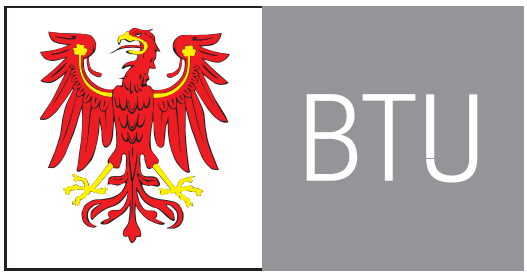
Brandenburgische Technische Universität Cottbus
Institut für Informatik
Lehrstuhl Graphische Systeme



Diplomarbeit

Beschleunigung von Matchingverfahren durch 3D-Laserdaten zur schnellen und präzisen Objektrekonstruktion

Dirk Frommholz
Matrikelnummer 2105709

14. Januar 2008

Betreuer:
Prof. Dr. Winfried Kurth (BTU Cottbus)

Zweitgutachter:
Dr. rer. nat. Karsten Scheibe (DLR Berlin)

Diplomarbeit
Beschleunigung von Matchingverfahren durch 3D-Laserdaten
zur schnellen und präzisen Objektrekonstruktion

Dirk Frommholz (dirk@csidev.net)
Studiengang Informatik
Matrikelnummer 2105709
Rostocker Straße 21
03046 Cottbus



Prof. Dr. Winfried Kurth (wk@informatik.tu-cottbus.de)
Brandenburgische Technische Universität Cottbus
Lehrstuhl Graphische Systeme am Institut für Informatik
Ewald-Haase-Straße 12/13
03044 Cottbus



Dr. rer. nat. Karsten Scheibe (karsten.scheibe@dlr.de)
Deutsches Zentrum für Luft- und Raumfahrt e. V. (DLR)
Einrichtung Optische Informationssysteme am Institut für
Robotik und Mechatronik
Rutherfordstraße 2
12489 Berlin

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides statt, dass ich diese Diplomarbeit selbstständig verfasst und noch nicht anderweitig für Prüfungszwecke vorgelegt oder veröffentlicht habe. Für die Erstellung wurden ausschließlich die angegebenen Quellen und Hilfsmittel benutzt. Wörtliche und sinngemäße Zitate sind als solche gekennzeichnet.

Ort, Datum Unterschrift

Kurzfassung

Die vorliegende Diplomarbeit kombiniert die photogrammetrische mit der laserbasierten Objekterfassung. Das Ziel hierbei ist es, einen Algorithmus zu entwerfen und zu implementieren, der die spezifischen Vorteile der beiden Verfahren in Bezug auf die Ausführungsgeschwindigkeit und Genauigkeit in sich vereint und die Nachteile ausblendet. Hierzu wird der Suchbereich für die fensterbasierte Korrespondenzanalyse auf den Stereobildern mit Hilfe eines Raytracers, der auf dem 3D-Modell des Laserscanners operiert, auf eine Punktumgebung eingeschränkt. Wie die Evaluierung des entwickelten Softwareprototyps anhand einer künstlichen und einer realen Szene zeigt, verringert sich der Rechenzeitbedarf im Vergleich zur klassischen Zuordnungsbestimmung entlang der Epipolarlinien um ein Vielfaches, sofern die Voraussetzungen für den neuen Algorithmus erfüllt sind. Ebenso ergibt sich eine Qualitätssteigerung bei den lokalen Bildübereinstimmungen, die es ermöglicht, die Raumpunkte des vorhandenen 3D-Modells zu korrigieren und zusätzlich zu verdichten.

Inhaltsverzeichnis

1. Einleitung **8**
- 1.1. Motivation . . . 8
- 1.2. Ziel der Arbeit . . . 9
- 1.3. Inhaltliche Gliederung . . . 9
- 1.4. Notation . . . 9
  - 1.4.1. Mathematische Ausdrücke . . . 9
  - 1.4.2. Quelltextreferenzen und Benutzeroberflächenelemente . . . 10

2. Grundlagen **11**
- 2.1. Stereophotogrammetrie . . . 11
  - 2.1.1. Bilderfassung . . . 11
  - 2.1.2. Kameramodell . . . 13
    - 2.1.2.1. Planare Lochkamera . . . 13
    - 2.1.2.2. Zylindrische Lochkamera . . . 18
  - 2.1.3. Stereogeometrie . . . 21
  - 2.1.4. Objektrekonstruktion . . . 23
    - 2.1.4.1. Korrespondenzproblem und Matching . . . 24
    - 2.1.4.2. Sichtstrahlrekonstruktion . . . 25
    - 2.1.4.3. Auflösung und Zeitbedarf . . . 26
- 2.2. 3D-Laserscanning . . . 28
  - 2.2.1. Aufbau und Distanzbestimmung . . . 28
  - 2.2.2. Objektrekonstruktion . . . 29
- 2.3. Modellerzeugung . . . 31

3. Motivation und Konzeption **35**
- 3.1. Motiv der Verfahrenskombination . . . 35
- 3.2. Konzeption für einen integrierten Algorithmus . . . 36
  - 3.2.1. Ausgangssituation . . . 36
  - 3.2.2. Datenmanagement . . . 37
    - 3.2.2.1. Bilddaten . . . 37
    - 3.2.2.2. 3D-Modelle . . . 39
    - 3.2.2.3. Konfigurationen . . . 39
  - 3.2.3. Objektpunktschätzung . . . 40
  - 3.2.4. Matching . . . 41
  - 3.2.5. Modellverbesserung . . . 42

4. Implementierung **44**
- 4.1. Überblick über die Software . . . 44
- 4.2. Entwicklungsumgebung und Zielplattform . . . 45
- 4.3. Interner Aufbau der Anwendung . . . 45
- 4.4. Gemeinsam genutzte Komponenten . . . 47
- 4.5. Rahmenprogramm und Bedienkonzept . . . 49

4.6. Datenmanagement . . . 50
4.6.1. Bilddaten . . . 50
4.6.1.1. Speicherung und Datenzugriff . . . 50
4.6.1.2. Höhere bildverarbeitende Algorithmen . . . 55
4.6.1.3. Ein- und Ausgabe . . . 67
4.6.2. 3D-Modelle . . . 69
4.6.2.1. Speicherung und Datenzugriff . . . 69
4.6.2.2. Ein- und Ausgabe . . . 70
4.6.3. Konfigurationen . . . 73
4.6.3.1. Speicherung und Datenzugriff . . . 73
4.6.3.2. Ein- und Ausgabe . . . 74
4.6.4. Datenobjekte . . . 75
4.6.5. Algorithmusobjekte . . . 76
4.7. Funktionale Algorithmen . . . 76
4.7.1. Ein- und Ausgabe der Bilddaten, 3D-Modelle und Konfigurationen . . . 78
4.7.2. Visualisierung . . . 79
4.7.2.1. Bilddaten . . . 79
4.7.2.2. 3D-Modelle . . . 80
4.7.3. Integriertes Matching . . . 80
4.7.3.1. Kameramodelle und Stereogeometrie . . . 81
4.7.3.2. Objektpunktschätzung . . . 82
4.7.3.3. Korrespondenzanalyse . . . 91
4.7.4. Modellverbesserung . . . 98
4.7.4.1. Projektion der Modellpunkte . . . 99
4.7.4.2. Korrekturphase . . . 102
4.7.4.3. Korrekturstatistik . . . 102
4.7.5. Test und Evaluierung . . . 104
4.7.5.1. Testweltgenerator . . . 105
4.7.5.2. Modelldeformation . . . 106

5. Ergebnisse **107**
5.1. Künstliche Testwelt . . . 107
5.1.1. Matching . . . 108
5.1.1.1. Projektionsarten, Speichermethoden und Zuordnungsalgorithmen . . . 110
5.1.1.2. Struktur des 3D-Baums und Strahlverfolgung . . . 116
5.1.1.3. Genauigkeit der Korrespondenzen . . . 125
5.1.1.4. Relative Orientierung der Kameras zum Modell . . . 141
5.1.2. Modellverbesserung . . . 147
5.1.2.1. Modelldeformation . . . 147
5.1.2.2. Matching . . . 147
5.1.2.3. Korrektur des gestörten Modells . . . 155
5.2. Reale Szene . . . 158
5.2.1. Matching . . . 158
5.2.2. Modellverbesserung . . . 165

6. Zusammenfassung und Ausblick **170**

A. Anhang **172**
A.1. Künstliche Testwelt . . . 172

A.2. Struktur der VRML-Modellbeschreibungen 175
A.3. Struktur der XML-Konfigurationsdateien und Ausführungsprotokolle 176

Abbildungsverzeichnis **180**

Tabellenverzeichnis **183**

Literaturverzeichnis **187**

1. Einleitung

1.1. Motivation

Die Speicherung und Verarbeitung von dreidimensionalen Computermodellen realer Objekte hat mit der stetig steigenden Leistungsfähigkeit der Rechnersysteme stark an Bedeutung gewonnen. Neben der Unterhaltungsindustrie, die mit wirklichkeitsgetreuen Computerspielen große Erfolge aufweisen kann, haben auch Unternehmen aus anderen Bereichen der Wirtschaft das Potential dieser Technologie entdeckt. So werden etwa für die Rekonstruktion von historischen Gebäuden, die Analyse von Materialfehlern, die Ortsbestimmung und vielfältige Vermessungsaufgaben umfangreiche Mengen an qualitativ hochwertigen 3D-Daten benötigt.

Ein zentrales Problem besteht in allen Anwendungsfeldern darin, den jeweiligen Untersuchungsgegenstand räumlich zu erfassen. Heute kommen zu diesem Zweck überwiegend berührungslose Verfahren zum Einsatz (*remote sensing*), die elektromagnetische Wellen unterschiedlicher Energie und Frequenz verwenden. Im Bereich des sichtbaren Lichts ist es beispielsweise möglich, mit mehreren geeignet positionierten und kalibrierten Kameras einen Satz von überlappenden Bildern zu erhalten, anhand dessen sich die Entfernungen zum dargestellten Objekt und somit auch dessen Gestalt mikrometergenau berechnen lassen. Bei dieser photogrammetrischen Methode ist es allerdings erforderlich, diejenigen Bildbereiche in den einzelnen Aufnahmen zu identifizieren, die den gleichen Objektausschnitt darstellen (Korrespondenzanalyse). Dazu müssen zahlreiche Bildausschnitte aufwendig miteinander verglichen werden, was angesichts der großen Datenmengen bei der Verwendung von professionellem digitalen Aufnahmegerät recht zeitintensiv ist.

Eine andere oft genutzte Möglichkeit zur Erfassung der Objektgestalt besteht in der Verwendung eines 3D-Laserscanners. Dieser misst die Entfernung zum Untersuchungsgegenstand in diskreten Punkten, indem er den Zeit- oder Phasenunterschied der emittierten Laserimpulse ermittelt, aus der sich wiederum die Form des Körpers rekonstruieren lässt. Im Gegensatz zu den photogrammetrisch gewonnenen Objektdaten arbeiten Laserscanner zwar schneller, sind aber in der Regel weniger genau und erfassen abrupte Tiefensprünge nur unzureichend.

Beide Verfahren weisen offensichtlich gegensätzliche Stärken und Schwächen auf. Daher ist es naheliegend, zu untersuchen, ob die photogrammetrische Erfassung von dreidimensionalen Objekten in Kombination mit der durch den Laserscanner gewonnenen Repräsentation präzise *und* schnell realisiert werden kann.

1.2. Ziel der Arbeit

Ziel dieser Arbeit ist es, ein Verfahren zu entwerfen und in Form einer Software zu implementieren, das die Korrespondenzanalyse auf den mit einer Stereopanoramakamera gewonnenen Bildinformationen beschleunigt. Dabei soll auf ein 3D-Modell der abgebildeten Szene zurückgegriffen werden, welches sich aus einem oder mehreren vermaschten Laserscans zusammensetzt. Aufbauend darauf ist ein Algorithmus zu entwickeln und zu analysieren, mit dem sich die diskrete Modellpunktwolke durch die photogrammetrisch ermittelten Entfernungsangaben sowohl quantitativ als auch qualitativ verbessern lässt.

1.3. Inhaltliche Gliederung

Inhaltlich gliedert sich die Diplomarbeit in insgesamt sechs Kapitel. Auf diese Einleitung, die das erste Kapitel darstellt, folgt zunächst die Erläuterung der mathematisch-technischen Grundlagen der photogrammetrischen wie auch der lasergestützten Objekterfassung und -rekonstruktion. Anschließend werden im dritten Teil das Konzept für den integrierten Algorithmus zur Beschleunigung der Korrespondenzanalyse und eine Methode zur Modellverbesserung vorgestellt. Der vierte Hauptabschnitt widmet sich der prototypischen Implementierung des Entwurfs als Software, die daraufhin mit künstlichen und realen Beispieldaten evaluiert wird. Das sechste und letzte Kapitel fasst die Ergebnisse der Diplomarbeit noch einmal kurz zusammen und gibt einen Ausblick auf alternative Implementierungsansätze.

1.4. Notation

In dieser Ausarbeitung werden an zahlreichen Stellen mathematische Größen verwendet und Quelltextobjekte referenziert. Um Mehrdeutigkeiten zu vermeiden und die Verständlichkeit der getroffenen Aussagen zu erhöhen, kommt in allen Kapiteln die folgende einheitliche Notation zum Einsatz.

1.4.1. Mathematische Ausdrücke

In den Erläuterungen der Diplomarbeit sind skalare mathematische Ausdrücke stets in kursiven Kleinbuchstaben gehalten (Beispiel: f). Für geometrische Entitäten kommen durchgängig kursive Großbuchstaben zum Einsatz (Beispiel: P_k).

Ungebundene Vektoren und Raumpunkte (gebundene Vektoren, Ortsvektoren) werden im Fettdruck mit Klein- respektive Großschrift unterschiedlich dargestellt, bei Berechnungen jedoch äquivalent behandelt. Dabei ist zur Vermeidung von Verwechslungen der zugrundegelegte Raum wenn notwendig durch einen hochgestellten geklammerten Buchstaben angegeben. So bezeichnet c den dreidimensionalen Kameraraum (camera space, Beispiel: $\mathbf{X}_\mathbf{k}^{(\mathbf{c})}$), ein w das ebenfalls dreidimensionale Weltkoordinatensystem (world space, Beispiel: $\mathbf{X}_\mathbf{k}^{(\mathbf{w})}$) und i den zweidimensionalen Bildraum (image space, Beispiel: $\mathbf{X}_\mathbf{k}^{(\mathbf{i})}$) – das Grundlagenkapitel geht hierauf noch näher ein. Um identische Punkte in zwei oder mehreren

Räumen auszeichnen zu können, wird der aktuell betrachtete Primärraum um die Sekundärräume unter Verwendung der Einschränkungsnotation (senkrechter Strich, |) erweitert. Der 2D-Bildpunkt $\mathbf{X}_{\mathbf{k}}^{(\mathbf{i})}$ als Element des Kameraraums, also in 3D-Koordinaten, besitzt die Notation $\mathbf{X}_{\mathbf{k}}^{(\mathbf{c}|\mathbf{i})}$.

Matrizen, etwa zur Beschreibung von Abbildungen, sind analog zu den Raumpunkten fett und in großen Lettern gedruckt (Beispiel: $\mathbf{R}$). Eine Raumkennzeichnung ist hierbei nicht erforderlich. Einzelne Matrixelemente werden ebenso wie Vektorkomponenten mit runden Klammern selektiert, wobei x, y und z bei zwei- und dreidimensionalen Vektoren die Elemente 1, 2 und 3 repräsentieren (Beispiel: $\mathbf{v}^{(\mathbf{w})}(x)$ wählt die x-Komponente des Vektors $\mathbf{v}^{(\mathbf{w})}$ im Weltkoordinatensystem, $\mathbf{M}(1, 2)$ wählt das Element in Spalte 1 und Zeile 2 der Matrix $\mathbf{M}$).

1.4.2. Quelltextreferenzen und Benutzeroberflächenelemente

Im Rahmen der Implementierung des integrierten Algorithmus zur Korrespondenzanalyse und des Verfahrens zur Modellverbesserung im vierten Kapitel werden Quelltextverweise gleich welcher Art in Schreibmaschinenschrift ausgezeichnet (Beispiel: `core`, `platform.h`). Im Speziellen lassen sich objektorientierte Datentypen (Klassen) an einem führenden großen C im Namen erkennen (Beispiel: `CImage`). Klassenvariablen (Attributen) geht der Kleinbuchstabe m gefolgt von einem Unterstrich voraus (Beispiel: `m_sx1`). Funktionen und Methoden sind durch ein abschließendes Klammernpaar () unabhängig von der konkreten Parameterzahl gekennzeichnet (Beispiel: `computeCorrectedPoints()`).

Ebenfalls in Schreibmaschinenschrift ausgeführt sind die Interaktionselemente der verwendeten Benutzeroberfläche, wenn diese zur Erläuterung von Bedienvorgängen ihren Niederschlag im Text finden. Zur Abgrenzung von den Quellcodereferenzen wird hierbei jedoch stets der Kontext explizit gemacht (Beispiel: Schaltfläche `Assign Object`).

2. Grundlagen

Im Folgenden werden zunächst die Grundlagen der Stereophotogrammetrie und der laserbasierten Objekterfassung dargelegt, die für das Verständnis dieser Arbeit notwendig sind. Der Fokus liegt hierbei auf den geometrischen Merkmalen der Panoramakamera und den sich anschließenden Verarbeitungsschritten zur Raumpunktrekonstruktion aus den Entfernungsangaben.

2.1. Stereophotogrammetrie

Die historisch aus der Geodäsie (Erdvermessung und -abbildung) hervorgegangene Wissenschaft der Photogrammetrie beschäftigt sich mit der Gewinnung von verlässlichen Informationen über physikalische Körper und ihre Umgebung durch die Aufzeichnung, Ausmessung und Interpretation von Mustern elektromagnetischer Wellen, insbesondere von photographischen Bildern [1]. Das Ziel dabei ist, ein dreidimensionales Modell der aufgenommenen Objekte zu erhalten, das diese möglichst wirklichkeitsgetreu repräsentiert und sich zur Weiterverarbeitung für das jeweilige Anwendungsgebiet eignet.

Wenngleich die Definition des Begriffs keine Aussage über die Anzahl der für die Informationsgewinnung heranzuziehenden Bildaufzeichnungen trifft, beschränken sich die folgenden Ausführungen auf den für diese Arbeit relevanten Spezialfall der Stereophotogrammetrie im sichtbaren elektromagnetischen Spektrum. Hierbei werden, wie der Name suggeriert, zwei sich inhaltlich überlappende Aufnahmen einer Szene von unterschiedlichen Standorten angefertigt und ausgewertet.

2.1.1. Bilderfassung

Die Bilderfassung in der Photogrammetrie erfolgt heute in der Regel mit digitalen Kameras, bei denen Bildsensoren in Gestalt von integrierten Schaltkreisen die Rolle des früher verwendeten chemisch beschichteten Filmmaterials übernehmen. Ansonsten ähneln sowohl der mechanooptische Aufbau als auch die Funktionsweise dem klassischen Vorbild weitgehend.

Wird der Bildaufnahmeprozess initiiert, so gelangt zunächst das von den selbstleuchtenden oder angestrahlten Szenenobjekten reflektierte Licht in das Kameraobjektiv. Hier wird es durch eine herstellerspezifische Anordnung von Streu- und Sammellinsen gebündelt. Im Vergleich zum idealen Strahlenverlauf treten dabei durch die physische Ausdehnung der optisch wirksamen Bauelemente und konstruktive Unzulänglichkeiten Abbildungsfehler auf, die sich nur bedingt kompensieren lassen. Hierzu zählen beispielsweise die durch Dis-

persion[1] verursachte chromatische Aberration und verschiedenartige räumliche Verzeichnungen (Tonnen- bzw. Kisseneffekt, Asymmetriefehler).

Wenn das konzentrierte Licht das Objektiv passiert hat, trifft es nach der Filterung von unerwünschten ultravioletten und infraroten Anteilen sowie der Maskierung der additiven Grundfarben Rot, Grün und Blau (RGB) für die Farbdarstellung in einem bestimmten Abstand auf den Bildsensor. Ist die Kamera scharf gestellt, entspricht diese Distanz bei unendlich weit entfernten Motiven gerade der Brennweite des Linsensystems. Die Fokussierung auf nah gelegene Objekte vergrößert den Sensorabstand vom Brennpunkt. Der Chip selbst besteht zumeist aus einem in CCD- oder CMOS-Technik [2] gefertigten Satz von endlich vielen diskret angeordneten photosensitiven Halbleiterelementen, die als Sensorpixel bezeichnet werden. Die Anzahl dieser Pixel in horizontaler und vertikaler Richtung nach der Dezimierung durch die Farbmaske definiert die Bildauflösung der Kamera und somit die Fähigkeit, kleine Objektdetails wiederzugeben, ohne benachbarte Strukturen zu integrieren. Im bestmöglichen Fall liegt der Auftreffpunkt des Lichts in der Mitte des Sensorfeldes und fällt somit mit dem idealen Kamerahauptpunkt (Schnittpunkt der optischen Achse mit der Bild- respektive Sensorebene) zusammen. In der Praxis ist dies jedoch durch die bereits angesprochenen Abbildungsfehler und minimale Abweichungen bei der Montage nicht erreichbar.

Die Sensorpixel wandeln danach das einfallende Licht in eine zu dessen Energie proportionale Spannung um. Mit Hilfe eines Analog-Digital-Wandlers wird diese für jedes Halbleiterelement in ein ganzzahliges numerisches Datum umgerechnet, wobei es je nach Dynamikumfang zu einem mehr oder minder starken Informationsverlust durch Runden kommt (Quantisierung). Die so ermittelten diskreten Werte repräsentieren die Intensitäten im Bild für jede der drei Grundfarben. Abschließend erfolgt die Speicherung der Bildinformationen in einem geeigneten (Datei-)Format auf einem persistenten Massenspeicher, von dem sie sich bei Bedarf mit einer Verarbeitungssoftware einlesen und manipulieren lassen. Abbildung 2.1 veranschaulicht den Erfassungsvorgang.

Obgleich die Bildakquise bei allen digitalen Kameras letztlich auf dem erläuterten Grundprinzip beruht, lassen sich verschiedene Kameratypen unterscheiden, die auf ganz bestimmte Einsatzgebiete zugeschnitten sind. Neben den handelsüblichen Geräten für Photoamateure werden etwa für Überwachungsaufgaben bewusst verzeichnende Spezialobjektive in sphärischer Ausführung mit großem Blickwinkel verwendet. Luftaufnahmen dagegen erfolgen oftmals mit sogenannten *push broom*-Kameras, die eine oder mehrere dedizierte eindimensionale Bildzeilen anstelle von flächigen Sensorelementen aufweisen. Hier erzeugt die Flugbewegung die fehlende zweite Dimension.

Auch die im Rahmen dieser Arbeit verwendete Panoramakamera (Abbildung 2.2), die am Deutschen Zentrum für Luft- und Raumfahrt in Berlin mitentwickelt wurde, besitzt einen Satz von hochauflösenden Linienelementen für jede der drei Grundfarben. Diese werden für die Bilderfassung rotiert, wodurch sich eine zylindrische Aufnahmecharakteristik mit horizontalen Erfassungswinkeln von bis zu 360° ergibt. Tabelle 2.1 fasst ausgewählte technische Daten der Panoramakamera kurz zusammen.

[1] die Abhängigkeit der Brechzahl von der Wellenlänge, d.h. vereinfacht ausgedrückt wird Licht unterschiedlicher Farbe verschieden stark gebündelt

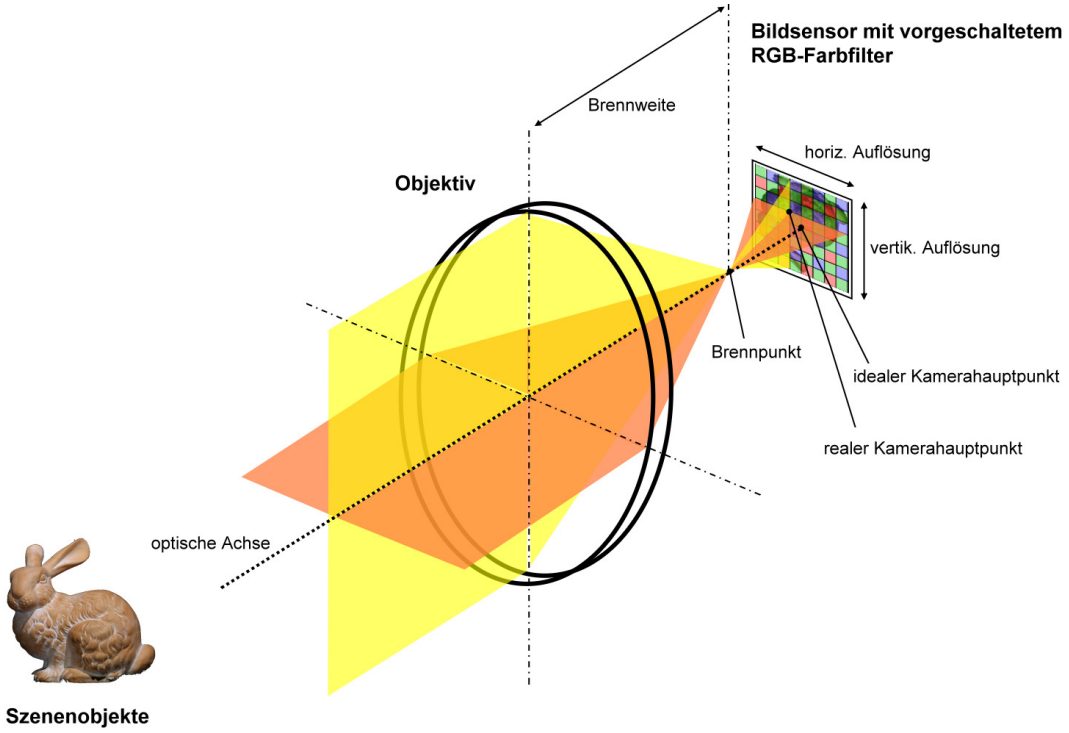


Abbildung 2.1.: Bilderfassung mit der digitalen Kamera

2.1.2. Kameramodell

Als Basis für die theoretischen Betrachtungen erweist sich der soeben skizzierte Kameraaufbau aufgrund seiner Komplexität etwa in Bezug auf die exakte Beschreibung des Objektivs als wenig geeignet. Für eine handhabbare kompakte mathematische Formulierung müssen daher vereinfachende Annahmen getroffen werden.

Prinzipiell lässt sich die Bilderfassung durch eine Kamera als Abbildung von einem dreidimensionalen Objektraum in einen zweidimensionalen Bildraum auffassen. Dabei kommt oftmals die Zentralprojektion zum Einsatz, die das Aufnahmegerät auf eine Lochkamera ohne Linsensystem reduziert und auch hier Verwendung finden soll [4]. Für den planaren Fall mit ebener Projektionsfläche, der sich zum Testen der Bildverarbeitungsalgorithmen besser eignet, und die zylindrische Panoramakamera werden die diesem Modell zugrundeliegenden Beziehungen im Folgenden hergeleitet.

2.1.2.1. Planare Lochkamera

Das planare Lochkameramodell (Abbildung 2.3) basiert darauf, dass alle von den Szenenobjekten emittierten oder reflektierten Lichtstrahlen in einem gemeinsamen Punkt $\mathbf{C}^{(c)}$ zusammenlaufen, der gleichzeitig den Ursprung des dreidimensionalen euklidischen Kamerakoordinatensystems bildet. Davor befindet sich im Brennweitenabstand f senkrecht zur Z-Achse die rechteckige Bildebene. Diese ist ihrerseits aus d_x x d_y rechteckigen Bildelementen, den Pixeln, zusammengesetzt. Jedes Pixel hat dabei eine Größe von d_{px} x d_{py} Längeneinheiten im Kamerasystem und kann gleichzeitig die drei Grundfarben Rot, Grün und Blau aufnehmen. Eine Spezialisierung der Sensorelemente und die damit einhergehen-



Abbildung 2.2.: Panoramakamera KST EyeScan M3D metric (entnommen aus [3])

Sensor	CCD, 3 Zeilen RGB, je 10.298 Pixel vertikal, 72 mm, integrierter UV-IR-Sperrfilter
Kontrastauflösung/Datenformat	42 bit RGB / 3 x 16 Bit, RAW unkomprimiert
Datenschnittstelle/Kamerasteuerung	Hot-Link, fiberoptisch, bidirektional
Frame Grabber	PCI Frame Grabber
Datenmenge	3 GB für 360° bei 60 mm Brennweite
Objektive	LINOS APO-Sironar digital HR mit 28, 35, 60, 100, 180 mm Brennweite
Kürzeste Scanzeit (360°, volle Auflösung)	180 s (bei Makropixel entsprechend kürzer)
Makropixel/schneller Pre-Scan	2x bis 5x
Empfindlichkeit	ISO 50 bis ca. ISO 200 pushbar
Winkelmesssystem	Auflösung 0,001°
Belichtungszeiten	1/250 s bis 1 s, stufenlos einstellbar (mit Shutter kürzere Zeiten möglich)

Tabelle 2.1.: Technische Daten der Panoramakamera EyeScan M3D metric (nach [3]). Die in der Quelle aufgeführte Zeilenauflösung von jeweils 10.200 Pixeln entspricht nicht den tatsächlich verfügbaren 10.298 Bildpunkten.

de Reduktion der nutzbaren Bildauflösung, wie sie in der Realität durch die vorgeschalteten Farbfilter auftritt, gibt es demnach im Modell nicht.

Jeder aufgenommene Objektpunkt $\mathbf{X_k^{(c)}}$ im Kamerasystem erzeugt nun entlang seines Sichtstrahls $R_k = \mathbf{C^{(c)}X_k^{(c)}}$ ausgehend vom Kamera- oder Projektionszentrum $\mathbf{C^{(c)}}$ einen zugehörigen Schnittpunkt $\mathbf{X_k^{(c|i)}}$ in der Bildebene. Er kann demzufolge auch durch zweidimensionale Bildkoordinaten als Pixelposition $\mathbf{X_k^{(i)}}$ lokalisiert werden. Dafür ist prinzipiell das kanonische Koordinatensystem mit den Basisvektoren $\hat{\mathbf{x}}^{(\mathbf{i})}$ bzw. $\hat{\mathbf{y}}^{(\mathbf{i})}$ und dem realen Kamerahauptpunkt $\mathbf{P^{(i)}}$ als Ursprung geeignet. Letzterer entspricht dem Schnittpunkt der senkrecht zur Projektionsfläche verlaufenden Z-Achse, die bestenfalls gleichzeitig die optische Achse repräsentiert. Eine Abweichung der Kreuzung von der Bildmitte, wo sich der ideale Kamerahauptpunkt befindet, ist hierbei zulässig und stellt den einzigen modellierbaren Abbildungsfehler dar. In der Praxis wie auch der vorliegenden Arbeit wird jedoch ein anderes Koordinatensystem mit der Orthonormalbasis $(\mathbf{x^{(i)}}; \mathbf{y^{(i)}})$ angenommen, bei dem sich der Bezugspunkt $\mathbf{B^{(i)}}$ in der linken oberen Bildecke befindet. Dies entspricht der Speicherkonvention der Graphikhardware sowie zahlreicher Bilddatenformate und hat den Vorteil, dass keine negativen Pixelpositionen auftreten können.

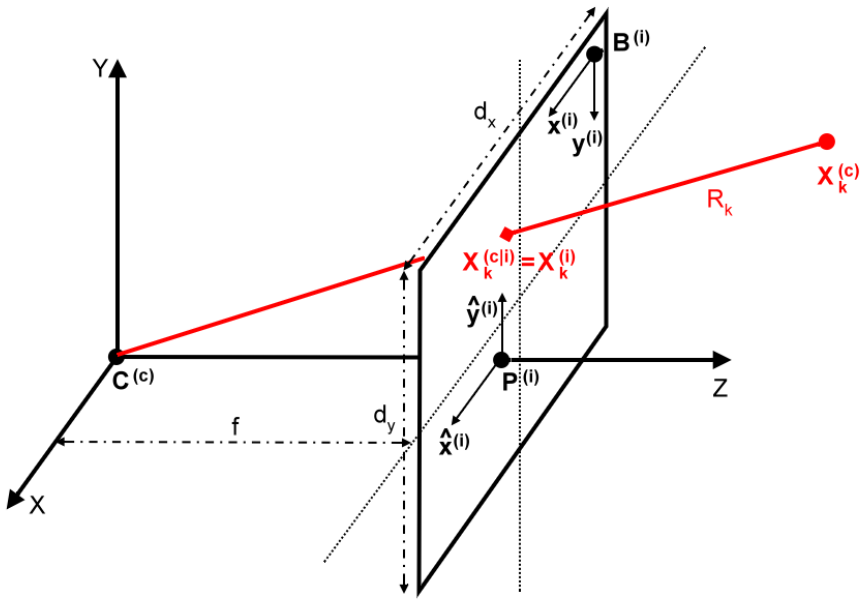


Abbildung 2.3.: Planares Lochkameramodell

Projektionsvorschrift Unter diesen Voraussetzungen leitet sich die Abbildungsvorschrift $M_{plproj} : \mathbb{R}^3 \rightarrow \mathbb{R}^2$ für die Objektpunkte $\mathbf{X_k^{(c)}}$ auf die Bildebene aus dem Strahlensatz ab. Es gilt demnach für die horizontale und vertikale Schnittpunktkomponente $\mathbf{X_k^{(c|i)}}(x)$ bzw. $\mathbf{X_k^{(c|i)}}(y)$ in Kamerasystemkoordinaten zunächst

$$\frac{\mathbf{X_k^{(c)}}(x)}{\mathbf{X_k^{(c)}}(z)} = \frac{\mathbf{X_k^{(c|i)}}(x)}{f} \qquad \mathbf{X_k^{(c|i)}}(x) = \frac{f\mathbf{X_k^{(c)}}(x)}{\mathbf{X_k^{(c)}}(z)} \tag{2.1}$$

und

$$\frac{\mathbf{X_k^{(c)}}(y)}{\mathbf{X_k^{(c)}}(z)} = \frac{\mathbf{X_k^{(c|i)}}(y)}{f} \qquad \mathbf{X_k^{(c|i)}}(y) = \frac{f\mathbf{X_k^{(c)}}(y)}{\mathbf{X_k^{(c)}}(z)} \ . \tag{2.2}$$

Hierbei sei neben einer positiven Brennweite jeweils $\mathbf{X_k^{(c)}}(z) \neq 0$ angenommen. Weil sich die Objektpunkte in der Realität selten innerhalb der Kamera befinden, kann man die zweite Einschränkung ohne Nebenwirkungen auch auf $\mathbf{X_k^{(c)}}(z) \geq f$ verschärfen. Damit werden gleichzeitig auch diejenigen unsichtbaren Szenenelemente von der Projektion ausgeschlossen, die sich hinter dem Kamerazentrum $\mathbf{C^{(c)}}$ befinden, wenngleich die Formeln auch in diesem Fall ein korrektes Ergebnis liefern.

Die Division durch die Pixelabmessungen überführt beide Werte in das hauptpunktbezogene Bildkoordinatensystem. Mit einer Translation unter Berücksichtigung von $\mathbf{P^{(i)}}$ und der Änderung der vertikalen Richtung durch Subtraktion der um ein Pixel erniedrigten Bildhöhe[2] ergibt sich schließlich die finale Pixelposition zur Basis $\mathbf{B^{(i)}}$. Man erhält also

$$\mathbf{X_k^{(i)}}(x) = \frac{\mathbf{X_k^{(c|i)}}(x)}{d_{px}} + \mathbf{P^{(i)}}(x) \tag{2.3}$$

$$\mathbf{X_k^{(i)}}(y) = d_y - 1 - (\frac{\mathbf{X_k^{(c|i)}}(y)}{d_{py}} + \mathbf{P^{(i)}}(y)) \ , \tag{2.4}$$

wobei im Allgemeinen reellwertige Resultate auftreten, die im Sinne einer hohen Rechengenauigkeit nicht gerundet werden sollten (*Subpixelpräzision*).

Inverse Projektionsvorschrift Im Gegensatz zur Projektion ist die Berechnung eines Objektpunktes aus einem Bildpunkt $M_{plproj^{-1}} : \mathbb{R}^2 \rightarrow \mathbb{R}^3$ nicht eindeutig. Die horizontale und vertikale Position auf der Projektionsfläche im Kamerasystem erhält man durch die Umkehrung der Gleichungen 2.3 und 2.4:

$$\mathbf{X_k^{(c|i)}}(x) = d_{px}(\mathbf{X_k^{(i)}}(x) - \mathbf{P^{(i)}}(x)) \tag{2.5}$$

$$\mathbf{X_k^{(c|i)}}(y) = d_{py}(d_y - 1 - \mathbf{X_k^{(i)}}(y) - \mathbf{P^{(i)}}(y)) \tag{2.6}$$

Zuerst werden die Bildkoordinaten in das hauptpunktrelative Bezugssystem umgerechnet, wobei die vertikale Orientierung wechselt. Anschließend erfolgt die Konvertierung der resultierenden Position in Längeneinheiten über die Pixelgröße. Die Rekonstruktion des ursprünglichen Objektpunktes $\mathbf{X_k^{(c)}}$ im Kamerasystem gelingt dann mittels der Normalisierung des Vektors $\mathbf{X_k^{(c|i)}} - \mathbf{C^{(c)}}$ und der Skalierung des Ergebnisses mit der anderweitig

[2] aufgrund des Unterschieds zwischen Pixelposition und -anzahl: die Anzahl entspricht der um ein Pixel inkrementierten absoluten Differenz zweier Positionen, wenn man das Start- und Endpixel mitzählt

gewonnenen Entfernung von $\mathbf{X}_\mathbf{k}^{(\mathbf{c})}$ zum Projektionszentrum. Damit ist der ursprüngliche Sichtstrahl rechnerisch wiederhergestellt.

Bisher wurde implizit davon ausgegangen, dass das Koordinatensystem des Objekt- oder Weltraums mit dem der Kamera zusammenfällt. Dies ist jedoch meist nicht der Fall. Wenngleich beide als euklidisch angenommen werden können und der Einfachheit halber die gleiche (metrische) Skalierung besitzen sollen, so unterscheiden sie sich typischerweise in ihren Ursprungspunkten und der Drehung zueinander. Die Ausrichtung der Lochkamera in der Szene erfolgt daher durch deren Rotation um die drei Objektraumachsen, ausgedrückt mit den Winkeln ω (X-Achse), ϕ (Y-Achse) und κ (Z-Achse), sowie die Translation des Projektionszentrums $\mathbf{C}^{(\mathbf{w}|\mathbf{c})}$, aufgefasst als Teil der Welt. Dabei wird ein linkshändiges Weltkoordinatensystem[3] zugrunde gelegt, bei dem die Koordinatenachsen kongruent zum Kamerasystem in Abbildung 2.3 angeordnet sind. Der räumliche Nullwinkel fällt dabei mit der Z-Achse zusammen, der Drehsinn ist mathematisch negativ (im Uhrzeigersinn).

Bei differierenden Bezugspunkten und Winkelangaben muss also *vor* der Anwendung der Projektionsvorschrift für jeden abzubildenden Objektpunkt $\mathbf{X}_\mathbf{k}^{(\mathbf{w})}$ eine Translation $-\mathbf{t}^{(\mathbf{w})} = \mathbf{O}^{(\mathbf{w})} - \mathbf{C}^{(\mathbf{w}|\mathbf{c})}$ des Weltkoordinatenursprungs $\mathbf{O}^{(\mathbf{w})}$ in den Kamerasystemursprung $\mathbf{C}^{(\mathbf{w}|\mathbf{c})}$ erfolgen, der sich eine Rotation $\mathbf{R}_- = \mathbf{R}_{-\omega}\mathbf{R}_{-\phi}\mathbf{R}_{-\kappa}$ anschließt. Man erhält:

$$\mathbf{X}_\mathbf{k}^{(\mathbf{c})} = \mathbf{R}_-(\mathbf{X}_\mathbf{k}^{(\mathbf{w})} - \mathbf{t}^{(\mathbf{w})}) \tag{2.7}$$

mit den bekannten Drehmatrizen

$$\mathbf{R}_\omega = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\omega & -\sin\omega \\ 0 & \sin\omega & \cos\omega \end{pmatrix} \quad \mathbf{R}_\phi = \begin{pmatrix} \cos\phi & 0 & \sin\phi \\ 0 & 1 & 0 \\ -\sin\phi & 0 & \cos\phi \end{pmatrix} \quad \mathbf{R}_\kappa = \begin{pmatrix} \cos\kappa & -\sin\kappa & 0 \\ \sin\kappa & \cos\kappa & 0 \\ 0 & 0 & 1 \end{pmatrix} .$$

Ebenso erfordert die inverse Projektion, die die Szenen- aus der Pixelposition berechnet, eine *nachgelagerte* Rotation $\mathbf{R} = \mathbf{R}_\kappa\mathbf{R}_\phi\mathbf{R}_\omega$ in umgekehrter Reihenfolge und die Translation $\mathbf{t}^{(\mathbf{w})} = \mathbf{C}^{(\mathbf{w}|\mathbf{c})} - \mathbf{O}^{(\mathbf{w})}$ des Kamerasystemursprungs in den Weltkoordinatenursprung:

$$\mathbf{X}_\mathbf{k}^{(\mathbf{w})} = \mathbf{R}\mathbf{X}_\mathbf{k}^{(\mathbf{c})} + \mathbf{t}^{(\mathbf{w})} . \tag{2.8}$$

Zusammengefasst lässt sich die planare Lochkamera demnach mit sechs internen Kameraparametern (innere Orientierung) und sechs Werten für die Positionierung und Drehung des Modells in der Szene (äußere Orientierung) beschreiben. Damit ergibt sich für die vollständige Abbildungsvorschrift in Hinrichtung

$$\mathbf{X}_\mathbf{k}^{(\mathbf{i})} = M_{plproj}(\mathbf{R}_-(\mathbf{X}_\mathbf{k}^{(\mathbf{w})} - \mathbf{t}^{(\mathbf{w})})) \tag{2.9}$$

bzw. in Rückrichtung bei bekannter Objektpunktdistanz

$$\mathbf{X}_\mathbf{k}^{(\mathbf{w})} = \mathbf{R}M_{plproj^{-1}}(\mathbf{X}_\mathbf{k}^{(\mathbf{i})}) + \mathbf{t}^{(\mathbf{w})} . \tag{2.10}$$

[3] Diese Wahl erfolgt im Hinblick auf die Validierung der Abbildungsvorschriften mit künstlich erzeugten Referenzaufnahmen.

2.1.2.2. Zylindrische Lochkamera

Ganz analog zur planaren Lochkamera verhält sich das zylindrische Modell, das die hochauflösende Panoramakamera vereinfacht nachbildet. Hier besteht die Projektionsfläche aus einem Teil eines geraden Kreiszylindermantels, der in aufgeschnittener und ausgebreiteter Form mit den gleichen Parametern wie die Bildebene zuvor beschrieben werden kann (Abbildung 2.4). Die bei diesem Kameratyp ebenfalls gebräuchliche Definition der Abbil-

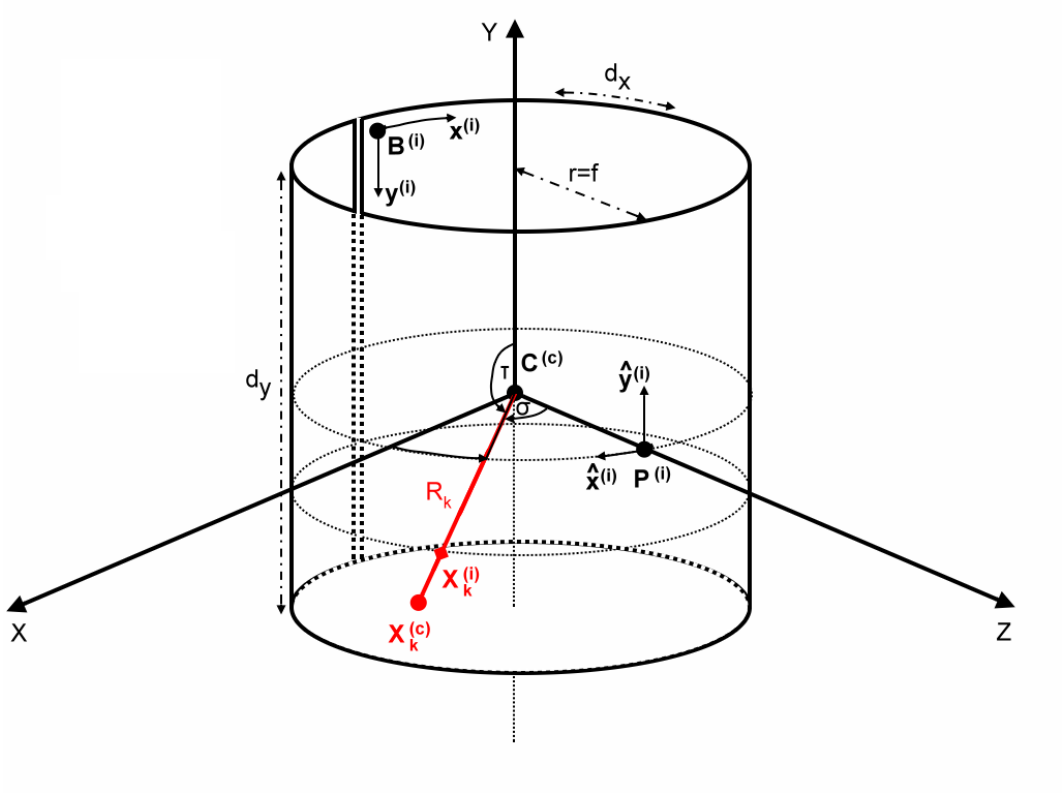


Abbildung 2.4.: Zylindrisches Lochkameramodell

dungsfläche über den horizontalen und vertikalen Sichtwinkel (horizontal/vertical field of view) σ und τ lässt sich über die trigonometrischen Winkelfunktionen sowie die Bogenlängenbeziehung $b = \alpha r$ mit der Bogenlänge b, dem Kreiswinkel α und dem Radius r auf die Anzahl der Pixel d_x bzw. d_y zurückführen zu

$$d_x = \frac{\sigma f}{d_{px}} \qquad\qquad d_y = \frac{2f \tan\frac{\tau}{2}}{d_{py}} \tag{2.11}$$

und in der Umkehrung

$$\sigma = \frac{d_x d_{px}}{f} \qquad\qquad \tau = 2\ \mathrm{atan2}(\frac{1}{2} d_y d_{py}, f)\ . \tag{2.12}$$

Dabei benötigt der Spezialfall des gestreckten vertikalen Winkels, für den die Tangensfunktion nicht definiert ist, eine gesonderte Behandlung. Weil im Rahmen dieser Arbeit ausschließlich sichtbare Objektpunkte verarbeitet werden, die sich nicht auf den Zylinderdeckflächen befinden, tritt er hier nicht auf. Der horizontale Winkel σ kann dagegen alle möglichen Werte im Intervall $[0, 2\pi)$ annehmen. Insbesondere sind auch überstumpfe

Sichtfelder zulässig, bei denen das planare Kameramodell versagt. Daher wird die Arkustangensfunktion durch die aus den Programmiersprachen entlehnte atan2-Funktion

$$\operatorname{atan2}(y,x) = \begin{cases} \arctan\frac{y}{x}, & x > 0, \\ \arctan\frac{y}{x} + \pi, & x < 0, y \geq 0 \\ \arctan\frac{y}{x} - \pi, & x < 0, y < 0 \\ +\frac{\pi}{2}, & x = 0, y > 0 \\ -\frac{\pi}{2}, & x = 0, y < 0 \\ 0, & x = 0, y = 0 \end{cases} \tag{2.13}$$

substituiert, die die Quadrantenbeziehung berücksichtigt und den Wertebereich $(-\pi, \pi]$ anstatt $(-\frac{\pi}{2}, \frac{\pi}{2})$ aufweist.

Streng genommen ist die Verwendung der Bogenlängenformel nicht ganz korrekt. Da diese „gebogene“ Pixel auf dem Zylindermantel induziert, die Bildpunkte den Kreisbogen jedoch linear approximieren, tritt ein Fehler auf. Für große Radien respektive Brennweiten im Vergleich zur Pixelgröße ist die Annäherung jedoch nahezu perfekt und kann in diesem Fall praktisch vernachlässigt werden. Da sie jedoch mit allen Umwandlungen von Winkeln in Pixel und umgekehrt einhergeht, sollte man sich dennoch stets vergewissern, dass es zu keiner exzessiven Fehleraddition kommt.

Projektionsvorschrift Die Projektionsvorschrift $M_{cylproj} : \mathbb{R}^3 \to \mathbb{R}^2$, die einen Objektpunkt $\mathbf{X}_\mathbf{k}^{(\mathbf{c})}$ im Kamerasystem in einen Bildpunkt $\mathbf{X}_\mathbf{k}^{(\mathbf{i})}$ auf dem Zylindermantel überführt, kommt in der Waagerechten ebenfalls nicht ohne die trigonometrischen Funktionen aus. So ergibt sich der Horizontalwinkel von $\mathbf{X}_\mathbf{k}^{(\mathbf{c})}$ mit der Z-Achse, welche per Definition den Kamerahauptpunkt $\mathbf{P}^{(\mathbf{i})}$ induziert, aus seinen Koordinaten zu

$$\tan\sigma = \frac{\mathbf{X}_\mathbf{k}^{(\mathbf{c})}(x)}{\mathbf{X}_\mathbf{k}^{(\mathbf{c})}(z)} \qquad \sigma = \operatorname{atan2}(\mathbf{X}_\mathbf{k}^{(\mathbf{c})}(x), \mathbf{X}_\mathbf{k}^{(\mathbf{c})}(z)) \tag{2.14}$$

Wenn man den Zylinder in Abbildung 2.4 von oben betrachtet, nimmt σ negative Werte links und positive Werte rechts von $\mathbf{P}^{(\mathbf{i})}$ an. Der Drehsinn ist also wiederum mathematisch negativ. Mit der durch den Winkel gegebenen Bogenlänge $b = f\sigma$ lässt sich ausgehend von $\mathbf{P}^{(\mathbf{i})}$ über die Pixelgröße d_{px} zunächst die hauptpunktbezogene Abszisse ermitteln. Die Translation des Ursprungspunkts von $\mathbf{P}^{(\mathbf{i})}$ nach $\mathbf{B}^{(\mathbf{i})}$ liefert alsdann die endgültige Bildkoordinate in der Waagerechten:

$$\mathbf{X}_\mathbf{k}^{(\mathbf{i})}(x) = \frac{b}{d_{px}} + \mathbf{P}^{(\mathbf{i})}(x) = \frac{f\sigma}{d_{px}} + \mathbf{P}^{(\mathbf{i})}(x) \ . \tag{2.15}$$

Die Ordinate kann man fast wie im planaren Fall mit dem Strahlensatz bestimmen, indem eine virtuelle Bildebene tangential am Schnittpunkt des Sichtstrahls mit dem Zylindermantel konstruiert wird. Diese liegt allerdings nicht zwingend parallel zur X-Y-

Koordinatenebene, weswegen man hier die „echte“ euklidische Objektdistanz ansetzen muss und nicht einfach mit $\mathbf{X}_\mathbf{k}^{(\mathbf{c})}(z)$ als Divisor abstrahieren kann. Damit folgt

$$\frac{\mathbf{X}_\mathbf{k}^{(\mathbf{c})}(y)}{\sqrt{\mathbf{X}_\mathbf{k}^{(\mathbf{c})}(x)^2 + \mathbf{X}_\mathbf{k}^{(\mathbf{c})}(z)^2}} = \frac{\mathbf{X}_\mathbf{k}^{(\mathbf{c}|\mathbf{i})}(y)}{f} \qquad \mathbf{X}_\mathbf{k}^{(\mathbf{c}|\mathbf{i})}(y) = \frac{f\mathbf{X}_\mathbf{k}^{(\mathbf{c})}(y)}{\sqrt{\mathbf{X}_\mathbf{k}^{(\mathbf{c})}(x)^2 + \mathbf{X}_\mathbf{k}^{(\mathbf{c})}(z)^2}} \tag{2.16}$$

für die hauptpunktbezogene vertikale Position, die sich anschließend durch

$$\mathbf{X}_\mathbf{k}^{(\mathbf{i})}(y) = d_y - 1 - (\frac{\mathbf{X}_\mathbf{k}^{(\mathbf{c}|\mathbf{i})}(y)}{d_{py}} + \mathbf{P}^{(\mathbf{i})}(y)) \tag{2.17}$$

subpixelgenau vom Ursprung $\mathbf{P}^{(\mathbf{i})}$ in das übliche Bildkoordinatensystem mit Zentrum $\mathbf{B}^{(\mathbf{i})}$ transformieren lässt.

Inverse Projektionsvorschrift Analog zur planaren Lochkamera gestaltet sich die Vorgehensweise bei der mehrdeutigen zylindrischen Rückprojektion $M_{cylproj^{-1}} : \mathbb{R}^2 \to \mathbb{R}^3$. In der Horizontalen muss dafür zunächst der Winkel mit der Z-Achse aus der waagerechten Pixelposition berechnet werden. Diesen erhält man nach der Rücktransformation in das hauptpunktbezogene Bildkoordinatensystem via

$$\sigma = \frac{d_{px}}{f}(\mathbf{X}_\mathbf{k}^{(\mathbf{i})}(x) - \mathbf{P}^{(\mathbf{i})}(x)) \ . \tag{2.18}$$

In der Vertikalen ergibt sich aus der Umkehrung von Gleichung 2.17 die Ordinate auf dem Zylindermantel zu

$$\mathbf{X}_\mathbf{k}^{(\mathbf{c}|\mathbf{i})}(y) = d_{py}(d_y - 1 - \mathbf{X}_\mathbf{k}^{(\mathbf{i})}(y) - \mathbf{P}^{(\mathbf{i})}(y)) \ . \tag{2.19}$$

Ist nun die Entfernung d des Objektpunktes $\mathbf{X}_\mathbf{k}^{(\mathbf{c})}$ von Kamerazentrum $\mathbf{C}^{(\mathbf{c})}$ bekannt, so lassen sich $\mathbf{X}_\mathbf{k}^{(\mathbf{c})}(x)$ und $\mathbf{X}_\mathbf{k}^{(\mathbf{c})}(z)$ anhand des Horizontalwinkels bestimmen. Es genügt, den Basiseinheitsvektor $\mathbf{e}_\mathbf{z}^{(\mathbf{c})}$ entlang der Z-Achse des Kamerasystems mit d zu skalieren und das Ergebnis anschließend um σ um die Y-Achse desselbigen zu rotieren oder äquivalent erst zu rotieren und dann zu skalieren:

$$\mathbf{X}_\mathbf{k}^{(\mathbf{c})}(x) = d\mathbf{R}_\sigma\mathbf{e}_\mathbf{z}^{(\mathbf{c})} = d\begin{pmatrix} \cos\sigma & 0 & \sin\sigma \\ 0 & 1 & 0 \\ -\sin\sigma & 0 & \cos\sigma \end{pmatrix}\mathbf{e}_\mathbf{z}^{(\mathbf{c})} \ . \tag{2.20}$$

Die noch fehlende Koordinate $\mathbf{X}_\mathbf{k}^{(\mathbf{c})}(y)$ wird rekonstruiert, indem man die Ordinate des Punktes $\mathbf{X}_\mathbf{k}^{(\mathbf{c}|\mathbf{i})}$) auf dem Zylindermantel ins Verhältnis zur Entfernung setzt. Weil sich die Projektionsfläche im bekannten Brennweitenabstand f befindet, ergibt sich

$$\mathbf{X}_\mathbf{k}^{(\mathbf{c})}(y) = \frac{d}{f}\mathbf{X}_\mathbf{k}^{(\mathbf{c}|\mathbf{i})}(y) \ , \tag{2.21}$$

was der Vorgehensweise zur Wiederherstellung des Sichtstrahls beim planaren Lochkameramodell ähnelt. Vollends identisch zur diesem gestaltet sich die Beschreibung der äußeren Orientierung als affine Transformation, die die Zylinderkamera in den dreidimensionalen Weltobjektraum einbettet.

2.1.3. Stereogeometrie

Für die Bestimmung der Objektentfernung und damit der Tiefenstruktur einer Szene sind bei der Photogrammetrie mindestens zwei Aufnahmen von unterschiedlichen Standpunkten erforderlich. Dabei ist es unerheblich, ob diese durch mehrere statische Kameras im Raum oder aus der Bewegung einer einzelnen Kamera resultieren. Beschränkt man sich auf die minimal notwendige Bildanzahl, wie es in dieser Arbeit durchweg der Fall ist, ergibt sich für ein solches Stereosystem der in Abbildung 2.5 skizzierte geometrische Zusammenhang.

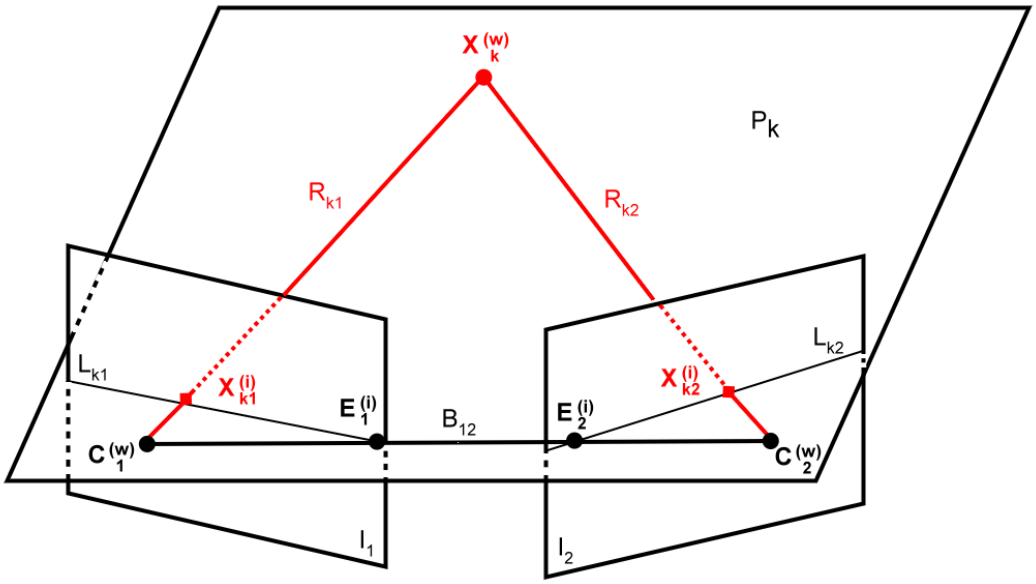


Abbildung 2.5.: Stereogeometrie mit planaren Projektionsflächen

Wie zu erkennen ist, sind die beiden real vorhandenen oder virtuellen Kameras (durch die Bewegung „erzeugt“) so in der Szene platziert, dass sie das gleiche Objekt erfassen. Aufgrund der unterschiedlichen Standorte hat dies zur Folge, dass die Szenenelemente in der zweiten Aufnahme I_2 im Vergleich zum ersten Bild I_1 verschoben sind. Diese Verschiebung ist entfernungsabhängig und wird als Disparität bezeichnet. Anhand des Daumensprungs (der senkrecht aufgestellte Daumen vor nur einem geöffneten Auge ändert seine Position, wenn man das geschlossene Auge öffnet und das vorher geöffnete Auge schließt) lässt sich der Effekt gut nachvollziehen.

Die Projektionszentren $\mathbf{C_1^{(w)}}$ und $\mathbf{C_2^{(w)}}$ der beiden Kameras in Weltkoordinaten befinden sich auf einer gedachten Verbindungslinie mit positiver Länge, der Basislinie B_{12}. Betrachtet man nun verschiedene Objektpunkte $\mathbf{X_k^{(w)}}$, so definieren diese zusammen mit der Basislinie jeweils eine Ebene P_k, in der sich als logische Konsequenz auch die Sichtstrahlen R_{k1} bzw. R_{k2} befinden. Vektoriell lässt sich diese Tatsache in Form der sogenannten *Komplanaritätsbedingung* ausdrücken:

$$\exists a, b, c \in \mathbb{R}; (a \neq 0) \vee (b \neq 0) \vee (c \neq 0) : a\mathbf{B}_{\mathbf{12}}^{(\mathbf{w})} + b\mathbf{R}_{\mathbf{k1}}^{(\mathbf{w})} + c\mathbf{R}_{\mathbf{k2}}^{(\mathbf{w})} = \mathbf{0}^{(\mathbf{w})} \ . \tag{2.22}$$

Übertragen auf den Bildraum führt die Komplanaritätsbedingung dazu, dass sich der Bildpunkt $\mathbf{X}_{\mathbf{k}}^{(\mathbf{i})}$ des Objektpunktes $\mathbf{X}_{\mathbf{k}}^{(\mathbf{w})}$ in der jeweils anderen Kamera nur entlang derjenigen eindimensionalen Kurve befinden kann, welche die Schnittmenge von P_k mit der jeweiligen Projektionsfläche repräsentiert. Legt man wie abgebildet das planare Lochkameramodell zugrunde, so besteht diese als Epipolarlinie (altgriechisch epi=auf) bezeichnete Kurve jeweils aus einer Geraden L_{k1} bzw. L_{k2}.

Wenn die Stereobilder mit zueinander geneigten Kameras aufgezeichnet werden, so schneiden sich alle Epipolarlinien einer Aufnahme I_1 bzw. I_2 in je einem speziellen Punkt, dem Epipol $\mathbf{E}_{\mathbf{1}}^{(\mathbf{i})}$ bzw $\mathbf{E}_{\mathbf{2}}^{(\mathbf{i})}$. Die Epipole entsprechen dabei gerade den Durchstoßpunkten der invarianten Basislinie B_{12} durch die Bildflächen und können sich außerhalb des Sichtfeldes befinden. Um $\mathbf{E}_{\mathbf{1}}^{(\mathbf{i})}$ bzw. $\mathbf{E}_{\mathbf{2}}^{(\mathbf{i})}$ zu berechnen, genügt es folglich, das komplementäre Projektionszentrum $\mathbf{C}_{\mathbf{2}}^{(\mathbf{w})}$ bzw. $\mathbf{C}_{\mathbf{1}}^{(\mathbf{w})}$ der entsprechenden (zugegebenermaßen leicht zweckentfremdeten) Abbildungsvorschrift $M_{plproj1}$ bzw. $M_{plproj2}$ für „gewöhnliche“ Objektpunkte zu unterwerfen. Den zweiten Punkt für die vollständige algebraische Beschreibung der Epipolarlinie gewinnt man aus dem zum Objektpunkt $\mathbf{X}_{\mathbf{k}}^{(\mathbf{w})}$ gehörenden Bildpunkt in Weltkoordinaten $\mathbf{X}_{\mathbf{k2}}^{(\mathbf{w}|\mathbf{i})}$ bzw. $\mathbf{X}_{\mathbf{k1}}^{(\mathbf{w}|\mathbf{i})}$ nach dem gleichen Prinzip.

Eine alternative Methode, um die Koordinaten diskreter Punkte auf L_{k1} respektive L_{k2} zu erhalten, basiert direkt auf der Epipolarebenendefinition. Sie ist insbesondere für Softwareimplementierungen prädestiniert und hat den Vorteil, mit beliebigen Projektionsflächen zu funktionieren. Dabei bestimmt man zunächst die auch als Epipolarebene bezeichnete Fläche P_k, etwa über die Basislinie und den Punkt $\mathbf{X}_{\mathbf{k1}}^{(\mathbf{w}|\mathbf{i})}$ bzw. $\mathbf{X}_{\mathbf{k2}}^{(\mathbf{w}|\mathbf{i})}$. Anschließend konstruiert man diejenige Bildspalte in Weltkoordinaten, für welche man den Epipolarlinienpunkt sucht. Hierzu werden zwei beliebige Spaltenpunkte mit identischer Abszisse und der Brennweite als Entfernung an die inverse Projektionsvorschrift $M_{plproj1^{-1}}$ bzw. $M_{plproj2^{-1}}$ übergeben. Die beiden resultierenden Koordinatentripel beschreiben eine dreidimensionale Gerade. Deren Schnittpunkt mit der Ebene P_k ergibt nach Projektion mittels $M_{plproj1}$ bzw. $M_{plproj2}$ zurück in Bildkoordinaten die gewünschte Position auf L_{k1} bzw. L_{k2}. Berechnet man zwei Epipolarlinienpunkte auf diese Weise, ist wiederum die gesamte Linie eindeutig bestimmt.

Wenn die Kameras nicht geneigt sind und sich die Bildebenen folglich in paralleler Lage befinden, dann verlaufen auch die Epipolarlinien in den Bildern I_1 und I_2 im gleichen Abstand zueinander. Die Epipole $\mathbf{E}_{\mathbf{1}}^{(\mathbf{i})}$ bzw. $\mathbf{E}_{\mathbf{2}}^{(\mathbf{i})}$ *können* sich dann im Unendlichen befinden. In diesem Fall lassen sie sich nicht durch die Projektion von $\mathbf{C}_{\mathbf{2}}^{(\mathbf{w})}$ respektive $\mathbf{C}_{\mathbf{1}}^{(\mathbf{w})}$ ermitteln.

Bei der Panoramakamera gestaltet sich die Sache komplizierter. Der hierbei durch die Epipolarebene induzierte Zylinderschnitt in Gestalt einer Ellipse ergibt in ausgebreiteter Form eine Sinuskurve [5]. Diese lässt sich grundsätzlich mit denselben Mitteln wie beim skizzierten planaren Modell konstruieren. Allerdings treten bei hinreichend großem horizontalen Aufnahmewinkel zwei Epipole je Kamera auf, durch welche sämtliche Epipolarlinien verlaufen, wenn man die Basislinie über das Projektionszentrum hinaus verlängert. Abbildung 2.6 veranschaulicht den Sachverhalt.

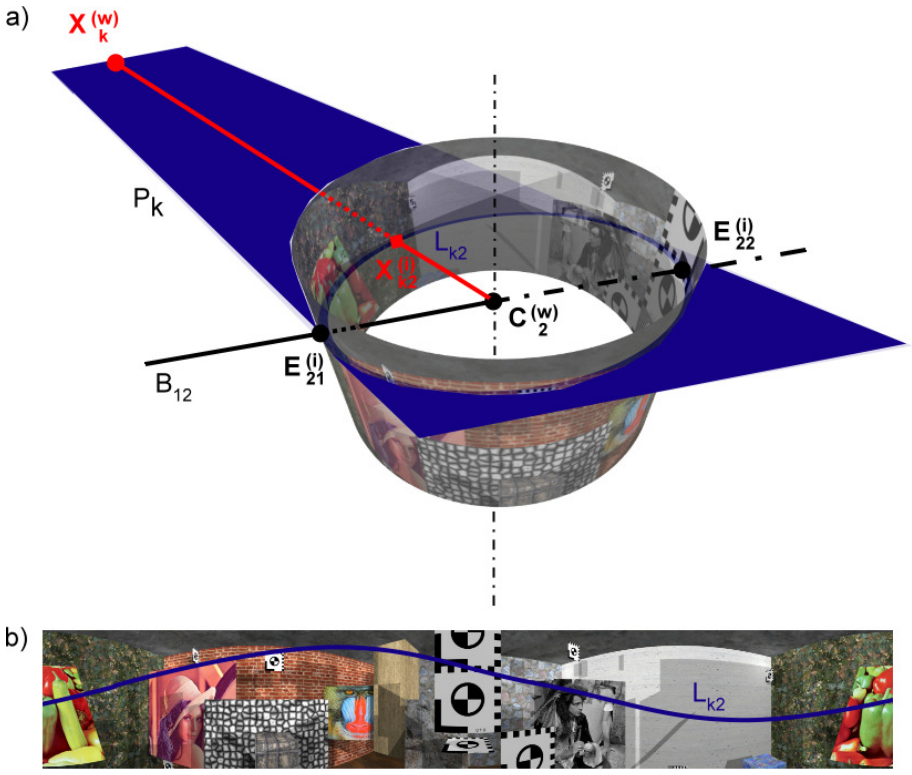


Abbildung 2.6.: (a) Schnitt der Epipolarebene mit der zylindrischen Bildfläche der zweiten Panoramakamera. Es treten zwei Epipole $\mathbf{E}_{\mathbf{21}}^{(\mathbf{i})}$ und $\mathbf{E}_{\mathbf{22}}^{(\mathbf{i})}$ auf, wenn man die Basislinie B_{12} verlängert. (b) Die sich daraus ergebende sinusförmige Epipolarlinie auf dem in die Ebene ausgebreiteten Bild

Aufgrund der Quadrik, welche als Schnittresultat entsteht, muss für die Herleitung der Epipolarlinien in geschlossener Parameterdarstellung ein weitaus größerer mathematischer Aufwand betrieben werden [5][6]. Die alternative Methode unter Verwendung der Bildspalten in Weltkoordinaten kann hier schneller ein in vielen Situationen hinreichend genaues Ergebnis liefern. Dafür müssen die planare Hin- und Rückprojektion lediglich durch ihre zylindrischen Gegenstücke substituiert werden. Die resultierenden diskreten Epipolarpunkte lassen sich bei Bedarf zu einer stetigen Funktion vervollständigen, etwa durch stückweise lineare oder polynomielle Interpolation.

