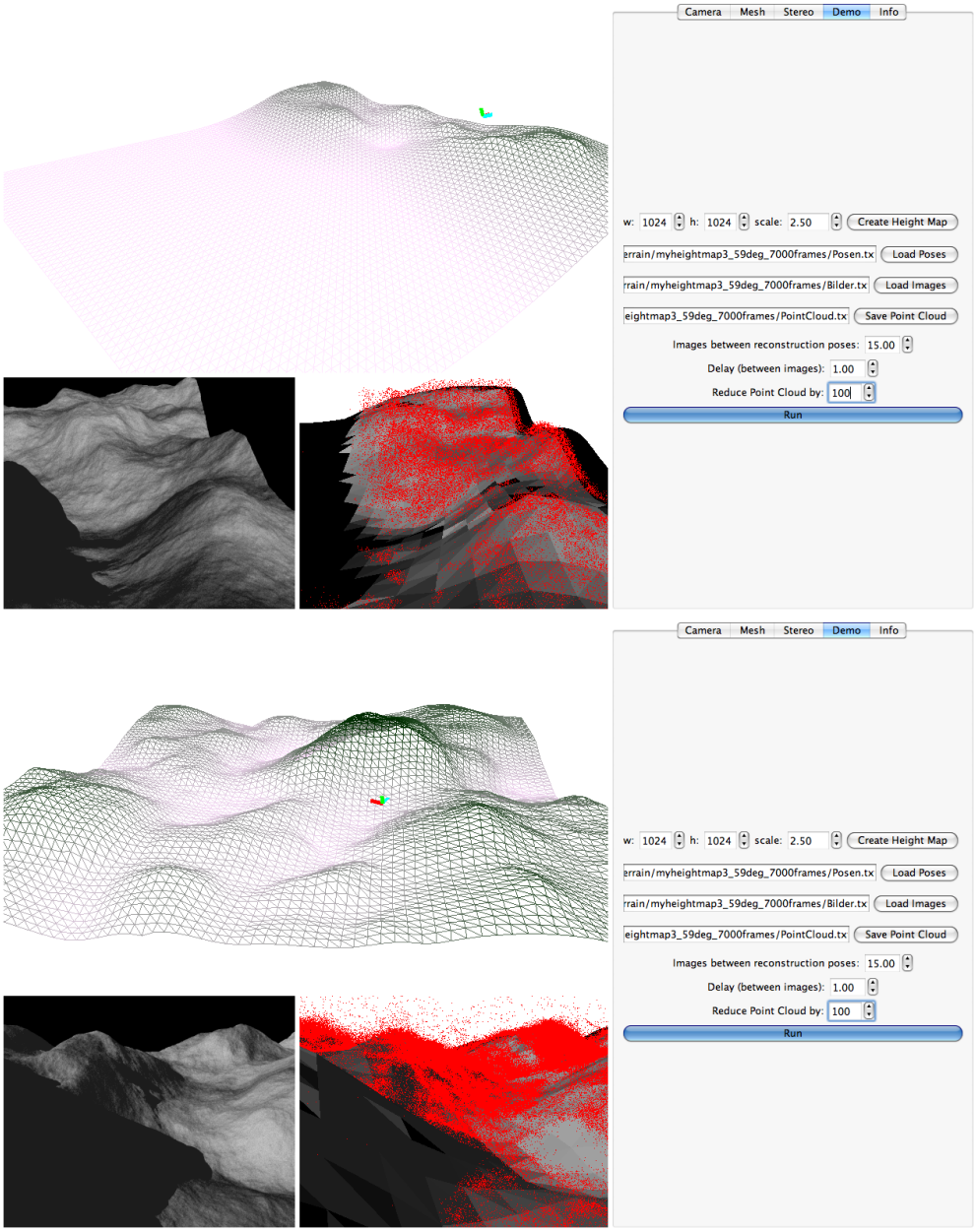


Abb. 6.9.: Das Programm *Presentation* während der Ausführung mit künstlichen Testdaten zu unterschiedlichen Zeitpunkten. Oben wird eine Vogelperspektive auf das Weltmodell angezeigt. Links unten das derzeitige Eingabebild. Rechts unten wird das Weltmodell in der derzeitigen Pose, mit eingefärbten Kacheln und der Punktwolke in rot dargestellt.

7. Fazit

Es wurde ein System entwickelt, mit dem sich ein Weltmodell an eine Punktwolke anpasst. Diese Punktwolke wird aus mehreren Bildern, aufgenommen durch eine kalibriertes Stereokamerasystem, gewonnen. Hierbei wird das zu rekonstrierende Gelände aus mehreren Blickwinkeln betrachtet. Für jede dieser Posen wird mittels Stereo-Matching eine Punktwolke generiert. Die Punktwolke aller Posen werden zusammen in einem Weltmodell integriert. Das Weltmodell ist ein regelmäßiges Dreiecksnetz. Unterschiedliche Methoden zur Anpassung des Weltmodells an die Punktwolke wurden erarbeitet und verglichen.