2.1.4. Objektrekonstruktion

Die Rückgewinnung der Raumpunkte eines Weltobjekts setzt voraus, dass sowohl die internen Parameter als auch die äußeren Orientierungen der beiden Kameras des Stereosystems bekannt sind. Um diese Angaben zur erhalten, ist es erforderlich, die Aufnahmegeräte zunächst mit speziellen Testmustern zu kalibrieren und eine Verbindung zu bekannten 3D-Positionen herzustellen. Hierfür gibt es eine Reihe von Methoden, denen sich die einschlägige Fachliteratur widmet [4][7][8][9][10] und die daher an dieser Stelle nicht weiter diskutiert werden sollen. Wenn beide Orientierungen berechnet sind, lassen sich die Objektpunkte $\mathbf{X}_{\mathbf{k}}^{(\mathbf{w})}$ allein aus ihren Abbildern $\mathbf{X}_{\mathbf{k1}}^{(\mathbf{i})}$ und $\mathbf{X}_{\mathbf{k2}}^{(\mathbf{i})}$ in den Aufnahmen I_1 bzw. I_2 rekonstruieren, sofern eine Zuordnung $\mathbf{X}_{\mathbf{k1}}^{(\mathbf{i})} \leftrightarrow \mathbf{X}_{\mathbf{k2}}^{(\mathbf{i})}$ gefunden werden kann (*Korrespondenzproblem*).

2.1.4.1. Korrespondenzproblem und Matching

Wenngleich Menschen in der Lage sind, das im Allgemeinen schlecht gestellte Korrespondenzproblem zumeist spielerisch einfach durch scharfes „Hinschauen“ zu lösen[4], ist es bis heute nicht gelungen, *den* generischen effizienten automatisierbaren Algorithmus für die qualitativ hochwertige Zuordnung von Bildpunkten anzugeben. Als Hauptschwierigkeiten bei der Entwicklung eines solchen Verfahrens erweisen sich vor allem unterschiedliche Lichtverhältnisse (Schatten), projektionsbedingte Verzerrungen und Verdeckungen in den Stereoaufnahmen. Um übereinstimmende Bildstrukturen zumindest näherungsweise detektieren zu können (zu *matchen*), ist man gezwungen, vereinfachende Kompromisse einzugehen, welche allerdings Zuordnungsfehler provozieren.

Die klassischen Matchingverfahren nehmen sich des Korrespondenzproblems durch einen Intensitäts- bzw. Farbwertvergleich an, der aufgrund des geringeren Mehrdeutigkeitspotentials nicht pixel- sondern blockweise erfolgt. Dabei wird ein einfach zu beschreibendes und deswegen meist rechteckiges Fenster W_{cp} um den Vorlagenpunkt $\mathbf{X}_{\mathbf{k1}}^{(\mathbf{i})}$ im Musterbild I_1 der ersten Kamera definiert und versucht, ein vom Inhalt her möglichst ähnliches Suchfenster W_{cs} mit denselben Abmessungen im Suchbild I_2 der zweiten Kamera zu finden. Dies geschieht durch das sukzessive Verschieben von W_{cs} in einem abgegrenzten Suchbereich W_s von I_2 und die Berechnung des Übereinstimmungsgrades zur Schablone W_{cp}. Damit kommt die Vorgehensweise praktisch der Anwendung eines digitalen Filters gleich [11].

Über die gezielte Beeinflussung der Matchingparameter, etwa der Fenstergrößen und der Suchschrittweite innerhalb von W_s, lässt sich die Qualität der Zuordnungen in Abwägung mit der quadratisch wachsenden und schrumpfenden Ausführungsgeschwindigkeit austarieren. Der Zeitbedarf ist dabei auch ausschlaggebend für die Wahl des eingesetzten Ähnlichkeitskriteriums. Um zu entscheiden, wie groß die Korrelation zwischen dem Bildinhalt des Muster- und Suchfensters ist, kommen oftmals relativ einfache und schnell zu berechnende Abstandsmaße wie die Summe der absoluten Differenzen (sum of absolute differences, SAD) oder die Summe der quadrierten Differenzen (sum of squared differences, SSD) zum Einsatz. Diese sind wie folgt definiert:

$$SAD = \sum_{x,y \in W_{cp}} |I_1(x,y) - I_2(\Delta x_{W_{cs}} + x, \Delta y_{W_{cs}} + y)| \tag{2.23}$$

$$SSD = \sum_{x,y \in W_{cp}} (I_1(x,y) - I_2(\Delta x_{W_{cs}} + x, \Delta y_{W_{cs}} + y))^2 \ . \tag{2.24}$$

Hierbei ist das aktuelle Vergleichsfenster W_{cs} im zweiten Bild zum Musterfenster W_{cp} im ersten Bild um $\Delta x_{W_{cs}}$ bzw. $\Delta y_{W_{cs}}$ Bildelemente in den Grenzen des Suchbereichs W_s versetzt. Die beste Übereinstimmung befindet sich an derjenigen Bildposition innerhalb des Suchbereichs, an welcher die Funktionen den kleinsten Betrag zurückliefern. Bei der Minimierung über W_s haben SAD und SSD in zahlreichen Tests vergleichbare Ergebnisse erzielt wie rechenintensivere Methoden, beispielsweise die normierte Kreuzkorrelation (normalized cross correlation, NCC), sofern nur geringe radiometrische Störungen auftre-

[4] extra zu diesem Zweck abgedruckte Suchbilder kann man manchmal auf den Kinderseiten von Zeitungen finden

ten [12]. Diese lassen sich notfalls durch Vorverarbeitungsschritte wie die Helligkeits- und Kontrastanpassung oder Rauschunterdrückung reduzieren.

Für die Festlegung von W_s bietet es sich an, die geometrischen Zusammenhänge im Stereosystem auszunutzen. Da sich alle korrespondierenden Bildpunkte auf der zum Vorlagenpunkt gehörigen Epipolarlinie im zweiten Kamerabild befinden müssen, genügt es, den Suchbereich W_s auf ein schmales Band entlang ihres Verlaufs zu beschränken. Mit zusätzlichem Wissen über die ungefähre Distanz und der sich daraus ableitenden Raumposition der Objektpunkte kann man W_s weiter auf ein kleines Gebiet um eine konkrete Epipolarlinienposition eingrenzen.

2.1.4.2. Sichtstrahlrekonstruktion

Hat man schließlich eine Zuordnung $\mathbf{X}_{\mathbf{k1}}^{(\mathbf{i})} \leftrightarrow \mathbf{X}_{\mathbf{k2}}^{(\mathbf{i})}$ ermittelt, so lassen sich die ursprünglichen Sichtstrahlen R_{k1} und R_{k2} von den Projektionszentren $\mathbf{C}_{\mathbf{1}}^{(\mathbf{w})}$ und $\mathbf{C}_{\mathbf{2}}^{(\mathbf{w})}$ durch die in Weltkoordinaten ausgedrückten Objektpunktprojektionen $\mathbf{X}_{\mathbf{k1}}^{(\mathbf{w}|\mathbf{i})}$ bzw. $\mathbf{X}_{\mathbf{k2}}^{(\mathbf{w}|\mathbf{i})}$ im Idealfall wiederherstellen. Der dabei entstehende Schnittpunkt entspricht dann dem ursprünglichen Objektpunkt $\mathbf{X}_{\mathbf{k}}^{(\mathbf{w})}$. Dessen Entfernung zu einem beliebigen anderen Raumpunkt $\mathbf{X}_{\mathbf{0}}^{(\mathbf{w})}$ ergibt sich aus dem Betrag des Differenzvektors $\mathbf{X}_{\mathbf{k}}^{(\mathbf{w})} - \mathbf{X}_{\mathbf{0}}^{(\mathbf{w})}$. Typischerweise verwendet man als Messpunkte die Projektionszentren der Kameras $\mathbf{X}_{\mathbf{0}}^{(\mathbf{w})} = \mathbf{C}_{\mathbf{1}}^{(\mathbf{w})}$ bzw. $\mathbf{X}_{\mathbf{0}}^{(\mathbf{w})} = \mathbf{C}_{\mathbf{2}}^{(\mathbf{w})}$ oder die Mitte der Basislinie $\mathbf{X}_{\mathbf{0}}^{(\mathbf{w})} = \mathbf{C}_{\mathbf{1}}^{(\mathbf{w})} + \frac{1}{2}(\mathbf{C}_{\mathbf{2}}^{(\mathbf{w})} - \mathbf{C}_{\mathbf{1}}^{(\mathbf{w})})$.

In der Praxis ist aufgrund der unvollkommenen Zuordnungen meist nur eine näherungsweise Rekonstruktion der Sichtstrahlen möglich, die dann in der Regel windschief verlaufen. Für diesen Fall gibt es keinen dem Objektpunkt $\mathbf{X}_{\mathbf{k}}^{(\mathbf{w})}$ entsprechenden Schnittpunkt. Um dennoch eine Distanz wie dargestellt berechnen zu können, wird daher ein virtueller Schnittpunkt $\tilde{\mathbf{X}}_{\mathbf{k}}^{(\mathbf{w})}$ eingeführt, der exakt auf der Mitte der kürzesten Verbindung $\mathbf{d}_{\tilde{\mathbf{R}}_{\mathbf{k1}}\tilde{\mathbf{R}}_{\mathbf{k2}}}^{(\mathbf{w})}$ der aneinander vorbeilaufenden Sichtstrahlen $\tilde{R}_{k1}$ und $\tilde{R}_{k2}$ liegt und so den Entfernungsfehler minimiert. Die Bestimmung von $\tilde{\mathbf{X}}_{\mathbf{k}}^{(\mathbf{w})}$ erfolgt über die Durchstoßpunkte $\mathbf{X}_{\mathbf{S1}}^{(\mathbf{w})}$ und $\mathbf{X}_{\mathbf{S2}}^{(\mathbf{w})}$ von $\tilde{R}_{k1}$ und $\tilde{R}_{k2}$ durch die Ebene, die von $\tilde{R}_{k2}$ und $\mathbf{d}_{\tilde{\mathbf{R}}_{\mathbf{k1}}\tilde{\mathbf{R}}_{\mathbf{k2}}}^{(\mathbf{w})}$ bzw. $\tilde{R}_{k1}$ und $\mathbf{d}_{\tilde{\mathbf{R}}_{\mathbf{k1}}\tilde{\mathbf{R}}_{\mathbf{k2}}}^{(\mathbf{w})}$ aufgespannt wird. Man erhält demnach $\tilde{\mathbf{X}}_{\mathbf{k}}^{(\mathbf{w})} = \mathbf{X}_{\mathbf{S1}}^{(\mathbf{w})} + \frac{1}{2}(\mathbf{X}_{\mathbf{S2}}^{(\mathbf{w})} - \mathbf{X}_{\mathbf{S1}}^{(\mathbf{w})})$ und daraus wie oben dargelegt die Entfernung.

Die Speicherung der rekonstruierten Punkte kann nach der Berechnung direkt als dreidimensionaler Koordinatensatz erfolgen. Aufgrund des hohen Platzbedarfs und der Erkenntnis, dass die Korrespondenzinformationen mit den Kameraparametern zur Wiederherstellung der Raumelemente ausreichen, werden oftmals lediglich die Beschreibung des Stereosystems und die Differenzen der Pixelpositionen im Muster- und Suchbild abgelegt. Diese Verschiebungs- oder Disparitätskarten (disparity maps) haben den Vorteil, in Form von (Gleitkomma-)Intensitätsbildern einer visuellen Bewertung nahezu direkt zugänglich zu sein (Abbildung 2.7). Daneben lassen sich die bekannten und erprobten Bildverarbeitungsalgorithmen zur Manipulation einsetzen.

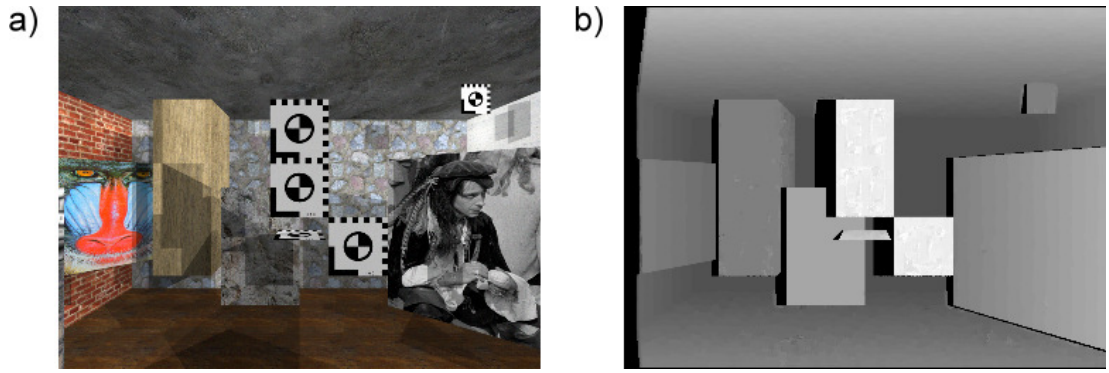


Abbildung 2.7.: (a) Musterbild eines planaren Stereosystems (b) die zugehörige visualisierte horizontale Disparitätskarte (keine Verschiebungen in der Vertikalen) nach dem Matching

2.1.4.3. Auflösung und Zeitbedarf

Ein wesentlicher Vorteil für den Einsatz von photogrammetrischen Verfahren zur Objekterfassung und anschließenden Rekonstruktion ist die erzielbare hohe räumliche Auflösung. Dies gilt sowohl für die Anzahl der 3D-Punkte als auch deren Abweichung von der realen Position. Maßgeblich verantwortlich für diesen positiven Aspekt ist die große Pixeldichte der modernen digitalen Bildsensoren mit horizontalen und vertikalen Abmessungen d_{px} bzw. d_{py} im Mikrometerbereich. Aber auch erweiterte Matchingalgorithmen, die unter günstigen Voraussetzungen subpixelgenaue Bildpunktzuordnungen ermitteln können, spielen hierbei eine nicht unbedeutende Rolle.

Auflösung Welche dreidimensionalen Auflösungen sich in Bezug auf die Szenenelemente theoretisch mit einem Stereosystem erzielen lassen, soll exemplarisch anhand einer ausgewählten Kamerakonfiguration abgeschätzt werden. Dabei sei ein zylindrisches Stereosystem angenommen, das parallel zur X-Z-Koordinatenebene liegt (Abbildung 2.8). Die Basislinie B_{12} der Länge d_B und der Sichtstrahl R_{k1} vom ersten Projektionszentrum $\mathbf{C_1^{(w)}}$ zum Objektpunkt $\mathbf{X_k^{(w)}}$ bilden einen rechten Winkel. Dieser soll ohne Beschränkung der Allgemeinheit im Weltkoordinatenursprung entlang der X- und Z-Achse liegen. Weiterhin werden die interne respektive externe Orientierung als exakt bekannt vorausgesetzt. Betrachtet man nun eine Fehlzuordnung durch das eingesetzte Matchingverfahren von n Pixeln ausschließlich in der Horizontalen, so ergibt sich über die Winkelbeziehungen und die Bogenlängenformel der Zusammenhang

$$\begin{aligned} \alpha_1 &= \operatorname{atan2}(d, d_B) & \tan\alpha_1 &= \frac{d}{d_B} \\ \alpha_2 &= \alpha_1 + \frac{n d_{px}}{f} & \tan\alpha_2 &= \frac{d'}{d_B} \end{aligned} \tag{2.25}$$

$$\Delta d = d' - d = d_B(\tan\alpha_2 - \tan\alpha_1) \ .$$

Man beachte hierbei, dass die projektionsbedingte Verzerrung bei dieser Konstellation *keine* Berücksichtigung finden muss, da bei der Zylinderkamera die horizontalen Pixelwinkel identisch sind. Für den vertikalen Zuordnungsfehler gilt dies nicht. Vielmehr *steigt* hier die

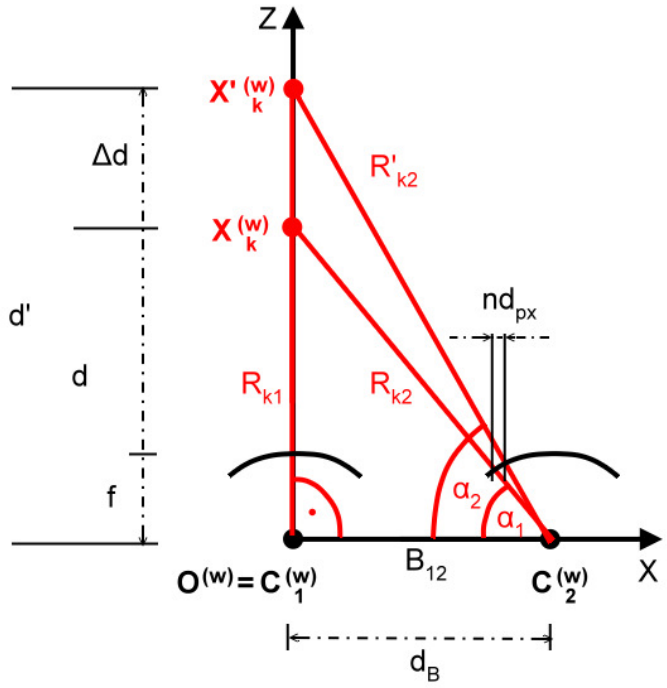


Abbildung 2.8.: Beispielkonfiguration zur Abschätzung der Auswirkung von fehlerhaften Punktkorrespondenzen

Auflösung zum oberen bzw. unteren Bildrand hin, da die Größe der Pixelwinkel abnimmt und so pro Bildpunkt ein kleinerer Szenenausschnitt abgebildet wird. Die minimale theoretische Präzision erhält man in der Senkrechten demzufolge in der unmittelbaren Umgebung des Kamerahauptpunktes. Beim planaren Lochkameramodell tritt dieser Effekt in beiden Bildrichtungen auf.

Zusätzlich zum konstanten Pixelwinkel garantiert die Beschränkung auf waagerechte Zuordnungen, dass der induzierte fehlerhafte Raumschnittpunkt $\mathbf{X}_{\mathbf{k}}^{\prime(\mathbf{w})}$ der Sichtstrahlen R_{k1} und R'_{k2} in der gleichen Epipolarebene liegt wie sein korrektes Gegenstück $\mathbf{X}_{\mathbf{k}}^{(\mathbf{w})}$. Berücksichtigt man auch Korrespondenzfehler in vertikaler Bildrichtung, ist abgesehen von der differierenden erreichbaren Raumpunktpräzision eine komplanare Lage der Strahlvektoren unwahrscheinlich. Die Verwendung eines virtuellen Raumschnittpunkts in der Mitte der Abstandsgeraden zur Fehlerminimierung lässt allerdings keine signifikante Abweichung zur Auflösung in Richtung der Z-Achse erwarten, wie sie aus einer horizontalen Fehlzuordnung resultiert.

Für die am DLR verwendete Panoramakamera ergeben sich auf Basis der technischen Daten in Tabelle 2.1 die Pixelabmessungen zu $d_{px} = d_{py} = 7\ \mu$m. Sie kann mit einem Objektiv der mittleren Brennweite $f = 60$ mm ausgestattet werden. Für die Länge der Basislinie B_{12} sei ein durchaus realistischer Wert von $d_B = 1$ m veranschlagt sowie eine Objektpunktentfernung $d = 10$ m zum Projektionszentrum $\mathbf{C}_{\mathbf{1}}^{(\mathbf{w})}$ vorgegeben. Mit dieser Konfiguration

berechnet sich die Abweichung, welche aus der horizontalen Falschzuordnung von einem Pixel resultiert, zu

$$\begin{aligned} \alpha_1 &= 84,289407^\circ \\ \alpha_2 &= \alpha_1 + 0,006685^\circ = 84.296092^\circ \\ \Delta d &= 0,011798 \text{ m} \approx 1,2 \text{ cm} . \end{aligned} \tag{2.26}$$

Die Raumpunktauflösung für diesen repräsentativen Parametersatz liegt demnach im einstelligen Zentimeterbereich. Eine weitere Steigerung ist durch den Gebrauch von subpixelgenauen Matchingverfahren möglich. Dabei werden auch die durch einen Interpolationsalgorithmus zwischen den vorhandenen Bildpunkten liegenden Pixel genutzt, wodurch die Granularität für die genannten Kamera- und Geometrieparameter schließlich bis in den Millimeterbereich hinein absinkt.

Zeitaufwand Die hohe Präzision, die sich mit der photogrammetrischen Objektrekonstruktion erreichen lässt, hat ihren Preis im zeitlichen Aufwand insbesondere für die Korrespondenzanalyse. Je nach Bildgröße können die zahlreichen pixelweise durchgeführten Vergleichsoperationen Stunden oder gar Tage in Anspruch nehmen. So sind für ein Stereosystem mit Aufnahmen von jeweils 1.000 x 1.000 Bildpunkten und Matchingfenstern von je 20 x 20 Pixeln pro Musterbildposition bereits $\approx 400.000 = 4 \cdot 10^5$ Suchbildpunkte aus dem Speicher zu lesen – in der beschleunigten Variante der Zuordnungssuche entlang der Epipolarlinie. Zusätzlich ist ≈ 1.000 mal die Abstandsfunktion zu berechnen. Sucht man für jedes Musterpixel eine Zuordnung, ergeben sich $\approx 4 \cdot 10^5 \cdot 10^6 = 4 \cdot 10^{11}$ Speicherzugriffe und $\approx 1.000 \cdot 10^6 = 10^9$ oder eine Milliarde Ähnlichkeitsberechnungen. Die quadratisch wachsende Komplexität in Bezug auf die Seitenlänge der Bilder und der für das Matching verwendeten Fenster fordert selbst bei den hohen Taktfrequenzen von modernen Computern im Gigahertzbereich ihren Tribut.

2.2. 3D-Laserscanning

Laserscanner (laser range finder, LRF) dienen zur direkten Erfassung von Entfernungen. Dazu verwenden sie zeitlich und räumlich kohärentes Laserlicht. Dieses weist sowohl eine gleichbleibende Schwingungsfrequenz und -phasenlage als auch eine geringe Strahlaufweitung (Divergenz) über weite Strecken auf, was für die Distanzmessung und die anschließende Objektrekonstruktion vorteilhaft ist.

2.2.1. Aufbau und Distanzbestimmung

Vom Aufbau her bestehen 3D-Laserscanner gewöhnlich aus einem Kopfmodul, das auf einem Stativ montiert ist und in horizontaler Richtung rotiert [13]. Die Drehung erfolgt dabei elektronisch durch präzise steuerbare Servomotoren, wobei sich der Rotationsbereich und die Winkelschrittweite vorgeben lassen. Innerhalb des Scankopfes befindet sich die Laserlichtquelle einschließlich der Fokussier- und vertikalen Ablenkoptik. Letztere ermöglicht die spaltenweise Objekterfassung und kann beispielsweise mit rotierenden Spiegeln oder nach dem Tachymeterprinzip realisiert sein. Auch hier besteht die Möglichkeit, die

gewünschte Auflösung in der Senkrechten vorzugeben. Im Zusammenspiel mit der horizontalen Drehbewegung wird somit bis auf die Standfläche praktisch das gesamte Umfeld des LRF erfasst. Abbildung 2.9 zeigt das am DLR verwendete Modell IMAGER 5003, für welches Tabelle 2.2 einige ausgewählte technische Daten aufführt.

Um nun die Entfernung zu den verschiedenen Szenenobjekten zu erhalten, wird für jede Horizontal- und Vertikalposition innerhalb des gewählten Aufnahmebereichs ein kurzer Laserimpuls ausgesendet. Trifft dieser auf einen Gegenstand[5], so wird abhängig vom Einfallswinkel und der Materialbeschaffenheit ein bestimmter Anteil der Lichtenergie reflektiert. Anhand der Laufzeit des Impulses oder der Phasenverschiebung bestimmt der LRF einen Entfernungswert. Für alle Strahlrichtungen zusammen ergibt sich eine zweidimensionale Distanzkarte, die vom optischen Eindruck der photogrammetrischen Disparitätskarte ähnelt. Einige LRF-Typen sind außerdem in der Lage, die Intensität der Reflexion in ein Graustufenbild umzusetzen. Abbildung 2.10 illustriert den Scanprozess.

Gut zu erkennen ist der diskrete Charakter der laserbasierten Objekterfassung im Gegensatz zur integrierenden Aufzeichnung mit dem Stereokamerasystem. Durch die winkeläquidistant arbeitende Auslenkungsmechanik reduziert sich zudem die Messpunktanzahl pro Fläche mit steigender Entfernung. Beides bewirkt, dass plötzliche Distanzunterschiede in der Szene (Objektkanten) unterabgetastet werden und somit Information verloren geht. Dies gilt auch für die farblichen Details der Szene und andere Materialeigenschaften (Textur, Reflektivität) aufgrund des aktiv emittierten kohärenten und damit monochromatischen Laserlichts. Entgegen der passiven Detektion von elektromagnetischen Wellen mit der Kamera reduziert die aktive „Beleuchtung“ allerdings den Einfluss der Umgebungsillumination. LRFs funktionieren daher auch in lichtarmen Innenräumen oder bei völliger Dunkelheit.

2.2.2. Objektrekonstruktion

Im Gegensatz zur photogrammetrischen Objektrekonstruktion ist beim Laserscanning keine nachgelagerte Korrespondenzanalyse notwendig. Vielmehr lassen sich die Raumpunkte aus den ermittelten richtungsbezogenen Entfernungen und der Laserposition praktisch direkt zurückgewinnen.

Die Rekonstruktion des ursprünglichen Objektpunktes $\mathbf{X}_{\mathbf{k}}^{(\mathbf{w})}$ erfolgt dabei ähnlich wie bei der inversen Projektion mit der zylindrischen Panoramakamera. Unter Annahme des linkshändigen Koordinatensystems für den Scanner wird zunächst die gemessene Distanz d als Vektor entlang der Z-Achse abgetragen. Anschließend berechnet man aus der Entfernungskartenposition und den Winkelinfomationen des LRF (absolute Angabe oder Inkrement) die horizontale und vertikale Strahlauslenkung σ respektive τ. Die Rotation des Distanzvektors um beide Winkel ergibt vorerst einen Raumpunkt relativ zum Standort des Laserscanners und nach der Einbettung in das Weltkoordinatensystem die finalen Koordinaten von $\mathbf{X}_{\mathbf{k}}^{(\mathbf{w})}$. Kehrt man den Prozess um, lässt sich aus dem Objektpunkt $\mathbf{X}_{\mathbf{k}}^{(\mathbf{w})}$ auch die zugehörige Position in der LRF-Distanzkarte zurückgewinnen.

Da diese Transformation vergleichsweise wenig Rechenaufwand erfordert, hängt der Zeitbedarf für die Gewinnung der Punktwolke einer Szene fast ausschließlich von der Durch-

[5] das Scannen von Lebewesen ist aufgrund der hohen Laserleistung der Gesundheit zuliebe nicht zu empfehlen



Abbildung 2.9.: Zoller und Fröhlich Laserscanner IMAGER 5003 (nach [14])

Laser	rot, 23 mW (Klasse 3R)
Strahldivergenz	0,22 mrad
Strahldurchmesser in 1 m Entfernung	3 mm kreisrund
Eindeutigkeitsbereich (maximale Messentfernung)	53,5 m
Minimale Messentfernung	1,0 m
Entfernungsauflösung	1,0 mm
Sichtfeld horizontal/vertikal	360°/310°
Winkelauflösung horizontal	0,01°
Winkelauflösung vertikal	0,018°
Scandauer für gesamtes Sichtfeld, mittlere Auflösung	100 s
Typische Datenerfassungsrate	125.000 Pixel/s
Maximalpixelzahl horizontal/vertikal auf 360°	20.000/20.000 Pixel

Tabelle 2.2.: Technische Daten des IMAGER 5003 mit Laser LARA 53500 (nach [14])

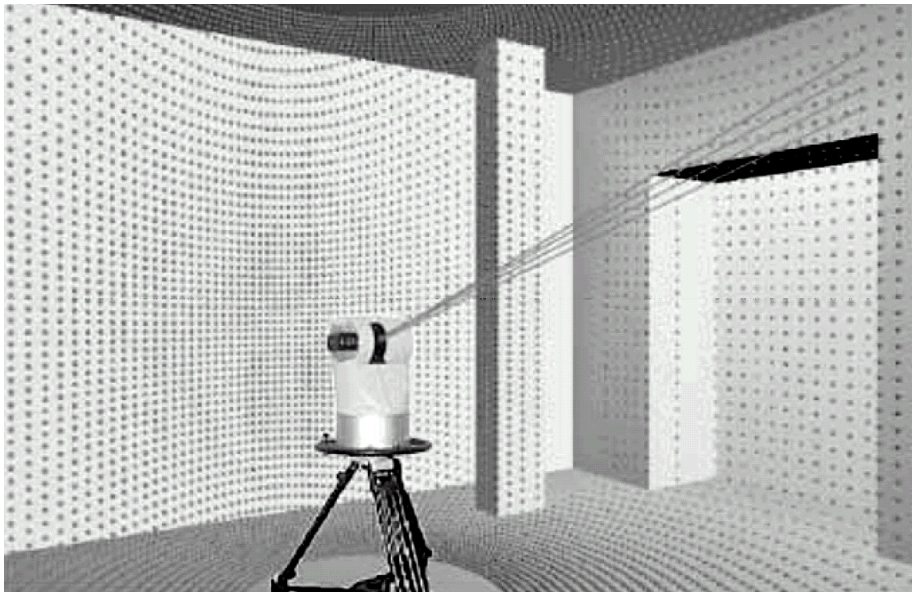


Abbildung 2.10.: Erstellung eines Laserscans (aus [13])

führung des Laserscans selbst ab. Entsprechend den technischen Daten des am DLR verwendeten LRF-Modells müssen für einen Durchgang je nach Punktzahl zwischen einer Minute bis hin zu knapp einer Stunde veranschlagt werden. Die erreichbare Entfernungsauflösung hängt dabei stark von den Materialeigenschaften und dem Einfallswinkel des Laserimpulses ab. Empirische Untersuchungen haben hierbei bezogen auf die Messentfernung von 10 m eine Standardabweichung von 5 mm bei günstigen Bedingungen ermittelt [15]. Flache Auftreffwinkel können diesen Wert allerdings negativ beeinflussen (Abweichungen um etwa 1 cm bei 20° Neigung gegen die Tangentialebene im Laserpunkt auf der Oberfläche). Gleiches gilt für (semi-)transparente und reflektierende Materialien. Hierfür zeigen die Messreihen Differenzen zum Sollwert von 1,5 cm für Styropor, bei einer metallenen Oberfläche kann aufgrund der hohen Intensität des zurückgeworfenen Laserstrahls auf 10 m allerdings kein gültiger Distanzwert ermittelt werden. Die horizontalen und vertikalen Winkelabweichungen sind meist geringer und liegen unter einem Millimeter [16].

2.3. Modellerzeugung

Aus den näherungsweise wiederhergestellten Objektpunkten des Laserscanners wie auch des Stereosystems können virtuelle dreidimensionale Modelle erstellt werden. Unter einem Modell versteht man dabei eine Menge von Polygonen, die die diskreten ausdehnungslosen Raumelemente vernetzen und zusätzlich mit einer Oberflächentextur versehen sein können. Für viele Anwendungen aus dem Vermessungs- und Rekonstruktionsbereich ist eine solche Darstellung aufgrund ihres geschlossenen Charakters besser geeignet als die schwer zu interpretierenden Punktwolken. Sie lässt sich mit entsprechender CAD-Software (computer aided design, computergestützter Entwurf) nicht nur auf intuitive Weise nachträglich editieren, sondern durch moderne beschleunigende Graphikprozessoren auch in Echtzeit visualisieren.

Um nun die von den Erfassungsverfahren gelieferten Objektpunkte in eine polygonale texturierte Repräsentation zu transformieren, müssen typischerweise drei Arbeitsgänge durchlaufen werden. Dazu gehören die Registrierung, die Vermaschung und abschließend die Texturierung.

Der erste Schritt, die Registrierung (registration), nimmt sich zunächst eines Grundproblems der berührungslosen Abtastung an. Prinzipbedingt sind sowohl der Laserscanner als auch die Stereokameras nicht in der Lage, die Objektpunkte von verdeckten Szenebereichen zu erfassen, sondern können ausschließlich die vom Gerätestandort sichtbare Oberflächenstruktur wiedergeben. Um jedoch auch Rauminformationen über die abgeschatteten Areale zu erhalten, behilft man sich mit mehreren überlappenden Aufnahmen von verschiedenen Positionen mit dem Ziel, die resultierenden Punktwolken im Anschluss geeignet zu kombinieren (zu registrieren).

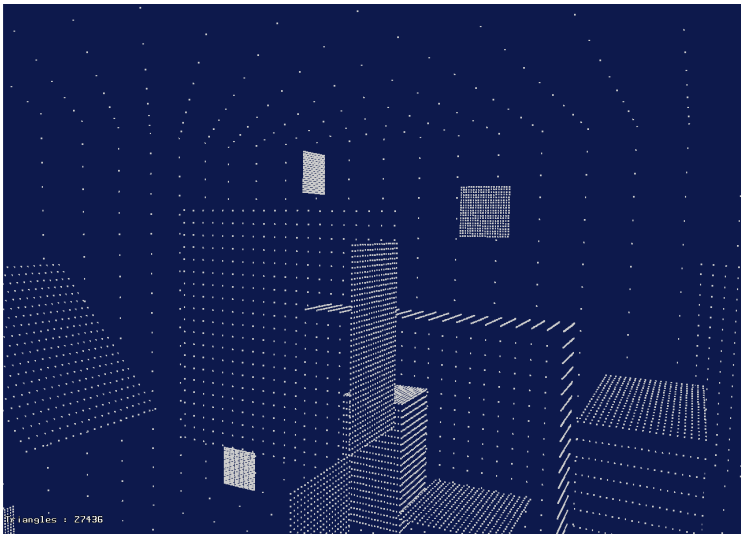
Dabei wird versucht, eine Zuordnung zwischen den gemeinsamen Elementen der Mengen herzustellen und eine Abbildungsvorschrift zu synthetisieren. In gewisser Weise ist die Registrierung demnach mit dem Matching von identischen Bildausschnitten verwandt. Geht man zur Vereinfachung des Problems von nicht deformierbaren Körpern unterschiedlicher Größe aus, was auf die erfassten Objekte praktisch immer zutrifft, dann sind die gesuchten Transformationen jeweils affin (Rotation, Translation, Skalierung). Sie lassen sich entweder zeitaufwendig manuell bestimmen oder automatisch berechnen, wofür sich etwa der iterative Algorithmus des nächsten Punktes (iterative closest point algorithm, ICP) [17] in seinen zahlreichen Varianten [18] eignet. Die prinzipielle Idee des Verfahrens ist es, den durchschnittlichen quadratischen Abstandsfehler der aktuell zu registrierenden Punktwolke im Vergleich mit der Mustermenge zu minimieren. Die sich dabei ergebenden vorläufigen Abbildungsparameter werden alsdann wieder auf die einzupassende Struktur angewandt und die Prozedur solange wiederholt, bis die gewünschte Präzision erreicht ist. Dabei garantiert der ICP-Algorithmus die monotone Konvergenz.

Wenn man die Punktwolken schließlich kombiniert hat oder mit einem einzigen Datensatz auskommt, erfolgt im nächsten Schritt die Vermaschung (meshing) der einzelnen Raumelemente mit Polygonen. Aufgrund ihrer Vorteile gegenüber komplexeren Strukturen, wie die garantierte Planarität, geschieht dies vielfach mit Dreiecken. Im Gegensatz zur Registrierung ist das manuelle Meshing wegen der hohen Anzahl an zu erstellenden Flächen nicht mehr praktikabel. Die für die Vernetzung entwickelten computerbasierten Algorithmen wie etwa das Crust- oder Cocone-Verfahren [19][20][21] basieren größtenteils auf der Delaunay-Triangulation bzw. dem dazu dualen Konzept der Voronoi-Diagramme. Sie liefern jedoch wegen der inhärenten Mehrdeutigkeiten im Ausgangsmaterial nicht immer das gewünschte Ergebnis. Falsch berechnete Dreieckseckpunkte, Lücken in der Oberfläche, singuläre Polygone und andere Artefakte müssen notfalls von Hand korrigiert werden.

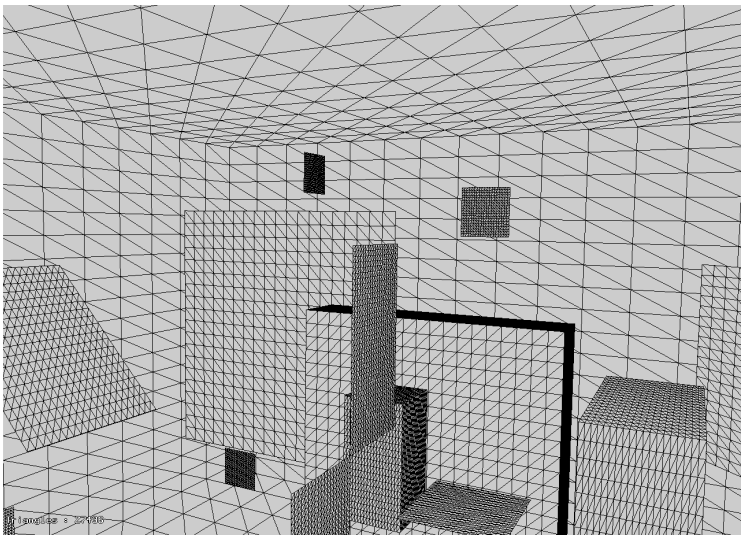
Auf die erzeugten Polygone lassen sich im letzten Arbeitsgang, der Texturierung (texture mapping), Bildinformationen aufbringen (die Texturen). Meist handelt es sich dabei um 2D-Bitmaps, die im Fall der Panoramakamera unmittelbar zur Verfügung stehen. Diese werden unter Angabe der gewünschten Bildkoordinaten für die Polygoneckpunkte mit unterschiedlichen Verfahren (planar, zylindrisch, sphärisch) auf die Modellvielecke projiziert [22]. Im Ergebnis erhält man ein photorealistisches Abbild der erfassten Szene, ohne geometrische Feinstrukturen (etwa rauhe Wandfassaden, Textilien, Bodenvegetation) oder unwesentliche Hintergrundobjekte aufwendig mit kleinen Flächen nachbilden zu müssen.

Abbildung 2.11 veranschaulicht die drei Verarbeitungsschritte anhand der für diese Arbeit entwickelten künstlichen Testszene, die zu einem späteren Zeitpunkt thematisiert wird.

a)



b)



c)



Abbildung 2.11.: Stufen der Modellerzeugung (a) Szene als Punktwolke, (b) als vermaschtes Drahtgitter und (c) nach der Texturierung

3. Motivation und Konzeption

Nach der Erläuterung der Grundlagen der photogrammetrischen und laserbasierten Objekterfassung sowie -rekonstruktion motiviert dieses Kapitel die Kombination der beiden Methoden. Dabei soll der Fokus auf der Verarbeitungsgeschwindigkeit und der erreichbaren Entfernungspräzision liegen. Es wird aufgezeigt, dass die Vereinigung positive Synergieeffekte verspricht, und die Konzeption für einen integrierten Algorithmus erarbeitet, welche als Grundlage für die Implementierung von 3D-Messanwendungen dienen kann.

3.1. Motiv der Verfahrenskombination

Die unterschiedlichen Herangehensweisen der Stereophotogrammetrie und des 3D-Laserscannings zur Gewinnung von Informationen über die Struktur einer Szene haben mehrheitlich divergierende Eigenschaften an den Tag gelegt. Dies gilt insbesondere für die Anzahl und Qualität der rekonstruierten Objektpunkte und den erforderlichen Rechenaufwand, der sich direkt in der benötigten Auswertungszeit niederschlägt. Wie ausführlich hergeleitet wurde, tendiert das photogrammetrische Verfahren zu einer höheren Raumpunktauflösung, speziell dann, wenn man auf subpixelgenaue Matchingalgorithmen zurückgreift. Außerdem liefert es die Texturinformationen für photorealistische Modelle ohne Zusatzaufwand gleich mit. Der Laserscanner kann unter günstigen Verhältnissen prinzipiell ein ähnlich akkurates Ergebnis erzielen. Abweichungen vom senkrechten Einfallswinkel und insbesondere reflektierende Materialen stellen ihn jedoch vor große Probleme. Dafür ist der LRF andererseits in der Lage, typische Erfassungsaufgaben in wenigen Minuten abzuschließen. Die bei den Stereoaufnahmen erforderliche Korrespondenzanalyse zur Ermittlung der 3D-Positionen dagegen nimmt selbst bei der Beschränkung des Suchbereichs auf die Epipolarlinien mehrere Minuten, Stunden oder gar Tage in Anspruch.

Möchte man nun weder auf die Genauigkeit der Photogrammetrie noch auf die Geschwindigkeit des 3D-Laserscanners verzichten, so stellt sich die Frage, wie die beiden Verfahren zweckmäßig kombiniert werden können. Eine mögliche Antwort hierauf hat bereits Abschnitt 2.1.4.1 angedeutet. Wenn es gelingt, die Korrespondenzanalyse zwischen den Stereoaufnahmen dadurch zu beschleunigen, dass man den Suchbereich auf ein kleines Gebiet entlang der Epipolarlinie begrenzt, dann erhält man möglicherweise ein Verfahren, das in kürzerer Zeit eine qualitativ hochwertige Punktwolke samt Texturinformationen zur Weiterverarbeitung liefert. Die Eingrenzung erfolgt dabei unter Verwendung eines vorweg angefertigten groben LRF-Modells, mit dessen Hilfe sich die Entfernungen und folglich die Raumpositionen der realen Weltobjekte abschätzen lassen.

3.2. Konzeption für einen integrierten Algorithmus

Eine Variante, wie die Suchbereichsdefinition zur zeiteffizienteren Korrespondenzanalyse mit einem lasergestützten 3D-Modell bewerkstelligt werden kann, skizzieren die folgenden Abschnitte. Darin wird zunächst die vom Verfahren erwartete Ausgangssituation fixiert. Anschließend erfolgt die Erläuterung der wesentlichen Aspekte des propagierten Algorithmus entsprechend ihrer Position in der Datenverarbeitungskette.

3.2.1. Ausgangssituation

Weil die Grundidee des Verfahrens zur Beschleunigung des Matchings eine große thematische Spannweite induziert, ist es vorab notwendig, den Ausgangspunkt dieser Arbeit realistisch abzustecken. Dabei schlägt sich die initiale Situation unmittelbar in den Anforderungen an die Eingabedaten des integrierten Algorithmus nieder. Die nötigen Vorleistungen, die für deren Erzeugung zunächst einmal erforderlich sind, sollen nur am Rande eine Rolle spielen und ansonsten als erbracht gelten.

Demnach wird festgelegt, dass die hier beschriebene Methode zwei vorverarbeitete Panoramabilder der gleichen Szene als Eingabedaten erhält, die in ihren Abmessungen differieren dürfen. Die innere Orientierung der verwendeten Aufnahmegeräte muss bekannt sein. Außerdem wird vorausgesetzt, dass ein dreidimensionales Szenenmodell existiert, welches auf die im Grundlagenkapitel beschriebene Weise aus einem oder mehreren Laserscans hervorgeht und mindestens die Koordinaten der Objektpunkte sowie darauf aufbauend eine Liste von Oberflächendreiecken enthält.

Um überhaupt einen Zusammenhang zwischen den Stereopanoramen und der dreidimensionalen LRF-Repräsentation herstellen zu können, ist es zudem notwendig, dass das Stereosystem an besagtem Modell vorab ausgerichtet wird. Nach der Bestimmung dieser *relativen* äußeren Orientierung (oftmals auch nur relative Orientierung genannt) der Kameras stimmt eine imaginäre Aufnahme des Modells näherungsweise mit dem Inhalt der Stereobilder überein. In der Reihenfolge Modell ⇒ Kameraausrichtung ausgeführt – man hätte auch die Raumpunktkoordinaten an die Position und Rotation des Stereosystems anpassen können – definieren die 3D-Daten gleichzeitig das Weltkoordinatensystem.

Dem Problem, die relative Orientierung vorab zu ermitteln, verleiht die Tatsache, dass es ohne die Bestimmung von korrespondierenden Punkten in dem hier dargestellten Szenario nicht automatisch gelöst werden kann, den gleichen subtilen Charme wie die Frage nach der Henne und dem Ei. Bei fehlender Orientierung lässt sich das Matching nicht beschleunigen, ohne die effiziente Korrespondenzanalyse kann man allerdings wiederum die Ausrichtung der Kameras nicht berechnen. Eine mögliche Lösungsstrategie für diese verfahrene Situation besteht in der interaktiven softwareunterstützten Anpassung des Stereosystems an das 3D-Modell. Dieser Prozess ist jedoch recht arbeitsintensiv und verursacht zusätzliche Personalkosten. In der Theorie lässt sich die relative Orientierung auch vollautomatisch ermitteln. Dazu löst man näherungsweise die im Allgemeinen nichtlineare Projektionsgleichung

$$\mathbf{X}_{\mathbf{k}}^{(\mathbf{i})} = M_{proj}(\mathbf{R}_{-}(\mathbf{X}_{\mathbf{k}}^{(\mathbf{w})} - \mathbf{t}^{(\mathbf{w})})) \tag{3.1}$$

iterativ mittels Ausgleichsrechnung nach der Drehmatrix $\mathbf{R}_-$ und dem Translationsvektor $\mathbf{t}^{(\mathbf{w})}$, wofür schon eine geringe zweistellige Zahl von frei wählbaren Modell-Bild-Korrespondenzen $\mathbf{X}_\mathbf{k}^{(\mathbf{w})} \leftrightarrow \mathbf{X}_\mathbf{k}^{(\mathbf{i})}$ genügt.

Um an die benötigten Zuordnungen für Gleichung 3.1 zu gelangen, werden zunächst ausgezeichnete Merkmalspunkte (features) wie etwa Ecken in beiden Datensätzen detektiert [23][24][25][26]. Dabei ist es vorteilhaft, die 3D-Punkte mittelbar aus den zweidimensionalen Distanz- oder Intensitätsinformationen des Laserscanners zu gewinnen, die inhaltlich mit dem auszurichtenden Panoramabild koinzidieren. Das Intensitätsbild lässt sich, sofern es nicht vom eingesetzten LRF ausgegeben wird, alternativ durch die Projektion der Punktwolke mit einer imaginären zylindrischen Panoramakamera im Laserstandort aus dem Modell erzeugen. Ordnet man dabei unterschiedlich weit entfernten Objektpunkten jeweils abgestufte Grauwerte zu, so sollte sich eine brauchbare zweidimensionale Bildvorlage für die Zuordnungssuche ergeben. Diese erfolgt nach der Featuredetektion ausschließlich an den markanten Stellen, was aufgrund des ausgedünnten Suchraums entsprechend zügig vonstatten geht (sparse matching). Damit initiiert die Korrespondenzanalyse im kleinen Maßstab letztlich das beschleunigte Matching im Großen. Die Vorgehensweise, bei der sich ein Prozess sprichwörtlich an den eigenen Haaren aus dem Sumpf zieht, bezeichnet man im Fachterminus auch als *bootstrapping*. Die praktische Umsetzung und Evaluierung der angedeuteten Methode zur automatischen Bestimmung der relativen Orientierung steht allerdings noch aus.

3.2.2. Datenmanagement

Aufgrund der hohen Anforderungen, die an die Eingabe, Ausgabe und Speicherung gestellt werden, kommt der Verwaltung der für das integrierte Matching benötigten Daten eine große Bedeutung zu. Im Wesentlichen sind hier drei primäre Arten von Informationen mit unterschiedlichen Eigenschaften zu behandeln. Zu den bislang erwähnten Panoramabildern und 3D-Modellen gesellen sich die Konfigurationen hinzu, welche zur Ablage von Algorithmuseinstellungen und Ausführungsprotokollen für die Reproduzierbarkeit sowie die Analyse der Ergebnisse essentiell sind.

3.2.2.1. Bilddaten

Die zu verarbeitenden Bildinformationen zeichnen sich dadurch aus, dass sie in mehreren Varianten auftreten. So liefert etwa die Panoramakamera farbiges RGB-Datenmaterial mit vorzeichenlosen ganzzahligen Pixelkomponenten und einer Auflösung von acht oder sechzehn Bit, während die Disparitätskarten aus atomaren Gleitkommawerten bestehen. Für die Visualisierung der Verschiebungswerte wiederum müssen die rationalen Bildpunkte in ein Integer-Graustufenbild konvertiert werden. Um diese offenkundige Vielfalt implementierungstechnisch unterstützen zu können, gibt es prinzipiell drei Möglichkeiten. Die Verwendung eines einzigen internen Datenformats, das als Obermenge alle Bildtypen aufnehmen kann, hat den Vorteil, die darauf aufbauenden Algorithmen zu vereinfachen. Allerdings wird bei diesem Ansatz mehr Speicherplatz als unbedingt notwendig belegt, und der Transport von großen Datenmengen wirkt sich tendenziell negativ auf das Laufzeitverhalten aus. Der Einsatz von spezialisierten statischen Bilddatenstrukturen kommt dagegen zwar mit der minimalen Menge an Hauptspeicher aus, erfordert aber angepasste Algorith-

men, die sich andererseits bestmöglich optimieren lassen. Der Erstellungs- und Wartungsaufwand ist bei dieser Variante beträchtlich, was der normalerweise angestrebten geringen Programmfehlerrate abträglich ist. Der Mittelweg zwischen den Extremen besteht in der Verwendung eines generischen Datentyps, der mit der Art der Pixelkomponenten spezialisiert wird und die anderen Metainformationen zur Laufzeit zugänglich macht. In diesem Fall ist es möglich, bei optimaler Platzausnutzung die semantischen Gemeinsamkeiten der bildpunktsensitiven Routinen auszunutzen, da diese lediglich der einmaligen Implementierung bedürfen.

Obwohl die effiziente Belegung des Hauptspeichers einem Mangel an selbigem mitunter vorbeugen kann, benötigen hochaufgelöste Panoramen nichtsdestotrotz große Mengen dieser Ressource. Insbesondere dann, wenn mehrere Bilder wie beim Stereosystem gleichzeitig vorgehalten werden müssen, ist mit hoher Wahrscheinlichkeit ein Engpass zu erwarten. Übersteigt der angeforderte Bedarf dabei die Menge an physisch installiertem RAM, so lagern moderne Betriebssysteme Teile der Bilder auf den langsamen Massenspeicher (in der Regel also die Festplatte) aus und lesen diese wieder ein, wenn sie erneut benötigt werden. Dieser als *swapping* oder *paging* bezeichnete Vorgang entzieht sich nahezu vollständig dem Einfluss des Benutzers und führt zu einem drastischen Einbruch der Verarbeitungsleistung, wenn die gerade verwendeten Speicherbereiche ausgelagert sind. Um diese Performanzeinbußen zu vermeiden, kann man den Auslagerungsmechanismus für die Bildinformationen vom Betriebssystem in die Anwendung hinein verlagern, wo er sich nahezu frei konfigurieren und an das oftmals bekannte Zugriffsmuster des ausgeführten Algorithmus anpassen lässt. Dadurch verringert sich zum einen die Anzahl der Zugriffe auf den Massenspeicher. Zum anderen besteht die Möglichkeit der Kopplung der Datenverwaltung mit dem Ein-/Ausgabesystem für die Panoramen. Anstatt diese naiv einzulesen und aus Speicherplatzmangel gleich wieder auszulagern, ist es denkbar, die originale persistent abgelegte Bilddatei zugleich auch als Auslagerungsobjekt zu instrumentalisieren (memory mapping). Der Zeitbedarf zum Laden sinkt praktisch auf Null, Änderungen des Bildinhalts werden allerdings unmittelbar auf der Vorlage auf dem Massenspeicher durchgeführt. Deswegen ist das Kopplungsverfahren eher für große Datenmengen prädestiniert, die wie bei der Korrespondenzanalyse keiner Modifikation unterliegen.

Die Verwendung unterschiedlicher Typen von Pixelkomponenten hat nicht nur einen Einfluss darauf, wie die Bildinformationen im volatilen Hauptspeicher abgelegt werden, sondern auch auf die Gestaltung der Ein- und Ausgabefunktionen vom bzw. zum nichtflüchtigen Medium. Dabei treten im Wesentlichen zwei Probleme auf. Zunächst einmal gibt es eine Vielzahl von Bitmapdatenformaten mit stellenweise recht kompliziertem Aufbau, der entsprechend analysiert und korrekt erzeugt werden muss. Außerdem sind die Pixelwerte darin nicht unbedingt so abgelegt, dass man sie ohne vorherige Umwandlung vom oder zum Hauptspeicher transferieren kann.

Ein Ansatz, um bei maximaler Flexibilität nicht für jedes Datenformat eigene Bildpunktkonvertierungsroutinen entwickeln zu müssen, besteht in der modularen Trennung der beiden Konzepte. Hierbei sind spezifische Lade- bzw. Speicherroutinen ausschließlich für die Analyse und Erstellung eines bestimmten Bitmapdatenformats verantwortlich, wofür gegebenenfalls auf existierende Bibliotheken von dritter Seite zurückgegriffen werden kann. Die Konvertierung der einzelnen Pixel übernehmen dagegen separate Codecs (zusammengesetztes Kunstwort aus encoder/decoder, Konvertierer/Rückkonvertierer), die je nach Bedarf an die Formatinterpreter übergeben werden. Dabei muss freilich für jede Kombination aus internem (Hauptspeicher) und externem (Massenspeicher) Bildpunktaufbau

ein passendes Codecmodul existieren, sofern man die jeweilige Pixeltransformation denn unterstützen möchte.

3.2.2.2. 3D-Modelle

Die Datenstruktur für die aus den Laserscans generierten 3D-Modelle wie auch die mit Hilfe des Stereosystems rekonstruierbare Punktwolke hat die Aufgabe, die dreidimensionalen Raumkoordinaten, die Oberflächenpolygone und die Angaben zur Texturierung vorzuhalten. Da es im Gegensatz zu den gerasterten Bildern schwierig ist, eine native Ordnung auf den Raumdaten zu definieren, die die räumliche Nähe der Szenenobjekte berücksichtigt, erfolgt die Speicherung in unsortierten Listen[6]. Deren Platzbedarf orientiert sich an der Auflösung der Panoramen, wodurch auch hier der Einsatz eines anwendungsgesteuerten Auslagerungsmechanismus gerechtfertigt ist. Die geringe Lokalität lässt jedoch einen negativen Einfluss auf die Vorhersagbarkeit der Zugriffsadressen erwarten, was auf einen verminderten Datendurchsatz hinausläuft.

Wie bei den Bildinformationen gibt es auch bei den 3D-Modellen eine Vielzahl von Datenformaten für die nichtflüchtige Ablage auf einem Massenspeicher. Aus Gründen der Kompatibilität mit anderen Werkzeugen sind hierbei offene standardisierte Repräsentationen zu bevorzugen. Die Auswahl an vorhandenen frei modifizierbaren Implementierungen ist diesbezüglich allerdings stark eingeschränkt, was die Entwicklung eines geeigneten Parsers für die Ein- und Ausgaberoutinen impliziert.

Da die Validierung der Algorithmen, welche die 3D-Daten manipulieren, allein anhand der rein numerischen Informationen ein schwieriges Unterfangen darstellt, ist es außerdem erforderlich, die Modelle in geeigneter Form zu visualisieren. Somit können fehlerträchtige Ausführungspfade graphisch begutachtet und Lösungsansätze bei erkannten Defekten auf Basis der 3D-Darstellung schneller abgeleitet werden. Für die Abbildung der Szene auf den (planaren) Bildschirm eignen sich prinzipiell die Projektionsvorschriften des Grundlagenkapitels. Allerdings verspricht die Hardwarebeschleunigung moderner Graphikkarten samt den dazugehörigen Anwendungsschnittstellen ein höheres Leistungspotential.

3.2.2.3. Konfigurationen

Die Konfigurationen speichern die Einstellungen der Algorithmen, die auf den Bildaufnahmen und 3D-Modellen operieren, und nehmen Statusmeldungen sowie anfallende Messwerte auf, die nach deren Terminierung weiter benötigt werden. In dieser Funktion bewerkstelligen sie die Übergabe von umfangreichen Parametersätzen, ermöglichen in persistenter Form die Reproduktion von Ergebnissen und stellen einen einheitlichen Protokollmechanismus bereit.

Strukturell bietet es sich an, die vergleichsweise kompakten Einstellungsdatensätze als Hierarchie zu organisieren. Damit werden die logischen Zusammenhänge zwischen den abgelegten Werten explizit gemacht. Außerdem lassen sich identische Parameterbezeichnungen in unterschiedlichem Kontext einsetzen und auf diese Weise Namenskonflikte vermeiden. Als Nachteil verkompliziert sich die Adressierung der einzelnen Konfigurations-

[6] Bei den Bitmapdaten identifizieren benachbarte Pixel immer benachbarte Bildobjekte oder das gleiche Bildobjekt im zweidimensionalen Bildraum.

elemente, da sich diese nun nicht mehr mit fortlaufenden Indizes ansprechen lassen. Gleiches gilt auch für die Ein- und Ausgabe der vollständigen oder partiellen Hierarchie vom bzw. auf den Massenspeicher, die jeweils einen speziellen Parser bzw. Zielformatprozessor voraussetzt. Die entsprechenden Komponenten müssen die Rangordnung beim Einlesen rekonstruieren und beim Schreiben geeignet linearisieren.

3.2.3. Objektpunktschätzung

Die anvisierte Beschleunigung der Korrespondenzanalyse lässt sich durch die Verkleinerung des Bereichs entlang der Epipolarlinie erreichen, in welchem das aktuelle Suchfenster verschoben wird. Einen Anhaltspunkt hierfür liefert das zu den Stereobildern relativ orientierte LRF-Modell, welches für die näherungsweise Rekonstruktion des realen Objektpunkts $\mathbf{X}_{\mathbf{k}}^{(\mathbf{w})}$ unter Verwendung der Notation aus dem Grundlagenkapitel benutzt wird.

Die Wiederherstellung von $\mathbf{X}_{\mathbf{k}}^{(\mathbf{w})}$ lässt sich bewerkstelligen, indem man den Sichtstrahl R_{k1} vom Projektionszentrum $\mathbf{C}_{\mathbf{1}}^{(\mathbf{w})}$ der ersten Kamera des Stereosystems durch das aktuell betrachtete Pixel in Weltkoordinaten $\mathbf{X}_{\mathbf{k1}}^{(\mathbf{w|i})}$ zurückverfolgt. Der nächstgelegene Schnittpunkt mit einem Oberflächendreieck des Modells liefert dann die gewünschte Raumposition. Dabei stellt die bekannte Ausrichtung der Szene zu den Stereoaufnahmen sicher, dass der Strahl dasjenige „richtige“ Objekt trifft, welches durch den passierten Bildpunkt wiedergegeben wird. Mit der anschließenden Projektion von $\mathbf{X}_{\mathbf{k}}^{(\mathbf{w})}$ in das komplementäre Bild I_2 der zweiten Kamera ergibt sich dann die finale Pixelposition, in deren Umgebung der mit dem Musterfenster korrespondierende Suchfensterinhalt lokalisiert sein muss. Die Rollen der beteiligten Aufnahmegeräte des Stereosystems sind bei der soeben umrissenen Objektpunktschätzung und -projektion austauschbar, so dass auch I_2 als Vorlagengeber für das Matching fungieren kann. Die damit einhergehende Verlagerung des Strahlursprungs in das Abbildungszentrum $\mathbf{C}_{\mathbf{2}}^{(\mathbf{w})}$ impliziert andere Sichtbarkeitsbereiche, ohne die gemeinsame Schnittmenge der Bilder zu tangieren, und bewirkt im Idealfall einen Vorzeichenwechsel der Disparitäten.

Wenngleich das Prinzip der Strahlverfolgung (Raytracing) zunächst nicht den Eindruck erwecken mag, technisch besonders anspruchsvoll zu sein, muss für die *effiziente* Umsetzung ein signifikanter zusätzlicher Implementierungsaufwand betrieben werden. Das Problem ist hierbei, das für die obligatorische Schnittpunktermittlung benötigte Dreieck in der ungeordneten Modellbeschreibung zu finden. Beim Einsatz des naiven Suchalgorithmus mit linearer Zeitkomplexität für jeden Strahl bzw. jedes Pixel steigt die Zahl der benötigten Rechenoperationen rasch an. Selbst für kleine Datensätze mit etwa 10.000 = 10^4 Polygonen und 1.000 x 1.000 = 10^6 Bildpunkten ergeben sich bereits Werte im Bereich von Zehnerpotenzen mit zweistelligem Exponenten.

Ohne eine geeignete räumliche Sortierung der Modelldreiecke und die damit verbundene Möglichkeit der binären Suche mit logarithmisch wachsendem Zeitaufwand führt die skizzierte Vorgehensweise offensichtlich nicht zu einer Beschleunigung der Korrespondenzanalyse. Setzt man eine ordnende Funktion ein, so ergibt sich andererseits ein gewisser Mehraufwand sowohl für die Klassifikation der Oberflächenprimitiven als auch für jeden einzelnen Suchaufruf. Letzterer besteht dabei nicht mehr nur aus dem Inkrementieren des Index der Dreiecksliste und einem Speicherzugriff, sondern beinhaltet auch die Eingrenzung der in Frage kommenden Polygone. Man muss demnach darauf achten, dass die

Sortierung und die wenigen komplexen Suchschritte pro Strahl respektive Pixel zusammen deutlich weniger Rechenzeit in Anspruch nehmen, als die Zuordnungsbestimmung entlang der gesamten Epipolarlinie erfordert.

3.2.4. Matching

Die Korrespondenzanalyse führt einen Vergleich der Pixel zwischen dem Muster- und dem Suchfenster mit Hilfe einer Ähnlichkeitsfunktion durch und akkumuliert die resultierenden Übereinstimmungswerte. Bei diesem blockorientierten Verfahren wird die unmittelbare Umgebung des Vorlagenpunkts in die Berechnung mit einbezogen, wodurch sich die zur Verfügung stehende Informationsmenge erhöht. Je größer dieser Umgebungsbereich gewählt wird, desto verlässlicher ist das Ergebnis in schwach gegliederten Bildarealen, da die Wahrscheinlichkeit zunimmt, ein strukturgebendes Element mit einzuschließen. Gleichwohl steigt bei großflächigen Suchfenstern der erforderliche Rechenaufwand für den Matchingschritt quadratisch mit der Seitenlänge an. Zudem kann es in den markanten Bildbereichen zu Fehlzuordnungen kommen, da die zusätzlichen Informationen hier entbehrlich sind und den Ähnlichkeitsgrad verwässern. Durch eine inhaltsbezogene Adaption der Fenstergröße besteht die Möglichkeit, die negativen Auswirkungen in beiden Fällen zu reduzieren.

Nicht vermeiden lässt sich dagegen das allen lokalen Verarbeitungsalgorithmen anhaftende Randproblem. Die üblichen Techniken, um auch an den Bildgrenzen konsistente Ergebnisse zu erhalten, funktionieren beim Matching nicht. So führt das Auffüllen mit den scheinbar neutralen Nullpixeln[7] zu verfälschten Übereinstimmungswerten. Bei der Anwendung der SAD- oder SSD-Funktion äußert sich dies in einem systematisch überhöhten Abstand durch die betragsmäßig größeren Pixeldifferenzen, wenn ein derart behandeltes Musterfenster mit einem zentral gelegenen Suchfenster verglichen wird und umgekehrt. In der Konsequenz sind die am Rand berechneten Ähnlichkeitsdaten ohne Korrektur nicht mehr mit denjenigen im Bildinnern vergleichbar, was die Qualitätsbewertung erschwert. Ähnlich verhält es sich, wenn man versucht, den fehlenden Inhalt für die Matchingfenster durch eine Spiegelung an der Bildbegrenzung oder die Fortsetzung mit den Pixeln der gegenüberliegenden Seite der Aufnahme („umklappen“, wrap-around) zu generieren. Die einzige praktikable Möglichkeit, den Bildrand adäquat zu berücksichtigen, besteht demnach in der Verschiebung des Bezugspunkts für den Korrelationsvorgang im Muster- und Suchfenster, so dass ausschließlich die existierenden Bildpunkte verwendet werden.

Um die Korrespondenzanalyse implementierungstechnisch zu beschleunigen, kann man wie für die Objektpunktschätzung mittels Strahlverfolgung ein hierarchisches Verfahren benutzen, welches sich mit logarithmischem Zeitbedarf der bestmöglichen Ausschnittsübereinstimmung annähert [7]. Hierbei besteht allerdings die Gefahr, dass durch die Neuabtastung des Bildes unabhängig von der obligatorischen Bandbreitenlimitierung durch einen Tiefpassfilter der falsche Bildausschnitt für den nachfolgenden feiner aufgelösten Matchingdurchlauf gewählt wird.

Verzichtet man auf die Zeitersparnis und setzt das hierarchische Verfahren lediglich zur Erzielung von präzisen Subpixelkorrespondenzen nach der bildpunktgenauen Analyse ein, tritt das Problem nicht mehr auf. In diesem Szenario wird keine Information „übersprungen“, sondern auf künstlich eingeführten interpolierten Farbwerten gearbeitet. Die Berechnung der Zwischenwerte gestaltet sich als lokale Operation allerdings aufwendiger als ein

[7] alle RGB-Farbkomponenten weisen den Wert null auf

simpler lesender Pixelzugriff. Wenngleich eine vorgelagerte Interpolation der Stereoaufnahmen aufgrund des immensen Speicherplatzbedarfs nicht zweckmäßig erscheint, können örtlich wirksame Puffermechanismen für die sonst während der Zuordnungsbestimmung wiederholt einzufügenden Bildpunkte einen signifikanten Leistungszuwachs bewirken.

3.2.5. Modellverbesserung

Die mit dem Stereosystem ermittelten Objektpunkte können aufgrund der zu erwartenden höheren Genauigkeit zur Verbesserung des ursprünglichen LRF-Modells benutzt werden. Im einfachsten Fall geschieht dies durch die vollständige Ersetzung der vorhandenen Punktwolke mit dem Ergebnis der photogrammetrischen Rekonstruktion. Da die Korrespondenzanalyse jedoch nicht vollkommen fehlerfrei arbeitet, kann aus der Modifikation der bereits existierenden Modellpunkte besonders an den für den Laserscanner problematischen Stellen möglicherweise ein qualitativ ansprechenderes Resultat entstehen.

Ein solcher Ansatz, der ausschließlich die vorhandenen 3D-Informationen verändert, ist die strahlgebundene Punktverschiebung. Dabei wird die hohe Winkelpräzision des LRF im Vergleich zur Entfernungsgenauigkeit ausgenutzt. Durch das Abtragen der mit dem Stereosystem gemessenen Entfernungen vom Kamerazentrum zu den Objektpunkten entlang der zugehörigen virtuellen Laserstrahlen lassen sich die mit dem Scanner bestimmten Distanzen nachträglich korrigieren. Allerdings muss hierzu der Standort des LRF im Modell bekannt sein. Bei Szenen, die durch die Registrierung aus mehreren Aufnahmen zusammengesetzt sind, ist das Verfahren nur dann anwendbar, wenn für jeden Objektpunkt, welcher durch die mustergebende Kamera des Stereosystems sichtbar abgebildet wird, auch die Raumposition des erzeugenden Scanners feststeht.

Sind die erforderlichen Standortinformationen dagegen nicht verfügbar, so kann man die Modifikation der existierenden nulldimensionalen Szenenelemente auch allein auf Basis der Bildaufnahmen umsetzen. Beim strahlunabhängigen Verschiebungsverfahren werden dazu die Objektpunkte des Modells an diejenigen Raumpositionen verlagert, die ihre photogrammetrisch gewonnenen Gegenstücke einnehmen. Dies setzt allerdings voraus, dass eine Zuordnung zwischen den vorhandenen ausdehnungsfreien LRF-Raumelementen und den Bildpunkten in der mustergebenden Stereoaufnahme bekannt ist, welche die Identifikation der passenden Disparitäten ermöglicht. Dabei sind exakte Angaben zur relativen Orientierung des Stereosystems zum Modell besonders wichtig. Anders als bei den vorangegangenen Verbesserungsansätzen und dem kamerabasierten Rekonstruktionsprozess allgemein geht eine eventuelle Abweichung von der korrekten Ausrichtung gleich doppelt als Fehler in die korrigierten Ortsangaben ein. Zur inakkuraten Position des maßgeblichen Kameraursprungs addiert sich bei der strahlunabhängigen Verschiebung die Entfernungsdifferenz aufgrund der ungenauen 3D-2D-Korrespondenzen. Abgesehen davon transformiert die Zuordnung mittels Projektion die Modellpunkte nicht unbedingt auf ganzzahlige Pixelpositionen im Bild der mustergebenden Kamera. Folglich erfordert die Berechnung der Entfernung der finalen räumlichen Ortsangaben einen Interpolationsschritt, der die Disparitätswerte in der Nachbarschaft des jeweiligen projizierten Objektpunkts mit einbezieht.

Eine weitere prinzipielle Unzulänglichkeit der soeben skizzierten strahlunabhängigen Modellverbesserungsmethode ist die Tatsache, dass die fehlerbehafteten LRF-Punkte eines Szenenobjekts auf ein anderes übergehen können. Besonders anfällig für diesen wörtlich zu verstehenden Effekt sind die Objektkanten, wo die im Vordergrund befindlichen unge-

nauen Raumpunkte eine „Korrektur“ mit der dem Bildhintergrund entsprechenden Distanz erfahren können (siehe Abbildung 3.1). Um derartige „Umverteilungen“ von Laserpunkten zu vermeiden, müssen die LRF-Daten an Ort und Stelle (in-place) unter Berücksichtigung des Objekts, zu dem sie gehören, verschoben werden. Wenngleich ein solches Vorgehen realisierbar sein mag, sprengen sowohl die Herstellung des Objektzusammenhangs über eine dreidimensionale Segmentierung der Szene als auch die schlecht gestellte Berechnung der Translationsvektoren für die fehlerhaften LRF-Punkte bei weitem den Rahmen dieser Arbeit.

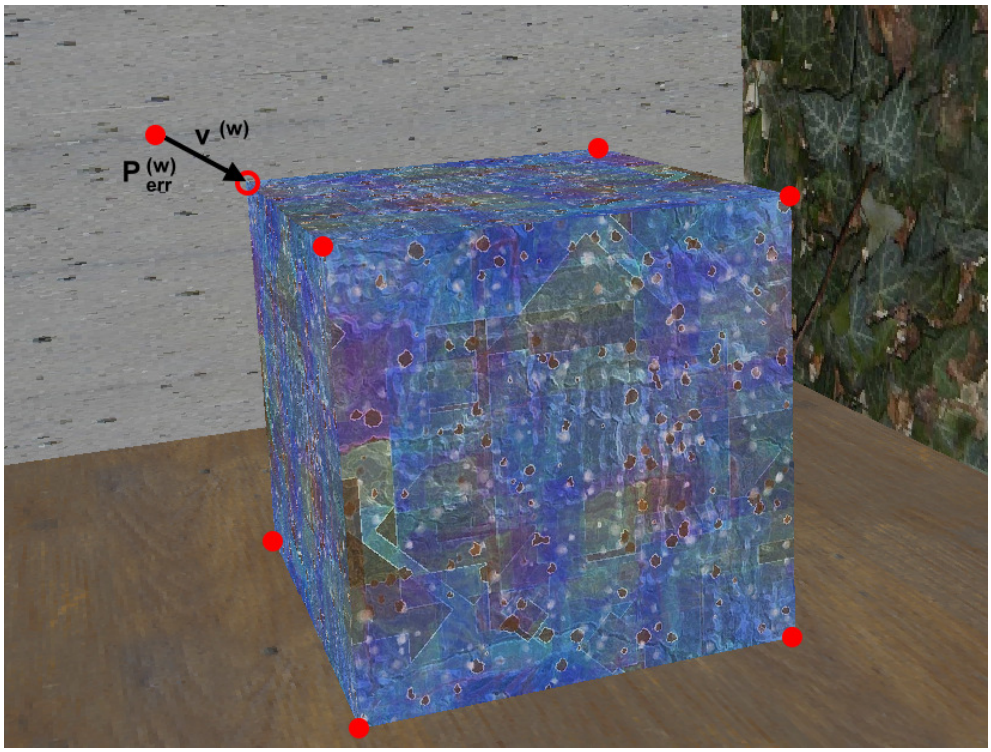


Abbildung 3.1.: Modifikation der vorhandenen 3D-Daten. Der ungenau gemessene LRF-Objektpunkt $\mathbf{P}_{\mathbf{err}}^{(\mathbf{w})}$ erscheint in der Aufnahme vor der grauen Hintergrundwand und wird daher selbiger bei der strahlunabhängigen Verschiebungsmethode zugeschlagen. Korrekt wäre die Platzierung auf dem Eckpunkt des blauen Batik-Quaders. Dazu müsste das Verfahren allerdings „wissen“, dass der Punkt zur Box gehört (Herstellung des Objektzusammenhangs) und um den unbekannten Translationsvektor $\mathbf{v}^{(\mathbf{w})}$ verlagert werden.

4. Implementierung

Inwieweit die im vorangegangenen Kapitel motivierte Kombination des photogrammetrischen Objekterfassungsverfahrens mit der laserbasierten Methode tatsächlich dazu beitragen kann, die Korrespondenzanalyse zu beschleunigen und ein qualitativ verbessertes 3D-Modell zu erzeugen, lässt sich erst anhand einer Beispielimplementierung evaluieren. Deshalb stellt dieses Kapitel mit der zugegebenermaßen etwas phantasielos DaTool (Abkürzung für Diplomarbeits-Tool) genannten Software ein prototypisches Werkzeug vor, das die angestellten konzeptionellen Überlegungen berücksichtigt. Im Mittelpunkt steht dabei die konkrete Umsetzung des integrierten Matchingalgorithmus und der flankierenden Funktionalität von der Datenverwaltung und -veranschaulichung bis hin zu den Testaspekten. Ferner werden dem Leser auch die getroffenen Designentscheidungen näher gebracht, die aufgrund der Größe des Gesamtsystems für die aufwandsarme erweiterungsfreundliche und portable Entwicklung von großer Bedeutung sind.

4.1. Überblick über die Software

Der im Rahmen der Diplomarbeit entwickelte Prototyp DaTool repräsentiert ein Softwaresystem, mit dem komplexe Arbeitsaufgaben auf theoretisch beliebigen heterogenen Daten als Folge von atomaren Teilschritten erledigt werden können. Er besteht im Wesentlichen aus einem Rahmenprogramm mit graphischer Benutzeroberfläche, einem Datenspeicher und einer Menge von Algorithmen, die die Berechnungen vornehmen. Letztere sind dabei als zur Laufzeit ladbare Module (Plugins) ausgeführt und lassen sich ohne Neuübersetzung der anderen Bestandteile hinzufügen oder entfernen. Prozessabläufe werden durch die weitestgehend frei wählbare sequenzielle Zusammenstellung einer Teilmenge der zum Zeitpunkt der Programmausführung vorhandenen Funktionen definiert. Beabsichtigt man die Implementierung einer neuen Handlungsvorschrift, so kann diese auf alle verfügbaren DaTool-Datentypen zurückgreifen und bei Bedarf an sich wesensfremde Informationen zusammenführen, wie es etwa beim propagierten Matchingverfahren der Fall ist. Für die beschleunigte Korrespondenzanalyse sind die konzipierten Speicherstrukturen und Algorithmusmodule bereits vorhanden.

Damit die beschriebene Flexibilität erreicht werden kann, bedient sich der Prototyp einer Reihe von Konzepten aus den höheren Programmiersprachen. Hierbei handelt es sich namentlich um die (starke) Typisierung, die Verwendung von Laufzeittypinformationen (runtime type information, RTTI) und die Gliederung der Berechnungsvorschriften in Unterprogramme mit Parametern und Rückgabewerten. Nach innen ermöglichen diese Denkmodelle die generische Datenhaltung und vereinheitlichen die DaTool-Algorithmusplugins in Bezug auf ihre Schnittstellen. In die Benutzeroberfläche integriert, definieren sie nach außen hin maßgeblich das Bedienkonzept der Anwendung, das ohne textuelle Kodierkenntnisse auskommt und treffend mit dem Schlagwort „graphische Programmierung“ charakterisiert werden kann.

4.2. Entwicklungsumgebung und Zielplattform

Die DaTool-Software ist in der Programmiersprache C++ geschrieben und macht intensiven Gebrauch von den objektorientierten Elementen [27]. Der Quellcode umfasst ohne den automatisch generierten Anteil über 20.000 Zeilen und kann mit den Compilern der Microsoft Visual Studio Suite 2003 (mit Service Pack 1) und 2005 [28] in nativen Maschinencode übersetzt werden. Dadurch wird die Leistungsfähigkeit der Intel-kompatiblen Zielprozessoren ohne Umwege über die Interpretation von Zwischenprodukten während der Ausführung direkt ausgenutzt. Abgesehen vom Compiler und der zugehörigen Laufzeitumgebung erfordert die Erzeugung einer funktionsfähigen Programmversion zusätzlich

- den Scannergenerator flex (Version 2.5.4) [29] und den Parsergenerator bison (Version 2.1) [30] für die Leseroutinen der 3D-Szenen,
- die TinyXML-Bibliothek (Version 2.5.3) [31] für die Ein- und Ausgabe der Algorithmuseinstellungen,
- das Qt-Toolkit (Version 4.2.2) [32] für die graphische Benutzeroberfläche (graphical user interface, GUI), das vor der Verwendung mit dem separat erhältlichen ACS-Paket (advanced compiler support) [33] modifiziert werden muss, sowie
- optional die STXXL-Bibliothek (Version 1.1.0) für umfangreiche Modelle [34][35][36], welche ihrerseits auf der Boost-Funktionssammlung (Version 1.34.1) [37] aufbaut [8].

Die genannten Applikationen und Bibliotheken sind dabei ohne Ausnahme im Quellcode erhältlich und werden als freie Software lizensiert, was im Übrigen auch für DaTool selbst gilt. An Zielplattformen unterstützt der Prototyp derzeit ausschließlich die 32-Bit-Windows-Betriebssysteme 2000, XP und Vista. Wegen der Ausgliederung der systemabhängigen Programmteile ist eine Portierung etwa auf Unix-Derivate jedoch nicht von vornherein ausgeschlossen. Die zugrunde liegende Hardware sollte in jedem Fall mit einer beschleunigenden 3D-Graphikkarte samt Treibern für die OpenGL-Schnittstelle [38] ausgestattet sein, da sonst die interaktive Visualisierung der räumlichen Szenen in akzeptabler Geschwindigkeit nicht möglich ist.

4.3. Interner Aufbau der Anwendung

Die interne Struktur der Software orientiert sich am Prinzip der lose gekoppelten Komponenten aus der Softwaretechnik [39]. Sie gliedert sich bei grober Betrachtung in vier Teile, die jeweils spezifische Aufgaben erfüllen. Unter Angabe der Stellen im DaTool-Quelltextbaum, die die Unterteilung wiederspiegeln, sind dies:

- die **gemeinsam genutzten Komponenten** (`src/common`)

  Hierunter fallen sowohl plattformspezifische Module als auch generischer Code, der an zahlreichen Stellen im Programm benötigt wird. Zu den ersteren gehören die Definitionen von Standardtypen und -funktionen (`src/common/platform`) als auch großen Datencontainern (derzeit auf STXXL-Basis, `src/common/lrg`), das system-

[8] Die in Klammern angegebenen Versionen wurden bei der Entwicklung von DaTool eingesetzt.

nahe Ein-/Ausgabe-Subsystem (`src/common/sysio`) und die Routinen für das Laden der Plugins zur Laufzeit (`src/common/dlfcn`). Diese Module stellen eine Abstraktionsschicht zwischen dem Zielsystem und der Applikation zur Verfügung. Sie müssen daher im Gegensatz zu den anderen DaTool-Komponenten bei einem Plattformwechsel angepasst werden. Der generische Code umfasst Routinen zur ausnahmebasierten Fehlerbehandlung (`src/common/exception`), einige mathematische Funktionen (`src/common/math`) sowie diverse Hilfswerkzeuge (`src/common/util`), etwa zur Typkonvertierung, Textausgabe und Traversierung zweidimensionaler Datenstrukturen.

In beiden Fällen wird durch die Zusammenstellung von normierten Funktionen an einem dedizierten Ort vermieden, dass verschiedene Ansätze zur Implementierung der genannten Bereiche über die gesamte Anwendung verteilt existieren. Damit verringert sich der Such- bzw. Korrekturaufwand im Fehlerfall.

- das **Rahmenprogramm** (`src/qtgui`)

  Das Rahmenprogramm initialisiert die graphische Benutzeroberfläche der DaTool-Software in Gestalt des Hauptfensters und übernimmt die grundlegende Ereignisbehandlung zur Reaktion auf die Eingaben des Anwenders. Dies beschränkt sich allerdings auf die Auswahl von Datenobjekten und den hierauf operierenden Verarbeitungsalgorithmen.

- das **Datenmanagement** (`src/data`, `src/core`)

  Hierzu gehören die Speicherstrukturen für die Bilder (`src/data/image`), 3D-Modelle (`src/data/model3d`) und Konfigurationen (`src/data/configuration`) einschließlich der Zugriffsfunktionen, Ein-/Ausgabemodule sowie höheren DaTool-unabhängigen Bearbeitungsverfahren. Daneben wird auch der Programmkern des Prototyps dieser Gruppe zugeordnet, weil er lediglich aus der einheitlichen Verwaltung von Instanzen der genannten heterogenen Datentypen und einer Liste der verfügbaren funktionalen Algorithmen besteht, die Informationen jedoch nicht selbst manipuliert. Die Zweiteilung bezüglich der Quellcodeorganisation resultiert aus der bislang nicht umgesetzten Überlegung, die Datenmodule wie schon ihre „rechnenden“ Verwandten dynamisch zur Laufzeit zu laden.

- und die **funktionalen Algorithmen** (`src/algorithm`)

  Die funktionalen Algorithmen enthalten die Programmlogik für die Berechnungen auf den drei primären Speicherstrukturen. Die dynamisch zur Laufzeit ladbaren Module untergliedern sich entsprechend ihrer vornehmlichen Aufgabe in die Kategorien Eingabe (`src/algorithm/in`), Ausgabe (`src/algorithm/out`), Visualisierung (`src/algorithm/view`) und Datenmanipulation (`src/algorithm/manipulate`).
  Trotz unterschiedlicher Aufgaben besitzen alle Algorithmen identische Schnittstellen und teilen sich bestimmte Komponenten, die in `src/algorithm/common` situiert sind.

Wie ersichtlich wird, besteht im großen Maßstab eine Trennung zwischen der Datenverwaltung, der Benutzerschnittstelle und der funktionalen Programmlogik. Der Vorteil hierbei ist, dass sich die einzelnen Makroelemente bei Bedarf austauschen lassen und die auszuführenden Codemodifikationen dennoch lokal begrenzt bleiben. So kann etwa das Rahmenprogramm von der graphikbasierten Interaktion mit dem Anwender auf eine konsolengestützte

Variante umgestellt werden, ohne dass dazu Anpassungen an der Datenverwaltung erforderlich sind. Daneben ist durch die Separation auch die Möglichkeit einer Wiederverwendung einzelner Komponenten in artverwandten Projekten gegeben. Nachteilig wirkt sich der erhöhte intermodulare Kommunikationsüberhang und der intellektuelle Mehraufwand für den Entwurf aus. Die positiven Aspekte überwiegen jedoch deutlich.

Auf der Modulebene setzt sich das Prinzip der losen Kopplung sowohl physisch (Quelltextdateien) als auch logisch (objektorientierte Klassen) aus den gleichen Gründen nahtlos fort. Dies wird beispielsweise bei der Datenverwaltung deutlich, die abgetrennt von der Speicherung der Informationen auch die zugehörigen Ein- und Ausgaberoutinen umfasst. Ganz ähnlich verhält es sich mit den funktionalen Algorithmen, um noch einmal auf das GUI-Beispiel zurückzukommen. Aufgrund ihres dynamischen Charakters müssen die Plugins grundsätzlich ihre eigenen Benutzerdialoge implementieren und können hierfür nicht auf das Rahmenprogramm zurückgreifen. Obwohl sie demzufolge von der angesprochenen Austauschbarkeit der primären graphischen Bedienoberfläche auf keine Weise profitieren, sind die Algorithmusklassen so organisiert, dass eine Umstellung auf die konsolenbasierte Interaktion mitnichten Änderungen am vorhandenen Code für die Berechnungen impliziert.

Mit Hinblick auf den integrierten Matchingalgorithmus gehen die folgenden Abschnitte nun genauer auf die Umsetzung der genannten DaTool-Komponenten ein. Dabei stehen primär die eingesetzten Datenstrukturen und Verarbeitungsmethoden im Vordergrund. Aus Platzgründen wird die konkrete Abbildung in C++-Code durch die Referenzierung der Quelltextdateien und Klassennamen im Fall von objektorientierten Elementen abgekürzt. Für Informationen zu den Feinheiten bei der Implementierung sei auf die jeweilige vollständige Deklaration bzw. Definition im maschinenlesbaren Original verwiesen.

4.4. Gemeinsam genutzte Komponenten

Die gemeinsam genutzten DaTool-Komponenten bewerkstelligen die Abstraktion von der Zielplattform, die eine Voraussetzung für die portable Softwareentwicklung ist. Sie vereinigen oft genutzte Funktionen an einem zentralen Ort und helfen mit, die inkonsistente Mehrfachimplementierung von ein- und demselben Konzept in disjunkten Quelltexten zu vermeiden.

Von der Vielzahl an Modulen wird ein Quartett überproportional häufig verwendet. Dazu gehören die Plattformdefinitionen, die in der C++-Deklaration (dem Header) `platform.h` aufgeführt sind. Alle zielsystembezogenen Einstellungen werden dort durch das Einbinden der entsprechenden Schnittstellen aus der Laufzeitbibliothek und über Makrodefinitionen vorgenommen. Dies schließt insbesondere die Abbildung der eingebauten Basisdatentypen mit variablem Wertebereich auf solche mit fester Breite gemäß dem C99-Standard [40] ein, welche in DaTool anstelle der Originale durchgängig Verwendung finden. Damit wird nicht nur Portierungsproblemen durch unerwartete Überlaufsituationen begegnet. Ohne die einheitliche Bereitstellung der „langen“ 64-Bit-Integerzahlen und zugehörigen maschinenorientierten Operationen, die in vielfältiger Form auch auf den gängigen 32-Bit-

Computersystemen angeboten werden, ist die Handhabung umfangreicher Datenmengen ≥ 4 GiB[9] schlicht nicht möglich.

Eine Komponente, die intensiven Gebrauch von den Plattformdefinitionen macht, ist das Ein/Ausgabe-Subsystem (E/A-Subsystem). Es kapselt die grundlegende Ein- und Ausgabefunktionalität unter Verwendung einer objektorientierten Klassenhierarchie und macht sie unabhängig vom eingesetzten Zielrechner. Es kann theoretisch an beliebige Massenspeicher angepasst werden und unterliegt wegen der durchgängigen Verwendung von 64-Bit-Positionszeigern keiner praktisch relevanten Größenbeschränkung, was die abgelegte Informationsmenge angeht. Das Subsystem stellt dem Benutzer eine virtuelle Basisklasse `CIO` zur Verfügung, welche die Schnittstelle für die üblichen Operationen zum Öffnen, Schließen, Lesen und Schreiben der Daten unabhängig vom Medium repräsentiert. Sofern anwendbar ist ebenso das Setzen und Auslesen der aktuellen Schreib- bzw. Leseposition, die Größenänderung und das Löschen eines vorhandenen Objekts möglich. Der persistente Datenspeicher muss durch Ableitung konkretisiert werden. Derzeit existiert eine auf lokale Dateien beschränkte Implementierung `CFileIO`, die alle genannten Operationen unterstützt. Daneben sind auch andere Varianten denkbar, etwa der Zugriff auf Netzwerkdatenströme über verschiedene Kommunikationsprotokolle. Die bestehende spezialisierte Klasse dient ihrerseits als Ausgangspunkt für die Umsetzung eines Typs, der temporäre Dateien mit zufällig generiertem Namen vorhält und automatisch löscht, wenn sie nicht mehr gebraucht werden (`CTempFileIO`). Er bildet die Grundlage des transparenten Auslagerungsmechanismus für umfangreiche Bilddaten.

Im Kontrast zum E/A-Subsystem sind das Versions- wie auch das Mathematikmodul des DaTool-Prototyps unabhängig von der Zielplattform. Ersteres speichert in Gestalt der Klasse `CVersion` die Laufzeit-Typinformationen zur Identifikation und Beschreibung von Objekten, was mit dem in C++ standardmäßig vorgesehenen RTTI-Mechanismus in dieser Form nicht möglich ist. Abgelegt werden die Versionsnummer, der Klartextname des Instanztyps und dessen Aufgabe sowie das verantwortliche Entwicklerteam und eine global eindeutige Zeichenkette in Gestalt einer Version-4-UUID (universally unique identifier) gemäß [42]. Damit ausgestattet sind ummantelnde uniforme Behälterobjekte in der Lage, das jeweils enthaltene Objekt auf Anfrage typisiert zurückzuliefern. Eingesetzt werden solche Container im Softwareprototyp für die Ablage der Algorithmusmodule als auch die Speicherung der zu verarbeitenden Bild-, Modell- und Konfigurationsdaten.

Das Mathematikmodul (`coolmath.h`), das keine objektorientierten Sprachmittel verwendet, enthält eine Reihe von mathematischen Funktionen, die teilweise parametrisierbar sind und häufig benötigt werden. Dazu gehören Extremwertberechnungen für diskrete Mengen (Bestimmung von Minimum und Maximum), die Konvertierung zwischen den Winkeldarstellungen und Ganzzahlarithmetik (Potenzen, Logarithmen). Zusätzlich stellt die Deklaration eine Reihe von „toleranten“ Vergleichsoperationen für Gleitkommazahlen und die Geman-McClure-Korrekturfunktion bereit. Während letztere erst im Kontext des Matchingalgorithmus in Abschnitt 4.6.1.2 thematisiert werden soll, gehen die erstgenannten auf die technisch bedingte limitierte Genauigkeit der Prozessoren zurück. Diese hat zur Folge, dass eigentlich identische reellwertige Ergebnisse durch internes Runden voneinander differieren können und umgekehrt.

[9] GiB steht für Gibibyte und entspricht exakt $2^{30} = 1.073.741.824$ Byte gemäß der verbindlichen Norm IEC 60027-2 [41]. Das sind etwa 7% mehr Daten als bei der missbräuchlichen Verwendung des SI-Präfixes „giga“ (1 GB $= 10^9 = 1.000.000.000$ Bytes).

Um beim Vergleichen vor derartigen Überraschungen gefeit zu sein, die letztlich zu einem schwer lokalisierbaren Fehlverhalten der DaTool-Software führen, wird der Begriff der numerischen Identität nicht über die kongruente Bitrepräsentation zweier Gleitkommazahlen definiert, sondern eine gewisse absolute und relative (prozentuale) Abweichung erlaubt [43]. Die absolute Toleranzgrenze für die Egalität, die deutlich unter Eins liegt, deckt dabei betragsmäßig kleine, die relative Schwelle dagegen große Werte ab. Eine Trennung ist notwendig, da das feste Limit, ab welchem auf Ungleichheit entschieden wird, im zweiten Fall aufgrund der verringerten „gleitenden“ Anzahl der Nachkommastellen wirkungslos bleibt.

4.5. Rahmenprogramm und Bedienkonzept

Das Rahmenprogramm der DaTool-Software fungiert als Einsprungpunkt nach dem Programmstart. Es initialisiert das Hauptfenster der graphischen Benutzeroberfläche und verarbeitet die getätigten Tastatur- und Mauseingaben. Die Implementierung basiert auf der offenen plattformübergreifenden GUI-Bibliothek Qt, welche selbst auf die betriebssystemspezifischen Zeichen- und Ereignisbehandlungsroutinen zurückgreift und über einen umfangreichen Satz von C++-Klassen zugänglich macht. Die optische Gestaltung des Hauptfensters wie auch aller anderen Bedienelemente erfordert dabei keine Programmierung, sondern erfolgt mit dem beigefügten Designerwerkzeug, dessen Beschreibungen (`.ui`-Dateien) anschließend mit dem sogenannten meta object compiler (moc) automatisch in „richtigen“ Quelltext übersetzt werden. Lediglich die Reaktionen auf die Benutzereingaben sind manuell unter Bezugnahme auf die generierten Ausgaben kodiert. Deswegen fällt die Hauptklasse `CMainWindow` des Rahmenprogramms vergleichsweise kompakt aus und hat wenig Entwicklungszeit in Anspruch genommen, was neben der Quellcodeverfügbarkeit die Wahl des GUI-Toolkits auch nachträglich rechtfertigt. Abbildung 4.1 zeigt einen Bildschirmabzug des Hauptfensters auf einem System mit Windows XP.

Die Oberfläche gliedert sich in vier Bereiche, mit denen der Aufbau der Software auf das Bedienkonzept übertragen wird – die Form folgt gewissermaßen der Funktion. Im linken Teil des Hauptfensters ist dabei die Liste der Datenobjekte (object set) lokalisiert, die mit den verfügbaren Algorithmen im rechten oberen Abschnitt bearbeitet werden können. Hierfür wählt man eine Funktion aus, woraufhin die mittig platzierte Parameterliste die formal benötigten Eingabedaten anzeigt. Unter Beachtung des Modus (`in` für reine Eingabedaten, `inout` für modifizierbare Eingabedaten), der Vielfachheit, der akzeptierten Typen und der Beschreibung kann jedem Eintrag ein Datum der Objektliste zugeordnet werden. Dies geschieht über die Selektion sowohl des formalen Arguments als auch des Objekts und die nachfolgende Betätigung der Schaltfläche `Assign Object`. Dabei werden ungeeignete tatsächliche Parameter (etwa ein Bild- anstelle eines 3D-Modellobjekts) abgewiesen, worauf die Statusleiste am unteren Rand mit einem Hinweis aufmerksam macht.

Haben alle Eingabewerte eine Zuweisung erhalten, so lässt sich der Algorithmus schließlich durch die Aktivierung des mit `Run!` beschrifteten Bedienelements starten. Nach der Terminierung erscheinen im rechten unteren Hauptfensterbereich die Ergebnisdaten (Rückgabewerte), die entweder der Objektliste hinzugefügt oder ungenutzt verworfen werden können (Schaltflächen `Move to Obj Set` und `Discard Object`). Die wiederholte Ausführung der genannten Schritte mit anderen Funktionen und Eingabeparametern erlaubt es, zusammengesetzte Verarbeitungsaufgaben nach und nach zu erledigen.

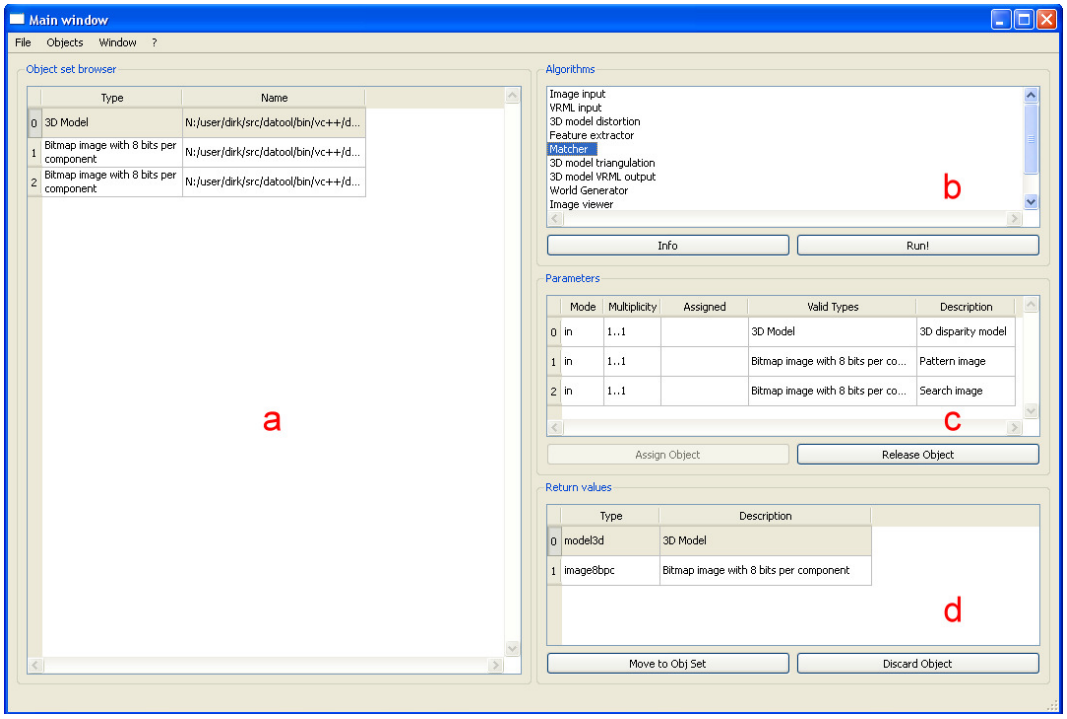


Abbildung 4.1.: Graphische Benutzeroberfläche (GUI) der DaTool-Anwendung (a) Liste der Datenobjekte (b) Algorithmusauswahl (c) Algorithmusparameter (d) berechnete Rückgabewerte

4.6. Datenmanagement

Die Makrokomponente für das Datenmanagement umfasst die objektorientiert umgesetzten dedizierten Datenstrukturen für die Speicherung und Verwendung der Bildinformationen, 3D-Modellbeschreibungen und Konfigurationen in der DaTool-Applikation. Daneben beinhaltet sie die erforderlichen Mantelklassen (wrapper) [44], die die Ablage der für sich genommen heterogenen Typen in einem homogenen Behälterobjekt ermöglichen, wie es etwa der Datenobjektliste im Hauptfenster zugrunde liegt. Diese Konstruktion bildet zusammen mit der Algorithmusliste, die als Speicherstruktur ebenfalls hier behandelt werden soll, den zentralen Kern der Software.

4.6.1. Bilddaten

Für die Verwaltung der Bildinformationen existiert ein umfangreiches Subsystem aus Klassen für die verschiedenartigen Teilaufgaben im Zusammenhang mit den zweidimensionalen Daten. Es deckt dabei neben der Speicherung auch die pixelbasierte und lokale Manipulation des Bildinhalts sowie die Ein- und Ausgabe vom und auf den Massenspeicher unter Berücksichtigung der Aspekte aus der Konzeption ab.

4.6.1.1. Speicherung und Datenzugriff

Die von der DaTool-Software genutzten Bilder werden mit einer Troika von Vorlagenklassen (Templateklassen) gespeichert, die hierfür unterschiedliche Mechanismen bemühen.

Allen Implementierungen gemein ist dabei die Verwendung der üblichen Bitmaprepräsentation und der Schnittstelle `CImage`, von der sie abgeleitet sind. Erstere definiert, dass die Farbkomponenten der Pixel nacheinander Zeile für Zeile abgelegt sein müssen, wobei die linke obere Bildecke den Ausgangspunkt bildet. Letztere verbirgt die Details der Speicherung und ermöglicht die einheitliche Behandlung der Bilddaten. Dies gilt insbesondere für die Metainformationen und den elementaren Pixelzugriff.

Die Metainformationen werden für die Konstruktion und Typidentifikation einer Bildinstanz unabhängig von der konkreten Ausgestaltung benötigt. Sie sind als Objekt der Klasse `CImageDescriptor` in `CImage` integriert und beschreiben die grundlegenden geometrischen, farblichen und technischen Eigenschaften. Hierzu gehören Angaben zur Bildgröße, der Anzahl der Farbkomponenten respektive -kanäle, dem Farbraum und dessen Auflösung (Bits pro Pixelanteil). Außerdem enthält jeder Deskriptor einen Hinweis auf den verwendeten Speichermechanismus nach der Erstellung eines Bildobjekts. Vor der Konstruktion mit einem unverbindlichen Vorschlag belegt, kann eine Anwendung diesen Eintrag optional dazu benutzen, die RAM-Verwaltung dem Benutzerwunsch anzupassen, wovon der DaTool-Prototyp derzeit jedoch keinen Gebrauch macht.

Nach dem erfolgreichen Aufsetzen des Objekts anhand der Metadaten ermöglicht die Schnittstelle schließlich den Zugriff auf die Farbanteile der Pixel. Die Parametrisierung von `CImage` erlaubt es hierbei, verschiedene Basistypen für die Bildpunktkomponenten zu verwenden. Damit lassen sich praktisch beliebige Farbtiefen umsetzen, insbesondere die gemäß der Konzeption benötigte Auflösung von acht bzw. sechzehn Bit pro RGB-Kanal. Auch die Realisierung von doppelt genauen Gleitkommabitmaps für die Disparitätskarten und andere zweidimensionale Ausgaben geschieht über das Vorlagenargument. Für den Zugriff auf die derart typisierten Pixel gibt es situationsabhängig mehrere Varianten. Alle Komponenten eines Bildpunkts können gleichzeitig, einzeln, für den Lesezugriff oder zum Schreiben angefordert werden. Außerdem ist die Selektion eines rechteckigen Ausschnitts variabler Größe mit der `getTile()`-Funktion für die direkte Manipulation durch lokale Bildverarbeitungsalgorithmen möglich. Hierbei gelten allerdings gewisse Einschränkungen, die von der zugrunde liegenden Speichermethode abhängen, womit sich der Kreis zur Konzeption erneut schließt.

Wie bei den Überlegungen zum Speicherplatzbedarf der hochauflösenden Panoramabilder und der daraus berechneten Ergebnisbitmaps festgestellt wurde, kann es vorkommen, dass der Datendurchsatz stark zurückgeht, wenn der betriebssysteminterne Auslagerungsmechanismus aufgrund der fehlenden Kenntnis über die Zugriffsmuster falsche RAM-Bereiche auf dem größeren, aber langsamen Massenspeicher ablegt. Daher gibt es insgesamt drei Implementierungen der `CImage`-Schnittstelle, die diesen Vorgang auf unterschiedliche Art und Weise in die DaTool-Anwendung verlagern, wo das Lese- bzw. Schreibverhalten der Algorithmen vielfach a priori bekannt ist.

RAM-basierte Speicherung Die erste Variante zur Ablage der Bitmapdaten, die Klasse `CRamImage`, zielt primär auf kleinformatige Bilder ab, die sich komplett im Hauptspeicher unterbringen lassen. Eine benutzerdefinierte Auslagerungsstrategie ist daher nicht vorhanden (triviale Lösung). Die Speicherung der zweidimensionalen Informationen erfolgt in einem monolithischen eindimensionalen Feld (Array), welches durch den Vorlagenparameter typisiert ist und dynamisch alloziert wird. Der Arrayindex i aus einer Pixelposition $\mathbf{P}^{(\mathbf{i})}$ relativ zum Start des konsekutiven Speicherbereichs leitet sich dabei aus der horizontalen

bzw. vertikalen Bildgröße d_x bzw. d_y, der Anzahl der Farbkanäle n_{col} und der Nummer der gewünschten Bildpunktkomponente c gemäß Gleichung 4.1 ab.

$$i = n_{col}(\mathbf{P^{(i)}}(y) \cdot d_x + \mathbf{P^{(i)}}(x)) + c \tag{4.1}$$

Diese explizit durchzuführende Linearisierung ist nicht ineffizienter als der Zugriff auf die Bilddaten in einem nichtkonsekutiven zwei- oder dreidimensionalen Feld (bei mehreren Farbkomponenten) mit verschachtelten Indexoperatoren, weil der Compiler hierfür implizit die gleiche Rechenvorschrift generiert. Mehrdimensionale Arrays haben jedoch den Nachteil, dass sie den Einsatz von maschinennah kodierten Assemblerroutinen erschweren, wie sie unter anderem die C++-Laufzeitumgebung mit `memcpy()` oder `memset()` (vgl. erneut [27]) bereitstellt. Dadurch induzieren sie gegenüber der eindimensionalen Repräsentation letztendlich ein ungünstigeres Laufzeitverhalten.

Aufgrund der Vorhaltung der Pixel ausschließlich im RAM[10] ist der Zugriff auf die Inhaltsinformationen sehr schnell. Blockorientierte Algorithmen können zudem über die `getTile()`-Funktion der übergeordneten `CImage`-Schnittstelle das gesamte Bild mit nur einem Aufruf anfordern und direkt im Speicher manipulieren. Diese Vorteile werden freilich mit dem hohen Platzbedarf erkauft, was den Einsatzbereich im Kontext des integrierten Matchingverfahrens je nach RAM-Ausbau auf mehr oder weniger kompakte Panoramen begrenzt.

Speicherung mit Auslagerung von uniformen Kacheln Bei der zweiten Klasse zur Speicherung der Bildinformationen deutet der Name `CGridImage` bereits auf den eingesetzten Mechanismus hin. Analog zum `CRamImage`-Typ wird auch hier ein eindimensionales Feld im RAM angelegt, welches allerdings nicht die gesamte Datenmenge aufnehmen kann und soll. Vielmehr beherbergt das Array eine begrenzte Zahl von Bildausschnitten, deren Auswahl und Form vom verwendeten Betriebsmodus abhängt. Diejenigen Informationen, welche sich nicht im RAM befinden, sind auf den Massenspeicher ausgelagert und werden bei Bedarf mit den gepufferten Bildteilen ausgetauscht. Den Datentransfer übernehmen dabei die Funktionen des E/A-Subsystems aus Abschnitt 4.4.

Der erste Betriebsmodus von `CGridImage`, welcher automatisch beim pixelweisen Zugriff über die geerbte `CImage`-Schnittstelle aktiviert wird, teilt das gesamte Bild in ein Gitter aus uniformen rechteckigen Kacheln ein. Deren Anzahl ergibt sich aus der frei konfigurierbaren Größe, wobei die Bildränder gegebenenfalls überragt werden, so dass jedes Pixel zu genau einem Ausschnittselement gehört. Von allen Kacheln speichert die Klasse lediglich eine wählbare Menge in dem eindimensionalen Feld, welches dazu eine passende Dimensionierung und Unterteilung in identische Abschnitte, sogenannte Slots, erfährt. Verwaltet werden die im Array verfügbaren Slots über eine Warteschlange `m_memTileInfo`, welche die Speicheradresse und den Zustand (leer, Kachel eingelagert oder Kachel beschrieben) nach dem FIFO-Prinzip[11] festhält. Da die Überprüfung des Zustands einer beliebigen Kachel anhand der Warteschlange jedoch eine lineare Zeitkomplexität aufweist – schlimmstenfalls ist hier ein vollständiger Durchlauf durch das Entnehmen von Elementen am Ende und Hinzufügen selbiger am Anfang erforderlich – gibt es parallel eine kompakte zweidimen-

[10] sofern das Betriebssystem seinerseits keine Auslagerung aufgrund von exzessiven Speicheranforderungen durchführt

[11] FIFO steht für „first in first out“. Zeitlich früher in die Warteschlange eingefügte Elemente können auch zeitlich früher entnommen werden.

sionale Verzeichnisstruktur `m_tileInfo`, die die Speicherorte und Zustände aller Kacheln protokolliert und in konstanter Zeit manipuliert werden kann.

Wenn der Benutzer nun ein bestimmtes Pixel zur Manipulation anfordert, so wird über dessen Koordinaten das zugehörige Gitterelement mittels ganzzahliger Division durch die Kachelgröße berechnet. Anschließend erfolgt die Überprüfung mit dem zweidimensionalen Verzeichnis, ob das ermittelte Rechteck bereits in einem Slot des eindimensionalen Feldes präsent ist. In diesem Fall kann der gewünschte Bildpunkt direkt anhand seiner Position innerhalb der Kachel zurückgeliefert und der Zustandseintrag sowohl in der Warteschlange als auch in `m_tileInfo` auf „Kachel geladen“ (Verwendung einer ausschließlich lesenden Schnittstellenoperation) oder „Kachel beschrieben“ (Verwendung einer modifizierenden `CImage`-Operation, auch wenn nichts geändert wird) aktualisiert werden.

Befindet sich das berechnete Gitterelement dagegen noch nicht im Speicher, gestaltet sich die Bereitstellung des angeforderten Bildpunkts ungleich zeitaufwendiger. In diesem Fall wird zunächst überprüft, ob noch mindestens ein freier Slot in der Warteschlange der eingelagerten Kacheln unbesetzt ist. Hat `m_memTileInfo` noch nicht die vorgegebene Maximallänge erreicht, so lädt `CGridImage` das zum Bildpunkt gehörende Rechteck an denjenigen Slot im Feld, welcher der um eins inkrementierten letzten Warteschlangenposition zugeordnet ist, und setzt den Kachelstatus wie oben erwähnt.

Sind schon alle Slots im Feld belegt, hat also die Warteschlange die vorgegebene Maximallänge erreicht, muss vor dem Einlesen der berechneten Kachel zunächst ein Gitterelement aus dem Hauptspeicher entfernt werden. Wegen des konstanten Zeitbedarfs für die Löschung verwendet die `CGridImage`-Klasse hierfür stets das erste Warteschlangenelement, was auf eine Rundlauf-Austauschstrategie (round robin) hinausläuft. Das betroffene Datum wird zuvor allerdings an die richtige Position auf dem Massenspeicher zurückgeschrieben, wenn es die Markierung „Kachel beschrieben“ aufweist, und der Status in der zweidimensionalen Verzeichnisstruktur `m_tileInfo` auf „leer“ im Sinne von „nicht geladen“ zurückgesetzt. Wurde die Kachel nicht verändert, kann sie direkt verworfen und der neue Inhalt wie beschrieben eingelagert werden, da die Warteschlange nach dem Entfernen nicht mehr voll besetzt ist.

Wie die Erläuterung des Auslagerungsmechanismus von `CGridImage` verdeutlicht, bedingt der für den Benutzer transparente pixelorientierte Zugriff auf umfangreiche Bilddatenmengen eine mehrstufige Fallunterscheidung *pro Punktoperation*, die selbst im günstigsten Fall mit einem Mehraufwand an Rechenzeit gegenüber der `CImage`-Klasse verbunden ist. Die tatsächliche Verzögerung im Vergleich zur komplett RAM-basierten Datenhaltung hängt dabei maßgeblich von der Anzahl und Größe der Kacheln und dem Zugriffsmuster ab. Das zeitraubende Auslagern wird, wie im Übrigen bei jeder Art von limitiertem Zwischenspeicher, durch örtliche Zugriffe minimiert, wie sie etwa bei digitalen Filtern und den Matchingalgorithmen zu verzeichnen sind. Zusätzlich steigert die Rundlaufstrategie die Effizienz von verteilt lokalen Bearbeitungsverfahren, die die Bildinformationen an mehreren abgegrenzten Stellen gleichzeitig auslesen und modifizieren. Diese Leistung vollbringt im Grunde genommen zwar auch das Betriebssystem, indem es typischerweise diejenigen Speicherblöcke einlagert, auf die gerade zugegriffen wird (demand paging). Dabei berücksichtigt es jedoch nicht die zweidimensionale Natur der Bilder. Bei großen 2D-Datensätzen und den üblichen Blockgrößen im Kibibytebereich [45] befinden sich bereits zwei vertikal angrenzende Bildpunkte nicht mehr auf der gleichen logischen Speicherseite und provozieren Aus- und Einlagerungsaktivität.

Sollte die mit den pixelweise operierenden Routinen erreichbare Verarbeitungsgeschwindigkeit zur Bildmanipulation für ein Anwendungsszenario nicht ausreichen, gibt es mit dem zweiten blockorientierten Betriebsmodus der `CGridImage`-Klasse die Möglichkeit, den im RAM zu puffernden Bildausschnitt selbst festzulegen. Dies geschieht wiederum über die von der `CImage`-Schnittstelle vermachte `getTile()`-Funktion zur Selektion eines rechteckigen Bereichs, die einen mit dem Vorlagenparameter typisierten Zeiger in das eindimensionale Pufferfeld zurückliefert. Im Gegensatz zu `CRamImage` ist die Größe des ansonsten frei wählbaren Fensters durch den vom Array belegten Speicherplatz begrenzt und wird bei einer Überschreitung mit dem ungültigen Nullpointer quittiert. Von der durch den typisierten Adresswert erhaltenen größenbeschränkten Zugriffsmöglichkeit auf die „rohen" Bilddaten kann anschließend mittels Zeigerarithmetik und Dereferenzierung lesend oder schreibend Gebrauch gemacht werden, ohne dass ein Zugriff auf den Massenspeicher erfolgt.

Dabei tut man – wie übrigens bei den anderen Bildspeicherklassen auch – gut daran, sich möglichst nicht zu verrechnen. Schon die kleinste Überschreitung des von der blockorientierten Zugriffsfunktion genehmigten Rechteckareals endet bestenfalls in einem Absturz der bildverarbeitenden Anwendung, kann aber auch nichtdeterministisch in undefiniertem Verhalten resultieren. Insbesondere der Wechsel der Bildzeile im bereitgestellten Fenster gemäß Gleichung 4.1 muss gut bedacht werden. Hierfür stellt die `getTile()`-Routine zur Anforderung des rechteckigen Ausschnitts über einen Referenzparameter das erforderliche Adressinkrement zur Verfügung. Damit ist es möglich, den korrekten Zeigerwert für die Erhöhung respektive Erniedrigung der vertikalen Position bei gleicher Abszisse zu bestimmen, auch wenn die Bildbreite nicht der horizontalen Fensterausdehnung entspricht (siehe Abbildung 4.2).

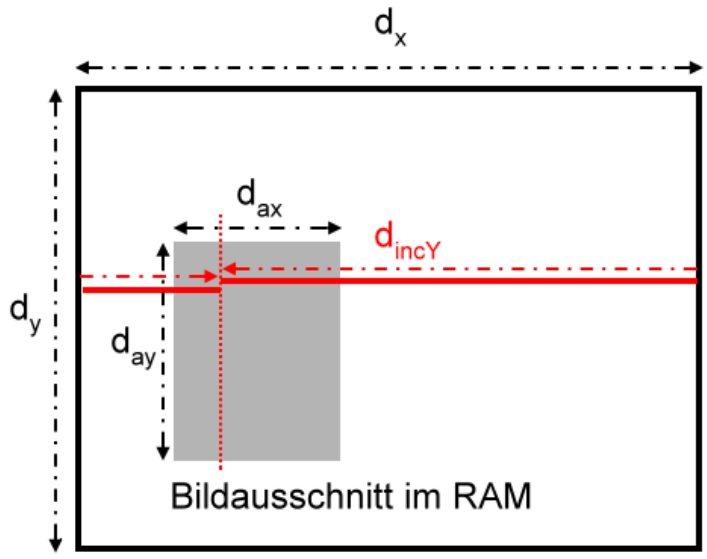


Abbildung 4.2.: Zeilenwechsel bei Verwendung der blockorientierten Zugriffsfunktion auf einem Graustufenbild. Das Adressinkrement d_{incY}, um bei gleicher Abszisse in die nächste Bildzeile zu gelangen, ist in der dargestellten `CRamImage`-Instanz nicht identisch mit der horizontalen Ausdehnung d_{ax} des angeforderten rechteckigen Bildareals (grau), sondern mit der waagerechten Bildabmessung d_x. Bei `CGridImage`-Instanzen liegt der Ausschnitt immer zeilenweise konsekutiv im RAM-Puffer vor, weswegen stets $d_{incY} = d_{ax}$ gilt. Werden Farbbilder verwendet, vervielfacht sich d_{incY} um die Zahl der Komponenten pro Pixel.

Wenn die `CGridImage`-Instanz schließlich zerstört wird, man einen neuen Bildausschnitt anfordert oder die pixelbasierten Schnittstellenoperationen verwendet, wird der Inhalt des selektierten Fensters in jedem Fall zurückgeschrieben. Die Verwendung mehrerer Ausschnitte für verteilt lokale Algorithmen ist aufgrund der dabei auftretenden Überlappungs- und Synchronisationsproblematik derzeit nicht explizit vorgesehen. Sofern alle durchzuführenden Berechnungen in den angeforderten Bildbereich fallen, ist diese Einschränkung unbedeutend. Andererseits können die Operationen, welche auf einzelnen Bildpunkten arbeiten, im Mischbetrieb mit der fensterorientierten Routine der `CImage`-Schnittstelle aufgerufen werden. Infolge der mit jedem Moduswechsel verbundenen zeitraubenden Auslagerungsvorgänge ist dies jedoch nur in wohldosierter Form empfehlenswert, etwa dann, wenn wenige Pixelzugriffe der anschließenden direkten Manipulation eines großen Bildausschnitts vorausgehen.

Die Vorteile des zweiten Betriebsmodus liegen auf der Hand. Die Selektion eines Fensters durch den Benutzer ermöglicht die optimale Anpassung an die lokal arbeitenden Algorithmen, da sich die Zahl der Massenspeicherzugriffe bestmöglich minimieren lässt. Zudem können in Assemblersprache kodierte Rechenvorschriften aufgrund der konsekutiven Lage des zurückgelieferten Bildausschnitts im Speicher geradlinig implementiert und eingesetzt werden. Nachteilig ist die begrenzte Tauglichkeit für verteilt lokale Bildverarbeitungsmethoden und das Fehlen von Schutzmechanismen, wenn durch Rechenfehler Pixelpositionen außerhalb des gewählten Rechtecks angesprochen werden. Daneben müssen die Algorithmen, welche Gebrauch von der blockorientierten `CImage`-Schnittstelle machen, in der Regel dahingehend angepasst werden, dass sie den während der Ausführung jeweils benötigten Bildinhalt ohne entbehrliche Überschneidungen anfordern. Ansonsten schwindet die Laufzeitersparnis durch verzichtbare Lese- und Schreibzugriffe vom und auf den Massenspeicher dahin.

Speicherung mit Auslagerung von uniformen Kacheln ohne Division Die dritte Möglichkeit zur Speicherung der Bilddaten über die `CImage`-Schnittstelle, die Klasse `CFastGridImage`, verwendet das gleiche gekachelte Funktionsprinzip wie `CGridImage`. Der einzige Unterschied besteht in der Limitierung der Größe der Gitterelemente, deren Abmessungen in horizontaler wie vertikaler Richtung eine Zweierpotenz ergeben müssen. Dadurch können die Divisionen sowohl bei der uniformen Aufteilung des Pufferfeldes wie auch der Abbildung einer Pixelposition auf die zugehörige Kachel vermieden und durch binäre Schiebeoperationen ersetzt werden. Diese benötigen weniger Taktzyklen und unterliegen, zumindest was die DaTool-Software angeht, der Parallelisierung auf Instruktionsebene durch die superskalaren Intel-kompatiblen Zielprozessoren [46][47]. In der Konsequenz erfahren die millionenfach aufgerufenen pixelbasierten Schnittstellenoperationen insgesamt eine Beschleunigung.

4.6.1.2. Höhere bildverarbeitende Algorithmen

Die erwähnten Speicherklassen ermöglichen über die gemeinsame `CImage`-Schnittstelle lediglich den elementaren Zugriff auf die Bildinformationen mit Pixelpräzision. Höhere Algorithmen, zu denen vereinfachende Makrofunktionen ebenso wie Subpixelroutinen, Farbraumkonvertierungsvorschriften und die lokalen Matchingschritte zählen, werden wegen der beabsichtigten Trennung von Datenhaltung und funktionaler Programmlogik und zur

Vermeidung einer überladenen Repräsentation nach außen[12] hingegen nicht abgedeckt. Obwohl sie streng genommen nicht zur Datenverwaltung gehören, werden diese Module aufgrund der Tatsache, dass sie unabhängig von der DaTool-Software eingesetzt werden können, und ihrer Quellcodezugehörigkeit an dieser Stelle erläutert.

Makrofunktionen zur vereinfachten Handhabung Die Flexibilität der Bilddatenhaltung mit den drei Speicherklassen und ihrer großen Zahl an Konfigurationsmöglichkeiten verkompliziert unweigerlich die Anwendung durch den Programmierer. So muss man für umfangreiche Bilder nicht nur die richtige Klasse auswählen und mit geeigneten Parametern instanziieren, um die abgelegten Informationen mit maximaler Geschwindigkeit manipulieren zu können. Zusätzlich ist es erforderlich, die Konstruktion und Zerstörung des für die Auslagerung essentiellen E/A-Objekts vorzunehmen. Ebenfalls aufwendig gestaltet sich das Kopieren zwischen zwei Klasseninstanzen mit verschiedenen Ablagemechanismen und, wie noch zu sehen sein wird, das Laden und persistente Speichern der Bilder.

Um die Benutzung der `CImage`-Derivate zu vereinfachen, sind die genannten kompositen Aktivitäten in einer einzigen Klasse namens `CImageUtility` in Form von Makrooperationen zusammengefasst. Für die Erzeugung und Zerstörung der Bildinstanzen genügt dadurch ein einziger Funktionsaufruf, infolgedessen automatisch die geeignete Speicherklasse bestimmt, falls nötig ein temporäres Auslagerungsobjekt des Typs `CTempFileIO` kreiert und selbiges nach Gebrauch wieder gelöscht wird. Analog verhält es sich mit der Bildduplikation und dem Datentransfer vom und zum Massenspeicher. Das Hinzufügen weiterer Makroroutinen ist in der Zukunft denkbar, allerdings sollte dabei die bisher noch nicht signifikante inhaltliche Überfrachtung von `CImageUtility` im Auge behalten werden. Gegebenenfalls ist zur Sicherstellung des vereinfachenden Charakters der Klasse eine funktionale Untergliederung zielführend.

Farbraumkonvertierung Bei der Verarbeitung von Bilddaten passiert es häufig, dass die Pixel vom ursprünglichen Farbraum in einen anderen übersetzt werden müssen. Im DaTool-Prototyp kommt diese Aufgabe der `CColorConverter`-Klasse mit ihrer Funktion `convert()` zu. Für die Konvertierung von RGB-Bildpunkten in Grauwerte L mit den perzeptuellen Gewichten nach [48]

$$L = 0,21 \cdot R + 0,72 \cdot G + 0,07 \cdot B \tag{4.2}$$

und in umgekehrter Richtung via Intensitätswertduplikation sind geeignete Spezialisierungen implementiert. Weil deren Definitionen zum Übersetzungszeitpunkt vollständig bekannt sind, kann der Compiler in beiden Fällen hochoptimierten Binärcode erzeugen und die maßgebliche `convert()`-Routine jeweils „inline“ an der Stelle ihres Aufrufs einfügen.

Subpixelzugriff Abgesehen von den akkuraten Matchingalgorithmen ist es in vielen Situationen vorteilhaft, den diskreten Charakter der Bilddaten zu ignorieren und den Farbwert an einer Position zwischen dem regulären Punktraster auszulesen. Für die Bestimmung eines solchen Subpixels existiert ein eigenes Modul mit der Funktionsklasse `CSubPixel`. Diese rekonstruiert das ursprüngliche kontinuierliche Bildsignal vor der Aufnahme

[12] Man spricht in diesem Zusammenhang von fetten Schnittstellen (fat interfaces, siehe [27]), die mehrere logisch trennbare Konzepte vereinen und dementsprechend viele unübersichtliche Operationen anbieten müssen.

durch die Kamera aus den vorhandenen Bildelementen und ist anschließend in der Lage, für jede Gleitkommaposition ein Farb- oder Grauwertdatum zu liefern.

Die Komplettierung der fehlenden Bildinformation zwischen den vorliegenden Pixeln wird dabei durch die Interpolation mit Hilfe von Tiefpassfiltern bewerkstelligt [49]. Weil der optimale Filter in Gestalt der normalisierten sinc-Funktion

$$sinc : \mathbb{R} \to \mathbb{R}, \quad sinc(x) = \begin{cases} \frac{\sin(\pi x)}{\pi x}, & x \neq 0 \\ 1, & x = 0 \end{cases} \tag{4.3}$$

aufgrund der unendlichen Ausdehnung praktisch unbrauchbar ist, kommen stattdessen Varianten mit kompaktem Träger zum Einsatz. Dies sind nach [50]:

- der Rechteck-, Box- oder nearest neighbour-Filter (verwendet das nächstgelegene Pixel)

$$f_{box}(x) = \begin{cases} 1, & |x| \leq \frac{1}{2} \\ 0, & sonst \end{cases} \tag{4.4}$$

- der Dreiecksfilter (entspricht der bilinearen Interpolation)

$$f_{triangle}(x) = \begin{cases} 1 - |x|, & 0 \leq |x| \leq 1 \\ 0, & sonst \end{cases} \tag{4.5}$$

- der kubische B-Spline-Filter (entspricht der stückweise polynomialen Interpolation)

$$f_{bspline}(x) = \begin{cases} \frac{1}{2}|x|^3 - 1|x|^2 + \frac{2}{3}, & |x| \leq 1 \\ -\frac{1}{6}|x|^3 + |x|^2 - 2|x| + \frac{4}{3}, & 1 < |x| \leq 2 \\ 0, & sonst \end{cases} \tag{4.6}$$

Die Qualität des näherungsweise rekonstruierten Bildsignals nimmt dabei tendenziell ebenso wie die benötigte Rechenzeit von oben nach unten zu, wobei die bilineare Funktion einen guten Kompromiss darstellt. Um nun mit einem der Filter f die Farbe des Subpixels $p_s(x, y, c)$ zu berechnen, verwendet man den Funktionswert als Wichtungsfaktor (reeller Filterkoeffizient) in einer zweidimensionalen Faltungsoperation. Für das Bild I ergibt sich mit der durch den Träger definierten quadratischen Umgebung der Größe d_f um p_s herum die Gleichung

$$p_s(x, y, c) = \sum_{l=-\frac{1}{2}d_f}^{\frac{1}{2}d_f} \sum_{k=-\frac{1}{2}d_f}^{\frac{1}{2}d_f} I(\lfloor x + l \rfloor), \lfloor y + k \rfloor, c) \cdot f(l) f(k) \ . \tag{4.7}$$

Bei der Applikation von Filterfunktionen, die negative Werte annehmen können oder nicht normalisiert sind (d.h. $\int_{-\infty}^{\infty} f(x)dx \neq 1$), hier nirgends der Fall), ist anschließend noch eine Skalierung mit dem Kehrwert der Summe der Wichtungsfaktoren unabdingbar. Andern-

falls treten bei den vorzeichenlosen ganzzahligen Pixelwerten sichtbare Unter- oder Überlaufeffekte auf. Implementierungstechnisch ist die Faltungsformel in der `CSubPixel`-Klasse durch ineinander verschachtelte for-Schleifen umgesetzt. Zusammen mit den als Traitklassen [51] ausgeführten Filterfunktionen, in denen die Größe des Trägers und eine Marke zur Normalisierung statisch definiert sind, hat der Compiler die Möglichkeit, die Schleifenkörper sequenziell auszuschreiben (loop unrollling-Optimierung) oder automatisch zu parallelisieren.

Analog zum Auslesen der Bilddaten gestaltet sich das Setzen von Subpixeln. Hierbei müssen alle Rasterpunkte im Bereich des Filterträgers mit den gewichteten Farbanteilen des neuen Bildelements additiv modifiziert werden. Wegen der fehlenden Notwendigkeit für den vorgeschlagenen Matchingalgorithmus ist das Einfügen von Zwischenpunkten derzeit allerdings nicht realisiert.

Lokale Matchingschritte Eine weitere Gruppe von höheren Algorithmen, die auf den Speicherklassen mit der einheitlichen `CImage`-Schnittstelle aufbauen, bilden die lokalen Matchingschritte, welche jeweils *einen* rechteckigen Musterbild- mit *allen* gleichförmigen Suchbildausschnitten innerhalb des Suchbereichs vergleichen. Die Bestimmung der besten Zuordnung ist dabei an unterschiedliche Erfordernisse in Bezug auf die Rechengeschwindigkeit und die Präzision der Korrespondenzen adaptierbar. Um den notwendigen Grad an Flexibilität mit möglichst geringem Implementierungsaufwand zu erreichen, bauen alle lokalen Verfahren auf der gleichen parametrisierbaren Codebasis auf. Neben der Schnittstelle für die eigentliche Korrespondenzanalyse in Gestalt der Klasse `CLocalMatcher` umfasst diese einen Satz von separaten Hilfsklassen `CLMConfig`, `CLMResult`, `CLMArea` und `CLMSimilarityMeasure`, die spezielle Speicher- und Rechenaufgaben übernehmen.

Die Schnittstelle der lokalen Matchingalgorithmen `CLocalMatcher` bietet Funktionen für die Konfiguration und den Start der Zuordnungssuche sowie zur Abfrage der besten gefundenen Übereinstimmung an. Über die Vorlagenargumente lässt sich definieren, mit welchem Auflösungstyp, welcher Art von Bilddaten und welcher Distanzfunktion der Analyseprozess bewerkstelligt werden soll. In Bezug auf die erste Angabe, die die theoretisch mögliche Entfernungspräzision begrenzt, aber auch einen maßgeblichen Einfluss auf die Matchinggeschwindigkeit ausübt, existieren zwei partiell spezialisierte Vorgaben. Die Klasse `CLocalMatcherStd` fixiert den Auflösungstyp auf ganze Zahlen und repräsentiert somit pixelgenaue Zuordnungsverfahren nach außen, die aufgrund der befehlsparallelen Integerarithmetik moderner Prozessoren entsprechend schnell arbeiten. Für die Implementierung der Suche nach inhaltlichen Übereinstimmungen mit Subpixelgenauigkeit ist die mit doppelt genauen Gleitkommazahlen operierende Schnittstelle namens `CLocalMatcherPrecise` angedacht. Der Vorlagentyp, der die Art der Bilddaten charakterisiert, entspricht dem Pixelkomponententyp der Bildinstanzen, welche Gegenstand der Korrespondenzbestimmung sind. Er wird – ebenso wie das dritte Argument in Form eines numerischen Bezeichners für die Abstandsfunktion – zur Instanziierung der Hilfsklassen und verschiedener lokaler Variablen in den Implementierungen der `CLocalMatcher`-Schnittstelle benötigt. Aus programmiertechnischen Gründen bedürfen beide Parameter dennoch der Spezifikation auf der höchsten Ebene der Klassenhierarchie.

Konsequent abhängig von den eingesetzten Vorlagenparametern, jedoch losgelöst von der konkret verwendeten Rechenvorschrift, gestaltet sich auch die Konfiguration der Matchingalgorithmen vor deren Aufruf. Die hierfür verantwortliche Funktion wird zwar von

`CLocalMatcher` bereitgestellt, jedoch sind die entsprechenden Angaben in der ausgelagerten Hilfsklasse `CLMConfig` respektive den vordefinierten Teilspezialisierungen `CLMConfigStd` und `CLMConfigPrecise` konzentriert. In Anlehnung an das Arbeitsprinzip aus dem Grundlagenabschnitt 2.1.4 und unter Berücksichtigung der konzeptionellen Aussagen aus Sektion 3.2.4 werden für die Einstellung sechzehn Attribute benötigt. Hierunter fallen mit der an den soeben referenzierten Stellen benutzten Notation und unter Angabe der Quelltextbezeichner in Klammern

- das Musterbild I_1 (`m_patternImage_p`),
- das Suchbild I_2 (`m_searchImage_p`),
- die Marke, ob die für das Matching verwendeten Intensitätswerte einem speziellen Farbkanal entnommen werden oder aus der perzeptuell gewichteten Kombination aller Pixelkomponenten hervorgehen sollen (`m_combineChannels`),
- der für die Zuordnungssuche gegebenenfalls zu benutzende spezielle Farbkanal (`m_channel`),
- der Vorlagenpunkt $\mathbf{X}_{\mathbf{k1}}^{(\mathbf{i})}$ im Musterbild (`m_patternX`, `m_patternY`),
- das geschätzte Suchbereichszentrum $\mathbf{X}_{\mathbf{k2}}^{(\mathbf{i})}$ im Suchbild (`m_searchX`, `m_searchY`),
- der Größe des rechteckigen Muster- und des Suchfensters W_{cp} bzw. W_{cs}, für welche die Korrelation berechnet werden soll (`m_corrDimX`, `m_corrDimY`),
- die Größe des rechteckigen Suchbereichs W_s (`m_searchDimX`, `m_searchDimY`),
- die Korrelationsschrittweite (`m_pixelStepX`, `m_pixelStepY`) und
- der Verfeinerungsfaktor, um den die Schrittweite für die Verschiebung des Suchfensters im Suchbereich geringer ausfällt als die Korrelationsschrittweite (`m_searchStepRefinementX`, `m_searchStepRefinementY`) .

Wenngleich diese Werte für sich genommen in ihrer Bedeutung wohldefiniert sind, besteht im Zusammenspiel bei der Korrespondenzanalyse noch der eine oder andere Interpretationsspielraum. Jede von `CLocalMatcher` abgeleitete Matcherimplementierung kann generell selbst festlegen, wie beispielsweise das Musterfenster in Relation zum Vorlagenpunkt lokalisiert ist oder ob sich alle Suchfenster zu jedem Zeitpunkt vollständig innerhalb des Suchbereichs befinden müssen.

Damit der Implementierungsaufwand auf das minimal erforderliche Maß beschränkt und die direkte Vergleichbarkeit der Zuordnungen gewährleistet bleibt, existiert mit `CLMArea` eine Template-Klasse, welche die Auslegung der Konfigurationsparameter für die DaTool-Software einheitlich festschreibt. Sie muss analog zu `CLMConfig` mit dem Auflösungstyp von `CLocalMatcher` instanziiert werden. Die einzige nichttriviale Funktion in `CLMArea` mit der Bezeichnung `computeLMArea()` erhält als Eingabe eine Konfigurationsinstanz. Daraus berechnet sie die vier absoluten Koordinaten `m_sx1`, `m_sy1`, `m_sx2` und `m_sy2` des rechteckigen Suchbereichs sowie vier Koordinaten `m_cx1`, `m_cy1`, `m_cx2` und `m_cy2` für das Korrelationsfenster relativ zum Vorlagenpunkt – letztere sind folglich nicht auf den positiven Wertebereich einschließlich der Null beschränkt.

Wie in Listing 4.1 dargestellt ist, können die selektierten Bildausschnitte mit diesen Angaben über ein Paar von jeweils zwei Schleifen durchlaufen und im innersten Codeblock beliebige Berechnungen (etwa die des Ähnlichkeitsgrades) ausgeführt werden. Der Ausdruck `ResType` steht dabei für den von `CLocalMatcher` übernommenen Auflösungstyp. Im Normalfall ergibt sich infolge der skizzierten Umsetzung eine Zentrierung des Vorlagenpunkts im Musterfenster, welche auf den Bezugspunkt für die Korrelation im Suchfenster übertragen wird. Ebenso liegt das Suchbereichszentrum mittig im Suchbereich. Beim Suchvorgang selbst wird das Zentrum des Suchfensters innerhalb des Suchbereichs bewegt, es können sich demnach Fensterteile außerhalb der Verschiebungsgrenzen befinden.

```
for(ResType sy=m_sy1; sy<=m_sy2; sy+=m_pixelStepY/
m_searchStepRefinementY) {

    for(ResType sx=m_sx1; sx<=m_sx2; sx+=m_pixelStepX/
    m_searchStepRefinementX) {

        for(ResType cy=m_cy1; cy<=m_cy2; cy+=m_pixelStepY) {
            for(ResType cx=m_cx1; cx<=m_cx2; cx+=m_pixelStepX) {

                // Operation auf den Punkten an Position
                //
                // (m_patternX+cx, m_patternX+cy) im Musterbild
                //
                // und
                //
                // (sx+cx, sy+cy) im Suchbild

            } // cx
        } // cy

    } //sx
} // sy
```

Listing 4.1: Beispielhafter Gebrauch der Ausgaben von `CLMArea`

Ein weiterer Aspekt von `CLMArea` ist, dass die Klasse das untrennbar mit den lokalen Bildverarbeitungsalgorithmen verbundene Randproblem berücksichtigt. Wie in der Konzeption festgestellt wird, sind die gebräuchlichen Methoden zur Behandlung der Bildgrenzen für die Ermittlung der Zuordnungen ungeeignet. Um alle Musterpixel verwenden zu können, bedient sich `CLMArea` der umrissenen Verschiebungstechnik. Dabei wird vom Normalfall der lokalen Musterumgebung abgewichen und der zentral befindliche Vorlagenpunkt, der im Musterfenster mit dem Bezugspunkt für die Korrelation zusammenfällt, durch eine Verschiebung des Rechteckareals an die Seite desselbigen bewegt. Diese Lageanpassung findet immer dann statt, sobald die Entfernung des Vorlagenpunkts zum Bildrand in seiner ursprünglichen mittigen Lage weniger als die Hälfte der horizontalen oder vertikalen Ausdehnung des Musterbereichs beträgt. Beide Koordinaten werden hierfür getrennt mit je einer dreistufigen Fallunterscheidung analysiert. Einschließlich der Zentrallage im Normalfall ergeben sich insgesamt neun mögliche Relativpositionen für den Bezugspunkt im Musterausschnitt (Abbildung 4.3(a)), die sich geradlinig auf das gleich große Suchfenster übertragen.