Die Evaluation zeigte, dass mit guter Geschwindigkeit und hoher Präzision ein statisches Terrain nachgebildet werden kann. Stereobilder in einer Auflösung von 640x480 und in Graustufen können mit eine Geschwindigkeit von etwa 1Hz als Punktwolke im Weltmodell registriert werden.

Die Implementierung des Systems lagert unterschiedliche Aufgaben in Threads aus. Dies erlaubt eine deutlich schnellere Datenverarbeitung. Ein Experiment zur Echtzeitfähigkeit des Systems zeigte, dass die erreichte Punktrekonstruktionsgeschwindigkeit für einen Datenstrom von 10 Bildern pro Sekunde ausreicht, wenn eine gleichmäßige, eher langsame Bewegung von der Kamera beschrieben wird. Durch eine Optimierung der Implementierung kann die Geschwindigkeit noch weiter verbessert werden. Zum Beispiel könnte die Anpassung der einzelnen Anker parallelisiert werden. Aber auch die GUI-Implementierung, für das *Presentation* Programm, kann deutlich schlanker und effizienter gestaltet werden.

Prinzipiell sollte über eine Methode zur Reduzierung der Punktanzahl nachgedacht werden, da diese das System verlangsamt. Eine Art "Aufräummechanismus", der Punkte in überbelegten Zellen zusammenfasst und gegebenfalls dann stärker gewichtet, wäre eine Idee in diese Richtung. Die Punkte von Zellen, die eine sehr hohe Standardabweichung aufweisen, könnten geringer gewichtet oder gar gelöscht werden, damit diese unsicheren Informationen weniger stark einbezogen werden.

Ein Mechanismus, der rekonstruierte Punkte miteinander vergleicht, würde helfen, korrekte und inkorrekten Punkte zu erkennen. Würde ein Punkt aus zwei Posen, in denen er sichtbar ist, gleich rekonstruiert werden, könnte er eine höhere Gewichtung bekommen. Ist es nicht möglich, einen Punkt aus einer zweiten Pose zu bestätigen, würde seine Gewichtung entsprechend sinken.

Die Wahl der Repräsentationsform des Systems bringt eindeutige Beschränkungen mit sich. Zum einen wird nicht zwischen Terrain und Objekten, die sich auf dem Terrain

befinden könnten, unterschieden. Außerdem würde die 2.5D-Repräsentation in realen Szenen schnell an ihre Grenzen stoßen: Das Weltmodell erlaubt für einen Punkt auf der Weltmodellebene nur einen Höhenwert. Für aus Höhenkarten generierte Terrains ist dies natürlich ausreichend, da solche Terrains ebenfalls nicht komplexer sein können.

Die computergenerierten Testdaten hatten natürlich den Vorteil, dass die Stereoalgorithmen so mit perfekt rektifizierten Bildern zweier, was die Bildverhältnisse angeht, perfekt identischen Kameras arbeiten konnten. Mit Punktwolken die von realen Kameras erzeugt werden ist mit mehr fehlerhaften Matches zu rechnen. Die Fehlertoleranz bei der Geländerekonstruktion sollte mit solchen Daten verifiziert werden.

Die dichte Rekonstruktion einer Welt ist ein interessantes und sinnvolles Arbeitsgebiet. Echtzeitanwendungen sind z.B. für autonome Roboter oder Benutzergesteuerte System wichtig. Um diese zu realisieren sind performante Repräsentationsformen und Algorithmen nötig. Diese Arbeit bietet für den Fall der Geländerekonstruktion einen Ansatz hierfür und zeigt, dass ausreichend schnelle Systeme für diese Anwendung prinzipiell möglich sind. Für die weitere Forschung sollten allerdings effiziente Methoden für mächtigere Repräsentationsformen untersucht werden. Eine volle 3D-Repräsentation ist wünschenswert für komplexere Umgebungen. Die Möglichkeit der Verarbeitung einer nicht statischen Welt ist für praktisch zu nutzende Systeme in realen Umgebungen Vorraussetzung.

Anhang

A. Kamerakalibrierungsmatrix aus OpenCV exportiert

Die exportierten Daten einer Kamerakalibrierung eines Stereokamerasystems bestehend aus zwei baugleichen Kameras mit folgenden Eigenschaften:

- Sensorgröße: 1/2”
- Pixelgröße: 8.3 x 8.3 µm
- Bildgröße: 782x582 Pixel
- Objektiv-Brennweite: 4,2mm

Intrinsische Parameter, linke Kamera

```
%YAML:1.0
Intrinsics_Left: !!opencv-matrix
   rows: 3
   cols: 3
   dt: d
   data: [ 531.056653127333, 0.,  379.800235607594, 0.,
        531.507093784907, 293.149746452392, 0., 0., 1. ]
```

Intrinsische Parameter, rechte Kamera

```
%YAML:1.0
Intrinsics_Right: !!opencv-matrix
   rows: 3
   cols: 3
   dt: d
   data: [ 530.690772822162, 0., 373.393192073844, 0.,
       530.693205176452, 289.773729097441, 0., 0., 1. ]
```

Verzerrungsparameter, linke Kamera

```
%YAML:1.0
```

```
Dist_Coef_Left: !!opencv-matrix
   rows: 1
   cols: 5
   dt: d
   data: [ -0.328554280867042, 0.093461400772124,
       -0.002491519754471, -0.000529261505434, 0. ]
```

Verzerrungsparameter, rechte Kamera

```
%YAML:1.0
Dist_Coef_Right: !!opencv-matrix
   rows: 1
   cols: 5
   dt: d
   data: [ -0.337731870768675, 0.106286842558592,
        -0.001643575070291,  -0.000377213853165, 0. ]
```

Stereo-Rotation

```
%YAML:1.0
Stereo_Rotation: !!opencv-matrix
   rows: 3
   cols: 3
   dt: d
   data: [ 0.999937356892409, 0.00370080543678748, 0.0105634431007743,
          -0.00366465220287168, 0.999987368995513, -0.00343979267757086,
          -0.0105760396773197, 0.00340086585323894,0.999938288843963 ]
```

Stereo-Translation

```
%YAML:1.0
Stereo_Translation: !!opencv-matrix
   rows: 3
   cols: 1
   dt: d
   data: [ -60.0919954410179, -0.107508427860513, -1.38378606260633 ]
```

B. Beispielhafte Eingabedaten für DemoThread

Bilddatei

```
TestData/terrain/myheightmap3_59deg_800frames/left000.png
TestData/terrain/myheightmap3_59deg_800frames/right000.png

TestData/terrain/myheightmap3_59deg_800frames/left001.png
TestData/terrain/myheightmap3_59deg_800frames/right001.png

TestData/terrain/myheightmap3_59deg_800frames/left002.png
TestData/terrain/myheightmap3_59deg_800frames/right002.png

TestData/terrain/myheightmap3_59deg_800frames/left003.png
TestData/terrain/myheightmap3_59deg_800frames/right003.png

TestData/terrain/myheightmap3_59deg_800frames/left004.png
TestData/terrain/myheightmap3_59deg_800frames/right004.png

TestData/terrain/myheightmap3_59deg_800frames/left005.png
TestData/terrain/myheightmap3_59deg_800frames/right005.png
```

Posendatei

```
<-0.00622043,0.367592,-0.929966,-3.47826>
<0,0.929984,0.367599,1.34948>
<0.999981,0.00228663,-0.0057849,-2.66667>
<0,0,0,1>

<-0.016742,0.371705,-0.9282,-3.36689>
<0,0.92833,0.371758,1.36663>
<0.99986,0.00622397,-0.0155421,-2.69262>
<0,0,0,1>

<-0.0275948,0.375892,-0.926253,-3.25552>
<0,0.926606,0.376035,1.38379>
<0.999619,0.0103766,-0.0255695,-2.71858>
<0,0,0,1>

<-0.0387906,0.380148,-0.924112,-3.14415>
<0,0.924808,0.380435,1.40094>
<0.999247,0.0147573,-0.0358739,-2.74453>
<0,0,0,1>
```

```
<-0.0503416,0.384472,-0.921763,-3.03278>
<0,0.922933,0.38496,1.4181>
<0.998732,0.0193795,-0.0464619,-2.77049>
<0,0,0,1>
```

Tabellenverzeichnis

Abbildungsverzeichnis

Literaturverzeichnis

[Bay et al., 2008] Bay, H., Ess, A., Tuytelaars, T., und Gool, L. V. (2008). Surf: Speeded up robust features. *Computer Vision and Image Understanding (CVIU)*, 110(3):346–359.

[Birchfield und Tomasi, 1999] Birchfield, S. und Tomasi, C. (1999). Depth discontinuities by pixel-to-pixel stereo. *Int. J. Comput. Vision*, 35(3):269–293.

[Bodenmueller und Hirzinger, 2004] Bodenmueller, T. und Hirzinger, G. (2004). Online surface reconstruction from unorganized 3d-points for the dlr hand-guided scanner system. In *3D Data Processing, Visualization and Transmission, 2004. 3DPVT 2004. Proceedings. 2nd International Symposium on*, pages 285 – 292.