Analog dazu darf die Umgebung im Suchbild, in welcher die Korrespondenz zum Vorlagenpunkt vermutet wird, einschließlich der über diesen Bereich hinausragenden Suchfenster-

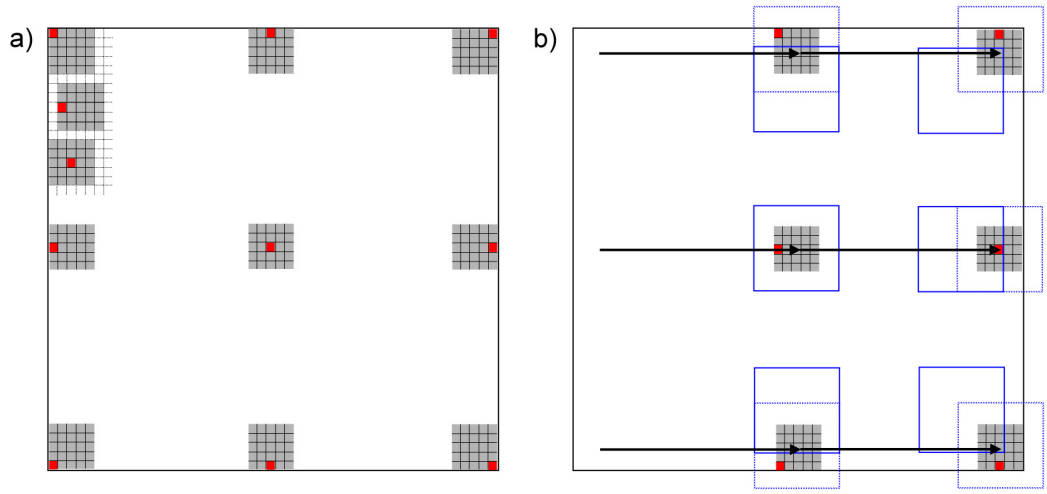


Abbildung 4.3.: (a) Verschiebung des Musterfensters (grau), so dass der Vorlagenpunkt (=Bezugspunkt für die Berechnung des Ähnlichkeitsgrades) in Bildrandnähe nicht mehr mittig darin liegt (b) Die ohne Größenänderung angepassten Suchbereiche (blau durchgehend) in den Randlagen des zu (a) gehörenden Suchbilds. Das Suchfenster (grau) ragt partiell über den Suchbereich hinaus, liegt aber im Gegensatz zur normalerweise eingesetzten zentrierten Anordnung (blau gestrichelt) stets im Bild.

teile die Bildgrenzen nicht verletzen. Weil die Maße des rechteckigen Suchbereichs gemäß den Benutzervorgaben auch hier konstant sein müssen, um vergleichbare Matchingresultate zu erhalten, wird bei einer Überschreitung der Ränder eine Verlagerung desselbigen in Richtung der Bildmitte vorgenommen (Abbildung 4.3(b)).

Sollten die Konfigurationsangaben zur Ausdehnung des designierten Zielgebiets aufgrund einer zu geringen Bildgröße allerdings schon von vornherein Makulatur sein, führt `CLMArea` im Gegensatz zum Muster- und Suchfenster eine nachträgliche achsenseparate Größenadaption des Suchbereichs durch. Die Verkleinerung erfolgt dabei durch die Translation der linken bzw. oberen Suchbereichskante in das Bild hinein, gefolgt von einer Kürzung derselbigen rechts respektive unten, wobei die überstehenden Suchfensterteile eingerechnet werden. Sie garantiert die uniforme Gestalt des geschrumpften rechteckigen Suchraums, solange die an `CLMArea` übergebene `CLMConfig`-Instanz identische Abmessungsangaben aufweist. Daher können die Matchingergebnisse auch für schmale horizontale oder vertikale Bildstreifen in Relation zueinander gesetzt werden.

Die Berechnung des Ähnlichkeitsgrades für jeweils ein Paar von Pixelkomponenten des aktuellen Muster- und Suchfensters, die mit `CLMArea` präzisiert sind, realisieren die `CLMSimilarityMeasure`-Traits. Dabei wird die Auswahl zwischen den verfügbaren Spezialisierungen für die SAD- und SSD-Abstandsfunktion über ganzzahlige Indizes, die durch den Aufzählungstyp `LMMetric` vorgegeben sind, zur Übersetzungszeit fixiert. Dies erleichtert abermals die Optimierung durch den Compiler. Neben der für den Distanzwert relevanten Funktion `computeMetric()` definieren die `CLMSimilarityMeasure`-Konkretisierungen auch Routinen für die Abfrage des initialen Übereinstimmungsgrades (`getInitialSimilarity()`), den theoretisch minimalen und maximalen Ähnlichkeitswert (`getWorstSingleSimilarity()`, `getBestSingleSimilarity()`) und den Vergleich zweier Abstände (`isMoreSimilar()`). Die zurückgelieferten Informationen beziehen sich dabei nicht auf die „rohen“ SAD- oder SSD-Beträge, sondern auf das Ergebnis der nachgeschalteten Ausrei-

ßerbehandlung, welche die Originalwerte jedoch bei zukünftigen Erweiterungen entgegen der aktuellen Quelltextversion auch unmodifiziert weiterreichen darf.

Wenn einige wenige Pixelkomponentenpaare bei der Korrelation überdurchschnittlich stark voneinander abweichen, kann dies dazu führen, dass eine qualitativ akzeptable Zuordnung degradiert und in der Folge von einer `CLocalMatcher`-Implementierung fälschlicherweise verworfen wird. Um diesen Effekt abzumildern, implementieren die existierenden `CLMSimilarityMeasure`-Spezialisierungen mit dem noch verbleibenden `handleOutlier()`-Unterprogramm eine Korrekturvorschrift, die den numerischen Ähnlichkeitswert begrenzt. Im Fall der monoton steigenden Differenzbeträge bzw. -quadrate des SAD- bzw. SSD-Übereinstimmungskriteriums übernimmt die Geman-McClure-Funktion [52]

$$f(\sigma, x) = \frac{x^2}{\sigma^2 + x^2} \tag{4.8}$$

aus dem Mathematikmodul der gemeinsam genutzten Komponenten die Ausreißerbehandlung. Dazu wird sie auf das Resultat jedes atomaren Vergleichs der Pixelkomponenten des Muster- respektive Suchfensters vor der Summenbildung angewandt. Mit der Notation aus dem Grundlagenabschnitt 2.1.4.1 erhält man somit für die korrigierenden Abstandsmaße SAD_{corr} und SSD_{corr}

$$SAD_{corr} = \sum_{x,y \in W_{cp}} f(\sigma, |I_1(x,y) - I_2(\Delta x_{W_{cs}} + x, \Delta y_{W_{cs}} + y)|) \tag{4.9}$$

$$SSD_{corr} = \sum_{x,y \in W_{cp}} f(\sigma, (I_1(x,y) - I_2(\Delta x_{W_{cs}} + x, \Delta y_{W_{cs}} + y))^2) \ . \tag{4.10}$$

Wie die zugehörige graphische Darstellung in Abbildung 4.4 verdeutlicht, wächst die Funktion $f(\sigma, x)$ mit zunehmendem Argument x immer langsamer und nähert sich asymptotisch dem Wert eins von unten an, ohne ihn jemals zu überschreiten. Der Steuerparameter σ für die Ausprägung der Limitierung muss dabei empirisch ermittelt und an `handleOutlier()` übergeben werden. Dabei besteht eine Abhängigkeit von der eingesetzten Ähnlichkeitsfunktion und den numerischen Beträgen der Farbwerte in den Bildern.

Zusammengenommen kapseln die `CLMSimilarityMeasure`-Traits alle relevanten Details einer beliebigen Metrik für die Korrespondenzanalyse, weswegen sich eine Umsetzung der Matcherschnittstelle nicht „by design“ auf eine konkrete Distanzfunktion festzulegen braucht. Die gewonnene Flexibilität geht dabei mit einem reduzierten Wartungsaufwand für die nun möglichen generischen `CLocalMatcher`-Derivate einher.

Bei einer Untersuchung der Funktionswerte der gängigen Ähnlichkeitskriterien stellt man fest, dass diese nicht nur für die SAD- und SSD-Rechenvorschriften aus einem ganzzahligen oder reellwertigen Datum bestehen und sich daher direkt auf die Basistypen der C++-Programmiersprache abbilden lassen. Zusammen mit der Bildposition der besten gefundenen Korrespondenz kann der numerische Übereinstimmungsgrad folglich in einem einheitlichen Template-Speicherverbund `CLMResult` untergebracht werden. Um daneben die Qualität der Zuordnungen bewerten zu können, enthalten die Objekte der öffentlichen Klasse außerdem den minimal und maximal möglichen Distanzwert bezogen auf die vom Matcher verwendete Muster- bzw. Suchfenstergröße. Effektiv werden diese Angaben durch die Multiplikation der Rechteckabmessungen, wie sie der `CLMArea`-Typ liefert, mit den theoretischen Extrema der verwendeten `CLMSimilarityMeasure`-Instanz berechnet.

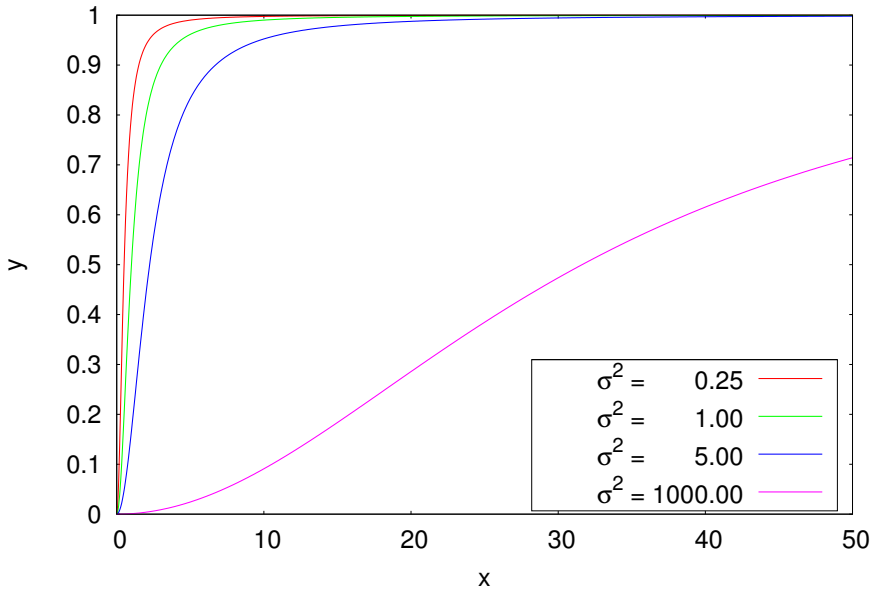


Abbildung 4.4.: Verlauf der Geman-McClure-Funktion für verschiedene Steuerparameter σ

In der Konsequenz ist die übergeordnete `CLocalMatcher`-Vereinbarung für die lokalen Matchingalgorithmen im Stande, das auf diese Weise gruppierte Zuordnungsresultat mit einer standardisierten Routine namens `getBestMatch()` anzubieten. Der Weg über implementierungsspezifische Abfragefunktionen, die das Wissen um den konkret verwendeten Matcherdatentyp voraussetzen und vor ihrem Aufruf eine explizite Typumwandlung der Basisklassenreferenz in selbigen zur Laufzeit bedingen, ist somit nicht zwingend erforderlich. Er bleibt für die Bereitstellung von zusätzlichen Informationen aber dennoch offen.

Der Ergebnistyp `CLMResult` wie auch die anderen Hilfsklassen `CLMConfig`, `CLMArea` und `CLMSimilarityMeasure` bilden zusammen die Grundlage für die konkrete Umsetzung der gemeinsamen Schnittstelle für die lokalen Matchingschritte. Diese erfolgt getrennt für die pixel- und subpixelgenaue Korrespondenzanalyse mit den Vorlagenklassen `COnePassMatcherStd` und `COnePassMatcherPrecise`, welche die Korrelationsberechnung mit einer Iteration (one pass) über jedes Muster- und Suchfensterpaar durchführen. Neben den Argumenten von `CLocalMatcher` müssen die beiden Varianten bei der Instanziierung mit einem geeigneten `CColorConverter`-Farbraumkonvertierungstyp und die präzise Ausführung zusätzlich mit dem Filterindex für die Subpixelinterpolation versehen werden. Um die Zahl der Templateparameter etwas zu verringern, fixieren die Klassen `CSADMatcherStd`/`CSSDMatcherStd` bzw. `CSADMatcherPrecise`/`CSSDMatcherPrecise` den Auflösungstyp mit Integer- respektive doppelt genauen Gleitkommazahlen sowie die verwendete Abstandsfunktion und machen letztere zudem im Namen explizit.

In der prinzipiellen Funktionsweise unterscheiden sich `COnePassMatcherStd` und `COnePassMatcherPrecise` nur in wenigen Details, welche jedoch die Separation der Quelltexte rechtfertigen. Nach der Konfiguration der Matchingparameter kann jeweils die Zuordnungssuche über die argumentfreie `run()`-Funktion gestartet werden. Die Korrespondenzanalyse für das selektierte Musterfenster besteht unabhängig von der gewünschten Ergebnispräzision aus insgesamt vier Teilschritten, die partiell der zeitlichen Optimierung dienen. In chronologischer Reihenfolge sind dies

- die Berechnung der Fenstergrenzen,
- die Initialisierung des Muster- und Suchpuffers,
- die Bestimmung der besten Zuordnung sowie
- die Umrechnung und Rückgabe der Korrespondenz .

Die Berechnung der Fenstergrenzen übernimmt wie beschrieben eine Instanz der Klasse `CLMArea`, welche unter Berücksichtigung des Randproblems die Einstellungswerte in handhabbare Fenstergrenzen umsetzt. Anstatt diese Koordinaten mit der dargelegten Schleifenkonstruktion direkt für die Zuordnungsberechnung einzusetzen, werden die miteinander zu vergleichenden Bildpunkte des Musterareals und *aller* Suchfenster lokal zwischengespeichert. Hierfür kommen zwei eindimensionale Felder, der Musterpuffer und der kombinierte Suchpuffer, zum Einsatz. Beide werden dynamisch mit Hilfe der Konfigurationsangaben angelegt und nur dann realloziert, wenn eine Änderung der Ausschnittsgrößen für die Korrelation oder der Suchbereichabmessungen mit einem erhöhten Speicherplatzbedarf verbunden ist.

Das Kalkül hinter dem Puffermechanismus ist es, die große Anzahl an sich teilweise wiederholenden Zugriffen auf die in Bearbeitung befindlichen Stereobilder über die verwendete pixelorientierte `CImage`-Schnittstelle zu reduzieren. Die beschleunigende Wirkung entfaltet sich dabei insbesondere dann, wenn die auslagernden Speicherklassen `CGridImage` und `CFastGridImage` die Daten bereitstellen oder die relativ langsamen Subpixeloperationen für hohe Zuordnungsauflösungen zum Einsatz kommen. Das für die Matchingalgorithmen prädestinierte blockorientierte Verfahren wird wegen der benötigten angepassten Routinen für jeden Pixelkomponententyp in der DaTool-Applikation derzeit nicht eingesetzt, obgleich es eine höhere Ausführungsgeschwindigkeit verspricht und die Pufferung zumindest im Fall der ganzzahligen Algorithmen überflüssig macht.

Für die Initialisierung der Zwischenspeicher bedienen sich die Matcherimplementierungen einer tranchierten Variante der in Listing 4.1 aufgezeigten Iterationsmethode über die gewählten Bildausschnitte. Der Muster- und der Suchpuffer werden mit separaten Schleifenkonstruktionen gemäß den vorgenommenen Einstellungen entweder mit dem dort definierten Farbkanal oder dem Ergebnis der gewichteten Kombination aller Farbkanäle belegt, wobei für die Adressierung der Arrays eine angepasste Variante der Linearisierungsformel 4.1 Anwendung findet. Die Umwandlung mehrerer Bildpunktkomponenten in Intensitätswerte – die einzige Art von Daten, die von den `COnePassMatcher`-Klassen verarbeitet werden kann – erfolgt mit einer lokal angelegten Instanz des als Vorlagenargument übergebenen Konvertertyps. Sie ist mit den vorhandenen `CColorConverter`-Spezialisierungen auf RGB-Pixel beschränkt. Weil die Wandlerklasse selbst ein Template in Bezug auf die Art der Pixelkomponenten darstellt, funktioniert die Konvertierung auch mit den interpolierenden Subpixelroutinen für die präzisen Matcher anstelle der elementaren `CImage`-Leseoperationen in der Standardversion.

Weil der Suchpuffer zur Vermeidung von unnötigen wiederholten Zugriffen auf die gleichen (Sub-)Pixel die Bilddaten aller Suchfenster im aktuellen Suchbereich simultan in einem einzigen diskreten Array bereitstellt, entsteht eine künstlich eingeführte Abhängigkeit zwischen den horizontalen bzw. vertikalen Such- und Korrelationsschrittweiten. Jeweils getrennt wählbar für beide Raumrichtungen definieren erstere dabei die minimalen Verschiebungsdifferenzen zwischen zwei Suchfenstern im Suchbereich und somit auch

die Anzahl der Bildausschnitte, die im aktuellen Matchingschritt mit dem Musterareal verglichen werden sollen. Letztere drücken dagegen den Abstand und die Menge der gegenüberzustellenden Bildpunkte *innerhalb* des Muster- und *eines* gleichgroßen Suchareals aus. Beide Angaben müssen aufgrund der Tatsache, dass die Intensitätswertpuffer für den Vergleich zwischen den Stereobildausschnitten exklusiv über Integerindizes angesprochen werden können, ganzzahlige Vielfache voneinander sein. Die Wahl, welcher Wert hierbei als numerisch kleinere Basis fungieren soll, hängt davon ab, ob man an der genauen Bestimmung des Ähnlichkeitsgrades (Suchschrittweite entspricht multipler Korrelationsschrittweite) oder der Zuordnungsposition (Korrelationsschrittweite entspricht vielfacher Suchschrittweite) interessiert ist. Da in dieser Arbeit primär die Lage der gefundenen Korrespondenzen relevant ist, werden die horizontale und vertikale Korrelationsschrittweite als Produkt der feineren Suchschrittweiten und den Faktoren `m_searchStepRefinementX` respektive `m_searchStepRefinementY` des übergebenen `CLMConfig`-Konfigurationsobjekts dargestellt.

Mit den initialisierten Zwischenspeichern, die die Randproblematik wie auch die unterschiedlichen Schrittweiten maskieren, kann jetzt die eigentliche Bestimmung der besten Zuordnung erfolgen. Die beiden `COnePassMatcher`-Klassen der DaTool-Software iterieren hierfür über den gesamten Suchpuffer, welcher zur Simplifikation der Schleifenrümpfe wieder in seiner ursprünglich zweidimensionalen Form betrachtet wird. In jedem Wiederholungsschritt, der letztlich einer bestimmten Verschiebung des Suchfensters entspricht, wird mit einem weiteren Schleifendoppel die Übereinstimmung mit dem Musterpuffer berechnet. Das Ähnlichkeitskriterium realisiert die `computeMetric()`-Routine der mit dem eingangs erwähnten numerischen Vorlagenparameter identifizierten und zuvor lokal instanziierten `CLMSimilarityMeasure`-Traitklasse. Die Funktion erhält das aktuelle Muster- und Suchfensterpixel als Argumente, welche mit einer an die Ausschnittsgrößen adaptierten Version der Linearisierungsformel 4.1 aus den insgesamt vier Zählervariablen respektive Pufferpositionen extrahiert werden. Danach durchläuft das Resultat die Ausreißerbehandlung mit der Geman-McClure-Funktion via `handleOutlier()` und wird anschließend zur Summe der bislang ermittelten Distanzwerte hinzuaddiert.

Wenn der Vergleich *aller* Pixel des aktuellen Suchfensters mit denen der Mustersektion abgeschlossen ist, entscheidet der Matchingalgorithmus anhand der aggregierten pixelweisen Unterschiede, ob der soeben analysierte Suchbildausschnitt der Vorlage stärker ähnelt als alle bislang untersuchten Kandidaten, die im Suchpuffer enthalten sind. Die hierfür erforderliche Ordnungsrelation stellt das instanziierte `CLMSimilarityMeasure`-Ähnlichkeitskriterium mit der `isMoreSimilar()`-Funktion bereit. Für die Betrags- bzw. Quadratsummen SAD respektive SSD der untersuchten Intensitätswerte, die es zu minimieren gilt, kommt der gewöhnliche <-Operator zur Anwendung.

Anhand eines Quelltextauszuges von `run()` aus der `COnePassMatcherPrecise`-Klasse verdeutlicht Listing 4.2, wie die Verschiebung des Musterfensters über den Suchbereich im DaTool-Prototyp umgesetzt ist. Man beachte, dass die Kopfzeilen der Schleifen und die Rücktransformation auf die eindimensionalen Pufferpositionen infolge der ganzzahligen Indizierung der lokalen Felder ohne zeitraubende Gleitkommaberechnungen auskommen.

In der noch verbleibenden vierten Ausführungsphase des lokalen Matchingschritts müssen die zweidimensionalen Koordinaten der gefundenen besten Zuordnung, die sich auf die Menge aller Suchfenster im Suchpuffer beziehen, in eine absolute Bildposition umgerechnet werden. Dies geschieht durch die Multiplikation mit den Suchschrittweiten für

```
// Legende der Variablen:
//
// m_patternCache_p - Musterpuffer
// m_searchCache_p  - kombinierter Suchpuffer (Suchbereich mit allen Suchfenstern)
//
// sx, sy           - Zähler für die Iteration über den Suchbereich im Suchpuffer
// maxSx, maxSy     - Abmessungen des Suchbereichs im Suchpuffer
// scDimX, scDimY   - Abmessungen des zweidimensional aufgefassten Suchpuffers
//
// cx, cy           - Korrelationszähler
// pcDimX, pcDimY   - Abmessungen des zweidimensional aufgefassten Musterpuffers
//                    (enstpr. Muster-/Suchfenstergröße)
//
// sPix, pPix       - miteinander zu vergleichende Intensitätswerte
//
// m_config_p       - Matchereinstellungen (CLMConfig)
// m_measure        - Abstandsfunktion (CLMSimilarityMeasure)
// similarity       - aufsummierte Distanzwerte für das Muster-/Suchfensterpaar
// result           - beste gefundene Zuordnung (CLMResult)
//
for (int32_t sy=0; sy<maxSy; ++sy) {
    for (int32_t sx=0; sx<maxSx; ++sx) {

        double similarity=0;

        for (int32_t cy=0; cy<pcDimY; ++cy) {
            for (int32_t cx=0; cx<pcDimX; ++cx) {

                // get pattern/searchwindow pixel from cache
                double pPix=m_patternCache_p[pcDimX*cy+cx];
                double sPix=m_searchCache_p[
                    scDimX*(sy+cy*m_config_p->m_searchStepRefinementY)+
                    sx+cx*m_config_p->m_searchStepRefinementX
                ];

                // sum up the similarity value
                similarity+=m_measure.handleOutlier(
                    m_measure.computeMetric(pPix, sPix)
                );

            } // cx
        } // cy

        // compare with the best match and update
        if (m_measure.isMoreSimilar(similarity, result.m_similarity)) {
            result.m_similarity=similarity;
            result.m_x=sx;
            result.m_y=sy;
        }

    } // sx
} // sy
```

Listing 4.2: Verschiebung des Musterfensters über den Suchbereich in der Klasse `COnePassMatcherPrecise`

beide Raumrichtungen (ausgedrückt als Quotienten der Korrelationsschrittweiten und der Verfeinerungsfaktoren) und einer anschließenden Translation in das Suchbereichszentrum. Die Lage des finalen korrespondierenden Punktes wird zusammen mit dem numerischen Übereinstimmungsgrad und den Extrema zur Qualitätsbewertung in einer Instanz des Ergebnistyps `CLMResult` gespeichert, die alsdann über die allen lokal operierenden Matcherimplementierungen gemeine Schnittstelle `CLocalMatcher` zugänglich ist.

4.6.1.3. Ein- und Ausgabe

Die Ein- und Ausgabe der Stereobilder geht mit dem Problem einher, verschiedene Dateiformate einlesen bzw. schreiben zu können und dabei zwischen dem extern vorliegenden Bildpunkttyp und dem der (teilweise) im RAM befindlichen `CImage`-Instanz zu übersetzen. Wie in der Konzeption dargelegt, begegnet die DaTool-Software dieser Schwierigkeit, indem die Pixelkonvertierung getrennt von der Formatanalyse erfolgt. Dadurch wird der Implementierungsaufwand für Erweiterungen und bei Wartungsarbeiten minimiert, weil der existierende Code von den erforderlichen Modifikationen nicht oder nur minimal betroffen ist. Abbildung 4.5 veranschaulicht das Zusammenspiel der Ein- und Ausgabemodule für die Bilddaten als statisches Diagramm in der Unified Modelling Language (UML) [53].

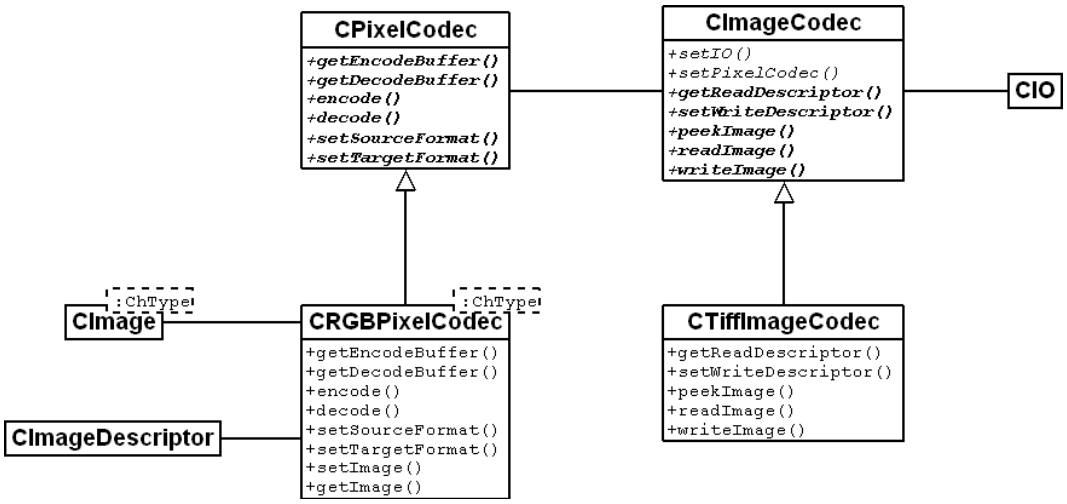


Abbildung 4.5.: UML-Diagramm des Ein- und Ausgabesystems für die Bilddaten

Der Datentransfer vom Massen- in der Hauptspeicher und umgekehrt wird mit insgesamt vier dedizierten Klassen bewerkstelligt. Die Analyse und Erstellung der Bilddateien übernehmen die Bildkodierer und -dekodierer (Bildcodecs) unter der gemeinsamen Schnittstellen `CImageCodec`, die derzeit alleinig durch den objektorientierten Typ `CTiffImageCodec` für das Tagged Image File Format (TIFF) [54] umgesetzt wird. Wegen der unzureichenden Anpassungsmöglichkeiten bei der Speicherverwaltung greift die Klasse dabei nicht auf externe Verarbeitungsbibliotheken eines Drittherstellers wie die de facto-Referenz LibTIFF [55] zurück, sondern stellt eine vollständige Eigenentwicklung dar. In der vorliegenden Implementierung unterstützt `CTiffImageCodec` eine Untermenge der sogenannten Baseline-Spezifikation. Dies umfasst im Besonderen unkomprimierte RGB- und Graustufenbilder mit acht oder sechzehn Bit pro Farbkomponente in der Intel-Bytereihenfolge (II-Bildtyp, geringstwertiges Byte an kleinster Adresse), die in mehrere Segmente (strips) aufgeteilt sein

dürfen bzw. in dieser Form zeilenweise geschrieben werden. Daneben werden auch doppelt genaue Gleitkommabitmaps mit 64 Bit pro Kanal (z.B. die Disparitätskarten) auf nicht standardisierte Weise transferiert und können folglich exklusiv mit der DaTool-Software geöffnet werden[13]. Die Ein- und Ausgabe (`readImage()`, `writeImage()`) der Bilddaten erfolgt geradlinig mit Hilfe einer übergebenen `CIO`-Instanz des gemeinsam genutzten E/A-Subsystems, wobei sich der für die systemnahen Operationen benötigte konsekutive Ziel- respektive Quelldatenpuffer nicht in `CTiffImageCodec` befindet. Der Speicherbereich, welcher den aktuell zu transferierenden rechteckigen Bitmapausschnitt beinhaltet, wird von der Klasse vielmehr über externe Pixelcodecs bezogen.

Die Pixelcodecs, welche über den objektorientierten Datentyp `CPixelCodec` nach außen repräsentiert werden, haben die Aufgabe, zwischen dem internen und externen Bildpunktformat im RAM und auf dem Massenspeicher zu konvertieren. Für die genannten RGB- und Graustufenauflösungen einschließlich der atomaren Gleitkommapixel zeichnet die `CRGBPixelCodec`-Implementierung verantwortlich, die alle auftretenden Umwandlungskombinationen beherrscht. Sie ermittelt die richtige Transformationsvorschrift anhand der übergebenen `CImage`-Bildinstanz, die gleichzeitig das Dekodierziel beim Lesen und die Enkodierquelle beim Schreiben darstellt, sowie zwei Metadatenobjekten vom Typ `CImageDescriptor` für beide Transfermodi, die die Daten auf dem persistenten Medium charakterisieren. Der Deskriptor für den Eingabevorgang muss dabei freilich mit den (Kopf-)Informationen in der Bilddatei korrespondieren, welche über die `peekImage()`- und `getReadDescriptor()`-Funktionen der Interpreterschnittstelle dediziert ausgelesen werden können. Die Metadaten für die Ausgabe können selbst über `setWriteDescriptor()` belegt werden, sollten jedoch die Einschränkungen des Pixel- wie auch des Bildcodecs wiederspiegeln. Ungültige oder inkompatible Angaben weist `CRGBPixelCodec` in beiden Fällen zurück.

Wenn alle Voraussetzungen erfüllt sind, kann ein beliebiger `CImageCodec`-Interpreter mittels der `getDecodeBuffer()`- bzw. `getEncodeBuffer()`-Funktion den vom E/A-Subsystem benötigten konsekutiven Datenpuffer anfordern, wobei er die Eckpunktkoordinaten des zu übertragenden rechteckigen Bildausschnitts als Argumente angibt. Nach dem erfolgreichen Aufruf erhält man abhängig von der Art der Konvertierung, der Größe des Bitmapbereichs und dem Typ der Speicherklasse einen direkten oder indirekten Verweis auf die Bilddaten. Im direkten Fall, der auch als *raw mode* bezeichnet wird, greift der Bildcodec ohne Umwege mit den blockbasierten Zugriffsfunktionen auf die Rohdaten der `CImage`-Instanz zu. Dies ist lediglich für die triviale Pixelumwandlung möglich (Quellpixelformat=Zielpixelformat) und setzt voraus, dass die Bildinstanz das angeforderte Rechteckareal in einem Stück bereitstellen kann – eine Prämisse, die nur bei `CRamImage` garantiert gegeben ist. Im indirekten Fall, dem *copy mode*, wird die Speicheranforderung dagegen mit einem in der Pixelcodecklasse angelegten lokalen Bildpunktfeld befriedigt. Für die Übertragung der darin abgelegten Daten in die Bildinstanz beim Lesen und die Initialisierung der intern gespeicherten 2D-Informationen vor dem Schreiben werden jeweils explizite Kopieroperationen benötigt (daher der Name des Modus), was vor allem in Verbindung mit den Farbwertumwandlungen Rechenzeit kostet.

Gestartet werden die Konvertierung und der Transfer der gepufferten Pixel in den Hauptspeicher, indem der Bildcodec die `decode()`-Funktion aufruft. Mit dem `encode()`-Un-

[13] Die standardkonforme Unterstützung von doppelt genauen Gleitkommabildern im TIF-Format ist zwar prinzipiell möglich, jedoch in der Umsetzung erheblich komplizierter als bei den verbreiteten Farbauflösungen. Abgesehen davon sind Softwareprodukte, die derartige Daten nativ verarbeiten können, kaum verfügbar, weswegen der gewählte proprietäre Ansatz insgesamt vertretbar ist.

terprogramm gelangen umgekehrt die intern abgelegten Bilddaten nach einer eventuellen Farbraumtransformation und Auflösungsanpassung blockweise auf den Massenspeicher. Die Umwandlung und zahlenmäßige Anpassung der Pixelkomponenten realisiert die `CRGBPixelCodec`-Klasse dabei über `CColorConverter`-Objekte, die mit dem Quell- und Zielfarbraum als Vorlagenargumente instanziiert werden. Zur Reduktion respektive Aufweitung des maximal möglichen Dynamikumfangs findet danach eine Skalierung der ganzzahligen Bildpunktelemente mit den festen Faktoren $\frac{1}{256}$ (16 Bit $\rightarrow$ 8 Bit) bzw. 256 (8 Bit $\rightarrow$ 16 Bit) statt. Die Gleitkommawerte bleiben dagegen in Ermangelung einer definierten maximalen Spannweite unverändert. Wenn die Pixelkomponenten den Wertebereich des internen respektive externen Zieldatentyps aufweisen, werden sie abschließend mit einer expliziten Typumwandlung in selbigen überführt. In der direkten Betriebsart werden beide Routinen `decode()` und `encode()` nicht benötigt. Hier übernimmt das E/A-Subsystem aufgrund der fehlenden Zwischenschicht unmittelbar die Kontrolle des Datentransfers.

Weil es sich mit dem dargelegten Design mitunter als kompliziert erweist, die TIFF-Bilddaten „mal schnell“ ein- oder auszugeben, vereinigt die Werkzeugklasse `CImageUtility` den Vorgang in Form der Template-Makrooperationen `loadImage()` und `saveImage()`. Diese benötigen lediglich das Ziel- bzw. Quellbild sowie das geöffnete E/A-Objekt als Parameter. Mit den Eingaben werden der Pixel- und Bildcodec automatisch so aufgesetzt, erprobt und für den Datentransfer genutzt, dass im Ergebnis alle Bildpunkte gemäß den Metadaten der übergebenen `CImage`-Speicherstruktur im RAM oder auf dem persistenten Medium vorliegen.

4.6.2. 3D-Modelle

Die Verwaltung für die 3D-Modelle stellt die erforderlichen Speicherstrukturen einschließlich der Zugriffsfunktionen und Ein-/Ausgaberoutinen für die aus den Laserdaten rekonstruierten Szenenelemente zur Verfügung. Aufgrund der abweichenden Eigenschaften kommen hierfür jedoch andere Techniken als bei den Bilddaten zum Einsatz.

4.6.2.1. Speicherung und Datenzugriff

Die Speicherung des Modells erfolgt im Gegensatz zur den Bildinformationen nicht monolithisch, sondern als Liste von Datensätzen des objektorientierten Typs `CData3D`, die in der Klasse `CModel3D` geführt wird. Dadurch lassen sich die Informationen der zusammengesetzten LRF-Scans auch noch nachträglich identifizieren und separat extrahieren. Jeder Datensatz speichert die Raumpunkte, die Ecken der Polygone und die Texturpositionen samt des Ablageortes der zugehörigen Bitmap. Während letzterer dabei durch eine Zeichenkette repräsentiert wird, liegen die drei erstgenannten Angaben wiederum listenartig in dynamischen Feldern des Typs `vector` aus dem `lrg`-Namensraum vor. Sie können in ihrer Länge geändert und mit den bekannten indexbasierten Zugriffsfunktionen `operator[]()` und `at()` wahlfrei modifiziert werden.

In der Ausgangskonfiguration von `lrg::vector` sind die Felder als Instanzen des `vector`-Containertyps der C++-Standardbibliothek ausgeführt, welcher im `std`-Namensraum angesiedelt ist (vgl. abermals [27]) und keine größeren Datenmengen bei geringem Hauptspeicherausbau aufnehmen kann. Daher kann die `lrg::vector`-Klasse über eine Makrodefinition zur Übersetzungszeit instruiert werden, anstelle der dem Compiler beiliegenden Im-

plementierung die `vector`-Klasse der STXXL-Bibliothek aus dem gleichnamigen Namensraum zu verwenden. Diese kann nach Angabe der Autoren in [36] Informationsvolumina im Bereich von dutzenden Tebibyte handhaben, sofern für den parallelen Auslagerungsmechanismus der alternativen Container genügend Kapazität auf dem Massenspeicher bereitsteht. Ein erfolgreich durchgeführter Schreib- und Lesetest mit $2 \cdot 10^9$ Integerzahlen à 64 Bit in einer `stxxl::vector`-Instanz (insgesamt ca. 16 GiB) stützt diese Aussage in abgeschwächter Form.

Punktkoordinaten Die Liste der Raumpunkte eines Datensatzes enthält Objekte der Klasse `CPoint3D`, in denen die drei Koordinatenwerte als doppelt genaue Gleitkommaattribute vorliegen. Neben dem Abruf der Raumposition bietet der Punkttyp einen Satz von arithmetischen Verknüpfungen und die klassischen Vektoroperationen (Skalarprodukt, Kreuzprodukt, Rotation um die Raumachsen) an, was ihn demzufolge auch für Berechnungen im $\mathbb{R}^3$ qualifiziert. Weil es im Vergleich zu den gerasterten Bilddaten keine native Ordnung auf der zusammengesetzten Punktmenge gibt, welche die darauf aufbauenden Algorithmen ausnutzen können, ist die Reihenfolge der Koordinatentripel in der Punktliste unbestimmt.

Polygone Ausgehend von den Punktkoordinaten definiert das dynamische Feld mit den Beschreibungen der Polygone eines Datensatzes die Oberfläche des gespeicherten 3D-Modells. In der gegenwärtigen Implementierung besteht diese stets aus Dreiecken und ist zudem nicht obligatorisch. Die Gestalt der Triangeln wird aus Speicherplatzgründen nicht direkt durch die insgesamt neun Eckpunktkoordinaten beschrieben, sondern durch drei Indizes in die Raumpunktliste mittelbar ausgedrückt. Eine Klasse namens `CTriangle3D` kapselt die Tripel als 64-Bit-Integerattribute ohne Vorzeichen.

Texturinformationen Im Kontrast zu den räumlichen Informationen enthält die Liste mit den Texturkoordinaten eines Datensatzes die zweidimensionalen Positionsangaben in eine durch die Pfadzeichenkette spezifizierte Bitmapstruktur, mit der jedem indizierten Raumpunkt der Polygonliste optional ein Farbwert zugeordnet werden kann. Die Reihenfolge und Anzahl der Angaben korrespondiert dazu mit den Dreiecksbeschreibungen. Repräsentiert werden die Positionstupel durch die Klasse `CPoint2D`, die keine ganzzahligen, sondern doppelt genaue Gleitkommaattribute beherbergt. Dies ist erforderlich, weil nicht jeder Polygoneckpunkt exakt auf eine Texturposition fällt. Zudem treten die Koordinaten in der Praxis üblicherweise in normalisierter Form im Bereich von [0 ... 1] auf und beziehen sich auf die Lage in Relation zu den Abmessungen des referenzierten Oberflächenbildes.

4.6.2.2. Ein- und Ausgabe

Der Transfer der 3D-Daten vom und zum Massenspeicher wird im Gegensatz zu den Bildinformationen nicht über eine einheitliche Schnittstelle abgewickelt, sondern muss für jedes Quell- bzw. Zieldatenformat individuell umgesetzt werden. Am DLR liegen die mit dem Laserscanner gewonnenen Modelle als VRML'97/2.0-Dateien im Klartext vor [56], weswegen sich die Unterstützung seitens der DaTool-Software auf diese Darstellung konzentriert. Dabei machen die persistenten Beschreibungen nicht von allen Knotentypen des Szenengraphen Gebrauch, den die Sprache zur hierarchischen Gruppierung der 3D-Objekte vorsieht,

sondern bewahren die Raum-, Polygon- und Texturinformationen kongruent zur Speicherung der Modelle in Form einer `IndexedFaceSet`-Struktur auf (siehe Anhang A.2). Auf dieser speziellen Teilmenge operiert das nachfolgend beschriebene Ein-/Ausgabesystem, bei dem aufgrund der technischen Unterschiede zwischen dem Lese- und Schreibvorgang die beiden Transferrichtungen getrennt realisiert sind.

Eingabe Die Eingabe der 3D-Modelle übernimmt ein compilerähnliches Subsystem, weil der naive Implementierungsansatz (feste Sequenz aus Leseprozessen und anschließender Zustandsabfrage) die Klartextdaten in einer bestimmten Reihenfolge erwartet, verschachtelte VRML-Elemente nicht gehandhabt werden können und die Ergänzung um neue Szenengraphknoten mit einem erhöhten Implementierungsaufwand verbunden ist. Anhand der Grammatik der Sprache wird daher zunächst die syntaktisch-semantische Analyse der VRML-Beschreibungen mit einem selbstentwickelten Scanner und Parser durchgeführt. Anschließend übernehmen die funktionalen Knoten des resultierenden Syntaxbaums die Initialisierung der Datensätze eines `CModel3D`-Speicherobjekts als Äquivalent zur Zielcodegenerierung. Die Umsetzung der einzelnen Arbeitsschritte orientiert sich dabei in hohem Maße an der Implementierung des Fehlerinjektionswerkzeugs FIT aus [57].

In Analogie zu den dort getroffenen Aussagen wird das Frontend des VRML-Laders, welches aus dem Scanner und Parser besteht, halbautomatisch mit den Generatorwerkzeugen flex und bison erstellt. Dadurch reduziert sich der Entwicklungsaufwand auf die Ausgestaltung der zugehörigen Definitionsdateien. Die Scannerbeschreibung `VRMLScanner.l` enthält eine Liste von regulären Ausdrücken, welche die Zeichenfolgen der kleinsten sinngebenden Einheiten (Token) im VRML-Quelltext darstellen und ihnen eine Bedeutung (etwa als Schlüsselwort, Literal oder Trennsymbol) verleihen. Die Tokenisierungsregeln sind dabei mit Aktionen verknüpft, die bei der Erkennung einer Sequenz ausgeführt werden und die Art sowie den numerischen bzw. textuellen Wert des lexikalischen Grundelements zurückgeben. Genutzt werden die Tokeninformationen anschließend in den Produktionen der VRML-Grammatik, die in der Definition des Parsers `VRMLParser.y` in der bison-eigenen Notation aufgeführt ist. Abgesehen von der Umwandlung der Erweiterten Backus-Naur-Form (EBNF) in die ursprüngliche Version der Metasprache haben die Regeln aus dem Standarddokument der VRML nur wenige Änderungen erfahren. Wie beim Scanner sind die Produktionen jeweils mit Handlungsvorschriften verknüpft, die während der Ableitung eines Wortes in Gestalt der Modellbeschreibung sukzessive den Syntaxbaum aufbauen.

Die Ableitung erfolgt mit dem in C++-Code übersetzten Frontenddefinitionen bottom-up. Ausgehend von den lexikalischen Token des Scanners, die die Terminalsymbole der Grammatik bilden, werden die rechten Seiten der Produktionen auf die linksseitigen Metasymbole reduziert. Der fertige Parser `VRMLParser.c` nutzt hierfür das LALR-Syntaxanalyseverfahren [58]. Es kommt ohne Backtracking aus und zieht für die Auswahl des nächsten Ableitungsschritts lediglich ein lexikalisches Vorschauelement vom Scanner `VRMLScanner.c` heran. Weil sich einige VRML-Regeln jedoch nur unter Berücksichtigung einer größeren Textmenge oder ihres Kontextes eindeutig identifizieren lassen, treten hierbei mitunter Fehlentscheidungen auf. Aus diesem Grund verwendet der Parser zusätzlich den in [57] beschriebenen Auflösungsmechanismus der Scanner-Parser-Rückkopplung (semantic parsing) und erweitert den LALR-Ansatz um die Strategie von Versuch und Irrtum zu einem generalized LR- oder GLR-Verfahren.

Während die Syntaxanalyse abläuft, wird in jedem Reduktionsschritt die der Produktion zugeordnete Aktion ausgeführt. Diese besteht zumeist darin, einen spezifischen Syntaxbaumknoten als Derivat der Klasse `CSyntaxTreeNode` dynamisch zu erzeugen und in die bislang aufgebaute hierarchische Struktur zu integrieren. Allen Baumelementen ist dabei gemein, dass sie über die vorhandene `run()`-Funktion gleichzeitig für die „Zielcodeerstellung“ verantwortlich sind und somit in der Gesamtheit das Backend bilden. Diesem Umstand ist die englische Bezeichnung als *handler* im Quelltext geschuldet. In der VRML-Implementierung generieren die funktionalen Knoten keine Maschinenbefehle, sondern belegen die Datensätze eines Speicherobjekts mit den Raumpunktkoordinaten, Polygonbeschreibungen und Texturpositionen des persistent vorliegenden 3D-Modells. Initiieren lässt sich die Synthesephase durch die Ausführung von `run()` auf dem Wurzelknoten. Der Funktionsaufruf wird dann rekursiv durch die Syntaxbaumebenen nach dem Postorder-Schema (Kindknoten vor den Vaterknoten) propagiert. Beim Besuch eines Knotens trägt dieser dann entsprechend seiner Position, die direkt die Rolle in der VRML-Grammatik wiederspiegelt, in unterschiedlichem Maß zur Komplettierung der Datensatzwerte bei. So speichern die Blätter des Baums beispielsweise Listen von numerischen Tokenwerten (`CXListHandler`), die dann auf höherer Ebene (von Knoten des Typs `CCoordinateHandler`) als Raumpunkte interpretiert, auf Plausibilität geprüft und schließlich in die Modellinstanz eingetragen werden.

Zur Vereinfachung des Ladevorgangs sind sowohl die Analyse- als auch die Synthesephase in der Klasse `CVRMLLoader` zusammengefasst. Diese versteckt die Funktionen des generierten Scanners und Parsers und übernimmt die Behandlung von syntaktischen und semantischen Fehlern. Abbruch- und Statusmeldungen werden dabei über eine Instanz des objektorientierten Datentyps `CVRMLProgress` in Textform mit Zeilennummer zugänglich gemacht, die bei der Erzeugung des Moduls zusätzlich zur Quelle der VRML-Beschreibung und dem obligatorischen Speicherobjekt übergeben werden kann.

Ausgabe Weniger komplex als die Eingabe der Modelle gestaltet sich die Ausgabe der im Hauptspeicher befindlichen 3D-Informationen. Die Datensätze eines übergebenen Speicherobjekts werden mit Hilfe der Klasse `CVRMLWriter` und dem E/A-Subsystem der DaTool-Software als VRML-Beschreibung auf dem Massenspeicher abgelegt. Hierbei findet der speziell auf Textausgaben zugeschnittene Datentyp `CTextIO` Anwendung, welcher im Kontrast zu den anderen `CIO`-Subklassen die Zeilen mit den betriebssystemabhängigen EOL-Symbolen (end of line) terminiert. Entsprechend der Syntax des `IndexedFaceSet`-Sprachelements erfolgt der Transfer der Punktkoordinaten, Dreiecksindizes und Texturpositionen auf das nichtflüchtige Medium nacheinander in drei Zahlengruppen. Diese erfahren wie die zwischendurch eingefügten VRML-Feldnamen und Trennzeichen der besseren Lesbarkeit wegen eine hierarchische Einrückung. Dabei stellt die Klasse `CIndent`, welche den gemeinsam genutzten Komponenten zugeordnet ist, die Konsistenz der Einzüge sicher. Sie verwaltet die Art und Anzahl der führenden Zeichen pro Textzeile nach dem Stapelprinzip und gibt diese aus. Analog zum Ladevorgang der Modelle verwaltet ein dedizierter Datentyp `CVRMLWriterProgress` den Ausgabefortschritt, der aufgrund der vorab bekannten Informationsmenge im Unterschied zum Parser prozentual quantifiziert wird.

4.6.3. Konfigurationen

Das integrierte Matchingverfahren und die Modellverbesserung weisen zahlreiche Kenngrößen auf, was die Implementierung einer graphischen Eingabemaske erschwert und die ergonomische interaktive Bedienung beeinträchtigt. Deshalb erfolgt die Übergabe der Kenngrößen für die genannten funktionalen Algorithmen wie auch die Erstellung der Ausführungsprotokolle mit statistischen Informationen für die Evaluierung über spezielle Datensätze, die Konfigurationen. Sie werden mit Hilfe der `CConfiguration`-Schnittstelle verwaltet und durch abgeleitete Klassen des objektorientierten Datentyps `CConfigIO` vom Massenspeicher eingelesen bzw. persistent auf diesem abgelegt.

4.6.3.1. Speicherung und Datenzugriff

Die Speicherung der einzelnen Konfigurationselemente für die funktionalen Algorithmen und der Zugriff auf die Daten beruhen entsprechend der Konzeption auf einem hierarchischen Organisationsprinzip. In einer n-ären Baumstruktur werden zusammenhängende Einstellungen als Bruderknoten eines (gegebenenfalls künstlich eingeführten) logisch übergeordneten Vaterknotens gruppiert, wobei die disjunkten Pfade zu jeder Baumkomponente die eindeutige Adressierung gewährleisten.

Im DaTool-Prototyp manifestiert sich die hierarchische Gliederung der einzelnen Konfigurationselemente in der `CConfiguration`-Schnittstelle, über die sie lesend oder schreibend angesprochen werden können. Die Basisklasse definiert hierfür ein Duo von rein virtuellen Funktionen, die durch Ableitung implementiert werden müssen. Mit der `getItem()`-Routine erhält man dasjenige Datum als Instanz der Speicherklasse `CConfigItem`, welches unter der als Zeichenkette übergebenen Baumposition vorliegt. Per Konvention verwendet DaTool für die Adressierung hierbei Pfadangaben im Unix-Stil, bei denen die Knotennamen auf dem Weg zum gesuchten Blatt durch Schrägstriche abgetrennt sind. So bezeichnet etwa `/Configuration/Matcher/Algorithm/HybridStd/metric` die Distanzfunktion für den ganzzahlig arbeitenden lokalen Matchingschritt bei der modellgestützten Korrespondenzanalyse.

Die Verwendung einer textuellen Darstellung zur Knotenlokalisation ist zwar technisch nicht vollkommen, weil die Zeichenfolgen letztlich als wahlfreie Iteratoren missbraucht werden, aber einfach umzusetzen und praxistauglich. Im Gegensatz zu einer Vereinbarung, die Zeiger auf spezielle objektorientierte Knotentypen entgegen nimmt und somit die klassische verkettete Umsetzung der Baumstruktur impliziert, schränkt `CConfiguration` die konkrete Implementierung der Hierarchie weniger stark ein. Dadurch eignet sich die Schnittstelle auch für alternative Baumrepräsentationen über konsekutive Felder oder dynamische Listen.

Das Einfügen von Einstellungswerten in die gedachte Baumstruktur gestaltet sich analog zur `getItem()`-Funktion über die namentlich verwandte `setItem()`-Routine. Sie erhält neben der obligatorischen Pfadangabe die aufzunehmende Konfigurationsinformation als `CConfigItem`-Objekt. Sollte dabei der als Speicherziel definierte Baumknoten noch nicht existieren, so wird er zusammen mit den noch nicht vorhandenen Vorgängern automatisch erzeugt.

Wie die Beschreibung zeigt, liegen die eigentlichen Algorithmuseinstellungen in der n-ären Hierarchie stets in gekapseltem Zustand vor. Die `CConfigItem`-Klasse ermöglicht dabei prinzipiell die Speicherung verschiedener Arten von Daten, wobei sich der konkrete Typ mit Hilfe einer internen Aufzählung festlegen lässt. In der derzeitigen Form beschränkt sich die Unterstützung auf Zeichenketten, bei denen es sich entweder um die Konfigurationsinformationen selbst oder erläuternde Kommentare handeln kann. Bei Bedarf muss diejenige Funktion, welche mittels `getItem()` oder `setItem()` lesend bzw. schreibend auf die Baumblätter zugreift, die Angaben kontextbezogen in eine andere Darstellung konvertieren (z.B. durch die Umwandlung der abgelegten Zeichen in einen numerischen Wert) bzw. umgekehrt selbige zunächst in eine Zeichenkette überführen.

Als Schnittstelle, die selbst keine Definition der deklarierten Routinen vornimmt, bedarf die `CConfiguration`-Klasse einer Implementierung, mit welcher die n-äre Hierarchie technisch konkretisiert wird. Im DaTool-Prototyp übernimmt diese Aufgabe der objektorientierte `CXMLConfiguration`-Datentyp, welcher sich der Bezeichnung nach den Konzepten der Extensible Markup Language (XML) zur Speicherung strukturierter Daten bedient. Im Zusammenspiel mit der freien TinyXML-Bibliothek baut die Klasse anhand der Pfadparameter der polymorphen `setItem()`-Funktion intern ein Document Object Model (DOM) auf. Dabei werden die Konfigurationsdaten konventionell in einer durch Zeiger verketteten Baumstruktur aus Knoten abgelegt, welche aus Instanzen der C++-Klassen von TinyXML bestehen. Mit der in `CXMLConfiguration` verfügbaren `getItem()`-Funktion kann anschließend in der Hierarchie navigiert werden. Die Traversierung erfolgt dabei mit schrittweisem „Durchhangeln“ durch die Kindknoten des DOM innerhalb einer Schleifenkonstruktion.

4.6.3.2. Ein- und Ausgabe

Die Ein- und Ausgabe der Konfigurationsinformationen vom bzw. auf den Massenspeicher ist von der hierarchisch angelegten internen Datenhaltung abgekoppelt und über die Schnittstelle `CConfigIO` zugänglich. Diese muss passend zur jeweiligen Baumstruktur implementiert werden und stellt mit `readConfiguration()` und `writeConfiguration()` Funktionen zum Lesen und Schreiben der Algorithmuseinstellungen und Ausführungsprotokolle in ihrer Gesamtheit bereit.

Für das DOM von `CXMLConfiguration` werden die vorhandenen Routinen der TinyXML-Bibliothek in der zugehörigen `CXMLConfigIO`-Klasse für den Datentransfer genutzt. Da XML von sich aus ein hierarchisches Austauschformat im Klartext definiert, ist bei der Ein- wie auch der Ausgabe der Baumelemente keine nach- bzw. vorgelagerte Linearisierung notwendig. Die zu speichernden atomaren Informationen befinden sich in ihrer nativen Form als Zeichenkette in den Attributen der XML-Start- oder Leerelement-Tags, wobei die Marken den Pfad zum jeweiligen Datum beschreiben. Sie lassen sich mit einem beliebigen Texteditor erstellen oder modifizieren. Die mit der `setItem()`-Funktion angegebenen Kommentare werden mit der üblichen Syntax `<!-- ... -->` zwischen den Start- und Endtags entsprechend ihrer Position in der Hierarchie platziert.

Prinzipiell können die funktionalen Algorithmen beim Speichern der Konfigurationen eine eigene Tagstruktur festlegen und müssen den Inhalt des eingelesenen n-ären Baums vor der Extraktion der Einstellungswerte überprüfen. Die DaTool-Software vereinheitlicht den Aufbau einiger XML-Dokumenttypen auf der Eingabeseite aus Interoperabilitätsgründen, so dass die gespeicherten Informationen zwischen dem integrierten Matcher und dem Mo-

dellverbesserungsplugin entweder ohne Änderungen (Kameraeinstellungen) oder nach dem manuellen Übertragen der gemeinsam genutzten XML-Blöcke ausgetauscht werden können. Anhang A.3 führt die Struktur aller vom Prototyp akzeptierten XML-Dateien mit erläuternden Kommentaren ebenso wie die spezifischen Ausführungsprotokolle exemplarisch auf.

4.6.4. Datenobjekte

Die einheitliche Speicherung der Bilder, 3D-Modelle und Konfigurationseinstellungen in der DaTool-Objektliste und der Austausch mit den funktionalen Algorithmen wird durch die grundverschiedenen Schnittstellen der drei heterogenen Datentypen stark erschwert. Weil im Hinblick auf die Erweiterbarkeit um neue Arten von Informationen ein getrennter Ablage- und Übergabemechanismus indiskutabel ist – jede neue Klasse von Daten bräuchte ihre eigene Containerinstanz – wird stattdessen wie beim Stückguttransport im internationalen Warenverkehr mit ISO-Großraumbehältern eine künstliche Homogenisierung vorgenommen. Dies geschieht durch Mantelklassen, die die Objekte eines konkreten Typs uniform „verpacken“. Um die originale Repräsentation zurückzuerhalten, ist eine explizite Typumwandlung zur Laufzeit anhand der mit dem Mantel abgelegten Typinformationen notwendig.

In der Implementierung des Prototyps fungiert die `CObject`-Klasse als Schnittstelle für die Umhüllung. Sie stellt die gekapselten Instanzen der Speicherstrukturen in Form eines unbestimmten Zeigers (void*) zur Verfügung. Daneben beinhaltet `CObject` die Versionsinformationen (als `CVersion`-Instanz, siehe 4.4), den Ablageort auf dem externen Medium, die Bedeutung und den Ein-/Ausgabemodus des eingeschlossenen Datums. Letzterer spezifiziert, ob die gespeicherten Informationen lesend oder schreibend genutzt werden sollen, woraufhin die funktionalen Algorithmen Modifikationen verweigern können.

Weil in der Praxis die Instanzen der `CObject`-Klasse die einzigen sind, welche einen Verweis auf das ursprünglich erzeugte und eingelagerte Speicherobjekt besitzen, sind sie allein für dessen Zerstörung nach dem Gebrauch verantwortlich. Die neutrale void*-Abstraktion macht den Aufruf des passenden Destruktors jedoch unmöglich. Dies gilt zumindest dann, wenn man hierbei ohne vorherige explizite Typumwandlung auskommen möchte, die zu einem Skalierungsproblem in der Bereinigungsfunktion führen kann. Aus diesem Grund hält die Mantelschnittstelle die Bilder, Modelle und Konfigurationen nicht selbst als Zeigerattribut vor, sondern delegiert diese Aufgabe an abgeleitete Klassen, derer es für jeden Speichertyp eine geben muss. Für die drei Datenarten der DaTool-Software sind dies die Konkretisierungen `CImageObject`, `CModel3DObject` und `CConfigurationObject`, wobei `CImageObject` als Vorlagenklasse universell für alle Arten von Pixelkomponenten ausgeführt ist. Die Derivate erzeugen und speichern das Mantelobjekt in der jeweiligen typisierten Ursprungsform und können es daher in ihrem Destruktor polymorph zerstören.

Ganz auf die von `CObject` abgeleiteten Klassen für jede Speicherstruktur lassen sich die Änderungen am Quelltext beim Hinzufügen neuer Informationsarten dann doch nicht begrenzen. Bei der Typprüfung in den Algorithmen, die die „verpackten“ Instanzen lesend oder schreibend bearbeiten, wird eine zur Übersetzungszeit bekannte konstante Größe als Vergleichswert benötigt. Die Versionsinformationen für die genannten Konkretisierungen sind daher als statische Attribute in dem Modul `CObjectDefs` abgelegt und somit direkt über den Klassennamen zugänglich. Bei der Einführung einer weiteren Speicherklasse muss

`CObjectDefs` mit dem zugehörigen Versionseintrag vervollständigt und die darin enthaltene Information bei einer Typabfrage von der neuen Wrapperklasse zurückgeliefert werden.

Mit der einheitlichen Mantelschnittstelle ist es nun möglich, einen homogenen C++-Standardcontainer für die Liste der in DaTool verfügbaren Objekte einzusetzen. Die Klasse `CObjectList` implementiert diesen mit einem dynamischen Feld (`std::vector`), das den wahlfreien Zugriff erlaubt. Weil genau eine programmweite Instanz benötigt wird und diese ohne die permanente Mitführung zugänglich sein soll, befindet sich die Liste als Attribut eingebettet in der Singletonklasse `CObjectSet` (vgl. wiederum [44]).

4.6.5. Algorithmusobjekte

Ebenso wie die Liste der Datenobjekte in der DaTool-Software ein Einzelstück ist, werden auch die verfügbaren funktionalen Algorithmen in einer einzigen globalen Instanz der Klasse `CAlgorithmSet` mit Hilfe eines dynamischen Feldes abgelegt. Die Elemente bestehen hierbei aus den in `CAlgorithmDLL` gruppierten Einsprungpunkten für die Konstruktion und Destruktion der Objekte für die einzelnen Berechnungsvorschriften, welche als Plugins in Form von dynamisch gebundenen Bibliotheken (dynamic link libraries, DLLs) vorliegen. Geladen werden die funktionalen Module während der Erstellung des Singletons und gleichsam bei dessen Zerstörung aus dem Speicher wieder entfernt. Die für diese betriebssystemspezifische Aufgabe benötigten Routinen sind Bestandteil der gemeinsam genutzten Komponenten.

4.7. Funktionale Algorithmen

Alle Berechnungen auf den Datenobjekten werden im DaTool-Prototyp durch die funktionalen Algorithmen ausgeführt. Die einzelnen Module bilden dabei das Konzept der Unterprogramme aus den höheren Programmiersprachen auf der Anwendungsebene nach. Im Gegensatz zur starren Implementierung durch eine Routine mit fester Signatur definieren sie selbst die Art und Anzahl der Eingabeparameter und Rückgabewerte, die dann mit Hilfe der Laufzeittypinformationen validiert und ausgewertet werden. In der Konsequenz gestaltet sich die Schnittstelle nach außen hin unabhängig von der konkreten Handlungsvorschrift, was den Vorteil der einheitlichen Nutzung mit sich bringt, die sich in DaTool bis zur graphischen Benutzeroberfläche erstreckt.

Vernachlässigt man die uniforme externe Repräsentation mit dem festgeschriebenen Austauschmechanismus für die zu verarbeitenden Informationen und die Ergebnisse der Berechnungen, so gibt es für die Algorithmen in Bezug auf ihren Aufbau und die Umsetzung prinzipiell keine Beschränkungen. Das im Rahmen dieser Arbeit verwendete Design (Abbildung 4.6) ist für alle realisierten Module vom Kern her dennoch identisch, weil es als Nebenbedingung die Erweiterbarkeit um andere Interaktionsformen mit dem Anwender berücksichtigen und die interne Datenhaltung zentralisieren soll.

Die gemeinsame Schnittstelle aller funktionalen Algorithmen `CAlgorithm` stellt Funktionen zur Abfrage der formalen Parameter und des Typs der Handlungsvorschrift bereit. Die erwarteten Eingaben werden dabei über eine Liste von `CFormalParameter`-Instanzen charakterisiert, die kongruent zur Benutzeroberfläche (vgl. 4.5) die Art des Datenobjekts, den

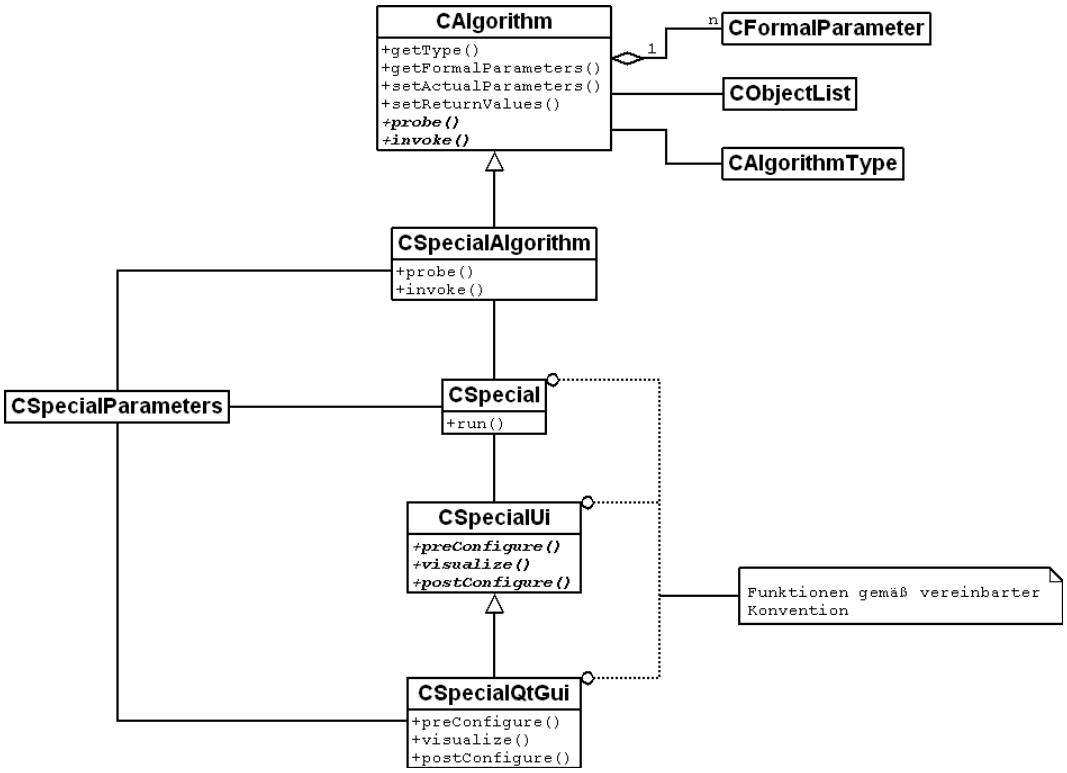


Abbildung 4.6.: Design der funktionalen Algorithmen als UML-Klassendiagramm

Modus, die Vielfachheit und eine textuelle Beschreibung speichern. Der Typ setzt sich aus der Versionsinformation als `CVersion`-Objekt und der Aufgabenkategorie (Eingabe, Ausgabe, Manipulation, Visualisierung) zusammen. Er dient der übergeordneten Anwendung zur eindeutigen Identifikation wie auch als Hinweis auf die Anwendbarkeit des Algorithmus im momentanen Bearbeitungskontext – will der Benutzer lediglich Daten laden, kann die Applikation die Betrachtungsmodule ausblenden.

Die Spezifikation der tatsächlichen Argumente und des Ablageortes der Ergebniswerte mit den Funktionen `setActualParameters()` bzw. `setReturnValues()` erfolgt bei den DaTool-Algorithmen über je eine Liste von `CObject`-Mantelinstanzen, wobei die letztgenannte initial leer sein darf. Vor der Ausführung der Berechnungen mittels `invoke()` können (und sollten) die Eingaben durch die `probe()`-Routine validiert werden. Den funktionalen Code für diese beiden rein virtuellen Operationen stellen dabei die von `CAlgorithm` abgeleiteten objektorientierten Module bereit, die im Diagramm exemplarisch den Namen `CSpecialAlgorithm` tragen. Während die Überprüfung der Parameter auf Korrektheit direkt in der Klasse erfolgt und die Argumente dabei implementierungsspezifisch mit oder ohne das umgebende Mantelobjekt in die algorithmusinterne Datenhaltung `CSpecialParameters` übertragen werden, ist die `invoke()`-Routine in einer getrennten Ausführungsschicht `CSpecial` realisiert. Diese Klasse lässt sich unabhängig vom `CAlgorithm`- und `CSpecialAlgorithm`-Oberbau verwenden und eignet sich demnach auch für den Einsatz außerhalb der DaTool-Software.