[Bouguet, 2010] Bouguet, J.-Y. (2010). Camera calibration toolbox for matlab. `http://www.vision.caltech.edu/bouguetj/calib_doc/` Abruf: 28.07.2010.

[Bradski und Kaehler, 2008] Bradski, G. und Kaehler, A. (2008). *Learning OpenCV - Computer Vision with the OpenCV Library*. O'Reilly, 1 edition.

[DFKI, 2010] DFKI (2010). Projekt semprom.

[Eichler et al., 1974] Eichler, H. J., Gobrecht, H., Hahn, D., Niedrig, H., Richter, M., Schoenebeck, H., Weber, H., und Weber, K. (1974). *Bergmann-Schaefer, Lehrbuch der Experimentalphysik Optik*, volume III. Walter de Gruyter, 6 edition.

[Hartley und Zisserman, 2008] Hartley, R. und Zisserman, A. (2008). *Multiple View Geometry in Computer Vision*. Cambridge University Press.

[Heikkila und Silven, 1997] Heikkila, J. und Silven, O. (1997). A four-step camera calibration procedure with implicit image correction. In *CVPR '97: Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR '97)*, page 1106, Washington, DC, USA. IEEE Computer Society.

[Hildebrandt, 2010] Hildebrandt, M. (2010). Dfki.

[Hirschmüller et al., 2005] Hirschmüller, H., Scholten, F., und Hirzinger, G. (2005). Stereo vision based reconstruction of huge urban areas from an airborne pushbroom camera (hrsc). In Kropatsch, W. G., Sablatnig, R., und Hanbury, A., editors, *Pattern Recognition*, volume 3663 of *Lecture Notes in Computer Science*, pages 58–66. Springer Berlin / Heidelberg.

[Jähne, 2005] Jähne, B. (2005). *Digitale Bildverarbeitung*. Springer-Verlag.

[Kagami et al., 2005] Kagami, S., Takaoka, Y., Kida, Y., Nishiwaki, K., und Kanade, T. (2005). Online dense local 3d world reconstruction from stereo image sequences. In *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 3858 – 3863.

[Kaustubh Pathak und Schwertfeger, 2007] Kaustubh Pathak, Andreas Birk, J. P. und Schwertfeger, S. (2007). 3d forward sensor modeling and application to occupancy grid based sensor fusion. *International Conference on Intelligent Robots and Systems (IROS)*.

[Kolmogorov und Zabih, 2002] Kolmogorov, V. und Zabih, R. (2002). Multi-camera scene reconstruction via graph cuts. In *ECCV '02: Proceedings of the 7th European Conference on Computer Vision-Part III*, pages 82–96, London, UK. Springer-Verlag.

[Lhuillier und Quan, 2005] Lhuillier, M. und Quan, L. (2005). A quasi-dense approach to surface reconstruction from uncalibrated images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(3):418 –433.

[NVIDIA, 2010] NVIDIA (2010). Nvidia cuda c programming guide. `http://developer.nvidia.com/object/gpucomputing.html` (Stand: 07.07.2010).

[Ohno und Tadokoro, 2005] Ohno, K. und Tadokoro, S. (2005). Dense 3d map building based on lrf data and color image fusion. In *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 2792 – 2797.

[Press et al., 2007] Press, W. H., Teukolsky, S. A., Vetterling, W. T., und Flannery, B. P. (2007). *Numerical Recipes - The Art of Scientific Computing*. Cambridge University Press.

[Scharstein und Szeliski, 2002] Scharstein, D. und Szeliski, R. (2002). A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47(1-3):7–42.

[Stam, 2008] Stam, J. (2008). Stereo imaging with cuda. `http://sourceforge.net/projects/openvidia/files/CUDA%20Stereo%20Camera/`.

[Strang, 2003] Strang, G. (2003). *Linear Algebra*. Springer-Verlag.

[Turk und Levoy, 1994] Turk, G. und Levoy, M. (1994). Zippered polygon meshes from range images. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 311–318, New York, NY, USA. ACM.

[Vogiatzis et al., 2008] Vogiatzis, G., Torr, P., Seitz, S., und Cipolla, R. (2008). Reconstructing relief surfaces. *Image Vision Comput., Vol. 26, No. 3. (2008), pp. 397-404. Image Vision Comput., Vol. 26, No. 3. (2008), pp. 397-404. Image Vision Comput.*, 26(3):397–404.

[Zhang und Kanade, 1998] Zhang, Z. und Kanade, T. (1998). Determining the epipolar geometry and its uncertainty: A review. *International Journal of Computer Vision*, 27:161–195.