Je nach der konkreten Umsetzung können die `CSpecial`-Klassen vor, während und nach der Verarbeitung der Informationen aus dem zentralen Datenlager `CSpecialParameters`

auf unterschiedliche Art und Weise mit dem Benutzer interagieren. So fragen beispielsweise die Ein-/Ausgabemodule für die 3D-Modelle den Lese- bzw. Speicherort für den Datentransfer ab, bevor sie aktiv werden, und zeigen während des Datentransfers den Verlaufsfortschritt an. Um sich in der DaTool-Software nicht in die Abhängigkeit von einer bestimmten Benutzeroberfläche zu begeben und neue Bedienungsmöglichkeiten aufwandsarm integrieren zu können, befinden sich die Operationen für die Kommunikation mit dem Anwender nicht in der `CSpecial`-Implementierung selbst. Stattdessen werden sie über eine separate Schnittstelle `CSpecialUI` angesprochen und diese etwa mit einer GUI-Bibliothek im Sinne des funktionalen Algorithmus umgesetzt. Die drei Funktionen `preConfigure()`, `visualize()` und `postConfigure()` repräsentieren die eingangs erwähnten Zeitpunkte, zu denen der Benutzer traditionell Eingaben vornimmt und Rückmeldungen erhält. Im DaTool-Prototyp nutzen die Handlungsvorschriften die Dialoge der Qt-Bibliothek, welche auch für das Hauptfenster des Rahmenprogramms Verwendung findet. Die konkrete Ausgestaltung der Bedienoberfläche spiegelt sich in der Bezeichnung der `CSpecialUI`-Derivate wieder, die für die stellvertretende Algorithmusimplementierung `CSpecialQtGUI` lautet.

Die fertig implementierten Module werden, wie bereits im Zusammenhang mit der Verwaltung der Algorithmusobjekte kurz erwähnt, als dynamisch gebundene Bibliotheken übersetzt. Dafür ist eine einheitliche Schnittstelle, wie sie die Klasse `CAlgorithm` darstellt, zwingend erforderlich. Der erzeugte relozierbare Binärcode kann danach als Plugin zur Laufzeit geladen und die konkrete `CSpecialAlgorithm`-Instanz über zwei spezielle C-Fabrikfunktionen mit fester Bezeichnung, die als Einsprungpunkte dienen, erzeugt (`create()`) und nach Gebrauch zerstört (`destroy()`) werden [59]. Die Vorteile des Plugin-Konzepts bestehen in der einfachen Aktualisierung der existierenden Module und der Ergänzung um neue Algorithmen, ohne dass dafür Änderungen am restlichen DaTool-Quellcode erforderlich sind. Daneben lassen sich anwendungsspezifische Binärdistributionen durch das Hinzufügen oder Weglassen von bestimmten dynamisch gebundenen Komponenten erstellen. Im Folgenden sollen die vorhandenen Plugins einzeln besprochen werden. Die Schwerpunkte liegen dabei auf dem integrierten Matchingalgorithmus und dem Modul zur Modellverbesserung.

4.7.1. Ein- und Ausgabe der Bilddaten, 3D-Modelle und Konfigurationen

Die Ein- und Ausgabeplugins für die Bilddaten, 3D-Modelle und Konfigurationen sind durch die Klassen mit den Präfixen `CImageIn...`, `CImageOut...`, `CVRMLIn...`, `CVRMLOut...`, `CConfigIn...` und `CConfigOut...` realisiert. Sie verwenden zum Transfer der Daten vom und zum Massenspeicher jeweils die mit den zugehörigen Speicherklassen eingeführten Subsysteme, welche dem Benutzer über einen graphischen Auswahldialog zur Abfrage des Quell- bzw. Zielorts der persistenten Informationen zugänglich gemacht werden. Die lesenden Vertreter sind in der Lage, mehrere selektierte Datenobjekte gleichzeitig in die `CImage`-, `CModel3D`- und `CConfiguration`-Strukturen im Hauptspeicher zu übertragen. Dabei werden die dreidimensionalen Modelle nachträglich an das Koordinatensystem des DaTool-Prototyps angepasst.

Gemäß der Sprachspezifikation (Sektion 4.4.5 – Standard units and coordinate system) liegen die VRML-Daten in einem kartesischen rechtshändigen dreidimensionalen Koordinatensystem vor. DaTool verwendet jedoch ein linkshändiges Positionsschema (Abbildung 4.7), weil das Werkzeug für die Erstellung von kalibrierten Referenzaufnahmen der später behandelten Testszene dieser Konvention folgt und somit die unmittelbare Vergleichbar-

keit der dort verwendeten externen Kameraparameter gegeben ist. Bei der Übersetzung zwischen den Koordinatensystemen erfahren die dreidimensionalen Raumpunkte (und nur diese) einen Vorzeichenwechsel in der Z-Komponente, wenn sie vom Massenspeicher geladen oder wieder dorthin zurück geschrieben werden. Man beachte, dass die Spiegelung an der X-Y-Ebene erst durch die VRML-Algorithmusplugins von DaTool erfolgt und nicht bereits beim Analysevorgang mit dem VRML-Parser. Die dortigen Ein- und Ausgabefunktionen für die 3D-Modelle bleiben somit für andere Applikationen und Koordinatensysteme ohne Anpassung einsetzbar.

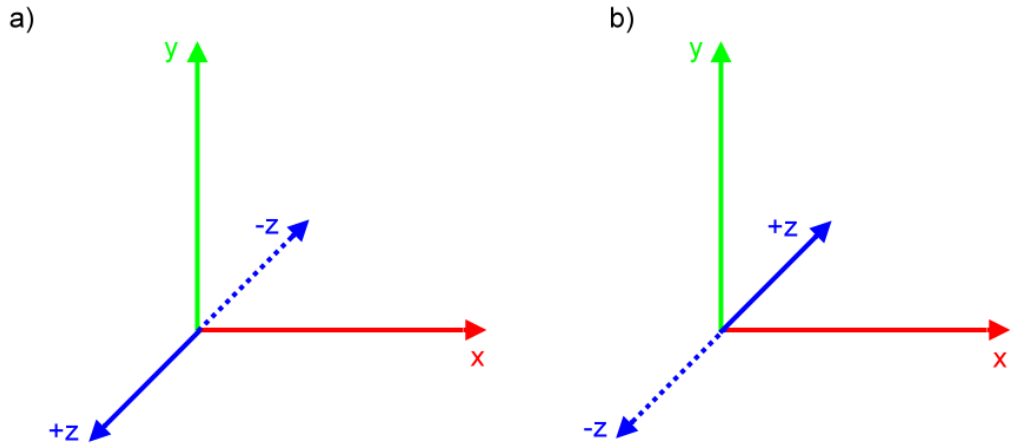


Abbildung 4.7.: (a) Rechtshändiges VRML-Koordinatensystem (b) linkshändiges DaTool-Koordinatensystem

4.7.2. Visualisierung

Die DaTool-Software umfasst zwei Plugins aus der Visualisierungskategorie zur interaktiven Darstellung der Bild- und Modellinformationen auf dem Bildschirm. Sie dienen primär dazu, die Auswirkungen der manipulierenden Algorithmen zu veranschaulichen, um auftretende Fehlfunktionen identifizieren und anschließend beheben zu können.

4.7.2.1. Bilddaten

Das Modul zur Betrachtung der Bilddaten besteht aus den `CImageView...`-Klassen und akzeptiert beim Aufruf Datenobjekte mit einer Auflösung von acht und sechzehn Bit pro Farbkanal. Zur Darstellung nutzt es die 2D-Graphikroutinen der Qt-Bibliothek. Während des Visualisierungsvorgangs wird das Bild des als Eingabeparameter übergebenen Datenobjekts pixelweise durchlaufen. Unter Berücksichtigung des einstellbaren Skalierungsfaktors sowie der vom Benutzer gewählten horizontalen und vertikalen Laufleistenposition zeichnen die Routinen `drawImage8()` und `drawImage16()` des benutzerdefinierten GUI-Elements `CImageViewWidget` den gewünschten Ausschnitt. Weil durch die Größenänderung im Allgemeinen gebrochene Bildpunktpositionen auftreten, werden zum Setzen der Farbwerte auf dem Bildschirm die subpixelgenauen Zugriffsfunktionen aus Abschnitt 4.6.1.2 aufgerufen. Dabei ist der Dreiecksfilter zur bilinearen Interpolation der Zwischenwerte fest vorgegeben.

4.7.2.2. 3D-Modelle

Das Plugin zur Visualisierung der 3D-Modelle mit den objektorientierten Datentypen der `CModel3DView...`-Familie basiert auf der OpenGL-Schnittstelle, die bei modernen Graphikkarten zum großen Teil in Hardware implementiert ist. Dadurch wird die Darstellung der Szene beschleunigt. OpenGL umfasst die benötigten Funktionen zur Definition von dreidimensionalen Primitiven in einem virtuellen Raum und zur Projektion desselbigen auf den zweidimensionalen Bildschirm.

Die Platzierung der Grundelemente, zu denen die 3D-Punkte und die Polygone (als Indizes in die Punktmenge) zählen, wie auch die Oberflächengestaltung der Szene mit den Texturbildern und -positionsangaben bewerkstelligen austauschbare Unterkomponenten des Algorithmusmoduls. Diese als Renderer bezeichneten Bausteine spezialisieren die gemeinsame Schnittstelle `CGLRenderer`. Zur Darstellung der Modelle dient der abgeleitete objektorientierte Typ `CGLBasicModelRenderer`. Er reicht die Informationen aus dem `CModel3D`-Parameterobjekt unverändert an die OpenGL-Implementierung durch, weil das von der 3D-Schnittstelle erwartete Datenformat für die Raumpunkte, Polygone und Texturen mit dem der Modellbeschreibung übereinstimmt. Um die Geschwindigkeit der Visualisierung zu steigern, erfolgt die Übergabe der Punktkoordinaten, Polygonindizes und Texturpositionen nicht einzeln, sondern blockweise mit lokalen konsekutiven Feldern über den `glDrawArrays()`-Mechanismus. Weiterhin lädt `CGLBasicModelRenderer` einmalig die Oberflächenbitmaps der einzelnen Datensätze der Speicherstruktur anhand des dort festgehaltenen Ablageortes und macht sie der Umsetzung der 3D-Schnittstelle bekannt.

Sind alle Informationen an das OpenGL-System übergeben, beginnt der Anzeigedialog des Visualisierungsplugins mit der zweidimensionalen Darstellung der 3D-Szene auf dem Bildschirm. Hierzu wird zunächst über die 3D-Schnittstelle eine virtuelle planare Kamera mit festen internen Projektionseigenschaften aufgesetzt, die durch Mausbewegungen interaktiv im Raum plaziert und rotiert werden kann. Dadurch ist es möglich, Details in der abgebildeten Szene zur genaueren Begutachtung gezielt anzusteuern. Die Berechnung der neuen externen Orientierung der Kamera, die im unteren Dialogbereich aufgeführt ist, erfolgt dabei mittels der Differenzstrecke, die der Mauszeiger im Vergleich zu seiner vorherigen Position zurückgelegt hat. Im Fall einer Änderung wird der Renderer erneut aufgerufen und das angezeigte Modellbild aktualisiert.

4.7.3. Integriertes Matching

Das Kernelement dieser Arbeit in Gestalt des Plugins für den integrierten Matcher führt die Korrespondenzanalyse auf zwei planaren oder zylindrischen Stereobildern durch. Die obligatorischen Aufnahmen werden hierfür zusammen mit dem zugehörigen 3D-Modell, das aus einem oder mehreren Laserscans hervorgeht, den Kameraeinstellungen und einem Konfigurationsobjekt mit den Parametern für den Zuordnungsprozess als Eingabedaten übergeben. Nach der erfolgreichen Ausführung des Plugins erhält man die horizontale und vertikale Disparitätskarte, ein Graustufen-Übersichtsbild, das die Entfernungen vom Projektionszentrum der ersten Stereokamera mit abgestuften Intensitätswerten darstellt, sowie ein Ausführungsprotokoll als Ergebnis.

Von der Funktion her basiert der mit den `CMatcher...`-Klassen umgesetzte Algorithmus auf den lokalen Zuordnungsschritten (Abschnitt 4.6.1.2), deren Suchbereich mit dem

Strahlverfolgungsverfahren aus der Konzeption eine Einschränkung erfährt. Dabei werden die ungeordnet vorliegenden Oberflächenpolygone zur Steigerung der Ausführungsgeschwindigkeit der Objektpunktschätzung im LRF-Modell vorab räumlich sortiert. Weil die mit der Beschleunigung des Matchings befassten Teilkomponenten vergleichsweise umfangreich ausfallen, sind diese in separate objektorientierte Datentypen ausgegliedert.

4.7.3.1. Kameramodelle und Stereogeometrie

Zwei Aspekte, die für die Einschränkung des Suchbereichs bei der Korrespondenzanalyse wie auch die nachgelagerte Rekonstruktion der Objektpunkte relevant sind, bestehen in der Kodierung der Kameramodelle und deren Komposition zum Stereosystem. Der funktionale Algorithmus deklariert erstere über die `CCamera`-Schnittstelle und letzteres über die `CMVG`-Vereinbarung.

Die gemeinsame Schnittstelle zu den Kameramodellen wird gegenwärtig durch Implementierungen für den planaren und zylindrischen Typ mit den Klassen `CPlanarCamera` und `CCylinderCamera` realisiert. Beide stellen Operationen für die Projektion von Objektpunkten auf die Bildebene bzw. den Zylindermantel und umgekehrt bereit. Dazu verwenden sie die im Grundlagenkapitel eingeführten Rechenvorschriften. Die mehrdeutige inverse Abbildung $\mathbb{R}^2 \rightarrow \mathbb{R}^3$ wird hierbei durch die explizite Festlegung der Entfernung vom Projektionszentrum eindeutig gemacht. Im Ergebnis befinden sich die aus den Pixelpositionen ermittelten 3D-Punkte stets auf der räumlich betrachteten Projektionsfläche.

Um die Abbildung durchführen zu können, benötigen die im Quelltext formulierten Kameramodelle Angaben zur inneren und äußeren Orientierung. Die intrinsischen Parameter werden in Instanzen der Klassen `CPlanarIntOrientation` und `CCylinderIntOrientation` für die ebene und zylindrische Kamera abgelegt, die gegenwärtig die gleichen Informationen speichern. In der Notation des Grundlagenabschnitts 2.1.2 gehören hierzu die Brennweite f, der Hauptpunkt $\mathbf{P^{(i)}}$, die Pixelabmessungen d_{px} bzw. d_{py} und die effektive Sensorauflösung (=Bildauflösung) d_x bzw. d_y. Die Tatsache, dass beide Module nicht zu einem einzigen zusammengefasst sind, ist durch die Überlegung motiviert, dass jedes Kameramodell bei Bedarf selektiv um zusätzliche Parameter erweiterbar sein soll. Im Kontrast dazu wird die äußere Orientierung, die unabhängig von der Art der Abbildung ist, einheitlich mit der `CExtOrientation`-Klasse gespeichert. Der objektorientierte Datentyp beschreibt die Lage einer Kamera mit den Rotationswinkeln ω, ϕ und κ um die drei Raumachsen sowie einer Translation $\mathbf{t^{(w)}}$ zum Ursprung des DaTool-internen linkshändigen Koordinatensystems. Die Angaben werden dabei in `CPoint3D`-Attributen zu Tripeln von doppelt genauen Gleitkommazahlen zusammengefasst, wie sie generell für die Speicherung der Modellpunktkoordinaten zum Einsatz kommen.

Basierend auf den `CCamera`-Operationen, welche sich auf eine einzelne Kamera beziehen, implementiert die Klasse `CMVG` (multiple view geometry) zwei Funktionen für die Berechnungen im Stereoverbund. Dies umfasst die punktweise Konstruktion der Epipolarlinie mit der alternativen Methode aus Abschnitt 2.1.3 über die `getEpipolarLinePoint()`-Routine und die näherungsweise Bestimmung des Raumschnittpunktes aus einem Paar von korrespondierenden Bildpunkten via `correspondenceToWorld()` gemäß Sektion 2.1.4.2 . Mit dem C++–Sprachmittel der Ableitung können die vorgegebenen Algorithmen, die vom konkret verwendeten Kameratyp abstrahieren, optional durch andere Implementierungsvarianten ersetzt werden.

4.7.3.2. Objektpunktschätzung

Das in der Konzeption beschriebene Verfahren zur Einschränkung des Suchbereichs für die Korrespondenzanalyse basiert darauf, für jeden Bildpunkt der ersten Kamera eines Stereosystems den Sichtstrahl vom Projektionszentrum aus zu rekonstruieren und den nächstgelegenen Schnittpunkt mit dem Modell zu berechnen. Wenn nach Voraussetzung die beiden Aufnahmegeräte zu der mit Hilfe des Laserscanners gewonnenen dreidimensionalen Szene orientiert sind, die Objektpunkte also jeweils auf das zugehörige photographisch festgehaltene Pixel abbilden, dann markiert die Projektion des Auftreffpunktes des Strahls im zweiten Kamerabild schließlich das Zentrum für die Zuordnungssuche.

Das Hauptproblem bei der skizzierten Strahlverfolgung ist das Auffinden der für die Schnittpunktberechnung heranzuziehenden Oberflächendreiecke im LRF-Modell. Die ungeordnet vorliegenden Punktkoordinaten und folglich auch die über Eckpunktindizes beschriebenen Triangeln müssen vor dem Raytracing räumlich sortiert werden. Das Plugin, das den beschriebenen Matchingalgorithmus umsetzt, untergliedert daher die gesamte Szene mit einer hierarchischen 3D-Baumstruktur, die als Spezialfall den Oktalbaum (Octree) beinhaltet. Dabei wird jedes Polygon sukzessive kleiner werdenden quaderförmigen Boxen zugewiesen (klassifiziert), welche sich analog zu den Bildpixeln durch ganzzahlige Koordinaten ansprechen lassen. Auf diese Weise entsteht wie im Zweidimensionalen eine Ordnung auf dem Modell, die die Nachbarschaftsbeziehungen der 3D-Primitiven berücksichtigt – angrenzende Boxen beherbergen stets räumlich benachbarte Raumpunkte und Polygone.

Das Kalkül hinter der hierarchischen Einteilung ist es, die für die Schnittpunktberechnung in Frage kommenden Modellpolygone über ein binäres Suchverfahren mit logarithmischem Zeitaufwand zu identifizieren. Für realistische Eingangsdaten ist diese Vorgehensweise trotz der vorab einmalig durchzuführenden Klassifikation signifikant effizienter, als wenn pro Bildpunkt jedes Dreieck der Szene auf einen Sichtstrahltreffer untersucht werden muss.

Struktur, Adressierung und Orientierung des 3D-Baums Die dreidimensionale Baumstruktur wird durch die Klasse `C3DTreeTracer` des Plugins für das integrierte Matching implementiert, die auch die Klassifizierung der Polygone und die Strahlverfolgung vornimmt. Sie unterteilt die Szene auf der obersten Ebene in Quader, welche sich wiederum rekursiv aus gleichartigen Teilquadern zusammensetzen. Die Definition der Hierarchie erfolgt manuell über die Anzahl der Unterteilungsstufen (Level), die Menge der Subquader separat für jede Raumdimension sowie die initialen Abmessungen der Boxen (Länge, Breite, Höhe) in Weltlängeneinheiten. Letztere werden dabei mit doppelt genauen Gleitkommazahlen spezifiziert und müssen nach Multiplikation mit der Subquaderzahl der Modellausdehnung in X-, Y- und Z-Richtung entsprechen oder diese übersteigen. Eine automatische Bestimmung der Partitionsparameter im Hinblick auf die schnelle Dreieckssuche ist aufgrund der Komplexität der Thematik (Abhängigkeit der Konfiguration nicht nur von der Modellgröße, sondern von der Ausdehnung der Polygone, deren Raumlage und Verteilung) im DaTool-Prototyp nicht implementiert.

Um im beschriebenen 3D-Baum navigieren und an die eingeordneten Dreiecksinformationen gelangen zu können, benötigt man ein Schema für die Adressierung der einzelnen Quader. In der `C3DTreeTracer`-Klasse werden dazu die Boxen jedes Hierarchielevels als diskreter dreidimensionaler Raum (level box space) aufgefasst, wodurch sie sich bei

der Klassifikation der Polygone, der Strahlverfolgung und anderen Algorithmen intuitiv über ein ganzzahliges XYZ-Koordinatentripel ansprechen lassen. Auch die Speicherung der Dreiecke, die den Quadern zugeordnet sind, profitiert von der Konvention. Nach Anwendung der dreidimensionalen Linearisierungsfunktion (vgl. Formel 4.1)

$$i(\mathbf{B}, l) = \mathbf{B}(z)n_y(l)n_x(l) + \mathbf{B}(y)n_x(l) + \mathbf{B}(x) \ , \tag{4.11}$$

die im Quelltext den bezeichnenden Namen `box2Index()` trägt und eine eindimensionale levelbezogene Boxnummer $i(\mathbf{B}, l)$ aus der Boxraumposition $\mathbf{B}$ ermittelt, können hierfür sequenzielle oder assoziative Datencontainer eingesetzt werden. Die Boxanzahlen pro Raumrichtung $n_x(l)$, $n_y(l)$, $n_z(l)$ leiten sich direkt aus der Baumkonfiguration ab und werden zur weiteren Verwendung in speziellen Leveldeskriptoren des Typs `CLevelDescriptor` der `C3DTreeTracer`-Klasse aufbewahrt. Dies gilt auch für die Quaderabmessungen $d_{boxX}(l)$, $d_{boxY}(l)$ und $d_{boxZ}(l)$ in Weltlängeneinheiten, über die man die Lage $\mathbf{B}_{\mathbf{tree}}(l)$ einer Box in Weltlängeneinheiten aus den diskreten ganzzahligen Raumkoordinaten relativ zum Baumursprung durch

$$\mathbf{B}_{\mathbf{tree}}(l) = \begin{pmatrix} \mathbf{B}(x) \cdot d_{boxX}(l) \\ \mathbf{B}(y) \cdot d_{boxY}(l) \\ \mathbf{B}(z) \cdot d_{boxZ}(l) \end{pmatrix} \tag{4.12}$$

erhält.

Bevor die Modellpolygone in die Baumstruktur einsortiert werden können, muss diese in der Szene verankert und orientiert werden. Damit sich die räumlichen Berechnungen auf der Hierarchie möglichst einfach gestalten, erfolgt die Ausrichtung der Gesamtmenge aller Quader an der achsenparallelen kastenförmigen Umschreibung (axis aligned bounding box, AABB) des 3D-Modells und ist demnach ebenfalls achsenparallel. Die AABB wird vollständig mit zwei Eckpunkten, dem Minimal- und Maximalpunkt, beschrieben, die sich aus den Minima bzw. Maxima aller Polygoneckpunkt*koordinaten* ergeben. Demnach geht die Ermittlung der Extrema mit einem initialen Lauf durch alle Dreiecksindizes der Szenenbeschreibung einher. Die AABB-Eckpunkte sind im `C3DTreeTracer`-Typ abgelegt, wobei der Minimalpunkt fortan als Bezugspunkt (Ursprung) des 3D-Baums fungiert. In der Konsequenz liefert Gleichung 4.12 nach der Translation des Funktionswerts in den Minimalpunkt (Funktion `box2World()`) die finale Lage eines Quaders in Weltkoordinaten, wobei es sich – präzise und anschaulich gesagt – um dessen „linken unteren vorderen" Eckpunkt im linkshändigen DaTool-Koordinatensystem mit Blickrichtung entlang der positiven Z-Achse handelt.

Speicherung der Dreiecke im 3D-Baum Die Speicherung der Dreiecke, die bei der Klassifikation in die Baumstruktur eingeordnet werden, erfolgt für jede Hierarchieebene in den Leveldeskriptoren. Für jede nichtleere Box, der also wenigstens ein Dreieck der Szene zugeordnet ist, existiert ein Eintrag in Form zweier ineinander verschachtelter dynamischer Felder. Diese enthalten die zugehörigen Polygone als Indizes in die Dreiecksliste der an den funktionalen Algorithmus übergebenen 3D-Modellinstanz getrennt für jeden Datensatz.

Um keinen Speicherplatz an leere Baumquader zu verschenken, werden die einzelnen Boxeinträge in der Implementierung der Deskriptoren nicht mit einem dritten dynamischen Feld, sondern in einer Hashtabelle vorgehalten. Das geringfügig schlechtere Laufzeitverhalten dieses assoziativen Containers pro Raytracingvorgang wird dabei in Kauf genommen.

Für umfangreiche Modelle ist dieser Ansatz letztlich alternativlos, weil er die minimal mögliche Informationsmenge speichert und dennoch den wahlfreien Zugriff mit nahezu konstantem Zeitaufwand erlaubt. Als Schlüssel zu den Indizes in die Dreiecksliste des Modells pro Datensatz dienen die linearisierten levelbezogenen Boxkoordinaten. Diese stellen – abgesehen von der obligatorischen Umwandlung in den Indextyp der Hashtabelle – gleichzeitig den Hashwert dar, zumal sie als Integerzahlen über einen nativen <-Vergleichsoperator verfügen.

In Bezug auf die Speicherkomplexität im schlimmsten Fall (Groß-O in der Landau-Symbolik [60]) hat die Wahl des assoziativen Containers anstelle eines unmittelbar indexadressierten dynamischen Feldes keine Auswirkungen. Sie entspricht der Zeitkomplexität der Polygonklassifikation.

Klassifikation der Modellpolygone Die Beschleunigung der Objektpunktschätzung mit dem 3D-Baum setzt voraus, dass die Oberflächenpolygone des LRF-Modells entsprechend ihrer Lage im Raum in die Quader einsortiert werden. Dieser Klassifikationsprozess wird einmalig vor der Strahlverfolgung und der damit verbundenen Suche nach den Dreieckskandidaten für die Schnittpunktberechnung über die `classify()`-Funktion des `C3DTreeTracer`-Datentyps durchgeführt.

Die Einordnung der Polygone erfolgt rekursiv von der obersten zur niedrigsten Baumebene. Zunächst werden dabei alle Level-0-Quader in der Ausgangsgröße durchlaufen und die diskrete Position im Boxraum in Weltkoordinaten umgerechnet. Anschließend findet eine Untersuchung aller Modelldreiecke daraufhin statt, ob sie vollständig im aktuell betrachteten Baumelement liegen oder teilweise in den Quader hineinragen. Den hierfür erforderlichen Dreieck-Box-Schnitttest stellt die Klasse `C3DTreeIntersector` bereit, wobei sie auf den Algorithmus aus [61] zurückgreift. Dieser setzt achsenparallele Boxen voraus, wie sie nach der Baumausrichtung an der Szene vorliegen.

Vor der Übergabe der Eckpunktkoordinaten an die Testroutine werden die Triangeln zentrisch mit dem festen Faktor $1,2$ skaliert. Dadurch lassen sich Fehlzuordnungen vermeiden, bei denen ein Dreieck die aktuell betrachtete Box haarscharf touchiert, wegen numerischer Ungenauigkeiten bei der Berechnung aber dennoch keine Überlappung eintritt. Auf der anderen Seite bewirkt die temporäre Vergrößerung der Polygonfläche, dass Dreiecke häufiger mehreren Boxen zugeordnet werden, was später die Sucheffizienz beeinträchtigt. Dieser Nachteil wiegt allerdings weniger schwer, als wenn der Strahlverfolgungsalgorithmus ein Polygon bei der Schnittpunktberechnung „übersieht“, weil es nicht korrekt klassifiziert ist. Ohnehin ist nicht zu erwarten, dass sich ein irregulär aufgebautes 3D-Modell ohne Mehrfachzuordnungen in eine Komposition aus regulären Quadern einfügt. Die Klassifikation eines Dreiecks in mehrere Baumelemente muss folglich in jedem Fall zulässig sein und bei der Implementierung der Objektpunktschätzung berücksichtigt werden.

Wenn aus der Schnittberechnung wenigstens ein positives Ergebnis hervorgeht, die aktuelle Box des nullten Baumlevels also nicht leer ist, werden die entsprechenden Polygone mit der Datensatznummer und dem Index in die Dreiecksliste des LRF-Modells in der Hashtabelle des Leveldeskriptors abgelegt. Anschließend wird die Prozedur für alle Subquader wiederholt, wobei jedoch nur noch die Dreiecke der Elternboxen Berücksichtigung finden. Durch die rekursive Propagation bis in die unterste Baumebene verringert sich die Zahl der zu überprüfenden Polygone für realistische 3D-Daten recht schnell, zumal die Wahrscheinlichkeit von leeren Subboxen bei sinkender Quadergröße zunimmt.

Die Zeitkomplexität der Polygonklassifikation hängt direkt mit dem Aufbau des 3D-Baums zusammen. Für die Abschätzung sei eine Hierarchie mit l Leveln, n Boxen auf der untersten Ebene, m Baumelementen im initialen Level, einer rekursiven Unterteilung in jeweils m Subquader sowie eine feste und von n linear abhängige Zahl von Modelldreiecken $n_t = k \cdot n$ angenommen ($l, m, n, n_t \in \mathbb{N}_+;\ k \in \mathbb{R}$). Da die Anzahl der untergeordneten Quader für jede Box rekursiv pro Baumebene um m ansteigt, gilt für die Menge der kleinsten Baumelemente $n = m^l$ und äquivalent $l = log_m n$. Während der Einsortierung in die Würfel des aktuellen Levels werden über den gesamten Baum betrachtet alle n_t Dreiecke genau einmal auf die Überlappung mit einer Box untersucht. Insgesamt finden bei l Hierarchieebenen also $n_t \cdot l = n_t \cdot log_m n = k \cdot n \cdot log_m n$ oder $O(n \cdot log\ n)$ Schnittberechnungen statt.

Rekursive Strahlverfolgung Ist die Klassifikation abgeschlossen, beginnt die beschleunigte Objektpunktschätzung mit dem Raytracingverfahren in der `C3DTreeTracer`-Klasse. Die naheliegende Methode (Funktion `traceRayRecursive()`) untersucht hierfür nach der Rekonstruktion des Sichtstrahls vom Projektionszentrum der ersten Kamera durch das aktuelle Pixel in Weltkoordinaten zunächst die Level-0-Boxen auf Strahlschnittpunkte, von denen es jeweils maximal zwei Stück geben kann. Diejenigen Quader, die der Strahl schneidet und die folglich einen Eintrag in der assoziativen Hashtabelle besitzen, werden nach dem euklidischen Abstand des nächstgelegenen Auftreffpunktes zum Projektionszentrum aufsteigend sortiert. Danach überprüft der Algorithmus die Subboxen der geordneten initialen Hierarchieelemente auf Kreuzungen mit der Halbgeraden und führt den beschriebenen Prozess rekursiv weiter bis zur untersten Ebene des 3D-Baums.

Bei den kleinsten Subquadern angekommen, werden schließlich die zugehörigen Polygonlisten aller Modelldatensätze aus dem Leveldeskriptor durchlaufen und die in die Box eingeordneten Dreiecke auf einen Strahlschnittpunkt getestet. Die Kreuzung, die sich in der geringsten euklidischen Entfernung zum ersten Abbildungszentrum befindet, ist der gesuchte Objektpunkt für die Projektion in das zweite Stereobild. Weil die Sortierung der Quader durchgängig sicherstellt, dass keine weiteren Triangeln in einem anderen Baumelement näher zur ersten Kamera platziert sein können, als in demjenigen, wo sich der erste Schnittpunkt zwischen einem Modelldreieck und dem Sichtstrahl befindet, müssen beim aktuellen Raytracinglauf keine weiteren Boxen mehr analysiert werden.

Wie die Beschreibung suggeriert, setzt das rekursive Verfolgungsverfahren zwei Arten von Schnittpunktformeln ein. Mit Hilfe des Box-Strahl-Tests wird die Zahl der zu untersuchenden Dreiecke auf diejenigen Polygone beschränkt, die den Quadern entlang des Sichtstrahls zugeordnet sind. Die Berechnung, ob und wo sich der Strahl mit dem Quader schneidet, übernimmt die von `C3DTreeIntersector` abgeleitete Klasse `CRay3DTreeIntersector`, die auf die geradlinige Lichtausbreitung spezialisiert ist – prinzipiell unterstützt die Objektpunktschätzung auch „krumme“ Ausbreitungsformen. Sie gliedert sich klassischerweise in zwei Phasen. Zunächst wird der Strahl, der sich aus dem Ursprungspunkt und einer Richtungsangabe zusammensetzt, als Gerade interpretiert und mit den sechs Quaderebenen geschnitten. Anschließend findet ein Vergleich der Kreuzungskoordinaten mit denen der Boxecken statt, welche die Grenzen der Quaderseiten definieren. Der numerischen Ungenauigkeit bei der obligatorischen Verwendung von Gleitkommawerten wird hierbei durch den Einsatz der toleranten Relationsoperatoren des Mathematikmoduls der gemeinsam genutzten Komponenten Rechnung getragen. Wenn die maximal mögliche Zahl von zwei (möglicherweise identischen) Schnittpunkten innerhalb der Flächenintervalle liegt, bricht der Vergleichsvorgang ab (early-exit-Strategie). Als Ergebnis erhält die aufrufende Instanz,

etwa der rekursive Raytracingalgorithmus, in jedem Fall die Menge und die Koordinaten derjenigen Kreuzungen zurück, die in Richtung des Sichtstrahls und nicht „hinter“ selbigem liegen.

Die zweite Schnittpunktberechnung verläuft zwischen dem Strahl und einem beliebig im Raum angeordneten klassifizierten Dreieck. Hierfür wird der in [62] aufgeführte Quellcode nachgenutzt. Die an den DaTool-Prototyp adaptierte Implementierung befindet sich wie schon der Box-Strahl-Test in der `CRay3DTreeIntersector`-Klasse.

In Bezug auf die Zeitkomplexität durchläuft der rekursive Raytracingalgorithmus pro Suchdurchlauf unter Beibehaltung der bei der Polygonklassifikation getroffenen Annahmen höchstens m Quader pro Hierarchielevel. Bei l Baumebenen erfordert dies $m \cdot l = m \cdot log_m n$ oder $O(log\ n)$ Box-Strahl-Schnittpunktberechnungen und in jedem Blattelement des 3D-Baums k Dreieck-Strahl-Tests. Für große Quaderzahlen n auf dem untersten Level und gleichmäßig über das Modell verteilte Dreiecke ($k \ll n$) ist die Polygonsuche mit einem Gesamtaufwand $O(log\ n) \ll O(n)$ wesentlich effizienter, als wenn pro Strahl jedes Dreieck naiv auf eine Kreuzung mit der Halbgeraden geprüft wird. Dies gilt auch dann, wenn man bei der zu erwartenden Menge von Raytracingdurchläufen den einmaligen Klassifikationsvorgang (Zeitkomplexität $O(n \cdot log\ n)$) einrechnet.

Inkrementelle Strahlverfolgung Neben der rekursiven Variante implementiert das Plugin für das integrierte Matching in der `C3DTreeTracer`-Klasse eine inkrementelle Raytracingmethode. Die Idee hierbei ist es, die Zahl der für die Berechnung des Objektschnittpunktes in Frage kommenden Oberflächenpolygone dadurch zu begrenzen, dass der kontinuierliche Sichtstrahl ausgehend vom Projektionszentrum der ersten Kamera durch die kleinsten Boxen des 3D-Baums diskret approximiert (quantisiert) wird. Dabei ist es entscheidend, *alle* Quader der tiefsten Hierarchieebene, durch die die Halbgerade verläuft, möglichst *effizient* zu ermitteln. Für den zweidimensionalen Fall ist das Rastern von kontinuierlichen Linien schon vor 45 Jahren elegant durch den Bresenham-Algorithmus gelöst worden [63]. Eine modifizierte und um die dritte Dimension erweiterte Variante dieses Verfahrens kommt daher für die inkrementelle Strahlverfolgung zum Einsatz. Abbildung 4.8 illustriert den Sachverhalt.

Die Diskretisierung von Kurven allgemein wird im DaTool-Plugin für die modellgestützte Korrespondenzanalyse durch die `CQuantizer`-Schnittstelle im Stil eines Vorwärtsiterators repräsentiert und ist für gerade Sichtstrahlen durch die Klasse `CLineQuantizer` umgesetzt. Bei der Initialisierung von Instanzen dieses Typs legt man zunächst den Bezugspunkt des 3D-Baums (Minimalpunkt der axis aligned bounding box der Szene), die Größe der kleinsten Quader, deren Anzahl pro Raumrichtung und den Ursprung sowie die Richtung der zu approximierenden Halbgeraden fest. Aus den Eingaben werden dann die Startquaderposition im Boxraum, der Mittelpunkt des Startquaders in Weltkoordinaten sowie die Positionsinkremente für den Quantisierungsvorgang berechnet. Diese legen fest, welchen Verlauf die dreidimensionale Rasterung nimmt und ergeben sich für jede Raumachse getrennt aus dem Vorzeichen der Strahlrichtung über die Signum-Funktion

$$sgn(x) = \begin{cases} 1, & x > 0 \\ 0, & x = 0 \\ -1, & x < 0 \end{cases} . \tag{4.13}$$

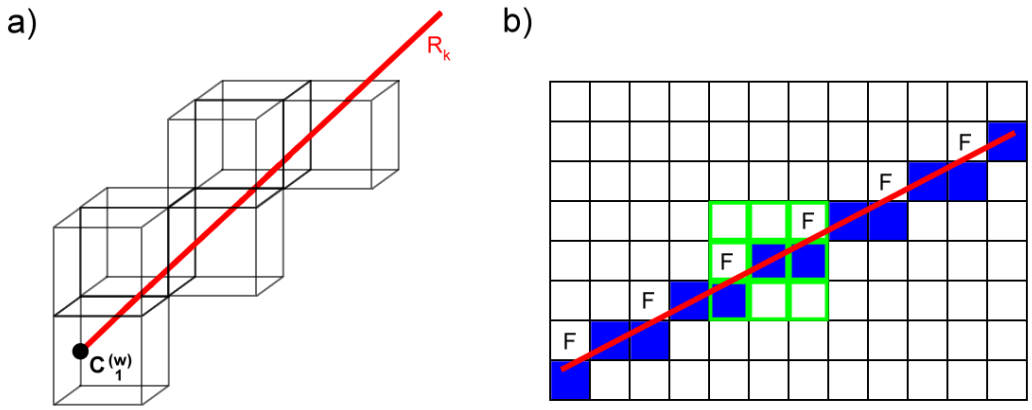


Abbildung 4.8.: (a) Korrekte Annäherung des Sichtstrahls beim inkrementellen Raytracing (b) Zweidimensionale Rasterung (blau) der kontinuierlichen Linie (rot) mit dem Bresenham-Algorithmus basierend auf der Achternachbarschaft (grün). Die mit F markierten Felder werden dabei vom Strahl geschnitten, jedoch erst auf Grundlage der Vierernachbarschaft ohne Diagonalzüge in die diskrete Ergebnismenge aufgenommen.

Mit den genannten Informationen ist die parameterlose `nextPoint()`-Funktion in der Lage, ausgehend von der Startbox die unterste Hierarchieebene entlang des Strahls schrittweise zu durchlaufen. Bei jedem Aufruf des Unterprogramms wird entschieden, welcher der drei benachbarten Quader nach Addition des X-, Y-, bzw. Z-Inkrements auf die aktuelle diskrete Baumposition die Abweichung zur kontinuierlichen Halbgeraden minimiert (Abbildung 4.9). Als Distanzkriterium fungiert der quadrierte euklidische Lotabstand d_i vom Mittelpunkt $\mathbf{M_i^{(w)}}$ des Kandidatenquaders $i \in \{1, 2, 3\}$ in Weltkoordinaten zum Sichtstrahl R_k mit dem Ursprung im Projektionszentrum der ersten Kamera $\mathbf{C_1^{(w)}}$ und der Richtung $\mathbf{r_{kd}^{(w)}}$. Er wird über den doppelt genauen Gleitkommaquotienten

$$d_i^2 = \frac{(\mathbf{r_{kd}^{(w)}} \times (\mathbf{C_1^{(w)}} - \mathbf{M_i^{(w)}}))^2}{\left|\mathbf{r_{kd}^{(w)}}\right|^2} \tag{4.14}$$

ermittelt [64]. Die Selbstmultiplikation im Zähler bzw. Nenner substituiert die normalerweise ausgeführten vergleichsweise langsamen Wurzelberechnungen, ändert aber nicht das Monotonieverhalten der Distanzfunktion und folglich auch nicht das qualitative Ergebnis der Minimalwertbestimmung.

Falls eine Anwärterbox durch die Addition des Nullinkrements identisch mit dem aktuell betrachteten Quader sein sollte, wird der resultierende Strahlabstand d_i^2 zum Boxmittelpunkt künstlich erhöht. Der aufaddierte „Strafbetrag“ garantiert, dass pro `nextPoint()`-Aufruf stets ein neues Baumelement entlang der Halbgeraden resultiert und die Funktion nicht permanent beim aktuellen Quader verharren kann. Der Fall, dass alle Inkremente weder positiv noch negativ sind (R_k mutiert zum richtungslosen Punkt), hat ein undefiniertes Verhalten zur Folge, ist aber praktisch irrelevant. Steht die minimale Abweichung zum Sichtstrahl plus dem eventuellen Penalty schließlich fest, wird die gegenwärtige diskrete Position auf der untersten Baumebene durch die Koordinaten der entsprechenden Nachbarbox ersetzt. Sie lässt sich effizient über die Funktion `getPoint()` auslesen. Den Terminierungszeitpunkt für die inkrementelle Berechnung der Baumquader signalisiert das

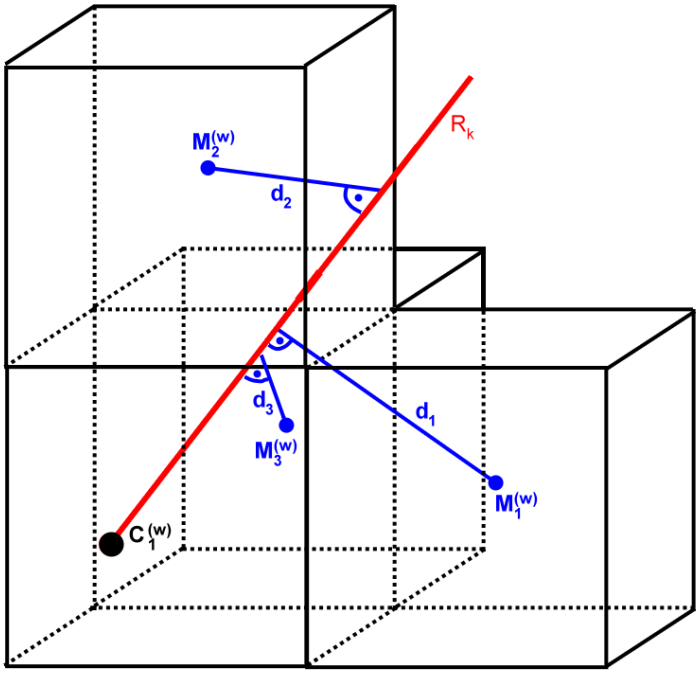


Abbildung 4.9.: Distanzberechnung zur Auswahl der Nachbarbox beim inkrementellen Raytracing

`hasNext()`-Unterprogramm, sobald ein weiterer Aufruf der `nextPoint()`-Routine zu einer Boxposition außerhalb des 3D-Baums führt. Als Grenzwerte dienen hierbei die Boxanzahlen für die drei Raumrichtungen, die bei der Initialisierung des Raytracingverfahrens übergeben werden.

Im Unterschied zum Bresenham-Verfahren, das im Zweidimensionalen prinzipiell jedes der insgesamt acht Nachbarpixel (Achternachbarschaft) als nächstes Rasterbildelement auswählen kann, vollführt der in DaTool eingesetzte Algorithmus keine Diagonalschritte. Er beschränkt sich bei der Bestimmung des nächsten Quaders auf die sechs orthogonalen Nachbarboxen. Dies ist der Notwendigkeit geschuldet, wirklich alle Baumelemente, die vom Sichtstrahl geschnitten werden, zu erfassen und in den Raytracingvorgang einzubeziehen. Andernfalls besteht die Gefahr, dass eine nicht geringe Zahl von Quadern samt den darin klassifizierten Dreiecken bei der Schnittpunktberechnung fälschlicherweise ausgelassen wird. Heruntergebrochen auf die Fläche sind die kritischen Rasterpunkte in Abbildung 4.8(b) exemplarisch mit dem Buchstaben F gekennzeichnet. Abgesehen von der fehlenden schrägen Approximation der Halbgeraden nutzt der 3D-Algorithmus mit dem Quadermittelpunkt einen anderen Bezugspunkt für die Distanzbestimmung zum kontinuierlichen Strahl als Bresenhams Methode und kommt bislang nicht ohne einfache Gleitkommaberechnungen aus.

Mit Hilfe der `CLineQuantizer`-Klasse kann die Objektpunktschätzung ohne einen rekursiven Baumdurchlauf erfolgen. Das `traceRayIncremental()` genannte Unterprogramm des objektorientierten Datentyps `C3DTreeTracer` konfiguriert hierfür den Strahlquantisierer mit den Informationen aus dem Leveldeskriptor der untersten Hierarchieebene des 3D-Baums. Beginnend beim Projektionszentrum der ersten Stereokamera fordert die Routine

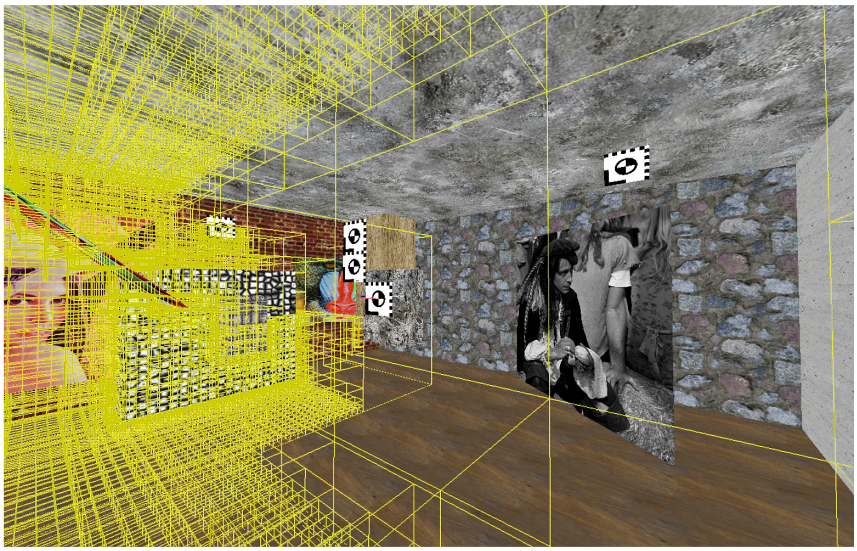
anschließend über `nextPoint()` schrittweise die Quader mit den kleinsten Abmessungen entlang des Strahls an und prüft, ob sie Oberflächenpolygone enthalten. Ist dies der Fall, so werden alle Modelldreiecke der aktuellen Box über den assoziativen Container des Leveldeskriptors extrahiert und auf einen Schnittpunkt mit der Halbgeraden untersucht. Analog zum rekursiven Raytracing kommt hierfür wiederum der in [62] beschriebene Dreieck-Strahl-Schnittpunktalgorithmus zum Einsatz. Sobald der dem Projektionszentrum nächste Auftreffpunkt ermittelt ist, wird dieser für die weitere Verwendung bei der Korrespondenzanalyse zurückgegeben. Andernfalls fährt `traceRayIncremental()` mit der Anforderung von Quadern entlang der Halbgeraden fort, bis die Funktion `hasNextPoint()` die Grenze des 3D-Baums anzeigt.

Für die Bestimmung der Zeitkomplexität des inkrementellen Raytracings unter den Voraussetzungen des Abschnitts über die Polygonklassifikation sind die Boxanzahlen n_x, n_y und n_z des 3D-Baums auf der untersten Ebene in den drei Raumrichtungen maßgeblich. Die maximale Quaderzahl, die für eine gegebene Hierarchiekonfiguration entlang des Sichtstrahls durchlaufen werden kann, entspricht der Menge der kleinsten Baumelemente entlang der Diagonalen der bounding box des Baums. Diese ist wiederum identisch mit dem Maximum von n_x, n_y und n_z, wenn man von einem konstanten Faktor absieht, der kleiner als der Unterteilungsgrad m der Boxen ist und aufgrund der orthogonalen Approximation mit der Sechsernachbarschaft multiplikativ einfließt. Unabhängig vom untergliederten Modell sind pro Raytracinglauf folglich höchstens $O(max(n_x, n_y, n_z))$ Inkrementoperationen erforderlich, für den Spezialfall einer kubischen Hierarchie benötigt man bei n Quadern auf dem untersten Level $O(\sqrt[3]{n})$ `nextPoint()`-Berechnungen. Die Dreieck-Strahl-Schnittpunkttests, die theoretisch auf jeder erreichten Box ausgeführt werden müssen, erhöhen den Gesamtaufwand kongruent zur rekursiven Methode. Diese ist in der Theorie für hinreichend große Werte n mit $O(log\ n)$ Suchschritten in der Hierarchie schneller als der inkrementelle Ansatz.

In der Praxis dürfte die Laufzeit des zweiten Strahlverfolgungsverfahrens stark vom verwendeten 3D-Modell abhängen. Einerseits kommt die Bestimmung des nächsten Quaders entlang der Halbgeraden pro `nextPoint()`-Schritt mit weniger Anweisungen aus als der Box-Strahl-Schnittpunkttest. Andererseits sind vor allem bei den relativ kleinen Quadergrößen auf der untersten Baumebene mehr Inkrementberechnungen als Kreuzungskalkulationen beim rekursiven Hierarchieabstieg zu erwarten.

Visualisierung Die Validierung der Polygonklassifikation wie auch der Strahlverfolgungsalgorithmen ist allein mit den numerischen Angaben (Polygonindizes, Boxraumpositionen, Schnittpunktkoordinaten) nahezu unmöglich. Um die korrekte Funktionsweise sicherzustellen und auftretende Fehler besser einordnen zu können, lässt sich die Objektpunktschätzung durch die Definition des Makros `ENABLE_3DTREE_VISUALIZATION` in der `CMatcher`-Klasse in einer speziellen Betriebsart übersetzen, die den Prozessablauf dreidimensional visualisiert. Hierfür werden die Routinen des Moduls zur Betrachtung der 3D-Modelle (siehe Abschnitt 4.7.2.2) nachgenutzt und um einen weiteren Renderer `CGL3DTreeRenderer` ergänzt. Dieser stellt den Sichtstrahl, die 3D-Baumquader und die Polygone unter Verwendung von OpenGL-Instruktionen dar, was allerdings deutlich auf Kosten der Laufzeit geht. Abbildung 4.10 zeigt exemplarisch die Baumstruktur und den Verlauf der inkrementellen Strahlverfolgung im Fehlerbehebungsmodus der Objektpunktschätzung.

a)



b)

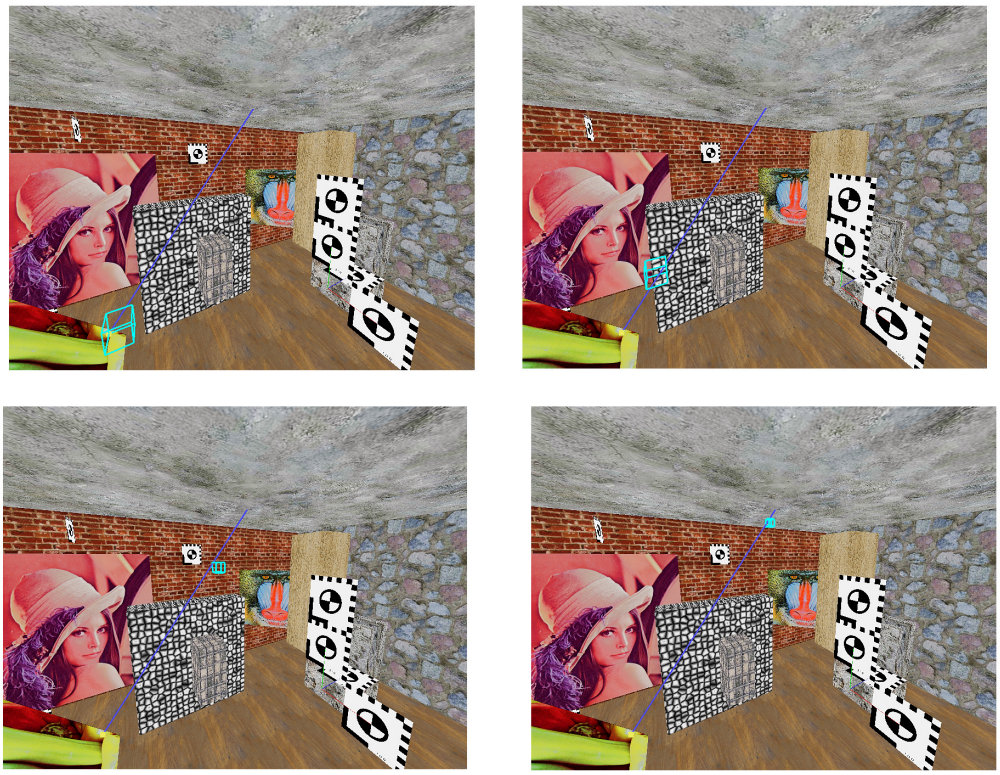


Abbildung 4.10.: (a) Aufbau eines sechsstufigen 3D-Baums mit Würfeln der Abmessungen $8-4-2-1-0,5-0,25$ Längeneinheiten und acht Subboxen pro Level (b) Inkrementelles Raytracing in der Testszene. Die türkis eingefärbten Baumquader werden schrittweise entlang des blauen Sichtstrahls berechnet.

4.7.3.3. Korrespondenzanalyse

Mit der Objektpunktschätzung und den lokalen Matchingschritten stehen die beiden Bausteine für die integrierte Korrespondenzanalyse zur Verfügung. Die `CMatcher`-Klasse des Plugins implementiert das Zusammenspiel der Komponenten, das im Folgenden beschrieben wird.

Funktionsprinzip Die Beschleunigung der Korrespondenzanalyse wird gemäß der Konzeption dadurch erreicht, dass der Suchbereich für die fensterbasierten Zuordnungsverfahren mit Hilfe der im Modell geschätzten Objektpunkte eingeschränkt wird. Ein Aufruf der `run()`-Funktion des `CMatcher`-Datentyps, die nach dem Algorithmusdesign den konkreten Einsprungpunkt für die Übereinstimmungssuche bildet, konstruiert daher zunächst Instanzen der Klassen für die lokalen Matchingschritte und die Kameraobjekte. Anschließend wird die Übereinstimmungssuche über die `modelAssistedMatching()`-Routine initiiert, welche die horizontale und vertikale Disparitätskarte anlegt, den 3D-Baum erzeugt, die Modellpolygone klassifiziert und die einzelnen Teilschritte der modellgestützten Zuordnungsbestimmung koordiniert. Aufgrund der einheitlichen Schnittstelle der Kameras vom Typ `CCamera` genügt hierbei eine Implementierung für die planare wie auch die zylindrische Projektion.

Nach der Initialisierungsphase durchläuft das Unterprogramm alle Pixelpositionen `(x1;y1)` des vom Anwender festgelegten relevanten Musterbildausschnitts (region of interest oder kurz ROI), welcher die Größe der Disparitätskarten bestimmt und standardmäßig mit den Abmessungen aus dem Bilddeskriptor übereinstimmt. In jeder Iteration wird das Tupel mit der inversen Projektionsvorschrift des ersten Kameraobjekts in das Weltkoordinatensystem des LRF-Modells umgerechnet und definiert danach zusammen mit dem Projektionszentrum aus der äußeren Orientierung einen Sichtstrahl für die sich anschließende Objektpunktschätzung. Wenn die mit dem rekursiven oder inkrementellen Raytracingalgorithmus verfolgte Halbgerade hierbei einen Schnittpunkt mit einem Oberflächenpolygon der Szene liefert, so kann `modelAssistedMatching()` dessen Koordinaten zur Suchbereichseinschränkung heranziehen. Andernfalls findet die Korrespondenzanalyse auf herkömmlichem Weg entlang der durch das Stereosystem induzierten Epipolarlinie im Suchbild statt.

Kreuzt der Sichtstrahl der ersten Kamera das 3D-Modell, dann wird der Schnittpunkt mit der Abbildungsvorschrift der zweiten Kamera in das zugehörige Suchbild projiziert. Die resultierenden Bildkoordinaten `(x2;y2)` repräsentieren das Zentrum des Suchbereichs für den lokalen Matchingschritt. Die Zuordnungsbestimmung selbst erfolgt allerdings nicht über einen direkten Aufruf der fensterbasierten Algorithmen, sondern wird mit einem hybriden hierarchischen Verfahren bewerkstelligt. Bevor die abgestufte Matchingprozedur zur Ausführung kommen kann, muss der Bildpunkt an der Position `(x2;y2)` zunächst eine Sichtbarkeitsprüfung positiv bestehen. Bei diesem Test wird durch eine weitere Strahlverfolgung vom Projektionszentrum der zweiten Kamera durch das designierte Suchbereichszentrum in Weltkoordinaten kontrolliert, ob der auf diese Weise gefundene zweite Objektschnittpunkt mit der ursprünglichen Kreuzung des Modells übereinstimmt und kein anderes Szenenelement den Weg versperrt (Abbildung 4.11).

Ist der Musterbildpunkt auch im Suchbild sichtbar und die hybride hierarchische Zuordnungsbestimmung mit dem Musterfenster um den Aufpunkt `(x1;y1)` und dem Suchbereichszentrum `(x2;y2)` daraufhin ausgeführt, wird der resultierende Abstandswert für das

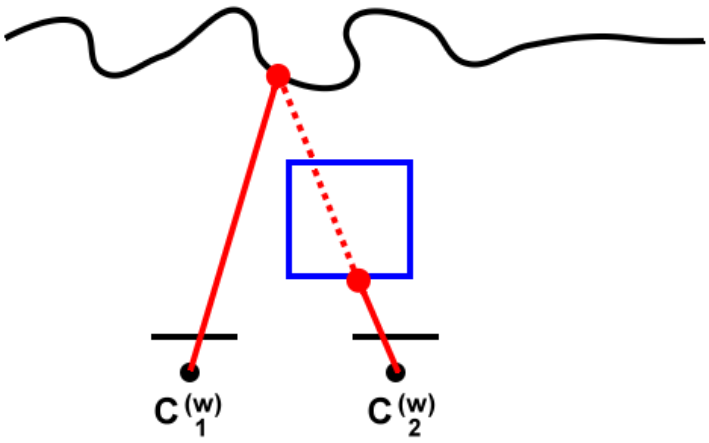


Abbildung 4.11.: Strahlverfolgung von der ersten bzw. zweiten Stereokamera. Bei Verdeckungen (blaues Objekt) treten unterschiedliche Schnittpunkte mit dem Modell auf.

am besten übereinstimmende Suchfenster einer Qualitätsprüfung unterzogen. Dabei darf der Übereinstimmungsgrad zum Musterareal einen benutzerdefinierten relativen Schwellwert nicht verletzen. Wenn eine qualitativ akzeptable Korrespondenz mit der Suchbildposition `(x2';y2')` vorliegt, wird die Differenz zum Musterfenster `(x2';y2')-(x1;y1)` getrennt nach Abszisse und Ordinate in der horizontalen bzw. vertikalen Disparitätskarte vermerkt. Außerdem vervollständigt der euklidische Abstand zwischen den Koordinatentupeln im Nachhinein eine für die schnelle visuelle Begutachtung gedachte Graustufen-Distanzkarte. Wenn das Musterpixel dagegen nicht im Suchbild sichtbar ist und folglich zwei differierende Objektschnittpunkte aus den beiden Raytracingläufen mit der Muster- und Suchbildkamera hervorgehen, wird keine Korrespondenzanalyse durchgeführt.

Wenn bereits der Sichtstrahl der ersten Kamera keinen Schnittpunkt mit einem Oberflächenpolygon des LRF-Modells erzeugt („Löcher“ in der Szene), ist eine Einschränkung des Suchbereichs nicht möglich. In diesem Fall wird versucht, das Musterfenster um die Position `(x1;y1)` entlang der zugehörigen Epipolarlinie im Suchbild wiederzufinden. Hierfür iteriert die von der `modelAssistedMatching()`-Funktion aufgerufene zuständige Subroutine `epilineMatching()` zur Zeitersparnis über eine äquidistante Teilmenge der Horizontalpositionen `ex` des Suchbildes und berechnet mit Hilfe einer lokalen `CMVG`-Instanz die fehlenden Ordinaten `ey` der Kurvenpunkte. Die Schrittweite, mit welcher die waagerechten Bildkoordinaten durchlaufen werden, ergibt sich aus der abgerundeten ganzzahligen Hälfte des Minimums der horizontalen und vertikalen Suchbereichsausdehnung, nimmt jedoch wenigstens den Wert eins an. Sie ist gerade so gewählt, dass bei der hybriden hierarchischen Zuordnungsbestimmung um die vollständigen Positionen `(ex;ey)` entlang der Epipolarlinie keine Lücken entstehen und möglichst wenige redundante Korrelationsberechnungen zwischen dem momentanen Musterbildausschnitt und der Suchfenstermenge des Suchareals stattfinden. Nach der Qualitätsprüfung wird die Differenz `(x2';y2')-(x1;y1)` der Musterbildposition `(x1;y1)` und der gefundenen „epipolaren“ Korrespondenz `(x2';y2')` analog zum modellgestützten Matching in den Disparitätskarten abgelegt. Der euklidische Abstand der Koordinatenpaare ergänzt später wiederum das Graustufen-Übersichtsbild.

Man beachte, dass bei der Zuordnungsbestimmung ohne Objektpunktschätzung entlang der Epipolarlinie *keine* Sichtbarkeitsanalyse stattfindet. Prinzipiell wäre eine inverse Validierung der Korrespondenzen durch einen Rollentausch von Muster- und Suchbild denkbar. Diese Methode verdoppelt allerdings den Zeitbedarf der ohnehin schon langsamen klassischen Übereinstimmungssuche und ist daher nicht implementiert. Verdeckungen in Bezug auf die abgebildeten Modellobjekte können folglich exklusiv über die Qualitätsprüfung der gefundenen Zuordnungen eliminiert werden, da davon auszugehen ist, dass einseitig sichtbare Szenenelemente den Ähnlichkeitsgrad negativ beeinflussen.

Zu Testzwecken lassen sich das modellgestützte wie auch das epipolare Verfahren in der `CMatcher`-Klasse über das Konfigurationsobjekt des Algorithmusplugins getrennt ausschalten (auch gleichzeitig). Ebenso kann die Sichtbarkeitsprüfung mit dem zweiten Raytracinglauf, welcher von der Suchbildkamera ausgeht, durch die Benutzervorgaben übergangen werden. Zum besseren Verständnis fasst Listing 4.3 den Ablauf der beschleunigten Korrespondenzanalyse unter Verwendung der eingeführten Bezeichner, die im realen Quelltext in dieser Form nicht auftauchen, als Pseudocode zusammen. Die Deaktivierungsoptionen bleiben dabei ebenso wie die nachgelagerte Erzeugung des Graustufen-Übersichtsbildes anhand der Disparitäten außen vor.

Hybride hierarchische Zuordnungsbestimmung Unabhängig davon, ob die Korrespondenzanalyse mit Hilfe der Objektpunktschätzung oder auf klassische Weise entlang der Epipolarlinien erfolgt, werden die lokalen Matchingalgorithmen aus Abschnitt 4.6.1.2 nicht direkt aufgerufen, sondern mittelbar über die `hierarchicalMatchingHybrid()`-Funktion der `CMatcher`-Klasse. Das Unterprogramm implementiert die Suche nach dem Musterfenster innerhalb des Suchbereichs als zweistufigen Prozess mit dem Ziel, subpixelgenaue Ergebnisse in angemessener Zeit zu erhalten. Zu diesem Zweck wird zunächst ein lokaler SAD- oder SSD-Matchingschritt mit den Integeralgorithmen auf Basis der `CLocalMatcherStd`-Schnittstelle durchgeführt, welcher die Korrespondenz im Suchbild mit einer maximalen Präzision von einem Pixel liefert. Aufgrund der vergleichsweise hohen Ausführungsgeschwindigkeit sind die Verfahren dabei in der Lage, ein größeres Suchareal relativ zügig zu bearbeiten.

Das Resultat dient anschließend als Eingabe für den zweiten Schritt der abgestuften Zuordnungsbestimmung. Hierbei kommt ein gleitkommagenauer Matchingalgorithmus des objektorientierten Basistyps `CLocalMatcherPrecise` zum Einsatz, dessen Ergebnis schließlich die gesuchte Subpixelposition der besten Übereinstimmung im Suchbild darstellt. Weil vor der Ausführung bereits feststeht, wo sich die gesuchte Korrespondenz ungefähr befindet, kann jetzt der Suchbereich im zweiten Kamerabild stark verkleinert werden. Dabei verringert sich die Zahl der benötigten „teuren“ Rechenschritte für das Erreichen der höheren Zuordnungsauflösung quadratisch in Bezug auf die Seitenlängen des Areals. Eine weitere Laufzeitreduktion lässt sich durch die simultane negative Größenänderung des Korrelationsfensters erreichen, welche wie die Anpassung des Suchbereichs über das Konfigurationsobjekt des Matcherplugins vorgenommen werden kann. In diesem Fall steigt allerdings das Risiko für Fehlzuordnungen im Subpixelbereich und damit die Gefahr, den mit dem zweiten Analyseschritt erzielten Genauigkeitsgewinn wieder zu neutralisieren.

In bestimmten Anwendungsszenarien werden keine gleitkommagenauen Korrespondenzen auf Subpixelebene benötigt, oder man möchte auf die lokalen Matchingschritte mit In-

```
FÜR alle vertikalen Pixelpositionen y1 in der ROI des Musterbildes {
    FÜR alle horizontalen Pixelpositionen x1 in der ROI des Musterbildes {

        # Verfolge den Strahl R1 vom Kamerazentrum des
          Musterbildes durch (x1,y1) in Weltkoordinaten

          WENN es einen Objektschnittpunkt gibt {

            // Matching mit Modellunterstützung

            # Projiziere den Objektschnittpunkt in das Suchbild und
              berechne so das Suchbereichszentrum (x2,y2)
            # Verfolge den Strahl R2 vom Kamerazentrum des
              Suchbildes durch (x2,y2) in Weltkoordinaten

            WENN sich R1 und R2 im gleichen Objektpunkt schneiden {

                # Führe die hybride hierarchische Korrespondenzanalyse
                  um (x2,y2) aus, Ergebnis ist die Zuordnung (x2',y2')
                  und der Ähnlichkeitsgrad d

                WENN d < Grenzwert für akzeptable Zuordnungen {

                    # Trage die Differenz (x2',y2')-(x1,y1) in die
                      horizontale bzw. vertikale Disparitätskarte
                      ein
                }
            }

        } SONST {

            // Matching ohne Modellunterstützung

            FÜR jede Abszisse ex des Suchbildes {

                # Berechne die Ordinate ey des Epipolarlinienpunkts
                  anhand der Stereogeometrie (MVG-Objekt)
                # Führe die hybride hierarchische Korrespondenzanalyse um
                  (ex,ey) aus, Ergebnis ist die Zuordnung (x2',y2')
                  und der Ähnlichkeitsgrad d

                WENN d < Grenzwert für akzeptable Zuordnungen {

                    # Trage die Differenz (x2',y2')-(x1,y1) in die
                      horizontale bzw. vertikale Disparitätskarte
                      ein
                }
            }
        }
    }
}
```

Listing 4.3: Integrierter Matchingalgorithmus im Pseudocode

tegerarithmetik verzichten. Daher lassen sich die beiden Stufen des hybriden Verfahrens wechselseitig, jedoch nicht gleichzeitig, deaktivieren.

Qualitätsprüfung der Korrespondenzen Nach der hybriden hierarchischen Zuordnungsbestimmung eignen sich nicht alle Bildpositionspaare für die Objektrekonstruktion. Aufgrund der im Grundlagenkapitel (Abschnitt 2.1.4.1) angesprochenen Unzulänglichkeiten der lokalen Matchingverfahren, welche letztlich nur eine Näherungslösung für das schlecht gestellte Grundproblem liefern, ist mit einer nicht unerheblichen Zahl von falschen bzw. ungenauen Korrespondenzen zu rechnen. Dieser Umstand spiegelt sich im berechneten Ähnlichkeitsgrad wieder, der praktisch niemals sein theoretisches Minimum annimmt.

Der Anwender des DaTool-Prototyps kann daher über das Konfigurationsobjekt des integrierten Matchingalgorithmus einen relativen Schwellwert (Prozentwert) zur Unterscheidung der akzeptablen von weniger geeigneten Zuordnungen definieren, woraufhin letztere bei der Korrespondenzanalyse verworfen werden. Der Verzicht auf die Verwendung der absoluten Abstandsbeträge soll deren Abhängigkeit von der Fenstergröße für die Korrelation eliminieren. Nichtsdestotrotz bleiben die Grenzwerte wie auch der für den Vergleich erforderliche Relationsoperator abhängig vom eingesetzten Matchingkriterium, wobei sich die prinzipielle Frage stellt, ob und wie man verschiedene Distanzfunktionen ohne einen Verlust ihrer Verlaufseigenschaften in Bezug auf die Qualitätsprüfung einheitlich betrachten kann.

Die relative Qualität jeder gefundenen Zuordnung wird über das Verhältnis des absoluten Ähnlichkeitsgrades zur Bandbreite der Distanzwerte für die Abstandsfunktion (SAD, SSD) und die konkret eingesetzte Muster- bzw. Suchfenstergröße im Unterprogramm `checkMatchQualityCreateDisparityEntry()` der `CMatcher`-Klasse berechnet. Der benötigte spezifische minimale und maximale Abstandsbetrag steht hierbei zusammen mit der numerischen Übereinstimmung und der Suchbildposition der Korrespondenz im jeweiligen Ergebnisobjekt des aktuellen lokalen Matchingschritts, das stets vom einheitlichen Datentyp `CLMResult` ist (siehe Abschnitt 4.6.1.2). Die Vergleichsfunktion für je zwei Distanzwerte liefern die `CSimilarityMeasure`-Traitklassen, die das Ähnlichkeitskriterium kodieren, mit der `isMoreSimilar()`-Routine.

Matchingstatistik Im Verlauf der Korrespondenzanalyse werden durch die `CMatcher`-Klasse statistische Informationen für die objektive Evaluierung der Matchingergebnisse erfasst und in einem Ausführungsprotokoll festgehalten. Hierzu gehören unter Angabe der Bezeichnungen in der XML-Ausgabe in Klammern (siehe Anhang A.3)

(1) die Start- und Endzeit für die Erzeugung des 3D-Baums und die Polygonklassifikation (`StartTime` und `EndTime` im `TreeClassification`-Block),

(2) die Start- und Endzeit für die Zuordnungsbestimmung selbst einschließlich der qualitativen Selektion (`StartTime` und `EndTime` im `Matching`-Block),

(3) der für die aktuelle Matcherkonfiguration konstante minimal und maximal mögliche Abstandswert (`BestPossibleSimilarity` und `WorstPossibleSimlarity`),

(4) der durchschnittliche Ähnlichkeitsgrad der Zuordnungen vor und nach der qualitativen Selektion (average similarity before/after quality check, `AvgSimilarityBeforeQC` und `AvgSimlarityAfterQC`),

(5) der mittlere Disparitätsfehler zur „wahren“ Punktverschiebung des geschätzten Suchbereichszentrums vor und nach der Qualitätsprüfung (average disparity error to raytracing ground truth before/after quality check, `AvgDisparityErrorToRTGTBeforeQC` und `AvgDisparityErrorToRTGTAfterQC`),

(6) die Standardabweichung des mittleren Disparitätsfehlers zur „wahren“ Punktverschiebung des designierten Suchbereichszentrums vor und nach der qualitativen Zuordnungsselektion (disparity error to raytracing ground truth standard deviation before/after quality check, `DisparityErrorToRTGTStdDeviationBeforeQC` und `DisparityErrorToRTGTStdDeviationAfterQC`),

(7) die Anzahl der Strahltreffer mit dem Modell, ausgehend vom Projektionszentrum der ersten und zweiten Kamera (`RayHitsCamera1` und `RayHitsCamera2`),

(8) die Anzahl der lokalen Matchingschritte mit Suchbereichseingrenzung durch das 3D-Modell (`ModelAssistedMatchingSteps`),

(9) die Anzahl der lokalen Matchingschritte, die eine Suche entlang der Epipolarlinie beinhalten (`EpipolarMatchingSteps`),

(10) die Anzahl der qualitativ akzeptablen Zuordnungen (`QualityMatches`) und

(11) die Anzahl der Matchingfehler (`MatchingErrors`).

Die Zeitstempel (1) und (2), die mit Hilfe der standardisierten `ctime()`-Funktion der C++-Laufzeitbibliothek sekundengenau ermittelt werden, belegen die Dauer der primären Arbeitsschritte des funktionalen Algorithmus. Eine objektive Aussage über die Güte der Korrespondenzen liefern das arithmetische Mittel des Übereinstimmungsgrades vor bzw. nach der Qualitätsprüfung (4), der durchschnittliche Disparitätsfehler (5) und dessen Standardabweichung (6) sowie die Anzahl derjenigen Zuordnungen, die den vom Anwender festgelegten Schwellwert nicht verletzen (10).

Die mittleren Matchingdistanzen (4) nach der Ausreißerkorrektur, die für alle respektive die qualitativen Zuordnungen bestimmt werden, bieten als Lageparameter einen Anhaltspunkt für den absoluten Betrag der Übereinstimmungswerte. Sie sind abhängig von den eingesetzten Zuordnungsparametern, insbesondere der Abstandsfunktion (SAD oder SSD) und den Fenstergrößen, jedoch nicht vom Verfahren der Korrespondenzanalyse (modellgestützt oder epipolar). Zur Berechnung werden die Einzeldistanzen parallel zur Ausführung der lokalen Matchingschritte akkumuliert und die Summen abschließend durch die Kardinalitäten (8)+(9) bzw. (10) dividiert.

Neben ihrer Funktion als Hilfsgröße stellt die letztgenannte Anzahl der akzeptablen Zuordnungen (10) gleichzeitig ein primitives Streuungsmaß dar, bei dem der aktuell gesetzte relative Grenzwert für die Güte der Korrespondenzen den Steuerparameter repräsentiert. Als solches gibt das durch simples Inkrementieren bei bestandener Qualitätsprüfung gewonnene Datum Aufschluss über die Verteilung aller Übereinstimmungswerte bezogen auf die minimal und maximal mögliche Distanz der Vorlage zum Suchfenster (3). Wie die Größe (4) hängt die konkrete Kardinalität (10) von den verwendeten Matchingparame-

tern ab, wodurch die Vergleichbarkeit der resultierenden Messergebnisse in beiden Fällen eingeschränkt ist. Die klassischerweise als Streuungsmaß eingesetzte Standardabweichung

$$\sigma_X = \sqrt{E((X - E(X))^2)} \tag{4.15}$$

eignet sich nicht zur Beschreibung der Dispersion der Ähnlichkeitswerte. Selbst bei einer adäquaten Approximation der unbekannten Zufallsverteilung und dem daraus resultierenden Wissen über die Vorschrift zur Bestimmung des Erwartungswertes E der Zufallsvariablen X besitzt sie keine Aussagekraft, da σ die Abweichung vom durchschnittlichen Übereinstimmungsgrad (4) beschreibt. Demgegenüber schlägt sich in den Quantilen wie dem Median, der die geordnete Stichprobe halbiert, zwar eine Änderung der Matchingqualität direkt in der Verschiebung weiter Bereiche des Ähnlichkeitshistogramms in Richtung des theoretischen Minimums (Verbesserung) oder Maximums (Verschlechterung) nieder. Weil jedoch der Rechenaufwand für die unumgängliche Sortierung der n inhaltlichen Abstände zwischen den Muster- und Suchfenstern mindestens $O(n \cdot log\ n)$ beträgt (siehe abermals [60]) und auch der Speicherplatzbedarf für deren Pufferung recht hoch ist, werden sie analog zur Standardabweichung der Distanzwerte nicht von der `CMatcher`-Klasse bestimmt.

Um zumindest auf künstlich generierten 3D-Modellen und Bildern die Güte der gefundenen lokalen Bildübereinstimmungen losgelöst von der für die Zuordnungsschritte eingesetzten Konfiguration evaluieren zu können, werden zusätzlich zum Distanzdurchschnitt und der Anzahl der akzeptablen Zuordnungen der mittlere Disparitätsfehler (5) und dessen Standardabweichung (6) vor und nach der Qualitätsprüfung bestimmt. Als Referenzdaten für die Berechnung dieser Werte dienen die „wahren“ Punktverschiebungen im Stereosystem (ground truth-Disparitäten), die sich aus den mittels Strahlverfolgung bestimmten Suchbereichszentren ergeben. Demnach sind die resultierenden Messergebnisse nur dann aussagekräftig, wenn die Objektpunktschätzung aktiviert ist, das eingesetzte LRF-Modell fehlerfrei erfasst wird und die exakte Orientierung der Kameras zur Szene feststeht – Bedingungen, die mit realen Eingabedaten nicht erfüllbar sind.

Wenn die genannten Voraussetzungen vorliegen, werden zur Bestimmung des arithmetischen Mittels der Disparitätsfehler (5) zunächst die Verschiebungen d_1 und d_2 zwischen dem aktuellen Musterbildpixel und der durch Matching bestimmten Zuordnung respektive dem aktuellen Musterbildpixel und dem mittels Raytracing geschätzten Suchbereichszentrum über die euklidische Abstandsformel ermittelt. Nach der Addition der Differenzbeträge $|d_1 - d_2|$ aller Pixelpaare vor bzw. nach der Qualitätsprüfung und der Division der Summe durch die Anzahl aller bzw. der qualitativ akzeptablen Zuordnungen *ohne* die mit Hilfe der Epipolarliniensuche gefundenen Korrespondenzen erhält man schließlich die finalen Durchschnittsabweichungen. Der neben Größe (8) benötigte Divisor für den Mittelwert des Disparitätsfehlers nach der Selektion bedingt dabei eine klasseninterne Aufspaltung der Menge aller qualitativen Übereinstimmungen (10) entsprechend den beiden Matchingmodi (modellgestützt oder epipolar).

Im Gegensatz zu den Abstandswerten kann die Dispersion der Disparitätsabweichungen (6) um den soeben beschriebenen Durchschnittsfehler (5) unter Annahme einer Normalverteilung mit Hilfe der Standardabweichung σ aussagekräftig geschätzt werden. Die `CMatcher`-Klasse implementiert hierfür das in Gleichung 4.16 aufgeführte Einpass-Äquivalent

der bekannten Rechenvorschrift für eine endliche Stichprobe mit dem Umfang N, wodurch die Bestimmung von σ ohne die *vorherige* Kenntnis des durchschnittlichen Verschiebungsfehlers $\bar{x}$ der einzelnen euklidischen Disparitätsdifferenzen x_i zu den ground truth-Daten auskommt [65]

$$\sigma_X = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2} = \sqrt{\frac{1}{N-1}(\sum_{i=1}^{N} x_i^2 - \frac{1}{n}(\sum_{i=1}^{N} x_i)^2)} \tag{4.16}$$

Abgesehen von ihrer Rolle als Operanden für die Ermittlung der Zuordnungsqualität charakterisieren einige der bereits erläuterten statistischen Angaben und die noch nicht genannten Größen die Struktur des an den Matcher übergebenen LRF-Modells. Mit (7) wird der Anteil der Verdeckungen zwischen den Bildern des Stereosystems ausgedrückt, während die Werte (8) und (9) neben ihrer Funktion als Hilfsgrößen die opaquen Bereiche und die „Löcher“ der Szene quantifizieren, sofern die Objektpunktschätzung bzw. das Epipolarverfahren im integrierten Algorithmus aktiviert sind. Die Zahl der Matchingfehler (11) gibt dem Benutzer schließlich eine Rückmeldung über das Scheitern von lokalen Korrespondenzanalyseschritten. Angaben ungleich null deuten auf eine fehlerhafte oder singuläre Konfiguration des funktionalen Algorithmus oder noch nicht behobene Programmfehler hin und identifizieren ungültige Ausführungsprotokolle.

Um den Kopiervorgang bei der modulinternen Übergabe der statistischen Angaben zu vereinfachen, deren Erfassung eine Akkumulation bei der Zuordnungsbestimmung bedingt, werden die betroffenen Werte (4)-(11) in einem gemeinsamen Verbund `CMatcherStats` zusammengefasst. Wenn der integrierte Algorithmus am Ende der Ausführung die Kontrolle an das DaTool-Rahmenprogramm transferiert, fügt die federführende `CMatcher`-Klasse die Attribute dem Ausführungsprotokoll in Gestalt eines Konfigurationsobjekts hinzu und stellt dieses dem Benutzer als Rückgabewert bereit.

4.7.4. Modellverbesserung

Das Plugin für die Modellverbesserung korrigiert die vorhandenen Objektpunkte der LRF-Szene und kann diese mit den photogrammetrisch rekonstruierten Raumelementen optional verdichten. Dazu verbindet es den Ansatz der strahlunabhängigen 3D-Punktverschiebung aus Abschnitt 3.2.5 der Konzeption mit dem Ersetzungsverfahren. Der Standort des LRF relativ zum Modell wird nicht benötigt.

Technisch betrachtet besteht der funktionale Algorithmus aus den Klassen der `CCorrect...`-Familie, benutzt aber auch die Module zur Strahlverfolgung. Er erwartet als Eingabedaten das zu berichtigende Modell, die horizontale und vertikale Disparitätskarte vom Matcher sowie drei Konfigurationsobjekte mit den Eigenschaften der Stereokameras und den Parametern für den Korrekturvorgang. Aus diesen Angaben werden ein Graustufen-Raytracingbild, ein Modell mit den in der ersten Aufnahme sichtbaren ursprünglichen Raumpunkten (Raytracingmodell), ein weiteres Modell mit den korrigierten Raumpunkten und das Ausführungsprotokoll erzeugt. Die Polygon- und Texturinformationen der originalen LRF-Szene überträgt das Verbesserungsverfahren dabei nicht auf die ausgegebenen 3D-Daten, weil gemäß der Konzeption keine Garantie besteht, dass die modifizierten Eckpunkte der Dreiecke zum gleichen Objekt wie vorher gehören. Außerdem kommen beim optionalen Einsatz des Ersetzungsverfahrens neue Raumelemente zur ursprünglichen Sze-

ne hinzu, die sich nur mit einem für diese Arbeit unvertretbar hohen Aufwand (Herstellung einer Raumpunkt-Flächen-Zuordnung) in das vorhandene Polygonnetz integrieren lassen.

Für die Korrektur mit dem Verschiebungsverfahren werden die vorhandenen LRF-Punkte in das Bild der ersten Kamera projiziert, welches bei der Korrespondenzanalyse die Rolle des Musterbildes übernimmt. Durch die Abbildung wird eine 3D-2D-Zuordnung der Raumpunkte zu den Pixelpositionen hergestellt. Diese induziert aufgrund des verfahrensbedingten Zusammenhangs zwischen der Stereoaufnahme und der horizontalen wie vertikalen Disparitätskarte des Matchers gleichwohl eine 3D-3D-Verbindung zwischen den LRF-Modellpunkten und den photogrammetrisch rekonstruierbaren Raumpositionen. Benutzt man letztere in der Korrekturphase als Ziel für die Verschiebung der Originalpunkte, erhält man die korrigierte Szene, sofern das Stereosystem präzise kalibriert ist und qualitativ hochwertige Matchingergebnisse vorliegen. Abbildung 4.12 zeigt die beschriebenen Verhältnisse schematisch.

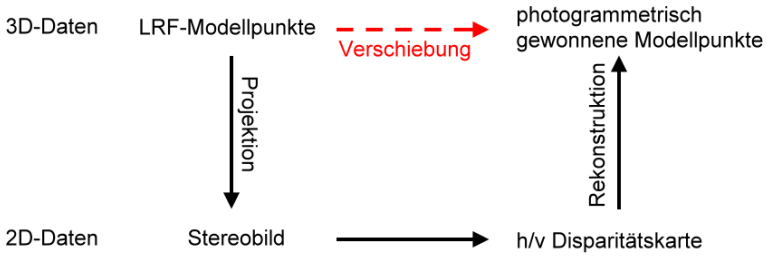


Abbildung 4.12.: Verhältnisse beim Verschiebungsverfahren

Die wahlfreie Verdichtung der vorhandenen Modellpunkte erfolgt durch die Rekonstruktion von ausdehnungslosen Raumelementen aus denjenigen Disparitäten, die nicht am Verschiebungsprozess beteiligt sind. Dazu wird die korrigierte Szene mit den neuen rein photogrammetrisch gewonnenen 3D-Punkten nachträglich ergänzt, was nur funktioniert, wenn aus der Zuordnungsanalyse nach der Qualitätsprüfung mehr Korrespondenzen und somit auch mehr Einträge in den Disparitätskarten resultieren, als nulldimensionale Modellelemente im ersten Kamerabild sichtbar sind.

4.7.4.1. Projektion der Modellpunkte

Die Projektion der Raumpunkte für die Modellverbesserung ist in der `raytraceModel()`-Routine der `CCorrect`-Klasse des Algorithmusplugins umgesetzt. Wie der Name vorgibt, nutzt die Funktion den rekursiven oder inkrementellen Strahlverfolgungsalgorithmus der Objektpunktschätzung, um diejenigen Teile der Szenenoberfläche zu bestimmen, welche im durch die region of interest des Matchers definierten Ausschnitt des ersten Kamerabildes sichtbar sind – für andere Bildbereiche existieren keine verwertbaren Disparitäten. Damit dies fehlerfrei funktioniert, muss die rechteckige ROI des Matchers in der Konfiguration des Modellverbesserungsalgorithmus erneut angegeben werden, da sich aus den übergebenen Disparitätskarten lediglich deren Größe, nicht jedoch die Position (Koordinaten der linken oberen Ecke) in der Aufnahme ergibt. Ohne das Raytracing ist man zum einen damit konfrontiert, *alle* diskreten Raumelemente abzubilden und auf die Lage innerhalb der Bildfläche zu untersuchen. Zum anderen werden möglicherweise auch unsichtbare LRF-Punkte projiziert, die von Modellpolygonen im Vordergrund verdeckt sind (Abbildung

4.13). Diese Raumelemente können nicht korrigiert werden, weil es hierfür keine gültigen Korrespondenzen im zweiten Kamerabild und somit auch keine verwertbaren Disparitäten gibt. Vielmehr führt eine Verwendung der maskierten 3D-Punkte im Rahmen der eingangs beschriebenen Verbesserungsmethode im Endergebnis zu starken Deformationsfehlern, sofern die Positionsverschiebung für ein sichtbares einem unsichtbaren Raumelement mit identischem Abbild zugeschlagen wird.

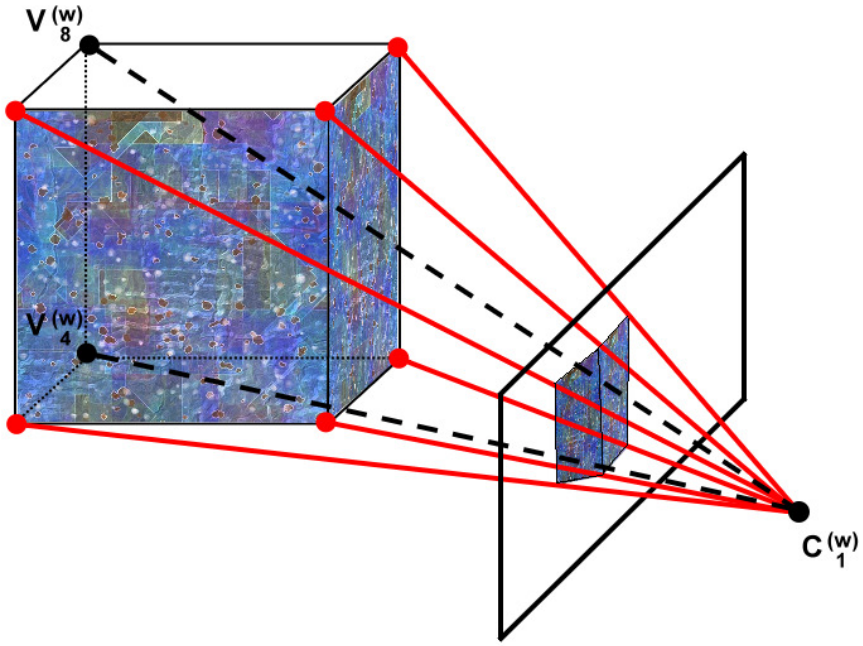


Abbildung 4.13.: Verdeckte Modellpunkte bei der Projektion zur Modellverbesserung. Die Raumelemente $\mathbf{V}_4^{(w)}$ und $\mathbf{V}_8^{(w)}$ sind im ersten Kamerabild nicht sichtbar, haben demzufolge keine korrespondierenden Projektionen in der zweiten Stereoaufnahme und können somit mangels zugehöriger Disparitätswerte nicht korrigiert werden.

Analog zur Vorstufe des integrierten Matchers setzt die Projektionsroutine zunächst einen 3D-Baum auf und sendet jeweils einen Sichtstrahl vom Zentrum der ersten Kamera durch alle Pixel in Weltkoordinaten desjenigen Teils der zugehörigen Aufnahme, welcher dem vom Matcher verwendeten relevanten Bildausschnitt entspricht. Wenn es einen Schnittpunkt mit einem Oberflächendreieck des LRF-Modells gibt, werden die Eckpunkte des getroffenen Polygons der Abbildungsvorschrift übergeben. Dabei entstammen der benötigte Kameratyp und die Angaben zur inneren wie äußeren Orientierung den Konfigurationsobjekten des Plugins, die mit denen der Korrespondenzanalyse identisch sein sollten. Diejenigen der drei resultierenden subpixelgenauen Bildpositionen, welche innerhalb der bei der Zuordnungsbestimmung definierten ROI der Aufnahme liegen, werden anschließend in Gestalt eines `CTraceEntry`-Verbunds in ein dynamisches Feld vom Typ `lrg::vector`, die Raytracingliste, übernommen. Jeder Listeneintrag enthält abgesehen von den obligatorischen Gleitkomma-Pixelkoordinaten die Datensatznummer sowie den 3D-Punkt- und Dreiecksindex in die Speicherstruktur des LRF-Modells. Gibt es keinen Strahltreffer, existiert kein Objektpunkt, der einer Korrektur bedarf.

Ein großes Problem des beschriebenen Strahlverfolgungsprozesses stellen doppelte Einträge in der Raytracingliste dar. Sie entstehen, wenn ein und dasselbe Oberflächenpolygon von mehreren Strahlen getroffen wird. In diesem Fall kommt es zu einer redundanten Ablage der gleichen Pixelposition und anderen `CTraceEntry`-Attribute, was je nach der Auflösung des Bildes und des Laserscanners zu einem beträchtlichen Mehrbedarf an Speicherplatz ohne jeden Nutzen führt. Daher erfolgt während der Strahlverfolgung eine Erkennung und Zurückweisung von Doppeleinträgen in der Raytracingliste anhand des ganzzahligen Dreiecksindex und der Datensatznummer, womit deren Existenzberechtigung im Verbund erklärt ist.

Normalerweise bedingt die Detektion der Doubletten eine zeitaufwendige Iteration über alle bisher gespeicherten Einträge des dynamischen Feldes. Weil jedoch mit einer gewissen Wahrscheinlichkeit davon ausgegangen werden kann, dass Sichtstrahlen durch benachbarte Pixel innerhalb einer Bildzeile der betrachteten Aufnahme auf das gleiche Modellpolygon treffen, lässt sich der Erkennungsprozess durch die Zwischenspeicherung (caching) des zuletzt touchierten Oberflächendreiecks beschleunigen. Die vorliegende Implementierung vergleicht daher erst einmal, ob der aktuelle Schnittpunkt auf dem gepufferten Triangel liegt. Ist dies der Fall, wird anhand der Disparitäten mit Hilfe eines `CMVG`-Objekts ein zusätzlicher Modellpunkt rekonstruiert, sofern die horizontale wie vertikale Positionsabweichung in den Stereobildern zu einer qualitativ akzeptablen Korrespondenz gehören und damit vorliegen[14]. Der Liste aller neuen Raumelemente hinzugefügt, dient die berechnete Primitive bei Bedarf in der Korrekturphase zur Verdichtung der Szene. Über eine positiv ganzzahlige Abtastrate n kann der Anwender des Plugins dabei den Grad der Modellergänzung steuern. Mit dem Wert wird aus Gründen der ökonomischen Speichernutzung bereits an dieser Stelle festgelegt, dass lediglich ein n-tel der Objektpunkte Einzug in das korrigierte Endergebnis des Algorithmus halten, welche aus multiplen Strahltreffern mit dem gleichen Dreieck hervorgehen.

Wird das zwischengespeicherte Polygon nicht vom Strahl getroffen, ist die Suche nach einem Doppeleintrag innerhalb der Raytracingliste nicht zu vermeiden. Wenn das Ergebnis hierbei positiv ausfällt, berechnet `raytraceModel()` wiederum einen zusätzlichen Raumpunkt anhand der Disparitätskarten und unter Berücksichtigung der Abtastrate. In beiden Fällen, dem Cache-Treffer und dem positiven Suchergebnis, wird anschließend unmittelbar ohne die Abbildung der Polygoneckpunkte der Doublette mit dem nächsten Strahl respektive Pixel in der Kameraaufnahme fortgefahren.

Mit der Raytracingliste liefert die Projektion der vorhandenen Modellpunkte im Endeffekt den Zustand vor der qualitativen Verbesserung durch die photogrammetrischen Informationen. Für die visuelle Begutachtung werden die mittels Strahlverfolgung berechneten Eckpunkte der Polygone im Anschluss an die Ausführung des funktionalen Algorithmus als diskrete 3D-Wolke zurückgegeben. Die Erstellung dieses elementaren Raytracingmodells übernimmt die `buildRaytraceModel()`-Funktion der `CCorrect`-Klasse, indem sie anhand der Datensatznummern und Punktindizes in den `CTraceEntry`-Listeneinträgen die zugehörigen 3D-Raumkoordinaten aus dem Eingabemodell extrahiert und neu zusammenstellt.

Das zweidimensionale Pendant zum Raytracingmodell bildet das Raytracingbild, das während der Modellpunktprojektion in der `raytraceModel()`-Routine erzeugt und vom Plug-

[14] Man beachte, dass die Schnittpunktkoordinaten der Doublette selbst nicht als unmittelbares Wiederherstellungsergebnis verwendet werden können, da sie als fehlerbehaftete Größe Gegenstand der qualitativen Verbesserung sind.

in ausgegeben wird. Es repräsentiert die euklidischen Distanzen der Schnittpunkte zum Projektionszentrum der ersten Kamera, die bei der Strahlverfolgung anfallen, mit unterschiedlichen Grauwerten. Durch die transparente Überlagerung der mustergebenden Stereoaufnahme mit dem Raytracingbild unter Verwendung einer geeigneten Bildverarbeitungssoftware lässt sich die Orientierung der Kameras zum originalen LRF-Modell visuell nachprüfen.

Listing 4.4 fasst die erläuterte Projektionsmethode als Pseudocode abschließend zusammen. Die Erstellung des Raytracingbildes ist der Übersichtlichkeit halber nicht aufgeführt. Man beachte, dass die äußere Iteration die Größenangaben der Disparitätskarten verwendet und nicht die als identisch angenommenen Abmessungen der region of interest des Matchers. Dadurch wird der Zugriff auf Pixel außerhalb der beiden Gleitkommabilder, der ein undefiniertes Verhalten des DaTool-Prototyps mit sich bringt, selbst dann unterbunden, wenn die ROI nicht die Werte bei der Korrespondenzanalyse aufweisen sollte (fail-safe-Philosophie).

4.7.4.2. Korrekturphase

Nach der Projektion der sichtbaren Oberflächenpunkte des LRF-Modells enthält die Raytracingliste alle Informationen für die Korrektur mit Hilfe der Disparitätskarten, die die Funktion `computeCorrectedPoints()` der `CCorrect`-Klasse durchführt. In einem sequenziellen Durchlauf durch das dynamische Feld werden die gespeicherten `CTraceEntry`-Verbundinstanzen ausgelesen und die Disparitäten zu den darin enthaltenen Bildkoordinaten ermittelt. Hierfür überprüft die Routine zunächst, ob an den vier ganzzahligen Positionen der horizontalen Disparitätskarte (und somit automatisch der vertikalen Entsprechung) um die subpixelgenaue Angabe aus dem Listeneintrag herum überhaupt gültige Verschiebungswerte vorliegen. Dies ist ausschließlich bei denjenigen Übereinstimmungen gegeben, die die Qualitätsprüfung der Korrespondenzanalyse bestanden haben. Im Fall eines positiven Resultats erfolgt die Berechnung der finalen Verschiebung, indem die Disparitäten an den vier Positionen sowohl in der horizontalen als auch vertikalen Karte mit den subpixelgenauen Zugriffsfunktionen aus Abschnitt 4.6.1.2 interpoliert werden.

Aus der interpolierten Verschiebung der Projektion des aktuellen Modellpunkts aus der Raytracingliste zwischen den Stereoaufnahmen wird anschließend mit einem `CMVG`-Objekt die zugehörige photogrammetrische Raumposition rekonstruiert. Diese dient aufgrund ihrer potenziell höheren Präzision als Korrekturwert und wird anstelle des im momentanen Listeneintrag vermerkten LRF-Punkts in das verbesserte Ausgabemodell eingetragen, was einer Verschiebung des ungenauen Originals in das genaue Raumelement gleichkommt. Wenn die `computeCorrectedPoints()`-Funktion alle Einträge der Raytracingliste abgearbeitet hat, werden abschließend die während der Strahlverfolgung neu gewonnenen ausdehnungsfreien 3D-Primitiven zur korrigierten Punktwolke hinzugefügt.

4.7.4.3. Korrekturstatistik

Wie der integrierte Matcher protokolliert auch das Plugin zur Modellverbesserung, genauer die `CCorrect`-Klasse, statistische Daten bei der Ausführung in einem speziellen Konfigurationsobjekt. Dazu gehören im einzelnen

```
# Abtastzähler=0
FÜR alle vertikalen Pixelpositionen y in den Disparitätskarten {
   FÜR alle horizontalen Pixelpositionen x in den Disparitätskarten {

       # Verfolge den Strahl R vom Kamerazentrum des Musterbildes
         durch die ROI-Position+(x,y) in Weltkoordinaten

       WENN es einen Objektschnittpunkt gibt {

           // Cache-Initialisierung
           WENN bislang noch kein Dreieck geschnitten wurde {
               # Setze voriges Dreieck dem aktuellen Dreieck gleich
           } SONST {

               // Überprüfe auf Cachetreffer
               WENN der Schnittpunkt im gleichen Modelldreieck wie der vorige liegt
               UND das aktuelle Dreieck nicht das erste geschnittene überhaupt ist {
                   # vorigesDreieckGetroffen=wahr
               } SONST {

                   WENN der Schnittpunkt in einem Modelldreieck der Raytracingliste
                   liegt {
                       # vorigesDreieckGetroffen=wahr
                   } SONST {
                       # vorigesDreieckGetroffen=falsch
                   }
               }
           }

           // aktualisiere Dreieck im Cache
           # Setze das vorige getroffene Dreieck dem aktuellen gleich

           // Behandle Doppeleintrag in der Raytracingliste
           WENN vorigesDreieckGetroffen {

               # Abtastzähler=(Abtastzähler+1) mod Abtastrate
               WENN Ersetzungsverfahren aktiv UND Abtastzähler==0 {

                   WENN die Horizontaldisparität an Kartenposition (x,y)
                   gültig ist {
                       # Rekonstruiere den Raumpunkt aus der horizontalen
                         und vertikalen Disparität an Position (x,y)
                         über ein MVG-Objekt
                       # Füge den rekonstruierten Raumpunkt in die Liste
                         der neuen Modellpunkte ein
                   }
               }

               # Nächster Schleifendurchlauf
           }

           // Behandle negativen Cachetreffer
           # Extrahiere die 3D-Eckpunkte v(1), v(2) und v(3) des getroffenen
             Dreiecks aus der Modellspeicherstruktur
           # Projiziere v(1), v(2) und v(3) in die Bildfläche,
             erhalte v(1)', v(2)', v(3)'

           WENN v(i)' (i=1...3) innerhalb der ROI liegt {
               # Füge v(i)' der Raytracingliste hinzu
           }
       } // Objektschnittpunkt
   } // x
} // y
```

Listing 4.4: Pseudocode für die Projektion der Modellpunkte bei der Modellverbesserung

(1) die Start- und Endzeit für das Raytracing zur Identifikation der Polygoneckpunkte des Modells, die in der ROI der ersten Stereoaufnahme liegen (`StartTime` und `EndTime` im `Raytracing`-Block),

(2) die Start- und Endzeit für die Erstellung des Raytracingmodells (`StartTime` und `EndTime` im `CreateRaytraceModel`-Block),

(3) die Start- und Endzeit für die Modellpunktkorrektur mittels Punktverschiebung (`StartTime` und `EndTime` im `Correction`-Block),

(4) die Anzahl derjenigen vorhandenen Modellpunkte ohne Doubletten, die mit den vorliegenden Stereoaufnahmen korrigierbar sind (`UniqueCorrectablePoints`),

(5) die Anzahl der tatsächlich korrigierten vorhandenen Modellpunkte (`CorrectedPoints`),

(6) die Anzahl der rein photogrammetrisch gewonnenen 3D-Punkte für die Verdichtung des Modells (`AdditionalPhotogrammetricPoints`) und

(7) die Anzahl der Raumpunkte des resultierenden 3D-Modells nach der Korrektur und Verdichtung (`OutputPoints`) ,

wobei die in Klammern angegebenen Bezeichnungen denen der XML-Ausgabe entsprechen (siehe Anhang A.3). Die Start- und Endzeiten für die Strahlverfolgung (1), die Erstellung des Raytracingmodells (2) und die eigentliche Korrektur der vorhandenen Modellpunkte (3) werden wie bei der Korrespondenzanalyse auf die Sekunde genau mit der Standardfunktion `ctime()` bestimmt. Das Ergebnis der Projektionsphase wird mit der Größe (4) quantifiziert, wobei sich nicht zwangsläufig alle in der ROI befindlichen Dreieckseckpunkte mit den photogrammetrisch bestimmten Verschiebungswerten korrigieren lassen. Die Zahl der 3D-Punkte (5), die in beiden Stereobildern auftreten und für die eine Disparitätsinterpolation mit anschließender Rekonstruktion möglich ist (keine Verdeckung, keine monokulare Randlage), ist in der Praxis stets kleiner. Wird die Verdichtung des LRF-Modells aktiviert, markiert (6) die Anzahl der neu hinzugefügten Raumelemente, die zusammen mit (5) schließlich die Mächtigkeit der finalen korrigierten Punktwolke (7) ergibt.

4.7.5. Test und Evaluierung

Für den Test und die qualitative Evaluierung des integrierten Matchingalgorithmus sowie der Modellverbesserung werden exakte[15] Referenzdaten sowohl für das LRF-Modell als auch die zugehörigen Stereobilder benötigt. Die Verwendung von Aufnahmen aus der realen Umgebung ist dabei problematisch, weil sie in jedem Fall durch Mess- und Abbildungsfehler verfälscht sind.

Wenn keine realen Referenzdaten für die Evaluierung verwendet werden können, bleibt als Ausweg die Erzeugung einer künstlichen Testwelt und die virtuelle Aufnahme von Stereobildern mit definierten Kameraparametern innerhalb dieser Szene. Hierfür implementiert die DaTool-Software einen speziellen Testweltgenerator. Außerdem existiert ein Algorithmusplugin für die quantifizierbare Störung des resultierenden 3D-Modells.

[15] im Sinne der numerischen Rechengenauigkeit, wie sie doppelt breite Gleitkommazahlen mit dem System ermöglichen, auf welchem die DaTool-Software läuft

4.7.5.1. Testweltgenerator

Der Testweltgenerator wird durch die Klassen der `CWorldGen...`-Familie implementiert. Sie kodieren einen künstlichen quaderförmigen Raum der Größe 20 x 10 x 20 Weltlängeneinheiten. Darin ist eine endliche Anzahl von Objekten an bekannten Positionen und mit festgelegten Abmessungen gemäß der im Anhang A.1 aufgeführten Karte platziert. Wie einige Beispielaufnahmen aus den vorangegangenen Sektionen dieser Arbeit gezeigt haben, handelt es sich im Speziellen um unterschiedlich texturierte Boxen und mit realen Bildaufnahmen überzogene begrenzte Ebenen, von denen einige zur besseren Orientierung und für Kalibrierungszwecke ein schwarz-weißes Passpunktmuster aufweisen.

Entsprechend ihrer Gestalt bestehen die Wände und die Gegenstände des Testraums aus einer oder mehreren zweidimensionalen Viereckflächen, was aus der Tatsache heraus motiviert ist, dass sie sich besonders leicht erzeugen lassen (Klasse `CWorldGenPlane`). Ausgehend von den Eckkoordinaten werden dabei zunächst die Raumpunkte entlang der Kantengeraden durch gleichmäßige Unterteilung berechnet. Anschließend erfolgt die Bestimmung der inneren Raumelemente als Kreuzung der beiden Geraden, die durch je zwei gegenüberliegende Kantenpunkte verlaufen, über die Lösung eines linearen Gleichungssystems nach der Regel von Cramer [66]. Dabei ist nach Konstruktion sichergestellt, dass ein Schnittpunkt selbst bei verdrehten Viereckseiten existiert.

Um letztendlich eine Fläche zu ergeben, werden die generierten 3D-Punkte anschließend mit Polygonen regulär vermascht. Dazu werden je vier Raumelemente zu zwei Dreiecken verbunden. Als Resultat erhält man die Referenzen auf die Eckpunkte für die Speicherstruktur des künstlichen Modells. Rückblickend auf Abschnitt 4.6.2 werden auch die Texturkoordinaten berechnet. Wegen der gleichmäßigen Struktur der Vierecks verteilen sich die Positionsangaben in die Bitmap für jeden Flächenpunkt äquidistant über den Bereich $[0..1] \times [1..0]$ relativ zur Texturgröße. Dabei korrigiert die Spiegelung in der Ordinate die vertikale Orientierung des Oberflächenbildes zum Viereck. Die Textur, deren Speicherort in einem Zeichenkettenattribut von `CWorldGenPlane` abgelegt ist, wird dadurch nicht kopfüber projiziert. Abbildung 4.14 veranschaulicht die Erzeugung der Objektflächen.

Aus den einzelnen und durch die Klasse `CWorldGenBox` zu quaderförmigen Sechsergruppen zusammengefassten Vierecken stellt die `CWorldGenWorld`-Klasse die gesamte Testwelt zusammen. Diese wird danach Fläche für Fläche im VRML-Format und als Szenendefinition für den POV-Ray Raytracer [67] auf dem Massenspeicher abgelegt. Die VRML-Repräsentation stellt hierbei das LRF-Modell des Testraums dar. Sie lässt sich direkt als Eingabedatenobjekt für den integrierten Matcher verwenden. Mit Hilfe der POV-Ray-Ausgabe werden die noch fehlenden Stereoaufnahmen erzeugt. Die Raytracing-Software, die im Gegensatz zur DaTool-Implementierung primär die Farbwerte in den Objektschnittpunkten bestimmt, liefert photorealistische Bilder mit Licht- und Schatteneffekten. Diese weisen weder Verzeichnungsartefakte noch Rauschanteile auf, welche sich allerdings nachträglich für mehr Praxisnähe durch eine Bildverarbeitungssoftware mit definierter Intensität aufschlagen lassen. Die Einstellungen für die virtuelle Kamera, die das Werkzeug für die Projektion von Szenenausschnitten verwendet und welche in Bezug auf den Typ (planar, zylindrisch) sowie die inneren und äußeren Parameter konfigurierbar ist, sind mit der DaTool-Software vollständig kompatibel. Sie werden bei der Erstellung der Szenendefinition über eine separat anzugebende „Headerdatei“ eingebunden und sind dadurch austauschbar. In eingeschränktem Maß gilt dies auch für die Objekttexturen im Testraum. Der Generator erwartet die Spezifikation des Pfades zu den Oberflächenbitmaps, die in der

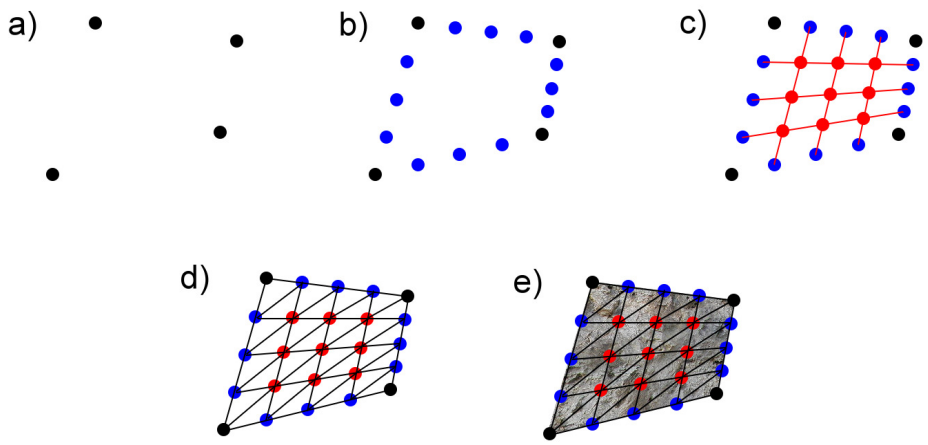


Abbildung 4.14.: Konstruktion einer Viereckfläche durch den Testweltgenerator (a) Ausgangssituation mit vier Eckpunkten (b) Berechnung der Punkte entlang der Kanten (c) Bestimmung der inneren Flächenpunkte als Kreuzung gegenüberliegender Kantenpunkte (d) Reguläre Vermaschung (e) Texturierung mit dem Bild einer Steinoberfläche

POV-Ray-Ausgabe zwar über feste Bezeichnungen referenziert werden, jedoch inhaltlich nicht auf das derzeit verwendete Design festgelegt sind.

4.7.5.2. Modelldeformation

Um zu untersuchen, inwieweit sich die 3D-Informationen mit den photogrammetrisch rekonstruierten Objektpunkten korrigieren lassen, müssen die exakten Referenzdaten mit einem definierten Störbetrag überlagert werden. Diese Aufgabe übernimmt das DaTool-Plugin zur Modelldeformation um die `CDistort...`-Klassen. Das übergebene Modellobjekt wird entweder mit dem achsen- oder dem strahlorientierten Störalgorithmus bearbeitet und als verfälschte Kopie zurückgeliefert.

Das achsenorientierte Verfahren addiert zufällig berechnete Werte zu den Punktkoordinaten der LRF-Szene. Die nullsymmetrischen Intervalle, in denen die Summanden liegen, werden dabei vorab vom Benutzer für jede Raumrichtung getrennt festgelegt, wodurch die hinzugefügten Rauschanteile eine Vorzugsrichtung erhalten können. Der strahlbezogene Algorithmus versucht dagegen, dass Verhalten eines realen Laserscanners zu emulieren. Nach der Angabe des LRF-Standorts $\mathbf{L}^{(\mathbf{w})}$ in der Szene und des Störintervalls wird jeder Objektpunkt $\mathbf{X}_{\mathbf{k}}^{(\mathbf{w})}$ um einen zufälligen Betrag entlang des virtuellen Laserstrahls $\mathbf{L}^{(\mathbf{w})}\mathbf{X}_{\mathbf{k}}^{(\mathbf{w})}$ verschoben. Diese Rauschcharakteristik korreliert dabei mit der Feststellung aus dem Grundlagenkapitel, wonach die LRF-Messfehler primär in der Ausbreitungsrichtung des emittierten Lichtimpulses auftreten.

5. Ergebnisse

Das vorletzte Kapitel befasst sich mit der Evaluierung des integrierten Matchingverfahrens im Hinblick auf die Ausführungsgeschwindigkeit und des funktionalen Algorithmus zur qualitativen Verbesserung des LRF-Modells durch die mit dem Stereosystem rekonstruierten Raumpunkte. Hierzu werden verschiedene Messungen an der vom DaTool-Prototyp generierten künstlichen Testwelt und einer realen Szene durchgeführt.

Als Testsystem für die Berechnung der Ergebnisse kommt ein handelsüblicher PC mit einem Athlon 64 X2 4400+ Doppelkernprozessor zum Einsatz, welcher mit 2,2 GHz pro CPU getaktet ist und auf insgesamt zwei Gibibyte DDR-SDRAM Hauptspeicher zurückgreifen kann. Daneben enthält der genutzte Rechner eine über die SerialATA-Schnittstelle angebundene Festplatte als Massenspeicher mit einer via h2benchw [68] praxisnah gemessenen mittleren Zugriffszeit von 14,3 ms und einer durchschnittlichen Datentransferrate von 18,8 MiB/s. Von den verfügbaren Zentraleinheiten nutzt DaTool als Singlethread-Applikation effektiv nur einen Kern, weswegen die zeitsparende parallele Ausführung von zwei Instanzen praktisch ohne wechselseitige Beeinflussung möglich ist und auch erfolgt. Den Softwareunterbau für das Programm, das für die einzelnen Untersuchungen teilweise mit speziellen Einstellungen übersetzt wird, bildet eine Installation des 32-Bit-Betriebssystems Windows XP mit aufgespieltem Service Pack 2 und den in Abschnitt 4.2 aufgeführten Bibliotheken und Werkzeugen.

5.1. Künstliche Testwelt

Die Untersuchungen an der künstlichen Testwelt und den mit POV-Ray angefertigten Bildaufnahmen zeigen auf, wie sich die Zuordnungsanalyse unter verschiedenen exakt definierten Ausgangsbedingungen verhält. Als dreidimensionale Testdatensätze für die Objektpunktschätzung werden zwei Varianten des generierten Raums mit unterschiedlichen Punktdichten verwendet, um den Einfluss der STXXL-Speicherverwaltung und die Eigenschaften der Strahlverfolgungsalgorithmen analysieren zu können. Die genauen Daten der 20 x 10 x 20 Längeneinheiten im 3D-Koordinatensystem (LE) umfassenden virtuellen Modelle A (grob) und B (fein) sind in Tabelle 5.1 zusammengefasst.

Für die Zuordnungsbestimmung werden aus der POV-Ray-Ausgabe des DaTool-Weltgenerators von Szene A[16] zwei Stereobildpaare mit planarer (Stereopaar 1) sowie zylindrischer Abbildungscharakteristik (Stereopaar 2) und identischen Beleuchtungsverhältnissen generiert. Tabelle 5.2 führt die Parameter der verwendeten virtuellen Kameras auf. Die dargestellten Werte existieren bis auf die platzbedingte Rundung in identischer Form als XML-Konfigurationen für den Prototyp.

[16] Die Punktdichte des Modells ist für die Erstellung der Bilder unerheblich.

Alle Einzelaufnahmen sind im Ausgangszustand exakt im Sinne der Präzision von doppelt genauen Gleitkommazahlen zum Modell ausgerichtet und mit einem uniformen Rauschen von zwei Prozent über alle RGB-Kanäle versehen. Dies entspricht bei der verwendeten Auflösung von acht Bit pro Farbkomponente einer maximalen Abweichung von 5,12 Intensitätsstufen vom Sollwert und dient zur Nachbildung der stochastischen thermoelektrischen Effekte im Sensor einer qualitativ hochwertigen Digitalkamera. Abbildung 5.1 stellt die Stereopaare graphisch dar.

Eigenschaft	**Modell A**	**Modell B**
Abmessungen	20 x 10 x 20 LE	
Punkte pro Flächenkante	20	100
Anzahl der Raumpunkte pro Objektfläche	400	10.000
Anzahl der Raumpunkte insgesamt	20.800	520.000
Punktabstand auf den Objektflächen	0,05 .. 1 LE	0,01 .. 0,2 LE
Anzahl der Dreiecke pro Objektfläche	722	19.602
Anzahl der Dreiecke insgesamt	37.544	1.019.304
Speicherplatzbedarf (VRML-Datei)	3,3 MiB	83,1 MiB

Tabelle 5.1.: Eigenschaften der für die Evaluierung verwendeten künstlichen 3D-Modelle

Parameter	**Stereopaar 1**		**Stereopaar 2**	
	Bild #1	**Bild #2**	**Bild #1**	**Bild #2**
Kamerazentrum	(0; 1; -4)	(0,5; 1; -4)	(-0,5; 1; -4)	(0,5 ;1; -4)
Rotation ω, ϕ, κ	(0°; 0°; 0°)	(0°; 0°; 0°)	(0°; 4,763°; 0°)	(0°; 355,236°; 0°)
Projektionstyp	planar		zylindrisch	
Sensor-/Bildpixel	1.024 x 768		1.024 x 384	
Pixelgröße	0,00001 x 0,00001 LE		0,00001 x 0,00001 LE	
Brennweite	0,005 LE		0,00488923984 LE	
Kamerahauptpunkt	(511,5; 383,5)		(511,5; 191,5)	
Sichtfeld	91,359° x 75,049°		120° x 42,880°	

Tabelle 5.2.: Interne und externe Kameraparameter für die Erzeugung der Stereoaufnahmen von der künstlichen Testwelt.

5.1.1. Matching

Die Evaluierung des integrierten Matchers erfolgt, indem das entsprechende DaTool-Algorithmusmodul mit den beschriebenen 2D- und 3D-Eingabedaten, den maschinenlesbaren Kameraparametern sowie einer Reihe von systematisch entwickelten Konfigurationen für die Programmkomponente selbst gestartet wird. Dabei stehen die Ausführungsgeschwindigkeit in den verschiedenen Berechnungsmodi und die erreichbare Genauigkeit der Zuordnungen im Mittelpunkt. Für die Auswertung der Testreihen werden die Graustufen-Übersichtsbilder, die der funktionale Algorithmus liefert, visuell begutachtet. Daneben fließen die statistischen Daten der Ausführungsprotokolle nach Abschnitt 4.7.3.3 in die Schlussfolgerungen ein.

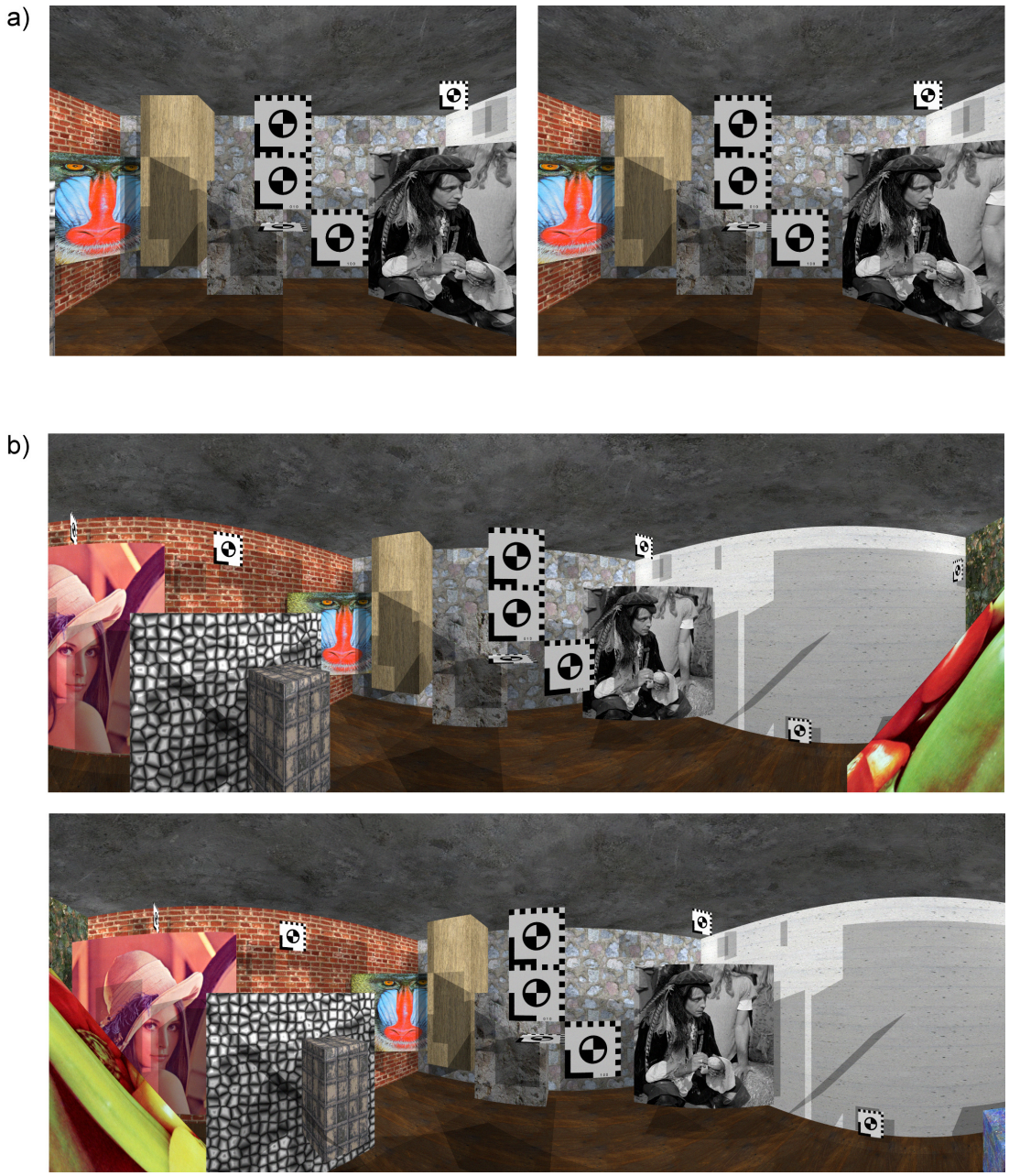


Abbildung 5.1.: (a) Erstes (links) und zweites Bild (rechts) des planaren Stereopaars 1 (b) Erstes Bild (oben) und zweites Bild (unten) des zylindrischen Stereopaars 2

5.1.1.1. Projektionsarten, Speichermethoden und Zuordnungsalgorithmen

Für die zylindrische und die planare Projektionsart werden zunächst die Arbeitsmodi des Matchers in Kombination mit den RAM-basierten und auslagernden Speichermethoden für die 2D- und 3D-Informationen eruiert. Dabei soll das zeitliche Verhalten im Vordergrund stehen. Als Eingabedaten dienen das fein aufgelöste Modell B und die beiden Stereobildpaare. Die Einstellungen für die Objektpunktschätzung mittels Strahlverfolgung wie auch die Korrespondenzanalyse sind konstant auf die Werte in Tabelle 5.3 fixiert und in entsprechenden Konfigurationsobjekten für den funktionalen Algorithmus abgelegt.

Objektpunktschätzung	
Level des 3D-Baums	5
Initiale Kantenlängen der Baumquader	8 x 8 x 8 LE
Unterteilungsfaktor	2 x 2 x 2
Kleinste Boxabmessungen	0,5 x 0,5 x 0,5 LE
Strahlverfolgungsmethode	rekursiv
Korrespondenzanalyse	
Region of Interest (ROI)	gesamtes Bild
Qualitätsschwellwert	1,0
Integer-Algorithmus	
Ähnlichkeitsfunktion	SAD
Verwendeter Farbkanal	RGB, perzeptuell gewichtet
Suchbereichsgröße	9 x 9 Pixel
Korrelationsfenstergröße	17 x 17 Pixel
Korrelationsschrittweite	1 x 1 Pixel
effektive Suchschrittweite	1 x 1 Pixel
Geman-McClure-Parameter σ	5,0
Gleitkomma-Algorithmus	
-	

Tabelle 5.3.: Parameter des integrierten Matchingalgorithmus für die Eruierung der Projektionsarten, Speichermethoden und Zuordnungsalgorithmen

Aufgrund der verwendeten effektiven Suchschrittweite (Korrelationsschrittweite dividiert durch den Verfeinerungsfaktor, vgl. Abschnitt 4.6.1.2) in beiden Bildrichtungen werden die Zuordnungen auf ein Pixel genau mit der SAD-Abstandsfunktion bestimmt. Eine Unterabtastung der Muster- und gleich großen Suchfenster bei der inhaltlichen Gegenüberstellung findet nicht statt. Der Geman-McClure-Parameter σ, der die Dämpfung des Einflusses von Ausreißern während der Kalkulation des jeweiligen Übereinstimmungsgrades festlegt, ist empirisch gewählt und beträgt $5,0$. Er gewährleistet eine signifikante Abschwächung hoher SAD-Differenzbeträge, welche für die generierten Aufnahmen stets im Bereich von 0 ... 255 liegen. Schlussendlich deaktiviert der relative Schwellwert $1,0$ die qualitative Selektion der Zuordnungen. Alle Korrespondenzen werden in den Disparitätskarten des funktionalen Algorithmus berücksichtigt.

Unter diesen Voraussetzungen ergeben sich für die Tests 5.1.1.1-a bis 5.1.1.1-e die Messdaten in den Tabellen 5.4 bis 5.8. Aus Platzgründen werden die aufgeführten statistischen

Werte des Matchers abgekürzt. Die Reihenfolge und Numerierung in der nachstehenden Übersetzungstabelle ist an die Aufzählung aus Abschnitt 4.7.3.3 angelehnt. Es bezeichnen

(1) *TreeClsDuration* den Zeitbedarf für die Klassifikation der Modelldreiecke mit der 3D-Baumstruktur (tree classification duration)

(2) *MatchDuration* den Zeitbedarf für die Korrespondenzanalyse einschließlich der qualitativen Selektion der Zuordnungen (matching duration),

(3) *BestPosSim* und *WorstPosSim* den für die aktuelle Matcherkonfiguration konstanten minimal und maximal möglichen Abstandswert,

(4) *AvgSimBefQC* und *AvgSimAfterQC* den durchschnittlichen Ähnlichkeitsgrad der Zuordnungen vor und nach der qualitativen Selektion,

(5) *AvgDispErrBefQC* und *AvgDispErrAftQC* den mittleren Disparitätsfehler zur „wahren“ Punktverschiebung des geschätzten Suchbereichszentrums vor und nach der Qualitätsprüfung,

(6) *DispErrSDvBefQC* und *DispErrSDvAftQC* die Standardabweichung des mittleren Disparitätsfehlers zur „wahren“ Punktverschiebung des designierten Suchbereichszentrums vor und nach der qualitativen Zuordnungsselektion,

(7) *RayHits1* und *RayHits2* die Anzahl der Strahltreffer mit dem Modell, ausgehend vom Projektionszentrum der ersten und zweiten Kamera,

(8) *MAMatchingSteps* die Anzahl der lokalen Matchingschritte mit Suchbereichseingrenzung durch das 3D-Modell,

(9) *EpiMatchingSteps* die Anzahl der lokalen Matchingschritte, die eine Suche entlang der Epipolarlinie beinhalten,

(10) *QualityMatches* die Anzahl der qualitativ akzeptablen Zuordnungen und

(11) *MatchingErrors* die Anzahl der Matchingfehler .

Test 5.1.1.1-a Im ersten Test wird die Korrespondenzanalyse mit aktivierter Objektpunktschätzung und eingeschalteter Suche entlang der Epipolarlinie durchgeführt. Weil das künstliche Modell rundum geschlossen ist, darf das letztgenannte Verfahren dabei allerdings nicht zur Anwendung kommen. Beide Stereobildpaare liegen jeweils komplett als `CRamImage`-Instanz im Hauptspeicher vor. Ebenso wird die 3D-Szene ohne Beteiligung der STXXL-Bibliothek vollständig im RAM abgelegt.

Test 5.1.1.1-b Die Objektpunktschätzung bleibt für diesen Test deaktiviert, weswegen die Korrespondenzen in der zweiten Stereoaufnahme ausschließlich entlang der Epipolarlinie, die mit jedem Musterbildpixel assoziiert ist, gesucht werden. Die Bilddaten und das 3D-Modell befinden sich wiederum vollständig im RAM.

Test 5.1.1.1-c Wie in Test 5.1.1.1-a sind der modellgestützte und der epipolare Matchingmodus für Untersuchung 5.1.1.1-c aktiviert. Die Stereobilder befinden sich vollständig

im RAM, das 3D-Modell wird mit Hilfe der STXXL-Container gespeichert. Dabei befinden sich mit den in der DaTool-Software verwendeten Einstellungen maximal 128 Kibibyte an Raumpunktdaten, 128 Kibibyte an Polygonindizes und 128 Kibibyte an Texturkoordinaten im Hauptspeicher, die sich auf jeweils acht Seiten zu vier Blöcken à 4096 Byte verteilen (vgl. die STXXL-Dokumentation [35]).

Test 5.1.1.1-d Im vierten Test dieses Abschnitts sind die Objektpunktschätzung und die Zuordnungssuche entlang der Epipolarlinie aktiviert. Die Stereobilder werden mit Instanzen der auslagernden `CGridImage`-Klasse verwaltet, wobei sich höchstens 192 Kibibyte in vier dedizierten Blöcken à 128 x 128 RGB-Pixel im Hauptspeicher befinden. Das 3D-Modell liegt vollständig im RAM vor.

Test 5.1.1.1-e Test 5.1.1.1-e unterscheidet sich von seinem Vorgänger lediglich darin, dass die Stereoaufnahmen mit Instanzen der `CFastGridImage`-Klasse gespeichert werden, die bei der Berechnung der ein- und auszulagernden Bildkacheln ohne Divisionsoperationen auskommt.

Stereopaar 1 (planar, 1.024 x 768 Pixel)

TreeClsDuration	MatchDuration	BestPosSim	WorstPosSim
19 s	1.229 s	0	289

AvgSimBefQC	AvgSimAfterQC	AvgDispErrBefQC	AvgDispErrAftQC
99,9478	99,9478	0,422068	0,422068

DispErrSDvBefQC	DispErrSDvAftQC	RayHits1	RayHits2
0,752941	0,752941	761.244	734.795

MAMatchingSteps	EpiMatchingSteps	QualityMatches	MatchingErrors
734.795	0	734.795	0

Stereopaar 2 (zylindrisch, 1.024 x 384 Pixel)

TreeClsDuration	MatchDuration	BestPosSim	WorstPosSim
19 s	666 s	0	289

AvgSimBefQC	AvgSimAfterQC	AvgDispErrBefQC	AvgDispErrAftQC
143,658	143,658	0,5819	0,5819

DispErrSDvBefQC	DispErrSDvAftQC	RayHits1	RayHits2
0,920602	0,920602	362.057	322.299

MAMatchingSteps	EpiMatchingSteps	QualityMatches	MatchingErrors
322.299	0	322.299	0

Tabelle 5.4.: Messdaten für Test 5.1.1.1-a: Beide Matchingmodi aktiviert, Bild- und Modelldaten vollständig im RAM

Stereopaar 1 (planar, 1.024 x 768 Pixel)

TreeClsDuration	MatchDuration	BestPosSim	WorstPosSim
0 s	61.201 s	0	289

AvgSimBefQC	AvgSimAfterQC	AvgDispErrBefQC	AvgDispErrAftQC
104,541	104,541	0	0

DispErrSDvBefQC	DispErrSDvAftQC	RayHits1	RayHits2
0	0	0	0

MAMatchingSteps	EpiMatchingSteps	QualityMatches	MatchingErrors
0	784.384	784.384	0

Stereopaar 2 (zylindrisch, 1.024 x 384 Pixel)

TreeClsDuration	MatchDuration	BestPosSim	WorstPosSim
0 s	28.011 s	0	289

AvgSimBefQC	AvgSimAfterQC	AvgDispErrBefQC	AvgDispErrAftQC
149,997	149,997	0	0

DispErrSDvBefQC	DispErrSDvAftQC	RayHits1	RayHits2
0	0	0	0

MAMatchingSteps	EpiMatchingSteps	QualityMatches	MatchingErrors
0	393.196	393.196	0

Tabelle 5.5.: Messdaten für Test 5.1.1.1-b: Matching entlang der Epipolarlinien, Bild- und Modelldaten vollständig im RAM

Stereopaar 1 (planar, 1.024 x 768 Pixel)

TreeClsDuration	MatchDuration	BestPosSim	WorstPosSim
202 s	13.924 s	0	289

AvgSimBefQC	AvgSimAfterQC	AvgDispErrBefQC	AvgDispErrAftQC
99,9478	99,9478	0,422068	0,422068

DispErrSDvBefQC	DispErrSDvAftQC	RayHits1	RayHits2
0,752941	0,752941	761.244	734.795

MAMatchingSteps	EpiMatchingSteps	QualityMatches	MatchingErrors
734.795	0	734.795	0

Stereopaar 2 (zylindrisch, 1.024 x 384 Pixel)

TreeClsDuration	MatchDuration	BestPosSim	WorstPosSim
201 s	10.758 s	0	289

AvgSimBefQC	AvgSimAfterQC	AvgDispErrBefQC	AvgDispErrAftQC
143,658	143,658	0,5819	0,5819

DispErrSDvBefQC	DispErrSDvAftQC	RayHits1	RayHits2
0,920602	0,920602	362.057	322.299

MAMatchingSteps	EpiMatchingSteps	QualityMatches	MatchingErrors
322.299	0	322.299	0

Tabelle 5.6.: Messdaten für Test 5.1.1.1-c: Beide Matchingmodi aktiviert, Bilddaten vollständig im RAM, Modelldaten in STXXL-Containern

Stereopaar 1 (planar, 1.024 x 768 Pixel)

TreeClsDuration	MatchDuration	BestPosSim	WorstPosSim
19 s	1.283 s	0	289

AvgSimBefQC	AvgSimAfterQC	AvgDispErrBefQC	AvgDispErrAftQC
99,9478	99,9478	0,422068	0,422068

DispErrSDvBefQC	DispErrSDvAftQC	RayHits1	RayHits2
0,752941	0,752941	761.244	734.795

MAMatchingSteps	EpiMatchingSteps	QualityMatches	MatchingErrors
734.795	0	734.795	0

Stereopaar 2 (zylindrisch, 1.024 x 384 Pixel)

TreeClsDuration	MatchDuration	BestPosSim	WorstPosSim
20 s	709 s	0	289

AvgSimBefQC	AvgSimAfterQC	AvgDispErrBefQC	AvgDispErrAftQC
143,658	143,658	0,5819	0,5819

DispErrSDvBefQC	DispErrSDvAftQC	RayHits1	RayHits2
0,920602	0,920602	362.057	322.299

MAMatchingSteps	EpiMatchingSteps	QualityMatches	MatchingErrors
322.299	0	322.299	0

Tabelle 5.7.: Messdaten für Test 5.1.1.1-d: Beide Matchingmodi aktiviert, Bilddaten in `CGridImage`-Instanzen abgelegt, Modelldaten vollständig im RAM

Stereopaar 1 (planar, 1.024 x 768 Pixel)

TreeClsDuration	MatchDuration	BestPosSim	WorstPosSim
19 s	1.249 s	0	289

AvgSimBefQC	AvgSimAfterQC	AvgDispErrBefQC	AvgDispErrAftQC
99,9478	99,9478	0,422068	0,422068

DispErrSDvBefQC	DispErrSDvAftQC	RayHits1	RayHits2
0,752941	0,752941	761.244	734.795

MAMatchingSteps	EpiMatchingSteps	QualityMatches	MatchingErrors
734.795	0	734.795	0

Stereopaar 2 (zylindrisch, 1.024 x 384 Pixel)

TreeClsDuration	MatchDuration	BestPosSim	WorstPosSim
19 s	684 s	0	289

AvgSimBefQC	AvgSimAfterQC	AvgDispErrBefQC	AvgDispErrAftQC
143,658	143,658	0,5819	0,5819

DispErrSDvBefQC	DispErrSDvAftQC	RayHits1	RayHits2
0,920602	0,920602	362.057	322.299

MAMatchingSteps	EpiMatchingSteps	QualityMatches	MatchingErrors
322.299	0	322.299	0

Tabelle 5.8.: Messdaten für Test 5.1.1.1-e: Beide Matchingmodi aktiviert, Bilddaten in `CFastGridImage`-Instanzen abgelegt, Modelldaten vollständig im RAM

Wie die Daten der tabellierten Ausführungsprotokolle belegen, führt die Korrespondenzanalyse auf den zylindrischen Stereoaufnahmen durchweg zu rund 45% höheren durchschnittlichen Abstandswerten gegenüber den planaren Bildern. Dieser Umstand ist zum einen auf die Verzerrung infolge der Projektion und der Kameraneigung zurückzuführen, an welche die einheitlich geformten quadratischen Suchfenster nicht hinreichend angepasst sind. Sie beinhalten dadurch einen größeren Anteil an Pixeln, die im aktuellen Musterfenster nicht auftreten. Zum anderen differieren die mittleren Helligkeiten zwischen den Bildpaaren, die sich direkt in den SAD-Beträgen niederschlagen. Bezogen auf die in Graustufen umgewandelten farbigen Projektionen belaufen sich diese auf 89,0 und 90,1 (planares Muster- und Suchbild) bzw. 115,6 und 111,9 (zylindrisches Muster- und Suchbild) Intensitätsstufen – eine Zunahme um etwa ein Viertel.

Die Klassifikationszeiten für das feine Modell, das mehr als eine Million Oberflächendreiecke aufweist, liegen im unteren zweistelligen Sekundenbereich, sieht man einmal von Test 5.1.1.1-c ab. Hier bewirkt die Speicherung der Szene in den auslagernden STXXL-Containern eine Verzehnfachung der Rechendauer, die sich auf den rekursiven Baumabstieg bei der Objektpunktschätzung durch die damit verbundenen Zugriffsoperationen auf die Polygone in den Boxen überträgt. In beiden Fällen ist dieses Laufzeitverhalten wohl der geringen Lokalität der Lesevorgänge auf den 3D-Daten geschuldet, die praktisch quasizufällig ausgeführt werden und die LED für die Festplattenaktivität des Testrechners permanent aufleuchten lassen. Der Gewinn an verfügbarem Hauptspeicher, der sich aus den Angaben des Windows-Taskmanagers während der Ausführung der DaTool-Applikation mit und ohne Unterstützung für umfangreiche Modelldaten ergibt, beträgt beim ersten Stereopaar 134.132 KiB - 116.444 KiB = 17.688 KiB und beim zweiten Stereopaar 124.596 KiB - 103.320 KiB = 21.276 KiB. Die Unterschiedsbeträge fallen für die Szene, die nach dem Laden ungefähr 60 MiB im RAM beansprucht, wegen des Überhangs durch den komplexen Swappingmechanismus der alternativen Datencontainer recht bescheiden aus. Der Einsatz der STXXL-Bibliothek rentiert sich bei dieser vergleichsweise geringen Informationsmenge noch nicht.

Die übrigen Testläufe weisen Matchingzeiten von ca. 21 (planar) bzw. 11 (zylindrisch) Minuten für die knapp 787.000 bzw. 400.000 Bildpunkte auf, sofern die Zuordnungsbestimmung modellgestützt erfolgt. Dies entspricht einem Durchsatz von etwa 640 bzw. 600 Pixeln/s, wonach die komplexere Projektion auf den Zylindermantel (Arkustangens-Funktion) um 7% langsamer arbeitet als die Rechenvorschrift für die ebene Abbildung. Die Verwendung der Bildspeicherklassen, die die zweidimensionalen Daten partiell auf dem Massenspeicher belassen, führt zu einem vergleichsweise geringen Zeitaufschlag von weniger als einer Minute. Bemerkenswert ist hierbei, dass sich die Substitution der Divisionsoperationen bei der Berechnung der Kachel mit dem aktuell in Bearbeitung befindlichen Pixel in einer um 3% gesteigerten Ausführungsgeschwindigkeit niederschlägt.

Der Speicherplatzgewinn bei Verwendung der `CGridImage`- und `CFastGridImage`-Klassen gegenüber der Datenhaltung im RAM mit `CRamImage` beläuft sich für die geeignet übersetzten DaTool-Varianten auf 134.132 KiB - 118.934 KiB = 15.198 KiB bzw. 134.132 KiB - 119.608 KiB = 14.524 KiB (Stereopaar 1) und auf 124.596 KiB - 116.652 KiB = 7.944 KiB bzw. 124.596 KiB - 118.060 KiB = 6.536 KiB (Stereopaar 2), wobei neben dem eingangs übergebenen Bildmaterial auch die Gleitkomma-Disparitätskarten (64 Bit pro Pixel) in die Differenzen einfließen. Betrachtet man den RAM-Bedarf der Rohdaten von allen unmittelbar beim Matching verwendeten oder erzeugten Bildern von 16.896 KiB (planar) respektive 8.448 KiB (zylindrisch), also ohne das im Nachhinein erzeugte Graustufen-Übersichtsbild,

so ist die Ersparnis trotz der geringen Informationsmengen und der Ungenauigkeiten des für die Messung verwendeten Windows-Taskmanagers signifikant.

Ohne die Objektpunktschätzung vervielfacht sich die für die Korrespondenzanalyse benötigte Rechendauer auf mehr als siebzehn bzw. knapp acht Stunden für die planaren bzw. zylindrischen Stereoaufnahmen. Der Quotient der Zeitangaben ist hierbei größer als das Verhältnis aus den zugehörigen Bildpixelzahlen von exakt $(1.024 \cdot 768) : (1.024 \cdot 384) = 2$. Dieser Umstand lässt darauf schließen, dass die mit der Zylinderprojektion verbundenen sinusförmigen Epipolarlinien häufiger außerhalb der zweiten Stereoaufnahme verlaufen als im planaren Fall und folglich weniger Matchingschritte ausgeführt werden können. Qualitativ führt das höhere Mehrdeutigkeitspotential der epipolaren Suche nach inhaltlichen Übereinstimmungen zwischen den Bildern zu schlechteren Ergebnissen, die sich in um 5% erhöhten SAD-Durchschnittswerten äußern.

Abbildung 5.2 zeigt die Graustufen-Übersichtsbilder nach der Anwendung des integrierten Matchingalgorithmus. Bei den Tests mit aktivierter Objektpunktschätzung (linke Hälfte) ist die Binärrepräsentation der Ausgaben jeweils identisch. Im planaren Übersichtsbild, das aus den Disparitäten eines Stereosystems mit virtuellen Aufnahmegeräten ohne Neigung zueinander resultiert, nehmen die Beträge der Verschiebungen und damit auch die Helligkeit mit steigender Distanz zum Standort der ersten Kamera ab. Einseitig sichtbare Bildbereiche sind schwarz dargestellt und fallen mit der Nulldisparität zusammen, die allerdings in der Praxis selten auftritt. Die Intensitäten der zylindrischen Übersichtsbilder weisen eine andere Verteilung auf, da die virtuellen Kameras im zugehörigen Stereosystem leicht zueinander verdreht sind. Der Schnittpunkt der Hauptpunktstrahlen, die für die Konfiguration aus Tabelle 5.2 mit den optischen Achsen zusammenfallen, liegt dabei im Raumpunkt (0; 1; 2) und somit fast mittig in der Szene. In diesem Punkt sind die Disparitäten und damit auch die Bildintensitäten minimal, während der Vorder- und Hintergrund jeweils eine größere Helligkeit aufweisen. Auch in den zylindrischen Projektionen sind diejenigen Bildbereiche, die ausschließlich von der ersten Kamera abgebildet werden, schwarz gefärbt.

Die Konturen der Szene sind in den inhaltlich gleichen Übersichtsbildern, welche sich aus den Testläufen mit aktivierter Objektpunktschätzung ergeben, deutlich zu erkennen. Durch die Pixelauflösung des Matchers entsteht in der geneigten Ebene rechts ein Stufenmuster. Außerdem sind kleinere Fehlzuordnungen, etwa im Bildzentrum oder an der Außenkante der länglichen vertikalen wood box auszumachen. Die aus der Zuordnungssuche entlang der Epipolarlinien resultierenden Graustufenbilder sind durch die stellenweise größeren Disparitäten normalisierungsbedingt dunkler und weniger kontrastreich. Daneben treten in den einseitig von der ersten Stereokamera aufgenommenen Bildteilen aufgrund der abgestellten Qualitätsprüfung offenkundig fehlerhafte Korrespondenzen zutage.

5.1.1.2. Struktur des 3D-Baums und Strahlverfolgung

Die nächste Testserie untersucht den Einfluss der Struktur des 3D-Baums auf die Geschwindigkeit der Dreiecksklassifikation und die Raytracingverfahren beim modellgestützten Matching. Dabei steht im Vordergrund, welcher Beschleunigungseffekt mit flachen bzw. tiefen Quaderhierarchien und verschiedenen Unterteilungsschemata einhergeht, und wie sich die rekursive und inkrementelle Strahlverfolgung zeitlich verhalten.

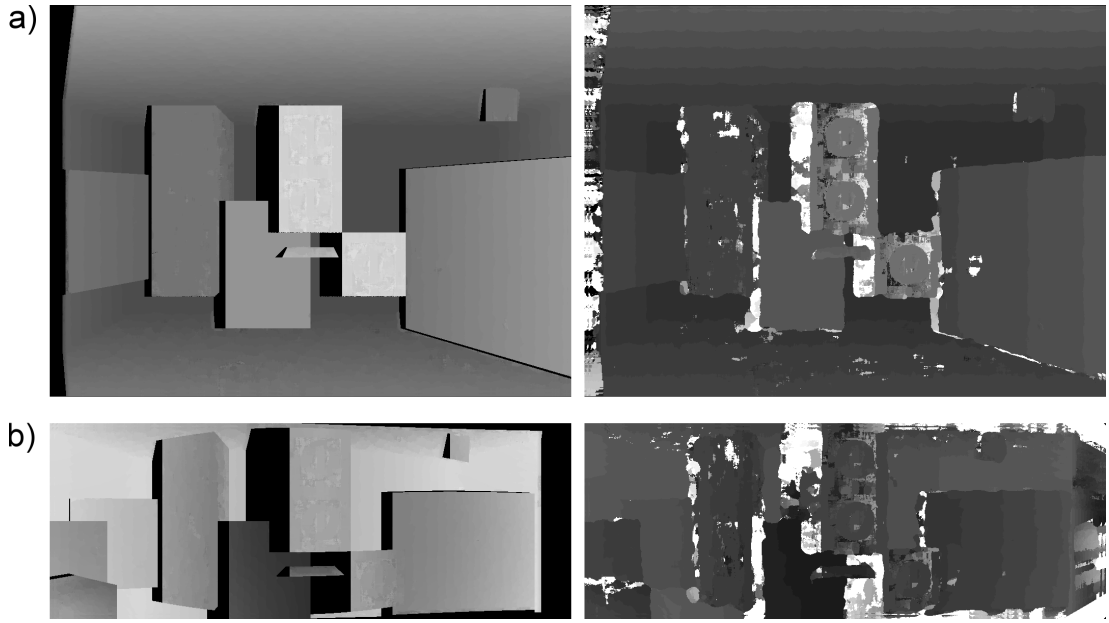


Abbildung 5.2.: (a) Identisches planares Graustufen-Übersichtsbild der Testläufe 5.1.1.1-a sowie 5.1.1.1-c bis 5.1.1.1-e (links) und planares Übersichtsbild der epipolaren Zuordnungssuche mit Test 5.1.1.1-b (rechts) (b) Zylindrische Übersichtsbilder für die Tests 5.1.1.1-a sowie 5.1.1.1-c bis 5.1.1.1-e (links) und 5.1.1.1-b (rechts)

Zur Beantwortung der Fragen werden die Anzahl der Level in der räumlichen Hierarchie als auch die initiale Boxgröße des 3D-Baums variiert. Die rekursive und inkrementelle Raytracingmethode kommen für das Matching mit aktivierter Objektpunktschätzung auf den künstlichen Szenen A und B zum Einsatz. Weil die Zuordnungsqualität bei den Tests 5.1.1.2-a bis 5.1.1.2-j weniger relevant ist, wird die Korrespondenzanalyse pixelgenau mit den Parametern aus Tabelle 5.3 durchgeführt, die während der gesamten Untersuchungsreihe konstant bleiben. Ebenso lässt die verwendete Projektionsvorschrift keine Auswirkungen auf die Strahlverfolgung erwarten, weswegen sich die Ergebnisse der Tests in den Tabellen 5.9 bis 5.18 stets auf das planare Stereobildpaar beziehen. Die ermittelten Messwerte entstammen den Ausführungsprotokollen des Matchers, die Abkürzungen für die Messgrößen sind in der vorangegangenen Sektion 5.1.1.1 erklärt.

Test 5.1.1.2-a Für Test 5.1.1.2-a werden beide 3D-Modelle mit einer Hierarchie von insgesamt sieben Ebenen unterteilt, wobei jedes der würfelförmigen Baumelemente aus acht Subwürfeln besteht (Oktalbaumstruktur). Die initiale Kantenlänge beträgt acht Einheiten im Weltkoordinatensystem, wonach die kleinsten Boxen Abmessungen von 0,125 LE in allen drei Raumdimensionen aufweisen. Dies entspricht dem zweieinhalbfachen (Modell A) bzw. zwölfeinhalbfachen (Modell B) minimalen Punktabstand, der nach Konstruktion der synthetischen Szene die kleinstmögliche Seitenlänge der Polygone definiert. Jeder nichtleere Würfel auf der tiefsten Baumebene enthält demnach höchstens ungefähr $2 \cdot (2,5-1)^2 \approx 5$ (Modell A) bzw. $2 \cdot (12,5-1)^2 \approx 265$ (Modell B) Dreiecke, wenn er von einer Fläche

des Modells parallel zu einer Ebene des Weltkoordinatensystems („gerade“) geschnitten wird[17]. Das Raytracing erfolgt rekursiv.

Test 5.1.1.2-b Test 5.1.1.2-b wird mit den Einstellungen von Test 5.1.1.2-a durchgeführt, das Raytracing erfolgt jedoch inkrementell.

Test 5.1.1.2-c Für Test 5.1.1.2-c werden die Oberflächenpolygone von Modell A und B mit einer fünfstufigen Hierarchie klassifiziert. Die Boxen des nächsttieferen Levels bestehen dabei aus jeweils $3^3 = 27$ statt wie bisher acht Würfeln bei einer initialen Kantenlänge von acht Einheiten. Auf der letzten Ebene besitzen die Baumelemente demnach Seiten zu je 0,0988 LE und enthalten, wenn sie von einer Modellviereckfläche parallel zu einer Ebene des Weltkoordinatensystems geschnitten werden, aufgerundet höchstens zwei Dreiecke (Modell A) bzw. 158 Triangeln (Modell B). Das Raytracing erfolgt rekursiv.

Test 5.1.1.2-d Der vierte Test der zweiten Serie wird, abgesehen von der inkrementellen Strahlverfolgung, mit der Konfiguration seines Vorgängers absolviert.

Test 5.1.1.2-e Die Oberflächenpolygone des groben und fein aufgelösten 3D-Modells A bzw. B werden in Test 5.1.1.2-e durch eine flache Oktalbaumhierarchie mit vier Ebenen unterteilt. Die initiale Kantenlänge der verwendeten Würfel beträgt wiederum acht LE, die kleinsten Würfel des letzten Levels messen daher eine Längeneinheit. Jedes Baumelement auf dem untersten Level enthält daher bei einem „geraden“ Schnitt mit einer Modellfläche höchstens 722 Dreiecke (Modell A) bzw. 19.602 Polygone (Modell B). Das Raytracing erfolgt rekursiv.

Test 5.1.1.2-f Testkonfiguration 2f entspricht Testkonfiguration 2e, einzig die Objektpunktschätzung erfolgt inkrementell.

Test 5.1.1.2-g Die Polygone der beiden LRF-Modelle werden im Testlauf 5.1.1.2-g mit Hilfe einer tiefen neunstufigen Oktalbaumhierarchie klassifiziert, die initiale Boxgröße beträgt acht LE pro Raumrichtung und die Kantenlänge der kleinsten Würfel des letzten Levels beläuft sich auf 0,03125 LE. Die Quader der untersten Ebene enthalten bei einem „geraden“ Schnitt mit einer Viereckfläche aufgerundet höchstens ein Polygon (Modell A) bzw. neun Dreiecke (Modell B). Das Raytracing erfolgt rekursiv.

Test 5.1.1.2-h Test 5.1.1.2-h wird mit dem Parametersatz von Test 5.1.1.2-g ausgeführt, für die Strahlverfolgung kommt jedoch der inkrementelle Algorithmus zum Einsatz.

[17] Ein Raster aus $n \times m$ Raumpunkten wird nach Konstruktion der künstlichen Szene mit $2(m-1)(n-1)$ Dreiecken regulär vermascht. Im Generatorplugin des DaTool-Prototyps ist es dabei bislang nicht möglich, die Werte m und n getrennt festzulegen. Vielmehr gilt stets m=n.

Test 5.1.1.2-i Der vorletzte Test dieser Serie benutzt ein flaches Oktalbaum-Klassifikationsschema „von unten“ mit vier Ebenen und einer initialen Boxgröße von zwei LE pro Raumdimension. Die Würfel der untersten Ebene haben die gleichen Abmessungen wie bei Test 5.1.1.2-a/b. Das Raytracing erfolgt rekursiv.

Test 5.1.1.2-j Test 5.1.1.2-j basiert auf den Einstellungen von Test 5.1.1.2-i, verwendet allerdings die inkrementelle Strahlverfolgungsmethode.

Modell A (37.544 Polygone)

TreeClsDuration	MatchDuration	BestPosSim	WorstPosSim
3 s	1.355 s	0	289

AvgSimBefQC	AvgSimAfterQC	AvgDispErrBefQC	AvgDispErrAftQC
99,9468	99,9468	0,422123	0,422123

DispErrSDvBefQC	DispErrSDvAftQC	RayHits1	RayHits2
0,753103	0,753103	761.244	734.792

MAMatchingSteps	EpiMatchingSteps	QualityMatches	MatchingErrors
734.792	0	734.792	0

Modell B (1.019.304 Polygone)

TreeClsDuration	MatchDuration	BestPosSim	WorstPosSim
41 s	1.396 s	0	289

AvgSimBefQC	AvgSimAfterQC	AvgDispErrBefQC	AvgDispErrAftQC
99,9478	99,9478	0,422068	0,422068

DispErrSDvBefQC	DispErrSDvAftQC	RayHits1	RayHits2
0,752941	0,752941	761.244	734.795

MAMatchingSteps	EpiMatchingSteps	QualityMatches	MatchingErrors
734.795	0	734.795	0

Tabelle 5.9.: Messdaten für Test 5.1.1.2-a: 7 Oktalbaumlevel, initiale Würfel mit 8 LE Kantenlänge, rekursives Raytracing

Modell A (37.544 Polygone)

TreeClsDuration	MatchDuration	BestPosSim	WorstPosSim
3 s	302 s	0	289

AvgSimBefQC	AvgSimAfterQC	AvgDispErrBefQC	AvgDispErrAftQC
99,9473	99,9473	0,422126	0,422126

DispErrSDvBefQC	DispErrSDvAftQC	RayHits1	RayHits2
0,753103	0,753103	761.244	734.796

MAMatchingSteps	EpiMatchingSteps	QualityMatches	MatchingErrors
734.796	0	734.796	0

Modell B (1.019.304 Polygone)

TreeClsDuration	MatchDuration	BestPosSim	WorstPosSim
42 s	333 s	0	289

AvgSimBefQC	AvgSimAfterQC	AvgDispErrBefQC	AvgDispErrAftQC
99,9482	99,9482	0,422071	0,422071

DispErrSDvBefQC	DispErrSDvAftQC	RayHits1	RayHits2
0,752941	0,752941	761.244	734.799

MAMatchingSteps	EpiMatchingSteps	QualityMatches	MatchingErrors
734.799	0	734.799	0

Tabelle 5.10.: Messdaten für Test 5.1.1.2-b: 7 Oktalbaumlevel, initiale Würfel mit 8 LE Kantenlänge, inkrementelles Raytracing

Modell A (37.544 Polygone)

TreeClsDuration	MatchDuration	BestPosSim	WorstPosSim
3 s	1.983 s	0	289

AvgSimBefQC	AvgSimAfterQC	AvgDispErrBefQC	AvgDispErrAftQC
99,947	99,947	0,422158	0,422158

DispErrSDvBefQC	DispErrSDvAftQC	RayHits1	RayHits2
0,753171	0,753171	761.244	734.790

MAMatchingSteps	EpiMatchingSteps	QualityMatches	MatchingErrors
734.790	0	734.790	0

Modell B (1.019.304 Polygone)

TreeClsDuration	MatchDuration	BestPosSim	WorstPosSim
52 s	1.985 s	0	289

AvgSimBefQC	AvgSimAfterQC	AvgDispErrBefQC	AvgDispErrAftQC
99,9479	99,9479	0,422103	0,422103

DispErrSDvBefQC	DispErrSDvAftQC	RayHits1	RayHits2
0,753009	0,753009	761.244	734.793

MAMatchingSteps	EpiMatchingSteps	QualityMatches	MatchingErrors
734.793	0	734.793	0

Tabelle 5.11.: Messdaten für Test 5.1.1.2-c: 5 Baumlevel, jeweils 27 Subwürfel, initiale Boxen mit 8 LE Kantenlänge, rekursives Raytracing

Modell A (37.544 Polygone)

TreeClsDuration	MatchDuration	BestPosSim	WorstPosSim
4 s	305 s	0	289

AvgSimBefQC	AvgSimAfterQC	AvgDispErrBefQC	AvgDispErrAftQC
99,9473	99,9473	0,422126	0,422126

DispErrSDvBefQC	DispErrSDvAftQC	RayHits1	RayHits2
0,753103	0,753103	761.244	734.796

MAMatchingSteps	EpiMatchingSteps	QualityMatches	MatchingErrors
734.796	0	734.796	0

Modell B (1.019.304 Polygone)

TreeClsDuration	MatchDuration	BestPosSim	WorstPosSim
50 s	315 s	0	289

AvgSimBefQC	AvgSimAfterQC	AvgDispErrBefQC	AvgDispErrAftQC
99,9482	99,9482	0,422071	0,422071

DispErrSDvBefQC	DispErrSDvAftQC	RayHits1	RayHits2
0,752941	0,752941	761.244	734.799

MAMatchingSteps	EpiMatchingSteps	QualityMatches	MatchingErrors
734.799	0	734.799	0

Tabelle 5.12.: Messdaten für Test 5.1.1.2-d: 5 Baumlevel, jeweils 27 Subwürfel, initiale Boxen mit 8 LE Kantenlänge, inkrementelles Raytracing

Modell A (37.544 Polygone)

TreeClsDuration	MatchDuration	BestPosSim	WorstPosSim
1 s	656 s	0	289

AvgSimBefQC	AvgSimAfterQC	AvgDispErrBefQC	AvgDispErrAftQC
99,9468	99,9468	0,422123	0,422123

DispErrSDvBefQC	DispErrSDvAftQC	RayHits1	RayHits2
0,753103	0,753103	761.244	734.792

MAMatchingSteps	EpiMatchingSteps	QualityMatches	MatchingErrors
734.792	0	734.792	0

Modell B (1.019.304 Polygone)

TreeClsDuration	MatchDuration	BestPosSim	WorstPosSim
14 s	1.693 s	0	289

AvgSimBefQC	AvgSimAfterQC	AvgDispErrBefQC	AvgDispErrAftQC
99,9478	99,9478	0,422068	0,422068

DispErrSDvBefQC	DispErrSDvAftQC	RayHits1	RayHits2
0,752941	0,752941	761.244	734.795

MAMatchingSteps	EpiMatchingSteps	QualityMatches	MatchingErrors
734.795	0	734.795	0

Tabelle 5.13.: Messdaten für Test 5.1.1.2-e: 4 Oktalbaumlevel, initiale Würfel mit 8 LE Kantenlänge, rekursives Raytracing

Modell A (37.544 Polygone)

TreeClsDuration	MatchDuration	BestPosSim	WorstPosSim
1 s	365 s	0	289

AvgSimBefQC	AvgSimAfterQC	AvgDispErrBefQC	AvgDispErrAftQC
99,9473	99,9473	0,422126	0,422126

DispErrSDvBefQC	DispErrSDvAftQC	RayHits1	RayHits2
0,753103	0,753103	761.244	734.796

MAMatchingSteps	EpiMatchingSteps	QualityMatches	MatchingErrors
734.796	0	734.796	0

Modell B (1.019.304 Polygone)

TreeClsDuration	MatchDuration	BestPosSim	WorstPosSim
14 s	2.797 s	0	289

AvgSimBefQC	AvgSimAfterQC	AvgDispErrBefQC	AvgDispErrAftQC
99,9482	99,9482	0,422071	0,422071

DispErrSDvBefQC	DispErrSDvAftQC	RayHits1	RayHits2
0,752941	0,752941	761.244	734.799

MAMatchingSteps	EpiMatchingSteps	QualityMatches	MatchingErrors
734.799	0	734.799	0

Tabelle 5.14.: Messdaten für Test 5.1.1.2-f: 4 Oktalbaumlevel, initiale Würfel mit 8 LE Kantenlänge, inkrementelles Raytracing

Modell A (37.544 Polygone)

TreeClsDuration	MatchDuration	BestPosSim	WorstPosSim
32 s	4.417 s	0	289

AvgSimBefQC	AvgSimAfterQC	AvgDispErrBefQC	AvgDispErrAftQC
99,9494	99,9494	0,422098	0,422098

DispErrSDvBefQC	DispErrSDvAftQC	RayHits1	RayHits2
0,753121	0,753121	761.244	734.710

MAMatchingSteps	EpiMatchingSteps	QualityMatches	MatchingErrors
734.710	0	734.710	0

Modell B (1.019.304 Polygone)

TreeClsDuration	MatchDuration	BestPosSim	WorstPosSim
137 s	4.391 s	0	289

AvgSimBefQC	AvgSimAfterQC	AvgDispErrBefQC	AvgDispErrAftQC
99,9502	99,9502	0,422043	0,422043

DispErrSDvBefQC	DispErrSDvAftQC	RayHits1	RayHits2
0,75296	0,75296	761.244	734.712

MAMatchingSteps	EpiMatchingSteps	QualityMatches	MatchingErrors
734.712	0	734.712	0

Tabelle 5.15.: Messdaten für Test 5.1.1.2-g: 9 Oktalbaumlevel, initiale Würfel mit 8 LE Kantenlänge, rekursives Raytracing

Modell A (37.544 Polygone)

TreeClsDuration	MatchDuration	BestPosSim	WorstPosSim
31 s	503 s	0	289

AvgSimBefQC	AvgSimAfterQC	AvgDispErrBefQC	AvgDispErrAftQC
99,9473	99,9473	0,422126	0,422126

DispErrSDvBefQC	DispErrSDvAftQC	RayHits1	RayHits2
0,753103	0,753103	761.244	734.796

MAMatchingSteps	EpiMatchingSteps	QualityMatches	MatchingErrors
734.796	0	734.796	0

Modell B (1.019.304 Polygone)

TreeClsDuration	MatchDuration	BestPosSim	WorstPosSim
140 s	539 s	0	289

AvgSimBefQC	AvgSimAfterQC	AvgDispErrBefQC	AvgDispErrAftQC
99,9482	99,9482	0,422071	0,422071

DispErrSDvBefQC	DispErrSDvAftQC	RayHits1	RayHits2
0,752941	0,752941	761.244	734.799

MAMatchingSteps	EpiMatchingSteps	QualityMatches	MatchingErrors
734.799	0	734.799	0

Tabelle 5.16.: Messdaten für Test 5.1.1.2-h: 9 Oktalbaumlevel, initiale Würfel mit 8 LE Kantenlänge, inkrementelles Raytracing

Modell A (37.544 Polygone)

TreeClsDuration	MatchDuration	BestPosSim	WorstPosSim
21 s	3.731 s	0	289

AvgSimBefQC	AvgSimAfterQC	AvgDispErrBefQC	AvgDispErrAftQC
99,9473	99,9473	0,422126	0,422126

DispErrSDvBefQC	DispErrSDvAftQC	RayHits1	RayHits2
0,753103	0,753103	761.244	734.796

MAMatchingSteps	EpiMatchingSteps	QualityMatches	MatchingErrors
734.796	0	734.796	0

Modell B (1.019.304 Polygone)

TreeClsDuration	MatchDuration	BestPosSim	WorstPosSim
526 s	3.811 s	0	289

AvgSimBefQC	AvgSimAfterQC	AvgDispErrBefQC	AvgDispErrAftQC
99,9482	99,9482	0,422071	0,422071

DispErrSDvBefQC	DispErrSDvAftQC	RayHits1	RayHits2
0,752941	0,752941	761.244	734.799

MAMatchingSteps	EpiMatchingSteps	QualityMatches	MatchingErrors
734.799	0	734.799	0

Tabelle 5.17.: Messdaten für Test 5.1.1.2-i: 4 Oktalbaumlevel, initiale Würfel mit 1 LE Kantenlänge, rekursives Raytracing

Modell A (37.544 Polygone)

TreeClsDuration	MatchDuration	BestPosSim	WorstPosSim
21 s	291 s	0	289

AvgSimBefQC	AvgSimAfterQC	AvgDispErrBefQC	AvgDispErrAftQC
99,9473	99,9473	0,422126	0,422126

DispErrSDvBefQC	DispErrSDvAftQC	RayHits1	RayHits2
0,753103	0,753103	761.244	734.796

MAMatchingSteps	EpiMatchingSteps	QualityMatches	MatchingErrors
734.796	0	734.796	0

Modell B (1.019.304 Polygone)

TreeClsDuration	MatchDuration	BestPosSim	WorstPosSim
526 s	318 s	0	289

AvgSimBefQC	AvgSimAfterQC	AvgDispErrBefQC	AvgDispErrAftQC
99,9482	99,9482	0,422071	0,422071

DispErrSDvBefQC	DispErrSDvAftQC	RayHits1	RayHits2
0,752942	0,752942	761.242	734.797

MAMatchingSteps	EpiMatchingSteps	QualityMatches	MatchingErrors
734.797	2	734.799	0

Tabelle 5.18.: Messdaten für Test 5.1.1.2-j: 4 Oktalbaumlevel, initiale Würfel mit 1 LE Kantenlänge, inkrementelles Raytracing

In Bezug auf die rekursive Strahlverfolgung erreicht die Oktalbaumkonfiguration des Tests 5.1.1.2-a mit der mittleren Hierarchietiefe eine Balance im Zielkonflikt zwischen dem Rechenaufwand für die Polygoneingrenzung und der Menge der auszuführenden Dreieck-Strahl-Schnittpunkttests für beide 3D-Modelle. Lediglich bei Test 5.1.1.2-e ergibt sich für den grob aufgelösten künstlichen Raum ein Geschwindigkeitsvorteil, der sich jedoch für die fein aufgelöste Szene in einen Nachteil verwandelt. Dieser Umstand deutet darauf hin, dass Würfel mit einer Kantenlänge von weniger als einer Längeneinheit den Suchraum der Dreiecke für die Schnittpunktberechnung in Modell A kaum noch verkleinern. Gleichzeitig nimmt der Zeitbedarf für die in der Tiefe wirkungslose rekursive Navigation durch den 3D-Baum zu. Bei Szene B ist der Umschwungpunkt, ab dem eine weitere hierarchische Unterteilung die Matchingdauer erhöht, aufgrund der fast 30fach höheren Polygonzahl erst bei 6-7 Hierarchiestufen zu erwarten. Die neun Ebenen von Durchgang 5.1.1.2-g sind in jedem Fall kontraproduktiv – auch, was die Klassifikationszeit von 32 s bzw. 137 s betrifft. Dass das rekursive Raytracing mit kleinen Boxen auf den groben 3D-Daten pro lokalem Matchingschritt ineffizient wird, bestätigen auch die weitgehend identischen Zeitangaben der Testläufe mit dem feinen Modell.

Verringert man die Levelzahl „von unten“ und belässt wie in Test 5.1.1.2-i die kleinstmögliche Boxgröße im Vergleich zu Untersuchung 5.1.1.2-a konstant, erhöht sich die Ausführungsdauer der integrierten Zuordnungsbestimmung gegenüber dem ersten Testlauf dieser Serie für beide Szenen. Die Überdeckung der Modellobjekte mit insgesamt $20 \cdot 10 \cdot 20 = 4.000$ Quadern in der Ausgangsgröße von einer Längeneinheit impliziert ebensoviele Box-Sichtstrahl-Berechnungen und verlängert daneben die Sortierung der Baumelemente nach der Entfernung zum Kamerazentrum. Die Applikation eines Klassifikationsschemas mit mehr als acht Subboxen geht beim rekursiven Raytracingalgorithmus nach den Ergebnissen

von Test 5.1.1.2-c trotz ähnlicher Quaderabmessungen auf der untersten Hierarchieebene wie bei Test 5.1.1.2-a pro Szene jeweils mit einem um ca. 50% höheren Rechenzeitbedarf einher. Abgesehen von der fehlenden Unterteilungswirkung der kleinen Boxen bei Modell A wird offenbar die schnellere Reduktion der Dreieckszahl pro Level und die feinere Anpassung der Baumstruktur an die Form der Modellobjekte durch die im Vergleich zum Oktalbaum größere Zahl an Abstiegsoperationen überkompensiert. Dies trifft vor allem auf die oberen Stufen der 3D-Rangordnung mit vielen nichtleeren Boxen zu.

Die inkrementelle Strahlverfolgung arbeitet fast immer vier- bis neunfach schneller als ihr rekursives Gegenstück. Die größte Matchinggeschwindigkeit ergibt sich für beide Modelle bei einer finalen Boxgröße von etwa 0,1 Längeneinheiten pro Raumdimension und liegt bei etwa 5-6 Minuten, wobei das Unterteilungsschema verfahrensbedingt irrelevant ist. Für größere Quader nimmt sie aufgrund der Vielzahl an Polygonen, die auf einen Strahlschnittpunkt abgeprüft werden müssen, zu. Hierbei wird deutlich, dass die erforderlichen Kreuzungsberechnungen wesentlich aufwendiger sind als die stückweise Quantisierung der Halbgeraden. Erst bei sehr kleinen Baumelementen, wie sie Test 5.1.1.2-h verwendet, macht sich die Approximation des Sichtstrahls zeitlich bemerkbar. Der Anstieg fällt dabei im Vergleich zu den kurzlaufenden Tests 5.1.1.2-b, d oder j szenenunabhängig mit etwa 70% weit weniger stark aus als beim rekursiven Raytracing, wo Test 5.1.1.2-g mit neun Hierarchiestufen dreimal länger rechnet als Test 5.1.1.2-a mit insgesamt sieben Ebenen.

Bei der Betrachtung der Qualitätsindikatoren und der Menge der Strahltreffer in den tabellierten Ausführungsprotokollen des integrierten Matchers treten geringe Schwankungen auf. Eigentlich sollten diese Werte nicht von der Struktur des 3D-Baums abhängen und zusammen mit den einheitlichen planaren Stereobildern identische Messergebnisse zur Folge haben. Die Ursache für die Abweichungen von den Resultaten der ersten Testserie aus Abschnitt 5.1.1.1 und untereinander sind numerische Instabilitäten bei den Schnittberechnungen, die trotz der diesbezüglich getroffenen Vorkehrungen (tolerante Vergleichsfunktionen, zentrische Skalierung der Boxen und Polygone bei der Klassifikation) vereinzelt auftreten. Eine Lösung dieses Problems, von dem auch die POV-Ray-Software betroffen ist, ist letztlich nur dadurch möglich, dass man bei der Implementierung des Raytracings auf Gleitkommazahlen verzichtet und für alle Aspekte ausschließlich Integerzahlen verwendet. Dies setzt eine vollständige Diskretisierung der 3D-Szene mit einer festen unteren Auflösung voraus und erfordert angepasste Algorithmen, die auf Divisionsoperationen und gebräuchliche reellwertige Funktionen verzichten. Abbildung 5.3 veranschaulicht das Problem exemplarisch anhand der Graustufen-Übersichtskarte von Test 5.1.1.2-g (grobes Modell), die den entsprechenden Ausgaben der anderen Läufe inhaltlich gleicht, wenn man jeweils von den unterschiedlich ausgeprägten punktuellen Störungen absieht.

5.1.1.3. Genauigkeit der Korrespondenzen

Neben der Ausführungszeit ist die Qualität der mit dem integrierten Matcher berechneten Korrespondenzen von entscheidender Bedeutung. Daher analysiert dieser Abschnitt die Auswirkung der Korrelationsfenstergröße auf die Zuordnungsgüte, den Unterschied zwischen den vom DaTool-Prototyp unterstützten Abstandsmaßen SAD und SSD und den Effekt der Geman-McClure-Funktion zur Ausreißerkorrektur. Überdies wird die Übereinstimmungssuche mit verschiedenen Auflösungen evaluiert und der Einfluss der Subpixelinterpolationsfilter untersucht.

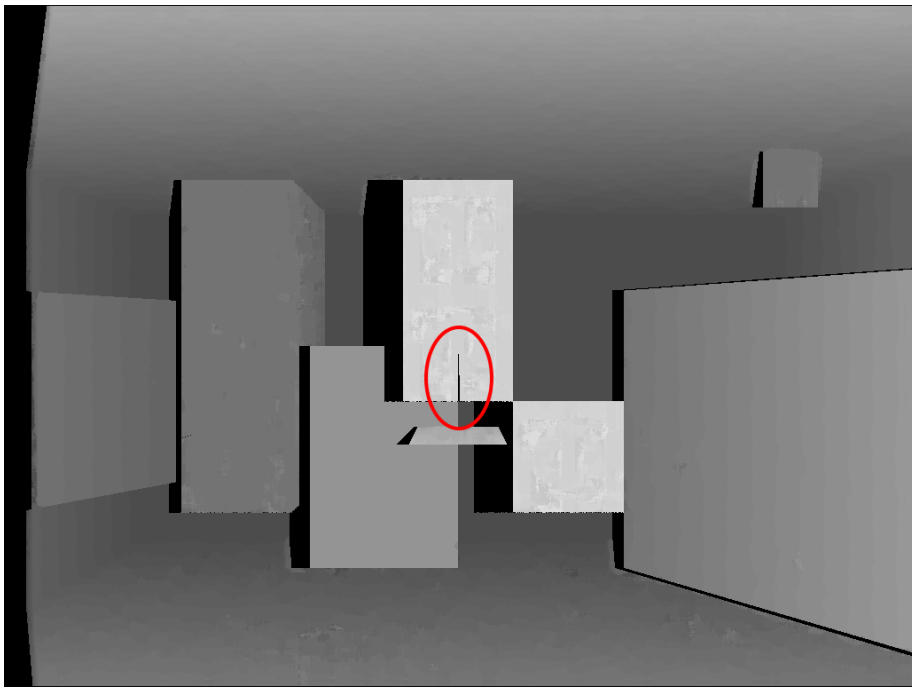


Abbildung 5.3.: Graustufen-Übersichtsbild von Test 5.1.1.2-g (9 Oktalbaumlevel, initiale Boxgröße 8 LE, rekursives Raytracing auf dem groben Modell). Die durch die numerischen Instabilitäten bei den Schnittberechnungen auftretenden Fehler, die die Zuordnungsbestimmung mit aktivierter Objektpunktschätzung punktuell unterbinden, sind rot markiert.

Bei allen Testläufen bleiben die Einstellungen für die aktivierte Objektpunktschätzung konstant, welche ausschließlich mit dem künstlichen 3D-Modell A erfolgt. Die verwendeten Hierarchieparameter sind in der Tabelle von Test 5.1.1.3-a aufgeführt, der als Bezugspunkt für die Läufe 5.1.1.3-b bis 5.1.1.3-k dient. In zweidimensionaler Hinsicht wird sowohl auf das planare auch das zylindrische Stereopaar zurückgegriffen. Die Ergebnisse der Untersuchungen sind in den Tabellen 5.23 bis 5.33 aufgeführt, wobei die Messwerte den zugehörigen Ausführungsprotokollen des integrierten Matchers entstammen und Abschnitt 5.1.1.1 die verwendeten Abkürzungen der Größen in den Kopfzeilen erläutert.

Man beachte in diesem Zusammenhang, dass die mittleren Distanzen AvgSimBefQC und AvgSimAfterQC nur dann direkt miteinander vergleichbar sind, wenn die Muster- bzw. Suchfenstergröße (Korrelationsfenstergröße), die Abstandsfunktion, die Korrelationsschrittweite und der Geman-McClure-Parameter σ identisch sind. Die ground truth-bezogenen Messwerte unterliegen derartigen Einschränkungen in diesem Abschnitt nicht.

Test 5.1.1.3-a Test 5.1.1.3-a verwendet die in Tabelle 5.19 festgehaltenen Einstellungen für die Objektpunktschätzung und die Bestimmung der Zuordnungen, die mit einem mittelgroßen Muster- bzw. Suchfenster pixelgenau durchgeführt wird. Die qualitative Prüfung und Selektion der Zuordnungen ist mit einem relativen Schwellwert von 0,25<1,0 bezogen auf die Bandbreite der SAD-Übereinstimmungsfunktion nach der Geman-McClure-Ausreißerkorrektur aktiviert. Zu Referenzzwecken erfolgt das Matching auch ohne wirksame Qualitätsprüfung. Die Ergebnisse hierfür lassen sich aus den tabellierten Werten

ableiten und liegen deshalb ausschließlich in maschinenlesbarer Form als XML-Protokoll vor.

Objektpunktschätzung	
Level des 3D-Baums	5
Initiale Kantenlängen der Baumquader	8 x 8 x 8 LE
Unterteilungsfaktor	2 x 2 x 2
Kleinste Boxabmessungen	0,5 x 0,5 x 0,5 LE
Strahlverfolgungsmethode	inkrementell
Korrespondenzanalyse	
Region of Interest (ROI)	gesamtes Bild
Qualitätsschwellwert	0,25
Integer-Algorithmus	
Ähnlichkeitsfunktion	SAD
Verwendeter Farbkanal	RGB, perzeptuell gewichtet
Suchbereichsgröße	9 x 9 Pixel
Korrelationsfenstergröße	17 x 17 Pixel
Korrelationsschrittweite	1 x 1 Pixel
effektive Suchschrittweite	1 x 1 Pixel
Geman-McClure-Parameter σ	5,0
Gleitkomma-Algorithmus	
-	

Tabelle 5.19.: Parameter für die Korrespondenzanalyse in Test 5.1.1.3-a, der die Referenz für die Läufe 5.1.1.3-b bis 5.1.1.3-k bildet

Test 5.1.1.3-b Test 5.1.1.3-b unterscheidet sich von Test 5.1.1.3-a lediglich im kleineren Korrelationsfenster mit 7 x 7 Pixeln.

Test 5.1.1.3-c Test 5.1.1.3-c weist mit 35 x 35 Pixeln ein größeres Korrelationsfenster auf, als die Konfiguration von Test 5.1.1.3-a festlegt, verwendet ansonsten aber die gleichen Einstellungen.

Test 5.1.1.3-d Der vierte Test dieser Serie dient zum Vergleich der SAD- mit der SSD-Ähnlichkeitsfunktion. Die Parameter von Test 5.1.1.3-a werden modifiziert, indem das SSD-Übereinstimmungskriterium die SAD-Abstandsfunktion ersetzt. Überdies wird der Geman-McClure-Parameter an die schneller ansteigenden quadrierten Einzeldifferenzen angeglichen und beträgt jetzt $\sigma = 25$. Die Qualitätsprüfung bleibt aktiviert, obwohl die Vergleichbarkeit der Qualitätsindikatoren zu den SAD-Werten der vorangegangenen Tests nur näherungsweise gegeben ist.

Test 5.1.1.3-e Test 5.1.1.3-e entspricht von den Einstellungen her Test 5.1.1.3-a, minimiert aber den Einfluss der Ausreißerkorrektur durch die Geman-McClure-Funktion (vgl. Abschnitt 4.6.1.2). Um dies ohne Quelltextänderungen zu erreichen, wird der Steuerparameter mit $\sigma = 1.000.000$ für SAD-Verhältnisse auf einen vergleichsweise hohen Wert

gesetzt. Dadurch nimmt die Funktion für die variablen Argumente x im relevanten Bereich von 0 ... 255 näherungsweise den angestrebten „neutralen“ linearen Charakter an. Weil die resultierenden Ähnlichkeitswerte mit dem modifizierten Argument σ sehr kleine Beträge aufweisen und stets unter der bisherigen Qualitätsschwelle liegen, wird bei diesem Lauf auf die Selektion der akzeptablen Zuordnungen verzichtet.[18]

Test 5.1.1.3-f Als Gegenstück zu Test 5.1.1.3-e kommt bei Test 5.1.1.3-f ein niedriger Steuerparameter $\sigma = 0,0625$ für die Geman-McClure-Funktion zur Anwendung. Die Limitierung von Ausreißern wird verstärkt, selbst kleine SAD-Beträge führen zu einer schnellen Annäherung an den theoretischen Maximalwert von 1,0. Die Qualitätsprüfung bleibt ähnlich wie bei der vorangegangen Untersuchung deaktiviert, da die resultierenden Disparitäten stets den bisherigen unangepassten Grenzwert verletzen.

Test 5.1.1.3-g Mit Test 5.1.1.3-g wird auf Grundlage der Konfiguration von Test 5.1.1.3-a die Auflösung für die Zuordnungsbestimmung verändert (Tabelle 5.20). Bei der Verschiebung des Suchfensters im Suchbereich findet nur noch jede zweite Bildzeile und -spalte Berücksichtigung. Zusätzlich werden das Muster- und Suchfenster unterabgetastet, weil nach den Ausführungen in Abschnitt 4.6.1.2 die Korrelationsschrittweite ein positives ganzzahliges Vielfaches der Suchschrittweite darstellen muss.

Korrespondenzanalyse	
Region of Interest (ROI)	gesamtes Bild
Qualitätsschwellwert	0,25
Integer-Algorithmus	
Ähnlichkeitsfunktion	SAD
Verwendeter Farbkanal	RGB, perzeptuell gewichtet
Suchbereichsgröße	9 x 9 Pixel
Korrelationsfenstergröße	17 x 17 Pixel
Korrelationsschrittweite	2 x 2 Pixel
effektive Suchschrittweite	2 x 2 Pixel
Geman-McClure-Parameter σ	5,0
Gleitkomma-Algorithmus	
-	

Tabelle 5.20.: Modifizierte Matchingparameter für Test 5.1.1.3-g

Test 5.1.1.3-h Test 5.1.1.3-h modifiziert die Einstellungen von Test 5.1.1.3-a dahingehend, dass die Korrespondenzanalyse subpixelgenau mit einer effektiven Suchschrittweite von einem viertel Pixel erfolgt. Daneben wird der ganzzahlige Pfad des hybriden hierarchischen Algorithmus deaktiviert. Die Bestimmung der Zuordnungen übernimmt exklusiv die Gleitkomma-Implementierung der lokalen Matchingschritte für die SAD-Distanzfunktion (Klasse `CSADMatcherPrecise`). Tabelle 5.21 enthält die angepassten Einstellungen für das Matching.

[18] Auf die Frage, ob und wie sich der Qualitätsschwellwert an die nichtlineare Geman-McClure-Funktion mit variablem Steuerparameter σ anpassen lässt und dabei die Vergleichbarkeit der Ergebnisse gewährleistet bleibt, soll an dieser Stelle nicht eingegangen werden.

Test 5.1.1.3-i Test 5.1.1.3-i zielt im Gegensatz zu seinem Vorläufer 5.1.1.3-h auf eine Zuordnungsauflösung von einem zehntel Pixel ab. Die lokalen Matchingschritte werden außerdem hybrid-hierarchisch ausgeführt. Dabei erfährt das mit der integerbasierten `CSADMatcherStd`-Klasse gewonnene pixelgenaue Resultat eine Verfeinerung mit dem gleitkommabasierten Algorithmus des `CSADMatcherPrecise`-Typs. Tabelle 5.22 konkretisiert die verwendeten Einstellungen.

Test 5.1.1.3-j Test 5.1.1.3-j nutzt die Einstellungen von Test 5.1.1.3-i nach, setzt jedoch den Boxfilter anstelle des Dreiecksfilters zur Subpixelinterpolation ein.

Test 5.1.1.3-k Test 5.1.1.3-k gleicht von den Parametern her Test 5.1.1.3-i. Die Subpixelinterpolation übernimmt abweichend jedoch der kubische B-Spline-Filter.

Stereopaar 1 (planar, 1.024 x 768 Pixel)

TreeClsDuration	MatchDuration	BestPosSim	WorstPosSim
1 s	275 s	0	289

AvgSimBefQC	AvgSimAfterQC	AvgDispErrBefQC	AvgDispErrAftQC
99,9473	66,714	0,422126	0,65166

DispErrSDvBefQC	DispErrSDvAftQC	RayHits1	RayHits2
0,753103	0,831941	761.244	734.796

MAMatchingSteps	EpiMatchingSteps	QualityMatches	MatchingErrors
734.796	0	99.821	0

Stereopaar 2 (zylindrisch, 1.024 x 384 Pixel)

TreeClsDuration	MatchDuration	BestPosSim	WorstPosSim
1 s	128 s	0	289

AvgSimBefQC	AvgSimAfterQC	AvgDispErrBefQC	AvgDispErrAftQC
143,658	62,6439	0,5819	1,23694

DispErrSDvBefQC	DispErrSDvAftQC	RayHits1	RayHits2
0,920602	1,27804	362.057	322.299

MAMatchingSteps	EpiMatchingSteps	QualityMatches	MatchingErrors
322.299	0	21.252	0

Tabelle 5.23.: Messdaten für Test 5.1.1.3-a: Integer-Matching, SAD, Suchbereich 9 x 9 px, Muster- und Suchfenstergröße 17 x 17 px

Korrespondenzanalyse	
Region of Interest (ROI)	gesamtes Bild
Qualitätsschwellwert	0,25
Integer-Algorithmus	
-	
Gleitkomma-Algorithmus	
Ähnlichkeitsfunktion	SAD
Verwendete Farbkanäle	RGB, perzeptuell gewichtet
Subpixelinterpolation	bilinear (Dreiecksfilter)
Suchbereichsgröße	9 x 9 Pixel
Korrelationsfenstergröße	17 x 17 Pixel
Korrelationsschrittweite	1 x 1 Pixel
effektive Suchschrittweite	0,25 x 0,25 Pixel
Geman-McClure-Parameter σ	5,0

Tabelle 5.21.: Modifizierte Matchingparameter für Test 5.1.1.3-h

Korrespondenzanalyse	
Region of Interest (ROI)	gesamtes Bild
Qualitätsschwellwert	0,25
Integer-Algorithmus	
Ähnlichkeitsfunktion	SAD
Verwendeter Farbkanal	RGB, perzeptuell gewichtet
Suchbereichsgröße	9 x 9 Pixel
Korrelationsfenstergröße	17 x 17 Pixel
Korrelationsschrittweite	1 x 1 Pixel
effektive Suchschrittweite	1 x 1 Pixel
Geman-McClure-Parameter σ	5,0
Gleitkomma-Algorithmus	
Ähnlichkeitsfunktion	SAD
Verwendete Farbkanäle	RGB, perzeptuell gewichtet
Subpixelinterpolation	bilinear (Dreiecksfilter)
Suchbereichsgröße	3 x 3 Pixel
Korrelationsfenstergröße	17 x 17 Pixel
Korrelationsschrittweite	1 x 1 Pixel
effektive Suchschrittweite	0,1 x 0,1 Pixel
Geman-McClure-Parameter σ	5,0

Tabelle 5.22.: Modifizierte Matchingparameter für Test 5.1.1.3-i

Stereopaar 1 (planar, 1.024 x 768 Pixel)

TreeClsDuration	MatchDuration	BestPosSim	WorstPosSim
1 s	98 s	0	49

AvgSimBefQC	AvgSimAfterQC	AvgDispErrBefQC	AvgDispErrAftQC
15,341	10,4404	0,600686	0,85511

DispErrSDvBefQC	DispErrSDvAftQC	RayHits1	RayHits2
0,887392	1,09916	761.244	734.796

MAMatchingSteps	EpiMatchingSteps	QualityMatches	MatchingErrors
734.796	0	237.306	0

Stereopaar 2 (zylindrisch, 1.024 x 384 Pixel)

TreeClsDuration	MatchDuration	BestPosSim	WorstPosSim
1 s	51 s	0	49

AvgSimBefQC	AvgSimAfterQC	AvgDispErrBefQC	AvgDispErrAftQC
21,2348	8,47725	0,810847	1,87522

DispErrSDvBefQC	DispErrSDvAftQC	RayHits1	RayHits2
1,09871	1,41276	362.057	322.299

MAMatchingSteps	EpiMatchingSteps	QualityMatches	MatchingErrors
322.299	0	33.837	0

Tabelle 5.24.: Messdaten für Test 5.1.1.3-b: Integer-Matching, SAD, Suchbereich 9 x 9 px, Muster- und Suchfenstergröße 7 x 7 px

Stereopaar 1 (planar, 1.024 x 768 Pixel)

TreeClsDuration	MatchDuration	BestPosSim	WorstPosSim
1 s	929 s	0	1.225

AvgSimBefQC	AvgSimAfterQC	AvgDispErrBefQC	AvgDispErrAftQC
475,989	288,645	0,627398	0,555062

DispErrSDvBefQC	DispErrSDvAftQC	RayHits1	RayHits2
1,55891	0,353156	761.244	734.796

MAMatchingSteps	EpiMatchingSteps	QualityMatches	MatchingErrors
734.796	0	30.588	0

Stereopaar 2 (zylindrisch, 1.024 x 384 Pixel)

TreeClsDuration	MatchDuration	BestPosSim	WorstPosSim
<1 s	420 s	0	1.225

AvgSimBefQC	AvgSimAfterQC	AvgDispErrBefQC	AvgDispErrAftQC
690,632	284,447	0,867823	0,464687

DispErrSDvBefQC	DispErrSDvAftQC	RayHits1	RayHits2
1,78534	0,428669	362.057	322.299

MAMatchingSteps	EpiMatchingSteps	QualityMatches	MatchingErrors
322.299	0	10.444	0

Tabelle 5.25.: Messdaten für Test 5.1.1.3-c: Integer-Matching, SAD, Suchbereich 9 x 9 px, Muster- und Suchfenstergröße 35 x 35 px

Stereopaar 1 (planar, 1.024 x 768 Pixel)

TreeClsDuration	MatchDuration	BestPosSim	WorstPosSim
1 s	269 s	0	289

AvgSimBefQC	AvgSimAfterQC	AvgDispErrBefQC	AvgDispErrAftQC
92,5752	59,5982	0,417121	0,460952

DispErrSDvBefQC	DispErrSDvAftQC	RayHits1	RayHits2
0,748927	0,605427	761.244	734.796

MAMatchingSteps	EpiMatchingSteps	QualityMatches	MatchingErrors
734.796	0	262.166	0

Stereopaar 2 (zylindrisch, 1.024 x 384 Pixel)

TreeClsDuration	MatchDuration	BestPosSim	WorstPosSim
1 s	120 s	0	289

AvgSimBefQC	AvgSimAfterQC	AvgDispErrBefQC	AvgDispErrAftQC
145,499	53,2271	0,590176	1,08986

DispErrSDvBefQC	DispErrSDvAftQC	RayHits1	RayHits2
0,929187	1,18368	362.057	322.299

MAMatchingSteps	EpiMatchingSteps	QualityMatches	MatchingErrors
322.299	0	33.272	0

Tabelle 5.26.: Messdaten für Test 5.1.1.3-d: Integer-Matching, SSD, Suchbereich 9 x 9 px, Muster- und Suchfenstergröße 17 x 17 px, Geman-McClure Steuerparameter $\sigma = 25$

Stereopaar 1 (planar, 1.024 x 768 Pixel)

TreeClsDuration	MatchDuration	BestPosSim	WorstPosSim
1 s	271 s	0	289

AvgSimBefQC	AvgSimAfterQC	AvgDispErrBefQC	AvgDispErrAftQC
$3,0026 \cdot 10^{-8}$	$3,0026 \cdot 10^{-8}$	0,488024	0,488024

DispErrSDvBefQC	DispErrSDvAftQC	RayHits1	RayHits2
0,88749	0,88749	761.244	734.796

MAMatchingSteps	EpiMatchingSteps	QualityMatches	MatchingErrors
734.796	0	734.796	0

Stereopaar 2 (zylindrisch, 1.024 x 384 Pixel)

TreeClsDuration	MatchDuration	BestPosSim	WorstPosSim
1 s	126 s	0	289

AvgSimBefQC	AvgSimAfterQC	AvgDispErrBefQC	AvgDispErrAftQC
$9,3608 \cdot 10^{-8}$	$9,3608 \cdot 10^{-8}$	0,723546	0,723546

DispErrSDvBefQC	DispErrSDvAftQC	RayHits1	RayHits2
1,14677	1,14677	362.057	322.299

MAMatchingSteps	EpiMatchingSteps	QualityMatches	MatchingErrors
322.299	0	322.299	0

Tabelle 5.27.: Messdaten für Test 5.1.1.3-e: Integer-Matching, SAD, Suchbereich 9 x 9 px, Muster- und Suchfenstergröße 17 x 17 px, Ausreißerkorrektur deaktiviert, keine Qualitätsprüfung

Stereopaar 1 (planar, 1.024 x 768 Pixel)

TreeClsDuration	MatchDuration	BestPosSim	WorstPosSim
1 s	281 s	0	289

AvgSimBefQC	AvgSimAfterQC	AvgDispErrBefQC	AvgDispErrAftQC
259,487	259,487	0,85031	0,85031

DispErrSDvBefQC	DispErrSDvAftQC	RayHits1	RayHits2
2,1627	2,1627	761.244	734.796

MAMatchingSteps	EpiMatchingSteps	QualityMatches	MatchingErrors
734.796	0	734.796	0

Stereopaar 2 (zylindrisch, 1.024 x 384 Pixel)

TreeClsDuration	MatchDuration	BestPosSim	WorstPosSim
1 s	126 s	0	289

AvgSimBefQC	AvgSimAfterQC	AvgDispErrBefQC	AvgDispErrAftQC
265,949	265,949	1,11157	1,11157

DispErrSDvBefQC	DispErrSDvAftQC	RayHits1	RayHits2
1,27566	1,27566	362.057	322.299

MAMatchingSteps	EpiMatchingSteps	QualityMatches	MatchingErrors
322.299	0	322.299	0

Tabelle 5.28.: Messdaten für Test 5.1.1.3-f: Integer-Matching, SAD, Suchbereich 9 x 9 px, Muster- und Suchfenstergröße 17 x 17 px, Ausreißerkorrektur verstärkt, keine Qualitätsprüfung

Stereopaar 1 (planar, 1.024 x 768 Pixel)

TreeClsDuration	MatchDuration	BestPosSim	WorstPosSim
1 s	68 s	0	64

AvgSimBefQC	AvgSimAfterQC	AvgDispErrBefQC	AvgDispErrAftQC
26,3737	14,1063	1,10393	1,11197

DispErrSDvBefQC	DispErrSDvAftQC	RayHits1	RayHits2
1,84206	1,17097	761.244	734.796

MAMatchingSteps	EpiMatchingSteps	QualityMatches	MatchingErrors
734.796	0	90.404	0

Stereopaar 2 (zylindrisch, 1.024 x 384 Pixel)

TreeClsDuration	MatchDuration	BestPosSim	WorstPosSim
1 s	37 s	0	64

AvgSimBefQC	AvgSimAfterQC	AvgDispErrBefQC	AvgDispErrAftQC
33,8279	13,0016	1,11465	1,72326

DispErrSDvBefQC	DispErrSDvAftQC	RayHits1	RayHits2
1,22203	1,43782	362.057	322.299

MAMatchingSteps	EpiMatchingSteps	QualityMatches	MatchingErrors
322.299	0	20.676	0

Tabelle 5.29.: Messdaten für Test 5.1.1.3-g: Integer-Matching, SAD, Suchbereich 9 x 9 px, Muster- und Suchfenstergröße 17 x 17 px, Korrelations- und Suchschrittweite 2 x 2 px

Stereopaar 1 (planar, 1.024 x 768 Pixel)

TreeClsDuration	MatchDuration	BestPosSim	WorstPosSim
1 s	15.137 s	0	289

AvgSimBefQC	AvgSimAfterQC	AvgDispErrBefQC	AvgDispErrAftQC
56,3078	42,3014	0,324982	0,26436

DispErrSDvBefQC	DispErrSDvAftQC	RayHits1	RayHits2
0,794226	0,479198	761.244	734.796

MAMatchingSteps	EpiMatchingSteps	QualityMatches	MatchingErrors
734.796	0	580.760	0

Stereopaar 2 (zylindrisch, 1.024 x 384 Pixel)

TreeClsDuration	MatchDuration	BestPosSim	WorstPosSim
1 s	6.623 s	0	289

AvgSimBefQC	AvgSimAfterQC	AvgDispErrBefQC	AvgDispErrAftQC
103,953	53,2351	0,492295	0,575022

DispErrSDvBefQC	DispErrSDvAftQC	RayHits1	RayHits2
0,934053	0,899342	362.057	322.299

MAMatchingSteps	EpiMatchingSteps	QualityMatches	MatchingErrors
322.299	0	66.245	0

Tabelle 5.30.: Messdaten für Test 5.1.1.3-h: Gleitkomma-Matching, SAD, Suchbereich 9 x 9 px, Muster- und Suchfenstergröße 17 x 17 px, Korrelations- und Suchschrittweite 0,25 x 0,25 px

Stereopaar 1 (planar, 1.024 x 768 Pixel)

TreeClsDuration	MatchDuration	BestPosSim	WorstPosSim
1 s	18.546 s	0	289

AvgSimBefQC	AvgSimAfterQC	AvgDispErrBefQC	AvgDispErrAftQC
55,1935	41,6757	0,339562	0,282127

DispErrSDvBefQC	DispErrSDvAftQC	RayHits1	RayHits2
0,788324	0,473415	761.244	734.796

MAMatchingSteps	EpiMatchingSteps	QualityMatches	MatchingErrors
734.796	0	588.158	0

Stereopaar 2 (zylindrisch, 1.024 x 384 Pixel)

TreeClsDuration	MatchDuration	BestPosSim	WorstPosSim
1 s	8.176 s	0	289

AvgSimBefQC	AvgSimAfterQC	AvgDispErrBefQC	AvgDispErrAftQC
102,286	53,2559	0,51401	0,583019

DispErrSDvBefQC	DispErrSDvAftQC	RayHits1	RayHits2
0,975761	0,919822	362.057	322.299

MAMatchingSteps	EpiMatchingSteps	QualityMatches	MatchingErrors
322.299	0	71.471	0

Tabelle 5.31.: Messdaten für Test 5.1.1.3-i: Hybrid-hierarchisches Matching, SAD, Suchbereich 9 x 9 / 3 x 3 px, Muster- und Suchfenstergröße jeweils 17 x 17 px, Korrelations- und Suchschrittweite 1 x 1 / 0,1 x 0,1 px, bilineare Subpixelinterpolation

Stereopaar 1 (planar, 1.024 x 768 Pixel)

TreeClsDuration	MatchDuration	BestPosSim	WorstPosSim
1 s	18.522 s	0	289

AvgSimBefQC	AvgSimAfterQC	AvgDispErrBefQC	AvgDispErrAftQC
164,32	68,458	0,443319	0,651264

DispErrSDvBefQC	DispErrSDvAftQC	RayHits1	RayHits2
0,800556	0,832534	761.244	734.796

MAMatchingSteps	EpiMatchingSteps	QualityMatches	MatchingErrors
734.796	0	99.821	0

Stereopaar 2 (zylindrisch, 1.024 x 384 Pixel)

TreeClsDuration	MatchDuration	BestPosSim	WorstPosSim
1 s	8.049 s	0	289

AvgSimBefQC	AvgSimAfterQC	AvgDispErrBefQC	AvgDispErrAftQC
204,181	68,9753	0,614422	1,23707

DispErrSDvBefQC	DispErrSDvAftQC	RayHits1	RayHits2
1,00281	1,27853	362.057	322.299

MAMatchingSteps	EpiMatchingSteps	QualityMatches	MatchingErrors
322.299	0	21.252	0

Tabelle 5.32.: Messdaten für Test 5.1.1.3-j: Hybrid-hierarchisches Matching, SAD, Suchbereich 9 x 9 / 3 x 3 px, Muster- und Suchfenstergröße jeweils 17 x 17 px, Korrelations- und Suchschrittweite 1 x 1 / 0,1 x 0,1 px, Subpixelinterpolation mit dem Box-Filter

Stereopaar 1 (planar, 1.024 x 768 Pixel)

TreeClsDuration	MatchDuration	BestPosSim	WorstPosSim
1 s	35.193 s	0	289

AvgSimBefQC	AvgSimAfterQC	AvgDispErrBefQC	AvgDispErrAftQC
48,8558	37,1281	0,296885	0,236432

DispErrSDvBefQC	DispErrSDvAftQC	RayHits1	RayHits2
0,783419	0,46612	761.244	734.796

MAMatchingSteps	EpiMatchingSteps	QualityMatches	MatchingErrors
734.796	0	618.252	0

Stereopaar 2 (zylindrisch, 1.024 x 384 Pixel)

TreeClsDuration	MatchDuration	BestPosSim	WorstPosSim
1 s	15.516 s	0	289

AvgSimBefQC	AvgSimAfterQC	AvgDispErrBefQC	AvgDispErrAftQC
93,8745	53,4229	0,480822	0,461213

DispErrSDvBefQC	DispErrSDvAftQC	RayHits1	RayHits2
0,97094	0,802043	362.057	322.299

MAMatchingSteps	EpiMatchingSteps	QualityMatches	MatchingErrors
322.299	0	104.808	0

Tabelle 5.33.: Messdaten für Test 5.1.1.3-k: Hybrid-hierarchisches Matching, SAD, Suchbereich 9 x 9 / 3 x 3 px, Muster- und Suchfenstergröße jeweils 17 x 17 px, Korrelations- und Suchschrittweite 1 x 1 / 0,1 x 0,1 px, kubische B-Spline-Subpixelinterpolation

Die in den Tests 5.1.1.3-a bis 5.1.1.3-c untersuchten Größenänderungen des Muster- bzw. Suchfensters bestätigen die in Abschnitt 3.2.4 der Konzeption getroffenen Aussagen. Mit der Verkleinerung von 17 x 17 auf 7 x 7 Bildpunkte in Test 5.1.1.3-b steigt der mittlere Disparitätsfehler bezogen auf die ground truth-Verschiebungen im Vergleich zum Referenzlauf vor und nach der Qualitätsprüfung sowohl für die planaren als auch die zylindrischen Aufnahmen. Gleichzeitig erhöhen sich die Standardabweichungen DispErrSDvBefQC und DispErrSDvAftQC, was darauf hindeutet, dass bei der Bewegung der kleineren Suchfenster über den jeweiligen Suchbereich des Suchbildes vermehrt lokale Mehrdeutigkeiten auftreten. Gestützt wird diese Interpretation durch den sichtbaren Grieseleffekt in den Graustufen-Übersichtsbildern des Matchers (Abbildungen 5.4(b) links und 5.5(b) links). Weil durch die kleineren Abmessungen der miteinander zu korrelierenden Fenster im unverändert großen Suchbereich mehr potentielle Korrespondenzkandidaten zur Verfügung stehen, nimmt die Anzahl der Zuordnungen, welche den relativen Abstandsschwellwert unterschreiten, gegenüber Test 5.1.1.3-a zu. Verstärkt wird dieser Effekt durch die mit der höheren Lokalität einhergehende verminderte Intensitätsvariation in den 7 x 7 Pixel messenden Bildausschnitten. Der geringere Kontrast hat hierbei niedrigere SAD-Werte zur Folge.

Die Vergrößerung des Korrelationsareals von 17 x 17 auf 35 x 35 Pixel liefert ebenfalls qualitativ schlechtere Übereinstimmungen, weil die entbehrlichen Zusatzinformationen in der lokalen Umgebung um das Musterpixel bzw. Suchfensterzentrum das Matching vor allem in den strukturierten Bildbereichen (Boden und Decke des Testraums, Objektkanten) sichtbar stören. Der durchschnittliche Disparitätsfehler und die zugehörige Standardabweichung sind bei Test 5.1.1.3-c vor der Qualitätsprüfung höher als für Test 5.1.1.3-a.

Nach dem Vergleich der SAD-Ähnlichkeitswerte mit dem relativen Schwellwert von $0,25$ und der Zuordnungsselektion ergibt sich beim Referenzlauf und Untersuchung 5.1.1.3-b paradoxerweise eine Verschlechterung der Messergebnisse gegen die mittels Raytracing gewonnenen ground truth-Daten, obwohl der mittlere Distanzwert „AfterQC“ fällt. Die Ursache hierfür liegt in den zentral gelegenen Passpunktfeldern und anderen partiell monochromen Stereobildbereichen begründet, die bei einer hinreichend kleinen Größe des Muster- bzw. Suchfensters in Ermangelung von geeigneten Strukturelementen gleich beim ersten Korrelationsvorgang eine höchstwahrscheinlich falsche Korrespondenz mit kleinem Distanzwert liefern. Wird die lokale Umgebung für die Zuordnungsbestimmung wie bei Test 5.1.1.3-c dagegen vergrößert, werden insbesondere bei den Kalibrationsmarkierungen stark kontrastierende Bildanteile im ansonsten einfarbigen Musterfenster eingeschlossen, die den Vorlagenausschnitt nahezu eindeutig identifizieren. In der Konsequenz gibt es beim Matching dieser Regionen weniger Fehlzuordnungen, und die Genauigkeit der Korrespondenzen übersteigt die vom ersten Test dieses Abschnitts.

In Bezug auf den Rechenzeitbedarf für die beschleunigte Übereinstimmungssuche bewahrheitet sich die theoretische Überlegung einer superlinearen Zunahme nach der Modifikation der Seitenabmessungen des Korrelationsbereichs auch in der Praxis. Berücksichtigt man den Aufwand für die zweifache inkrementelle Strahlverfolgung, der für die verwendete konstante Baumkonfiguration bei gemessenen 48 s (planar) bzw. 28 s (zylindrisch) liegt, ergeben sich für die Testläufe 5.1.1.3-b, 5.1.1.3-a und 5.1.1.3-c bereinigte Analysezeiten von 50 s $\rightarrow$ 227 s $\rightarrow$ 881 s (planar) bzw. 23 s $\rightarrow$ 100 s $\rightarrow$ 392 s (zylindrisch). Die ungefähre Verdoppelung einer Seitenlänge des quadratischen Muster- bzw. Suchfensters mit 7 $\rightarrow$ 17 $\rightarrow$ 35 Bildpunkten vom schnellsten zum langsamsten der drei Läufe zieht näherungsweise eine temporale Vervierfachung nach sich.

Das alternative Ähnlichkeitskriterium, die Summe der quadrierten Differenzen der Pixel des aktuell betrachteten Muster- und Suchfensterpaars, hat kaum Auswirkungen auf die Matchingqualität, sofern der Geman-McClure-Parameter σ an das quadratische Verhalten der SSD-Funktion angepasst wird. Dass die Ausreißerkorrektur einen merklichen Einfluss auf die Güte der Zuordnungen hat, zeigen die Ergebnisse der Tests 5.1.1.3-e und 5.1.1.3-f. Sowohl die de facto-Deaktivierung als auch die Forcierung der Abstandswertdämpfung führen zu höheren mittleren Disparitätsfehlern und vergrößern die Streuung der Verschiebungen gegenüber den ground truth-Daten im Vergleich zu Lauf 5.1.1.3-a. Während das Abgleiten der Zuordnungsgüte nach der Abschaltung auf die fortan ungewichtet in den Korrelationsprozess einfließenden Ausreißer zurückgeführt werden kann, ist nach der Verstärkung der Distanzwertabschwächung durch die Geman-McClure-Funktion offensichtlich die Unterscheidbarkeit von vormals abweichenden Fensterinhalten auf globaler Ebene beeinträchtigt. In den zugehörigen vollständigen Graustufen-Übersichtsbildern 5.4(e)/(f) und 5.5(e)/(f) äußern sich die modifizierten Steuerparameter σ in verwaschenen Objektkanten respektive undifferenziert fleckigen Oberflächen und untermauern diese Vermutungen.

Die Testergebnisse der verbleibenden Läufe 5.1.1.3-g bis k verdeutlichen, dass die Zuordnungsqualität von einer höheren Matchingauflösung signifikant profitiert, andererseits jedoch die benötigte Rechenzeit stark zunimmt. Bei der Unterabtastung des Muster- und Suchfensters wie auch des Suchbereichs mit Lauf 5.1.1.3-g ergeben sich prinzipbedingt und wegen der auftretenden Aliasingartefakte mittlere Disparitätsabweichungen von mehr als einem Bildpunkt relativ zu den ground truth-Daten und ähnlich hohe Standardabweichungen. Demnach eignet sich der mit 68 s (planar) bzw. 37 s (zylindrisch) (ohne

Strahlverfolgung 20 s bzw. 9 s) überaus schnelle Lauf lediglich als Ausgangspunkt für die weitere Eingrenzung des Suchbereichs.

Die subpixelgenauen Gegenstücke 5.1.1.3-h bis k benötigen jeweils mehrere Stunden für die Ausführung und erzielen fast alle Disparitätsfehler um 0,3 Pixel, welche vor der Qualitätsprüfung um ca. 25% (planar) bzw. 13% (zylindrisch) und nach selbiger um etwa 40% bzw. 45% unter denen des Referenztests 5.1.1.3-a liegen. Auch die durchschnittlichen SAD-Abstände fallen teilweise bis auf die Hälfte des Bezugswertes. Wie ein Vergleich der Qualitätsindikatoren von Test 5.1.1.3-h und 5.1.1.3-i für beide Stereoaufnahmen zeigt, hängt die konkret erreichbare Güte der Zuordnungen jedoch vom eingesetzten Zwischenwertfilter und dessen Abstimmung mit der angestrebten Auflösung ab. Obwohl sich die Präzision des erstgenannten Versuchs zum letztgenannten Lauf von 0,25 auf 0,1 Pixel mehr als verdoppelt, sind die entsprechenden Messwerte fast identisch. Der Informationsgehalt der 17 x 17 Bildpunkte fassenden Suchfenster bei Untersuchung 5.1.1.3-i mit den bilinear interpolierten Grauwerten auf Basis von gerade einmal 1,7 x 1,7 existierenden horizontalen bzw. vertikalen Pixeln reicht anders als bei Test 5.1.1.3-h mit 4,25 tatsächlich vorhandenen Bildpunkten in X- und Y-Richtung scheinbar nicht aus, um die richtige Korrespondenz treffsicher zu identifizieren.

Bestätigt wird diese Vermutung durch die Tests 5.1.1.3-j und 5.1.1.3-k, die den Box- respektive B-Spline-Filter zur Zwischenwertberechnung verwenden. Die durch Pixelverdoppelung erzeugten Intensitäten führen zu durchschnittlichen Disparitätsfehlern, Standardabweichungen und mittleren Abstandswerten „BeforeQC“, die für beide Aufnahmen noch hinter den Resultaten des pixelgenauen Referenzlaufs 5.1.1.3-a zurückbleiben. Mit der kantenerhaltenden kubischen B-Spline-Interpolation werden dagegen selbst mit der minimalen Ausgangsinformation ablesbar die qualitativ besten Korrespondenzen in dieser Arbeit bestimmt. Aufgrund des Trägers mit 4 x 4 Filterkoeffizienten (Box- und Dreiecksfilter je 2 x 2 Filterkoeffizienten) ist jedoch die benötigte Rechenzeit von fast zehn bzw. 4,5 Stunden für die planaren bzw. zylindrischen Stereoaufnahmen mit ihren 786.432 respektive 393.216 Bildpunkten inakzeptabel hoch.

Wenn man die Ausführungsdauer der Tests 5.1.1.3-h und 5.1.1.3-i analysiert, so wird der beschleunigende Effekt des hybriden hierarchischen Zuordnungsverfahrens deutlich. Die Zeitangaben des auf den Gleitkommaalgorithmus beschränkten Laufs unterscheiden sich auf hohem Niveau nicht in dem Maß von seinem zweistufig vorgehenden Pendant, wie es das Verhältnis der Matchingauflösungen von 2,5 eigentlich erwarten lässt. Ein negativer Einfluss der hybrid-hierarchischen Korrespondenzanalyse auf die Güte der Übereinstimmungen ist nicht unmittelbar erkennbar. Präzisionsbedingte Falschzuordnungen auf der pixelgenauen Ebene, die sich auf die subpixelgenaue Zuordnungsbestimmung übertragen, sind gleichwohl nicht auszuschließen. Sie dürften jedoch einen relativ geringen Qualitätsverlust implizieren.

Die Abbildungen 5.4 und 5.5 zeigen abschließend die Graustufen-Übersichtsbilder des Matchers für die dritte Testserie. Sofern vorhanden, sind die Bitmaps vor und nach der Qualitätsprüfung aufgeführt.

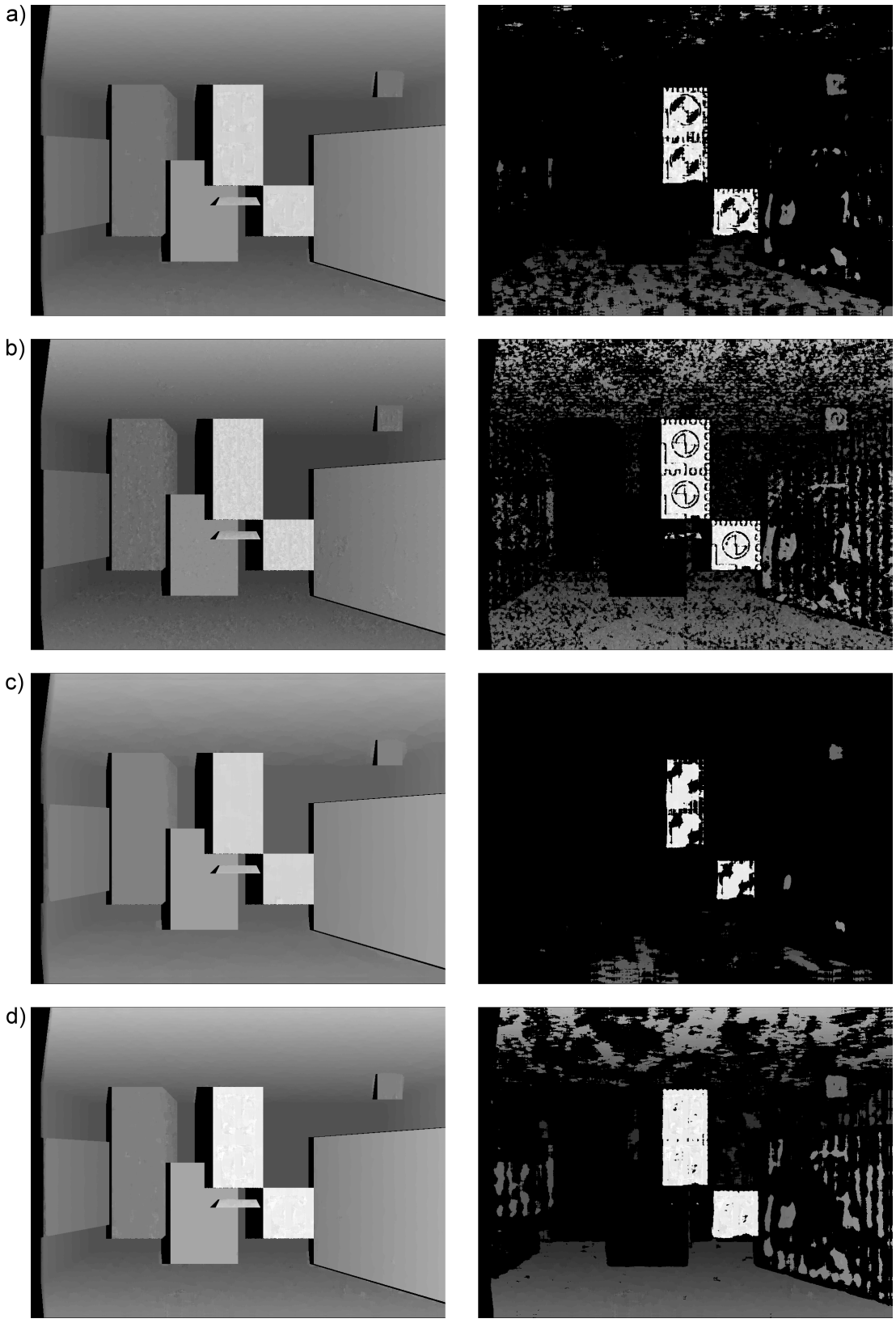


Abbildung 5.4.: Planare Graustufen-Übersichtsbilder der Testläufe 5.1.1.3-a bis 5.1.1.3-k ohne (links) und mit (rechts) aktivierter Qualitätsprüfung gemäß den Einstellungen des jeweiligen Laufs und sofern vorhanden. Die Buchstaben vor den Bildern decken sich mit denen der Testbezeichnungen.

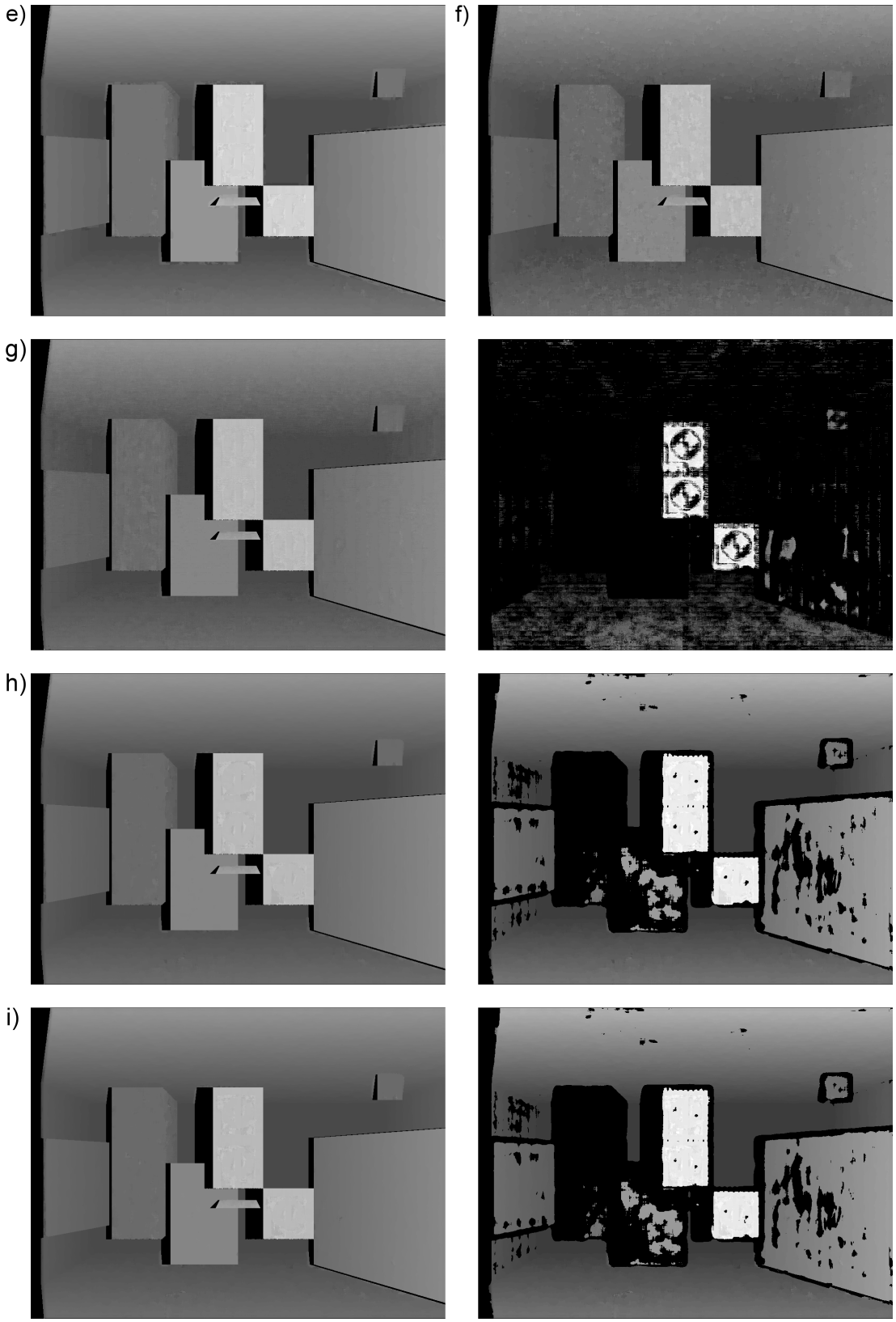


Abbildung 5.4.: Planare Graustufen-Übersichtsbilder der Testläufe 5.1.1.3-a bis 5.1.1.3-k (Fortsetzung)

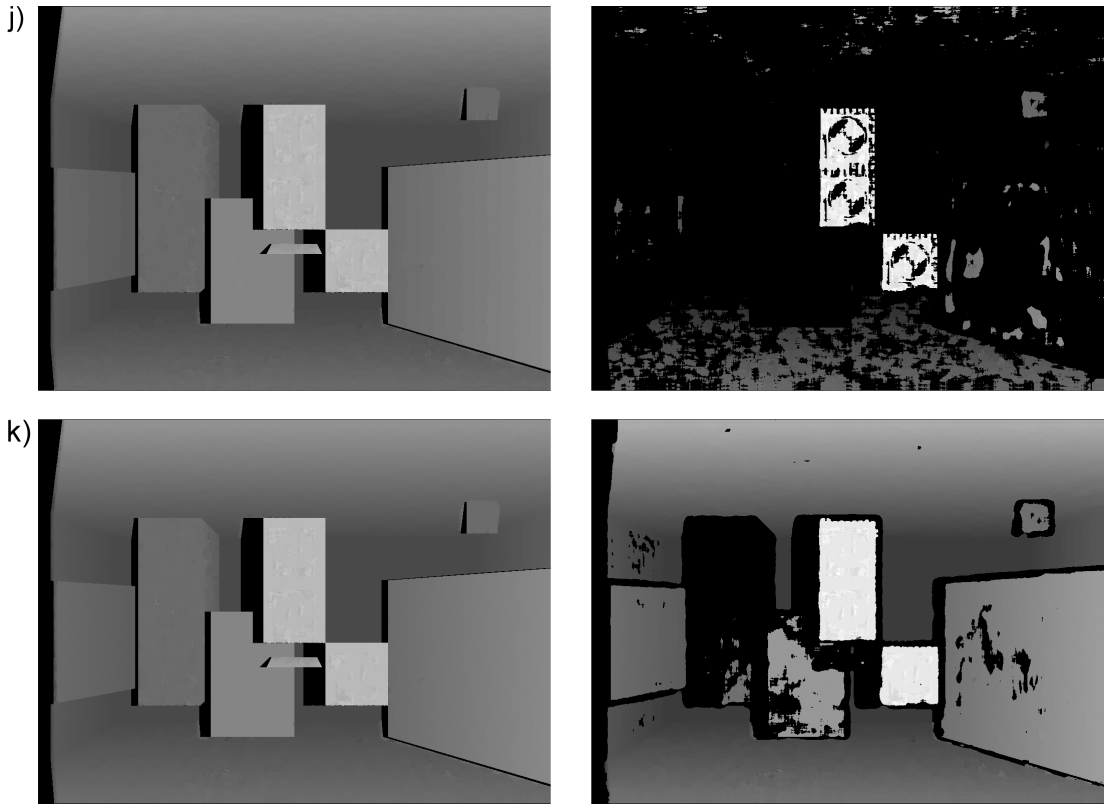


Abbildung 5.4.: Planare Graustufen-Übersichtsbilder der Testläufe 5.1.1.3-a bis 5.1.1.3-k (Fortsetzung)

5.1.1.4. Relative Orientierung der Kameras zum Modell

Entsprechend der Konzeption muss für die korrekte Funktionsweise der Objektpunktschätzung als Teil des integrierten Matchers die relative äußere Orientierung des Stereosystems zur 3D-Szene bekannt sein. Anders als bei der künstlichen Testszene lässt sich die Ausrichtung in der Realität jedoch nicht exakt bestimmen, weil die hierfür verwendeten Ausgangsdaten, das LRF-Modell und die Kameraaufnahmen, ihrerseits mit Fehlern behaftet sind. Rekapituliert man den Strahlverfolgungsprozess, so verursachen ungenaue Orts- und Rotationsangaben der Bildaufzeichnungsgeräte eine Abweichung der realen von den exakten Suchbereichszentren für die Korrespondenzanalyse. Die Positionsdifferenzen lassen sich in der Theorie mit einer Ausdehnung des Suchbereichs auffangen, wenn dieser den unbekannten Mittelpunkt desjenigen Gebiets beinhaltet, in welchem das Suchfenster bei korrekter Ausrichtung verschoben werden muss. Einen Anhaltspunkt, um die erforderliche Erweiterung abschätzen zu können, liefert typischerweise die eingesetzte interaktive Software zur Bestimmung der Ausrichtung oder das verwendete (halb-)automatische Orientierungsverfahren (auch die in Abschnitt 3.2.1 propagierte Methode). Grundsätzlich ist allerdings zu beachten, dass sich die anhand der Bilder rekonstruierten 3D-Raumpunkte, auf die unter anderem auch der Modellverbesserungsalgorithmus zurückgreift, in jedem Fall auf die fehlerbehaftete externe Orientierung beziehen. Dies gilt insbesondere für die bei der Entfernungsmessung relevanten Kamerastandorte.

Im Folgenden wird die theoretische Kompensationsfähigkeit des Matchers für zwei suboptimale relative Orientierungen praktisch untersucht. Dabei erfolgt die Bestimmung der

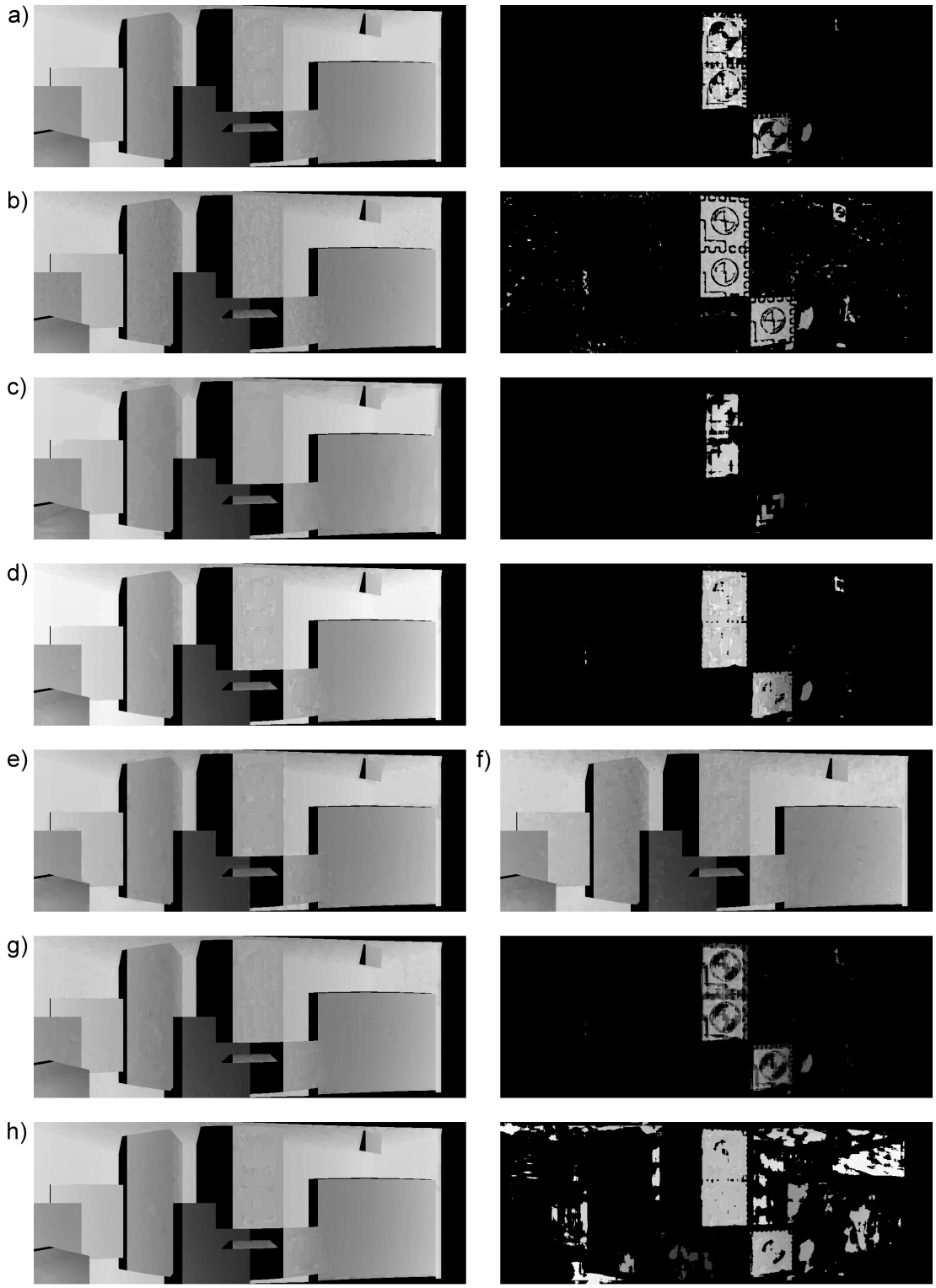


Abbildung 5.5.: Zylindrische Graustufen-Übersichtsbilder der Testläufe 5.1.1.3-a bis 5.1.1.3-k ohne (links) und mit (rechts) aktivierter Qualitätsprüfung gemäß den Einstellungen des jeweiligen Laufs und sofern vorhanden. Die Buchstaben vor den Bildern decken sich wiederum mit denen der Testbezeichnungen.

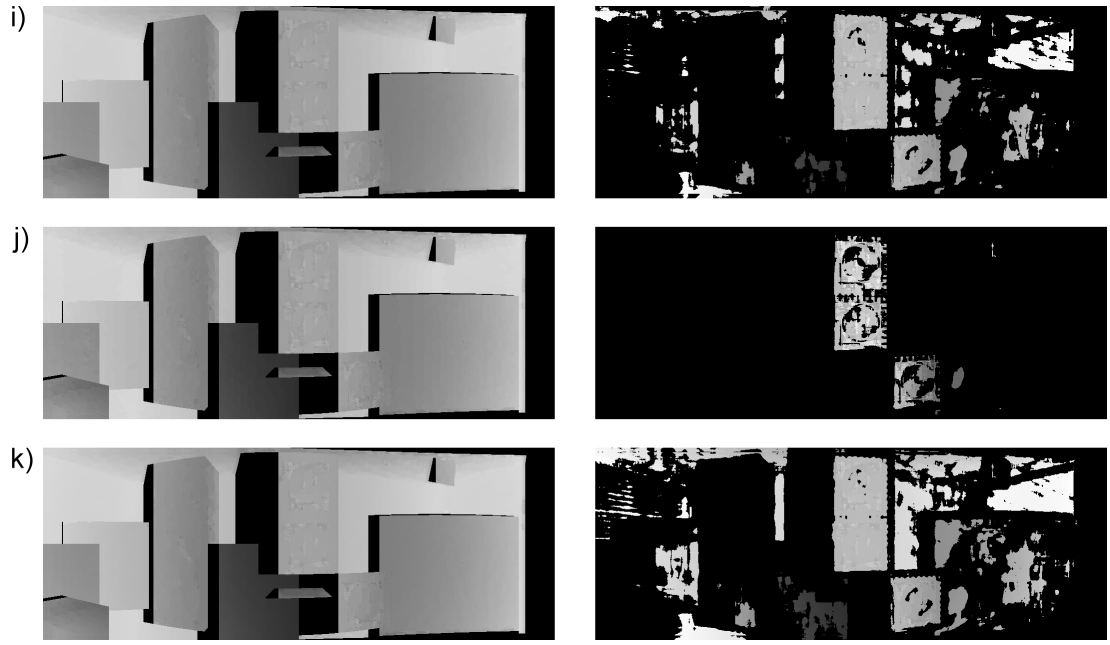


Abbildung 5.5.: Zylindrische Graustufen-Übersichtsbilder der Testläufe 5.1.1.3-a bis 5.1.1.3-k (Fortsetzung)

Zuordnungen mit dem groben LRF-Modell A und dem originalen Stereopaar 1, da man die Ausrichtung erst nach der photographischen Aufnahme bestimmt und diese demnach keinen Einfluß auf den Inhalt der Bilder hat. Die externen Kameraparameter für den integrierten Algorithmus werden in den Tests 5.1.1.4-a bis 5.1.1.4-d geändert, so dass sie nicht mehr zur Projektion „passen“. In den Tabellen 5.36 bis 5.39 sind die zugehörigen Ergebnisse der Korrespondenzanalyse aufgeführt, wie sie die Ausführungsprotokolle des integrierten Matchers maschinenlesbar festhalten. Die Bedeutung der abgekürzten Messgrößen erläutert Abschnitt 5.1.1.1.

Test 5.1.1.4-a Für Test 5.1.1.4-a wird die erste planare Kamera verschoben, die neue externe Orientierung des Stereosystems ist in Tabelle 5.34 aufgeführt. Die Korrespondenzanalyse erfolgt mit den Einstellungen von Lauf 5.1.1.3-a des letzten Abschnitts bei deaktivierter Qualitätsprüfung. Demnach sind die Abmessungen des Suchbereichs auf 9 x 9 Pixel festgelegt und berücksichtigen die modifizierte Orientierung nicht.

Parameter	**Stereopaar 1**	
	Bild #1	**Bild #2**
Kamerazentrum	(-0,2; 1,1; -3,8)	(0,5; 1; -4)
Rotation ω, ϕ, κ	(0°; 0°; 0°)	(0°; 0°; 0°)

Tabelle 5.34.: Geänderte äußere Kameraorientierung für Test 5.1.1.4-a zur Simulation von Translationsfehlern

Test 5.1.1.4-b Test 5.1.1.4-b nutzt eine Stereokonfiguration, bei der die erste Kamera im Vergleich zur korrekten Ausrichtung ohne Drehung um jeweils 2° im Uhrzeigersinn

um die X- und Y-Achse des DaTool-Koordinatensystems rotiert ist (Tabelle 5.35). Die Korrespondenzanalyse erfolgt wiederum mit den Einstellungen von Lauf 5.1.1.3-a ohne Qualitätsprüfung und ohne Kompensation der geänderten Orientierung durch eine Vergrößerung des 9 x 9 Pixel umfassenden Suchbereichs.

Parameter	**Stereopaar 1**	
	Bild #1	**Bild #2**
Kamerazentrum	(0; 1; -4)	(0,5; 1; -4)
Rotation ω, ϕ, κ	(2°; 2°; 0°)	(0°; 0°; 0°)

Tabelle 5.35.: Geänderte äußere Kameraorientierung für Test 5.1.1.4-b zur Simulation von Rotationsfehlern

Test 5.1.1.4-c Test 5.1.1.4-c entspricht von den Kameraparametern und Matchingeinstellungen her Test 5.1.1.4-a, benutzt jedoch einen größeren Suchbereich mit 81 x 42 Pixeln. Dieser soll die fehlerbehaftete Orientierung, die in einer Verschiebung der ersten Stereokamera besteht, kompensieren. Den Anhaltspunkt für die maximale Abweichung des Suchbereichszentrums, die die obigen Rechteckabmessungen motiviert, liefern die versuchsweise via POV-Ray gerenderten Bilder der künstlichen Szene unter Verwendung der modifizierten Ortsangaben. Im Vergleich zum originalen Stereopaar, für welches die exakte Kameraposition bekannt ist, ergibt sich am projizierten Passpunkt mit den Raumkoordinaten (0; 1; 0) (siehe Anhang A.1) eine Abweichung von (27; 14) Pixeln für das erste Bild. Das zweite Bild entspricht dem Original und trägt nicht zum Betrag der Gesamtdifferenz bei. Nach der Verdoppelung des horizontalen und vertikalen Positionsunterschieds, die aufgrund der mittigen Anordnung des Suchbereichs um das bei der Objektpunktschätzung berechnete Suchbereichszentrum erforderlich ist, beträgt die Größe des Areals, in dem sich die korrekte Zuordnung befindet, zunächst 54 x 28 Pixel. Ein Sicherheitsfaktor von 1,5 erweitert die Abmessungen schließlich auf die erwähnten 81 x 42 Bildpunkte. Die gewählte Kalibrationsmarkierung befindet sich recht nah an den Kamerastandorten, weil die Abweichung des Suchbereichszentrums bei den planaren Aufnahmen analog zu den Stereodisparitäten mit steigender Entfernung vom Betrachter abnimmt. Benutzt man dagegen einen Punkt aus dem Bildhintergrund zur Festlegung der Größe des Zielareals für die Zuordnungssuche, so hat diese Wahl eine systematische Unterschätzung der Abmessungen zur Folge.

Test 5.1.1.4-d Test 5.1.1.4-d verwendet die gleichen Kameraparameter wie Test 5.1.1.4-b. Die Suchbereichsgröße wird wie bei Lauf 5.1.1.4-c anhand der Abweichung des projizierten (0; 1; 0)-Passpunktes in den versuchsweise mit den modifizierten Drehwinkeln gerenderten Bildern relativ zu den künstlichen Originalaufnahmen abgeschätzt. Sie beträgt nach der obligatorischen Verdoppelung und der Beaufschlagung des Sicherheitsfaktors (51; 51) Pixel in der Horizontalen respektive Vertikalen.

Stereopaar 1 (planar, 1.024 x 768 Pixel)

TreeClsDuration	MatchDuration	BestPosSim	WorstPosSim
1 s	303 s	0	289

AvgSimBefQC	AvgSimAfterQC	AvgDispErrBefQC	AvgDispErrAftQC
171,347	171,347	2,75709	2,75709

DispErrSDvBefQC	DispErrSDvAftQC	RayHits1	RayHits2
1,39548	1,39548	762.871	725.839

MAMatchingSteps	EpiMatchingSteps	QualityMatches	MatchingErrors
725.839	0	725.839	0

Tabelle 5.36.: Messdaten für Test 5.1.1.4-a: Translationsfehler in der äußeren Orientierung der ersten Kamera, Integer-Matching, SAD, Suchbereich 9 x 9 px, Muster- und Suchfenstergröße 17 x 17 px

Stereopaar 1 (planar, 1.024 x 768 Pixel)

TreeClsDuration	MatchDuration	BestPosSim	WorstPosSim
1 s	300 s	0	289

AvgSimBefQC	AvgSimAfterQC	AvgDispErrBefQC	AvgDispErrAftQC
195,881	195,881	2,75073	2,75073

DispErrSDvBefQC	DispErrSDvAftQC	RayHits1	RayHits2
1,66075	1,66075	756.000	729.367

MAMatchingSteps	EpiMatchingSteps	QualityMatches	MatchingErrors
729.367	0	729.367	0

Tabelle 5.37.: Messdaten für Test 5.1.1.4-b: Rotationsfehler in der äußeren Orientierung der ersten Kamera, Integer-Matching, SAD, Suchbereich 9 x 9 px, Muster- und Suchfenstergröße 17 x 17 px

Stereopaar 1 (planar, 1.024 x 768 Pixel)

TreeClsDuration	MatchDuration	BestPosSim	WorstPosSim
<1 s	8.469 s	0	289

AvgSimBefQC	AvgSimAfterQC	AvgDispErrBefQC	AvgDispErrAftQC
101,99	101,99	14,7419	14,7419

DispErrSDvBefQC	DispErrSDvAftQC	RayHits1	RayHits2
9,76473	9,76473	762.871	725.839

MAMatchingSteps	EpiMatchingSteps	QualityMatches	MatchingErrors
725.839	0	725.839	0

Tabelle 5.38.: Messdaten für Test 5.1.1.4-c: Translationsfehler in der äußeren Orientierung der ersten Kamera, Integer-Matching, SAD, Suchbereich 81 x 42 px, Muster- und Suchfenstergröße 17 x 17 px

Stereopaar 1 (planar, 1.024 x 768 Pixel)

TreeClsDuration	MatchDuration	BestPosSim	WorstPosSim
1 s	6.730 s	0	289

AvgSimBefQC	AvgSimAfterQC	AvgDispErrBefQC	AvgDispErrAftQC
125,151	125,151	9,52665	9,52665

DispErrSDvBefQC	DispErrSDvAftQC	RayHits1	RayHits2
6,45404	6,45404	756.000	729.367

MAMatchingSteps	EpiMatchingSteps	QualityMatches	MatchingErrors
729.367	0	729.367	0

Tabelle 5.39.: Messdaten für Test 5.1.1.4-d: Rotationsfehler in der äußeren Orientierung der ersten Kamera, Integer-Matching, SAD, Suchbereich 51 x 51 px, Muster- und Suchfenstergröße 17 x 17 px

Wie die Messwerte bestätigen, führen die ungenauen relativen Orientierungen zu einer Abnahme der Zuordnungsqualität. Die durchschnittlichen Abstandswerte der Tests 5.1.1.4-a und 5.1.1.4-b liegen um mehr als 70% bzw. 96% über denen des Laufs 5.1.1.3-a, der die gleiche Matchingkonfiguration bei korrekter Ausrichtung der Kameras zum Modell verwendet. Durch die Vergrößerung des Suchbereichs verbessert sich jeweils der mittlere Übereinstimmungsgrad, erreicht jedoch aufgrund der gestiegenen Wahrscheinlichkeit, dass mehrere Suchfenster der Musterumgebung ähneln, nicht die Zuordnungsgüte des Referenztests aus dem vorangegangenen Abschnitt.

Dass der mittlere Disparitätsfehler und die zugehörige Standardabweichung nach der Suchbereichserweiterung in den Läufen 5.1.1.4-c und 5.1.1.4-d nicht absinken, hängt damit zusammen, dass sich die ground truth-Werte auf die fehlerbehafteten externen Parameter der virtuellen Kameras beziehen. Weil die verwendeten Bilder jedoch auf der korrekten Orientierung basieren, beschreiben die Angaben *nicht* die Differenz und deren Streuung der mittels Matching gefundenen zu den „wahren falschen“ Verschiebungen, wie sie der DaTool-Raytracer anhand der modifizierten Standort- und Rotationsdaten berechnet. Vielmehr charakterisieren die Messwerte den zweidimensionalen Positionsfehler, welchen die ungenaue Ausrichtung in den Stereoaufnahmen hinterlässt. Dies gilt jedoch nur dann, wenn die orientierungsbedingte Ortsabweichung in den Bildern die halbierte horizontale und vertikale Suchbereichsgröße unterschreitet, also ein Fehlerkompensationsversuch erfolgt. Bei den Läufen 5.1.1.4-a und 5.1.1.4-b, die mit einem Suchbereich von 9 x 9 Pixeln die Bedingung nicht erfüllen, bewegen sich die Ergebnisse unter AvgDispErrBefQC, AvgDispErrAftQC, DispErrSDvBefQC und DispErrSDvAftQC dagegen praktisch zufällig in einem Intervall von 0 ... 4,5 Pixeln.

Selbst wenn man versucht ist, einen erhöhten durchschnittlichen Disparitätsfehler und dessen Standardabweichung vom orientierungsbedingt fehlerbehafteten Modell bei einem hinreichend großen Suchbereich als „Fehler vom Fehler“ und damit positiven Indikator für die Güte der Zuordnungen zu interpretieren, eignen sich die ground truth-bezogenen Messwerte dennoch nicht als Qualitätskriterium. Da sie mit den inkorrekten Stereoverschiebungen von falschen Korrespondenzen suchbereichsabhängig überlagert sind, ist nicht mehr erkennbar, ob die Zuordnung stimmig und ein Anstieg allein durch die ungenügende Ausrichtung der Aufnahmen zur 3D-Szene verursacht wird (positiv) oder die scheinbar gefundene inhaltliche Übereinstimmung nicht mit dem Musterbildausschnitt korreliert (negativ).

Abbildung 5.6 zeigt die Graustufen-Übersichtsbilder für alle Tests zur relativen Orientierung. In der oberen Hälfte werden dabei die korrekt ausgerichteten Originalaufnahmen denjenigen Bildern gegenübergestellt, die sich beim Rendern mit dem POV-Ray-Werkzeug unter Verwendung der fehlerbehafteten externen Kameraparameter ergeben.

5.1.2. Modellverbesserung

Der Untersuchung des integrierten Matchers schließt sich die Evaluierung des Verfahrens zur Modellverbesserung an. Die hierfür benötigten Eingabedaten – eine deformierte Variante der künstlichen 3D-Szene und ein Satz von Disparitätskarten – müssen zunächst in vorgelagerten Arbeitsschritten generiert werden.

5.1.2.1. Modelldeformation

Für die Erzeugung der gestörten 3D-Daten wird die exakte grob aufgelöste Szene A mit dem DaTool-Plugin zur Modelldeformation bearbeitet, wobei realistischerweise eine Kombination aus dem strahl- und dem achsenorientierten Störverfahren zum Einsatz kommt. Während mit ersterem zunächst diejenigen LRF-Erfassungsfehler simuliert werden, welche durch das Material oder ungünstige Auftreffwinkel des Laserimpulses entstehen, soll letzteres anschließend die weniger starken Ungenauigkeiten der Auslenkungsmechanik des Scanners nachbilden. Tabelle 5.40 listet die verwendeten Deformationsparameter für beide Rauschgeneratoren auf, und Abbildung 5.7 zeigt die Gestalt der 3D-Szene aus Sicht des DaTool-Raytracers.

Strahlorientierte Deformation	
Standort des Laserscanners im Modell	(2; 0; -5)
Zufallsintervall für die Punktverschiebung entlang des Laserstrahls	-0,15 ... 0,15 LE
Achsenorientierte Deformation	
Zufallsintervall in X-Richtung	-0,05 ... 0,05 LE
Zufallsintervall in Y-Richtung	-0,05 ... 0,05 LE
Zufallsintervall in Z-Richtung	-0,05 ... 0,05 LE

Tabelle 5.40.: Parameter für die Modelldeformation zur Evaluierung des Modellverbesserungsalgorithmus

5.1.2.2. Matching

Mit der entstellten 3D-Szene und den zylindrischen Stereoaufnahmen wird anschließend die Korrespondenzanalyse bei aktivierter Objektpunktschätzung durchgeführt. Um die Deformation beim Matching zu kompensieren, bedarf die Suchbereichsgröße einer Anpassung an die Stärke der Modellstörung und die minimale Objektentfernung von der zweiten Stereokamera.

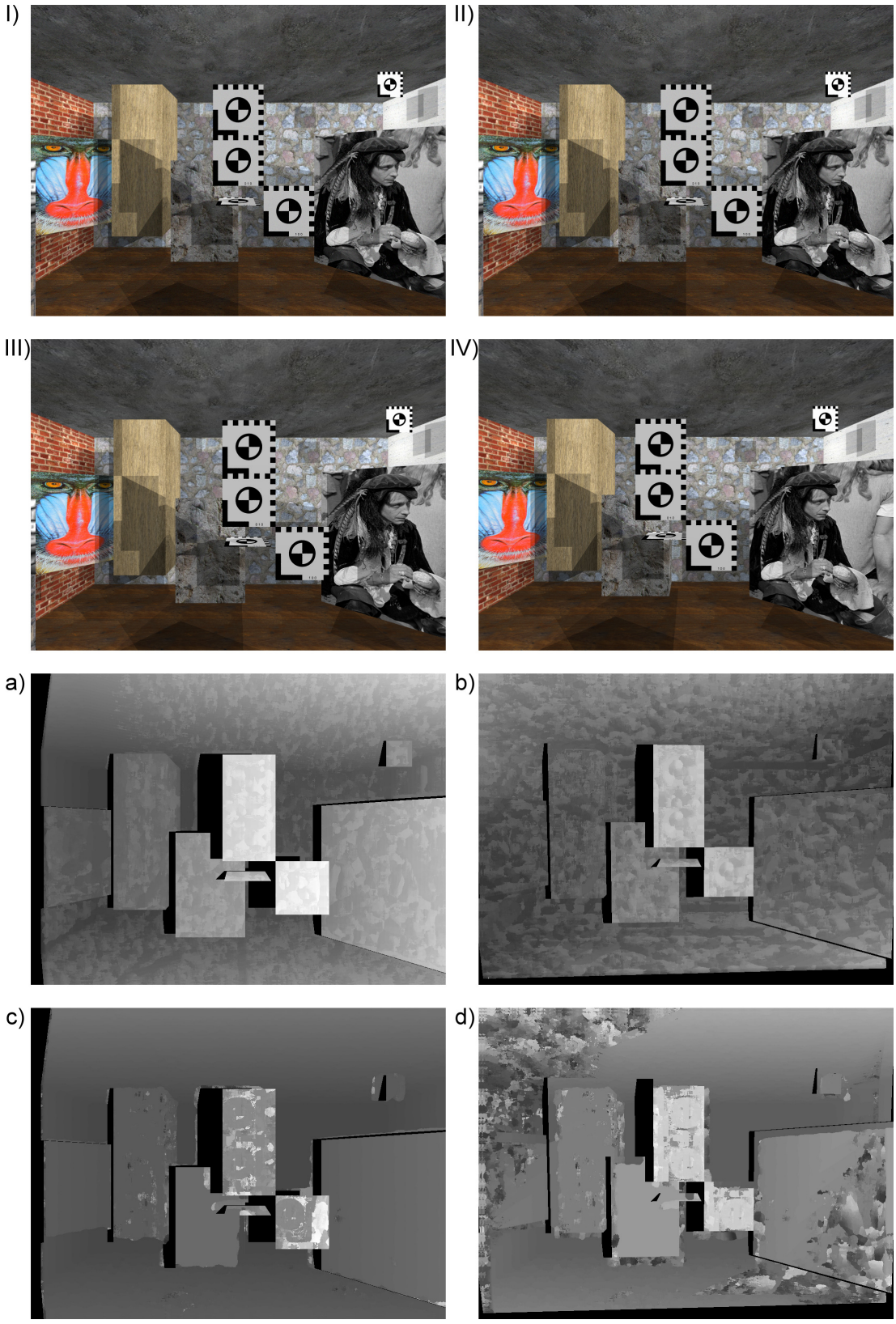


Abbildung 5.6.: Planare Graustufen-Übersichtsbilder der Testläufe 5.1.1.4-a bis 5.1.1.4-d. Die Buchstaben fallen mit denen der Testbezeichnungen zusammen. (I) und (II) stellen beide das Bild #1 des Stereopaars 1 dar, (III) und (IV) darunter sind mit der modifizierten Kameraposition bzw. den geänderten Drehwinkeln gerendert.

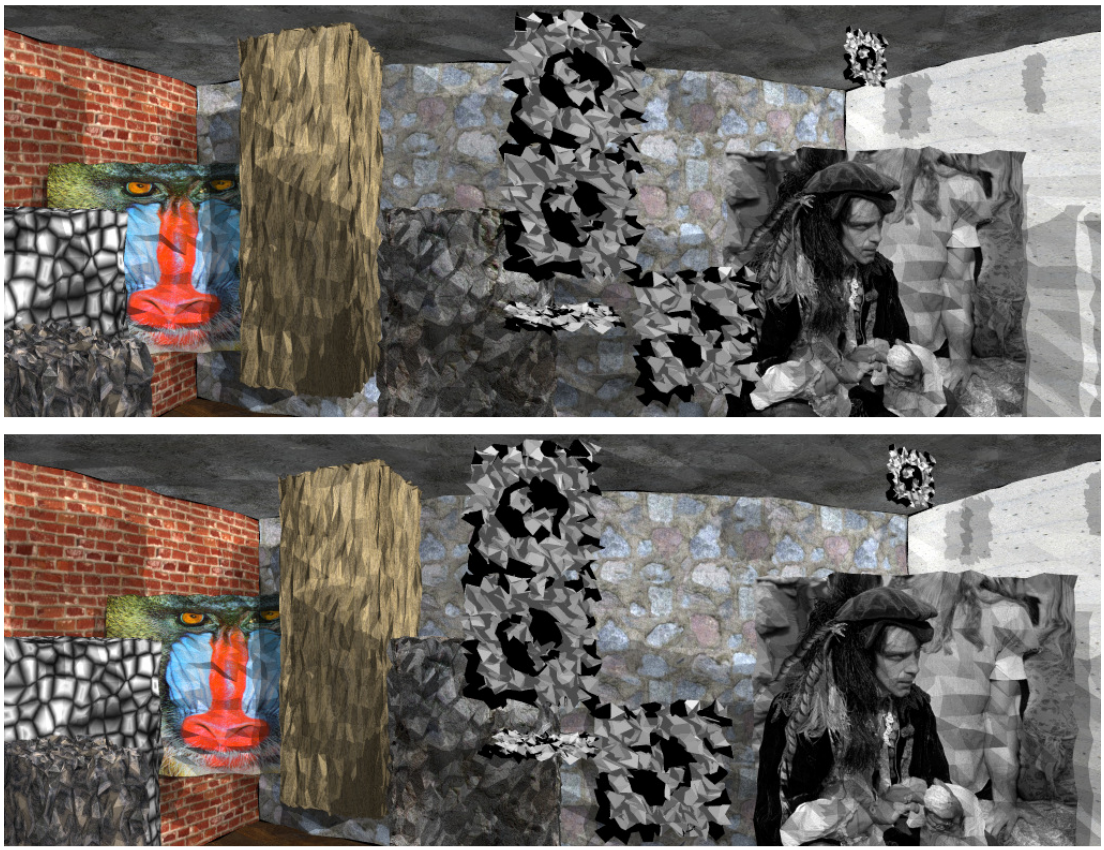


Abbildung 5.7.: Deformiertes LRF-Modell aus Sicht des DaTool-Raytracers für die erste (oben) und zweite virtuelle Stereokamera (unten).

Unter Verwendung der Notation aus dem Grundlagenkapitel sei für die Schätzung des Rechteckareals, in dem das Suchfenster verschoben wird, angenommen, dass die Sichtstrahlen R_{k1} bei der Objektpunktschätzung vom Zentrum der ersten Kamera durch das aktuelle Musterbildpixel verlaufen und auf das LRF-Modell treffen (Abbildung 5.8). Dabei bewirkt die Deformation, dass der berechnete fehlerbehaftete Schnittpunkt $\mathbf{X}_\mathbf{k}^{\prime(\mathbf{w})}$ von der Kreuzung $\mathbf{X}_\mathbf{k}^{(\mathbf{w})}$ mit dem unverfälschten realen Objekt, wie es das Stereosystem weitestgehend abbildet, abweicht. Diese Differenz überträgt sich auf das zweite Kamerabild und bewirkt dort eine Positionsveränderung des Suchbereichszentrums $\mathbf{X}_\mathbf{k}^{(\mathbf{i})}$ nach $\mathbf{X}_\mathbf{k}^{\prime(\mathbf{i})}$. Demnach kann die korrekte Zuordnung wie beim Ausgleich einer ungenauen relativen Orientierung der Kameras zum LRF-Modell nur dann bestimmt werden, wenn der Suchbereich um $\mathbf{X}_\mathbf{k}^{\prime(\mathbf{i})}$ den Punkt $\mathbf{X}_\mathbf{k}^{(\mathbf{i})}$ mit einschließt.

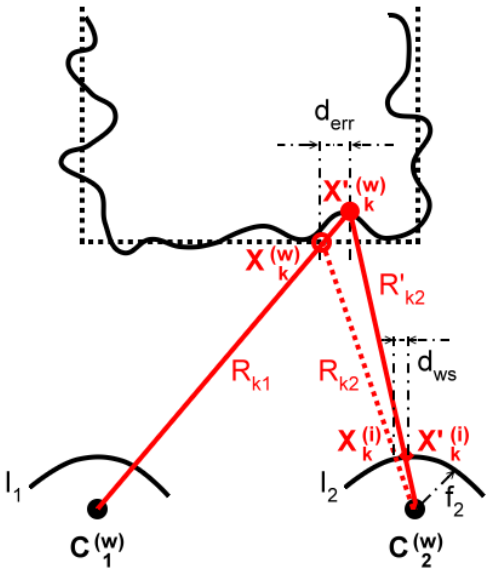


Abbildung 5.8.: Schätzung der Suchbereichsgröße für die Korrespondenzanalyse bei einem deformierten Modell. Im Vergleich zum tatsächlichen Szenenobjekt (schwarz gestrichelt) resultiert die fehlerhafte Erfassung (schwarz durchgezogen) in einem abweichenden Modellschnittpunkt, der als verschobenes Suchbereichszentrum $\mathbf{X}_\mathbf{k}^{\prime(\mathbf{i})}$ im zweiten Kamerabild in Erscheinung tritt.

Sofern der Höchstbetrag der Deformation respektive der Schnittpunktdifferenz d_{err} im Modell näherungsweise bekannt ist und diese sich maximal in der zweiten Stereoaufnahme manifestiert (Entstellung primär in der Höhe/Breite wie in Abbildung 5.8, keine merkliche Verkleinerung durch die projektive Verzerrung), kann man die Suchbereichsgröße aus der maximalen Lageverschiebung des Suchbereichszentrums d_{ws} mit dem Strahlensatz überschlagen. Es gilt

$$\frac{d_{err}}{d_{min}} = \frac{\hat{d}_{ws}}{f_2} \qquad \hat{d}_{ws} = f_2 \frac{d_{err}}{d_{min}} \qquad d_{ws} = \frac{\hat{d}_{ws}}{d_{pmax}} \tag{5.1}$$

wobei f_2 die Brennweite, $d_{pmax} = max(d_{px}, d_{py})$ das Maximum der horizontalen und vertikalen Pixelgröße und $d_{min} = min_k(\{\left|\mathbf{X}_\mathbf{k}^{\prime(\mathbf{w})} - \mathbf{C}_\mathbf{2}^{(\mathbf{w})}\right| \mid \mathbf{X}_\mathbf{k}^{\prime(\mathbf{i})} \in I_2\})$ den ungefähren minimalen euklidischen Abstand aller im Suchbild I_2 sichtbaren Szenenobjekte von der zweiten Kamera bezeichnet.

Diese Rechnung geht nahezu fehlerfrei auf und liefert ein aussagekräftiges Ergebnis, wenn die Tangente zur Bildfläche im Projektionspunkt $\mathbf{X}_{\mathbf{k}}^{\prime(\mathbf{i})}$ parallel zum Differenzvektor $\mathbf{X}_{\mathbf{k}}^{(\mathbf{w})} - \mathbf{X}_{\mathbf{k}}^{\prime(\mathbf{w})}$ ist und dessen Maximalbetrag d_{err} sehr viel kleiner als d_{min} ist. Beim Raytracing auftretende deformationsbedingte Strahltreffer auf nahen Objekten, die bei Verwendung eines exakten Modells eigentlich auf entfernt liegenden Gegenständen zustande kommen und umgekehrt, deckt Formel 5.1 ebensowenig ab wie der aus der Bestimmung der relativen Orientierung der Kameras zur 3D-Szene resultierende ungefähre Positionsfehler im Bild. Die sich in diesen Situationen vor allem an den Objektkanten ergebenden Disparitätsabweichungen können den berechneten Wert d_{ws} abhängig von der Entfernungsdifferenz um ein Vielfaches übersteigen, wie sich mit der in Abschnitt 2.1.4.3 entwickelten Formel zur Auflösungsschätzung nachvollziehen lässt.

Vernachlässigt man die Verschiebungsanteile durch den soeben skizzierten Typ von Fehltreffern, lässt sich die erforderliche Ausdehnung des Suchbereichs für die Kompensation der simulierten Erfassungsfehler beim Matching für das mit den oben aufgeführten Einstellungen gestörte Testraummodell berechnen. Das Einsetzen der Deformationsstärke aus Tabelle 5.40, der Kameraparameter für das zylindrische Stereopaar aus Tabelle 5.2 und der minimalen Objektentfernung, die aus der Karte der künstlichen Szene in Anhang A.1 hervorgeht (metal box), führt auf

$$\begin{aligned} f_2 &= 0,00488923984 \text{ LE} & d_{err} &= 0,2 \text{ LE} \\ d_{min} &\approx 3 \text{ LE} & d_{pmax} &= 0,00001 \text{ LE} \end{aligned} \tag{5.2}$$

und somit

$$d_{ws} \approx 33 \text{ px} . \tag{5.3}$$

Dabei setzt sich d_{err} im schlimmsten Fall additiv aus der strahlunabhängigen und der strahlbezogenen Störung zusammen. Unter der Annahme, dass sich die Modelldeformation gleichmäßig in der Horizontalen und Vertikalen des Suchbildes auswirkt, muss das Suchfenster in jedem lokalen Matchingschritt im Intervall von ±33 Pixeln um die aus der Objektpunktschätzung resultierende Suchbildposition verschoben werden. Beaufschlagt man den anderthalbfachen Betrag als Sicherheitsfaktor, ergibt sich eine Suchbereichsgröße von 99 x 99 Bildpunkten. Dies ist angesichts der geringen Störung von akkumuliert einem Prozent bezogen auf das Maximum der Testraumabmessungen ein beachtlicher Wert, der einen nicht unerheblichen Zeitaufwand für die Korrespondenzanalyse impliziert. Um die Zahl der mit dem Musterfenster zu korrelierenden Suchfenster zu reduzieren, kann es daher sinnvoll sein, die Abmessungen des Suchareals für jede Bilddimension getrennt zu bestimmen oder die Anordnung der Stereokameras auszunutzen, die die Vorzugsrichtung der Epipolarlinien festlegt, entlang welcher die Korrespondenzen lokalisiert sind.

Die folgenden Tests 5.1.2.2-a bis 5.1.2.2-c untersuchen die Kompensationswirkung beim Matching anhand von drei verschiedenen Suchbereichsabmessungen, darunter auch die geschätzte Größe von 99 x 99 Pixeln. In den Tabellen 5.42 bis 5.44 sind die Angaben aus den Ausführungsprotokollen des DaTool-Plugins für die integrierte Zuordnungsbestimmung aufgeführt. Abschnitt 5.1.1.1 erläutert die Abkürzungen der verwendeten Messgrößen.

Test 5.1.2.2-a Test 5.1.2.2-a verwendet die in Tabelle 5.41 zusammengefassten Einstellungen für den integrierten Matcher. Die Suchbereichsgröße für den hybriden hierarchi-

schen Algorithmus gleicht mit initial 9 x 9 Pixeln die Deformation des Modells gemäß der vorgenommenen Schätzung nicht aus. Weil die resultierenden Disparitätskarten auch für den Verbesserungsprozess unter Berücksichtigung des gesamten abgebildeten Teils der künstlichen Szene verwendet werden sollen, bleibt die Qualitätsprüfung mit einem Schwellwert von 1,0 deaktiviert. Außerdem wird die epipolare Zuordnungssuche zugelassen, weil durch die Entstellung „Löcher“ im Testraum auftreten und dessen Abgeschlossenheit aufbrechen.

Objektpunktschätzung	
Level des 3D-Baums	5
Initiale Kantenlängen der Baumquader	8 x 8 x 8 LE
Unterteilungsfaktor	2 x 2 x 2
Kleinste Boxabmessungen	0,5 x 0,5 x 0,5 LE
Strahlverfolgungsmethode	inkrementell
Korrespondenzanalyse	
Region of Interest (ROI)	gesamtes Bild
Qualitätsschwellwert	1,0
Integer-Algorithmus	
Ähnlichkeitsfunktion	SAD
Verwendeter Farbkanal	RGB, perzeptuell gewichtet
Suchbereichsgröße	9 x 9 Pixel
Korrelationsfenstergröße	17 x 17 Pixel
Korrelationsschrittweite	1 x 1 Pixel
effektive Suchschrittweite	1 x 1 Pixel
Geman-McClure-Parameter σ	5,0
Gleitkomma-Algorithmus	
Ähnlichkeitsfunktion	SAD
Verwendete Farbkanäle	RGB, perzeptuell gewichtet
Subpixelinterpolation	bilinear (Dreiecksfilter)
Suchbereichsgröße	3 x 3 Pixel
Korrelationsfenstergröße	17 x 17 Pixel
Korrelationsschrittweite	1 x 1 Pixel
effektive Suchschrittweite	0,25 x 0,25 Pixel
Geman-McClure-Parameter σ	5,0

Tabelle 5.41.: Parameter für den integrierten Matchingalgorithmus in Test 5.1.2.2-a

Test 5.1.2.2-b Test 5.1.2.2-b verwendet die Einstellungen seines Vorgängers, erweitert jedoch den Suchbereich für die ganzzahlige Stufe der hybriden hierarchischen Zuordnungsbestimmung auf 99 x 99 Bildpunkte.

Test 5.1.2.2-c Test 5.1.2.2-c wird im Wesentlichen mit den Einstellungen von Test 5.1.2.2-a durchgeführt, jedoch beträgt die Suchbereichsgröße für die erste Stufe des hybridhierarchischen Matchingverfahrens 201 x 21 Pixel. Die Abmessungen sollen die bei der Objektpunktschätzung auftretenden deformationsbedingten Strahltreffer, welche fälschlicherweise zu nah am oder zu weit entfernt vom Stereosystem liegen, über die Zuordnungs-

suche kompensieren. Weil die sinusförmigen Epipolarlinien aufgrund der waagerechten Anordnung der virtuellen Zylinderkameras innerhalb der Horizontalausdehnung des Fensters einer Kurzuntersuchung nach recht flach verlaufen, ist die Reduktion der Vertikalausdehnung auf 21 Bildpunkte zur Verringerung des Zeitbedarfs der Korrespondenzanalyse praktisch verlustfrei. Außerdem bleibt die optionale und standardmäßig aktivierte Sichtbarkeitsprüfung mittels des Raytracings von der zweiten Stereokamera aus deaktiviert. Dadurch sollen auch diejenigen Bildbereiche beim Matching berücksichtigt werden, die zwar in beiden Aufnahmen erscheinen, bei der Sichtbarkeitsprüfung auf dem deformierten Modell jedoch irrtümlich als verdeckt angenommen werden. Um die in den Bildern vorhandenen tatsächlichen Abschattungen zu filtern, beträgt der empirisch festgelegte relative Qualitätsschwellwert für die SAD-Distanzen nach der Ausreißerkorrektur 0,4 , die Qualitätsprüfung der Zuordnungen ist also aktiviert. Zur visuellen Begutachtung wird in einem separaten Matchinglauf ohne Korrespondenzselektion außerdem das vollständiges Übersichtsbild berechnet, ohne dass die resultierenden Messergebnisse an dieser Stelle aufgeführt sind.

Stereopaar 2 (zylindrisch, 1.024 x 384 Pixel)

TreeClsDuration	**MatchDuration**	**BestPosSim**	**WorstPosSim**
1 s	2.038 s	0	289

AvgSimBefQC	**AvgSimAfterQC**	**AvgDispErrBefQC**	**AvgDispErrAftQC**
105,026	105,026	0,713191	0,713191

DispErrSDvBefQC	**DispErrSDvAftQC**	**RayHits1**	**RayHits2**
1,07316	1,07316	361.553	319.110

MAMatchingSteps	**EpiMatchingSteps**	**QualityMatches**	**MatchingErrors**
319.110	476	319.586	0

Tabelle 5.42.: Messdaten für Test 5.1.2.2-a: Hybrid-hierarchisches Matching, SAD, Suchbereich 9 x 9 / 3 x 3 px, Muster- und Suchfenstergröße jeweils 17 x 17 px, Korrelations- und Suchschrittweite 1 x 1 / 0,25 x 0,25 px, bilineare Subpixelinterpolation

Stereopaar 2 (zylindrisch, 1.024 x 384 Pixel)

TreeClsDuration	**MatchDuration**	**BestPosSim**	**WorstPosSim**
1 s	11.996 s	0	289

AvgSimBefQC	**AvgSimAfterQC**	**AvgDispErrBefQC**	**AvgDispErrAftQC**
104,044	104,044	3,39963	3,39963

DispErrSDvBefQC	**DispErrSDvAftQC**	**RayHits1**	**RayHits2**
10,7132	10,7132	361.553	319.110

MAMatchingSteps	**EpiMatchingSteps**	**QualityMatches**	**MatchingErrors**
319.110	471	319.581	0

Tabelle 5.43.: Messdaten für Test 5.1.2.2-b: Hybrid-hierarchisches Matching, SAD, Suchbereich 99 x 99 / 3 x 3 px, Muster- und Suchfenstergröße jeweils 17 x 17 px, Korrelations- und Suchschrittweite 1 x 1 / 0,25 x 0,25 px, bilineare Subpixelinterpolation

Stereopaar 2 (zylindrisch, 1.024 x 384 Pixel)

TreeClsDuration	MatchDuration	BestPosSim	WorstPosSim
2 s	7.712 s	0	289

AvgSimBefQC	AvgSimAfterQC	AvgDispErrBefQC	AvgDispErrAftQC
111,062	81,2101	6,55044	3,22274

DispErrSDvBefQC	DispErrSDvAftQC	RayHits1	RayHits2
16,6187	10,3467	361.553	361.553

MAMatchingSteps	EpiMatchingSteps	QualityMatches	MatchingErrors
361.553	476	210.878	0

Tabelle 5.44.: Messdaten für Test 5.1.2.2-c: Hybrid-hierarchisches Matching, SAD, Suchbereich 201 x 21 / 3 x 3 px, Muster- und Suchfenstergröße jeweils 17 x 17 px, Korrelations- und Suchschrittweite 1 x 1 / 0,25 x 0,25 px, bilineare Subpixelinterpolation, Qualitätsschwellwert 0,4

Der resultierende mittlere Distanzfehler der Zuordnungen von Test 5.1.2.2-a entspricht recht genau dem Ergebnis von Test h aus Abschnitt 5.1.1.3 vor der Qualitätsprüfung, der bei nahezu identischer Konfiguration auf dem ungestörten künstlichen Modell operiert. Wie die zugehörige Graustufen-Übersichtskarte in Abbildung 5.9 vermuten lässt, genügt der relativ kleine Suchbereich offenbar, um die vornehmlich in der Tiefe wirksamen Deformationen im dominierenden Innenbereich der Objektflächen weitestgehend zu kompensieren. Diejenigen Modellstörungen, welche sich in den Stereoaufnahmen in horizontaler und vertikaler Richtung manifestieren und vor allem als Zacken an den Kanten der abgebildeten Testraumgegenstände hervortreten, werden dagegen beim Matching mit dem 9 x 9 Pixel großen Areal um die bei der Objektpunktschätzung berechnete Bildposition nicht ausgeglichen.

Der auf die abgeschätzten 99 x 99 Pixel vergrößerte Suchbereich in Test 5.1.2.2-b korrigiert mehr Störungen, was sich in einem geringfügig niedrigeren mittleren SAD-Wert im Vergleich zu seinem Vorgänger wiederspiegelt. Die angestiegenen ground truth-bezogenen Messergebnisse sind aufgrund des entstellten Modells gemäß der Argumentation des vorangegangenen Abschnitts abermals nicht aussagekräftig. Dass die positive Zahl der epipolaren Matchingschritte, die die Existenz von „Löchern“ in der deformierten Szene belegt, trotz der identischen 3D-Eingabedaten bei diesem Lauf von den anderen Untersuchungen abweicht, ist auf die von der Suchbereichsgröße abhängige Abtastung der Epipolarlinien im zweiten Kamerabild zurückzuführen und stellt keine Fehlfunktion dar.

Die Ausführungsdauer von Test 5.1.2.2-b wächst wegen der zahlreichen Suchfenster, die in jedem lokalen Matchingschritt mit dem aktuellen Musterbildausschnitt verglichen werden, auf das Sechsfache in Relation zum ursprünglichen 9 x 9 Pixel fassenden Suchbereich an. Sie skaliert damit weitaus besser, als die Verelffachung der Seitenlänge des quadratischen Zielgebiets für die Zuordnungsbestimmung eigentlich erwarten lässt. Offenbar tragen die Subpixelinterpolation und die Farbraumumwandlung bei der Initialisierung der klasseninternen Zwischenspeicher in den `CMatcher`-Implementierungen im Gegensatz zur Distanzwertberechnung mit den ganzzahlig indizierten Schleifen erheblich zum gesamten Rechenaufwand bei. Die Verwendung des lokalen Puffermechanismus für die miteinander zu korrelierenden Bildinformationen ist demnach gerechtfertigt.

Test 5.1.2.2-c erweitert den Suchbereich auf 201 Pixel in der Waagerechten und kompensiert bei Betrachtung der zugehörigen Graustufen-Übersichtsbitmaps auch diejenigen Modellstörungen, die bei der Objektpunktschätzung zu Strahltreffern auf fälschlicherweise

zu nah bzw. zu weit entfernt gelegenen Gegenständen in der 3D-Szene führen. Der dadurch verursachte zusätzliche Disparitätsbetrag vor allem an den Objektkanten der mittig gelegenen Passpunktmarken wird aufgefangen, weshalb die Zacken verschwinden. Allerdings zeigt der im Vergleich zu den anderen Läufen dieses Abschnitts gestiegene mittlere SAD-Abstand eine Verschlechterung der Güte der Korrespondenzen vor der Qualitätsprüfung an, die primär auf die deaktivierte Sichtbarkeitsanalyse und weniger auf die gestiegene Wahrscheinlichkeit einer Fehlzuordnung aufgrund der großen Zahl an Suchfensterkandidaten zurückzuführen ist. Dennoch führt letztere zu den auch bei Test 5.1.2.2-b auftretenden Artefakten vor allem im Deckenbereich, auf den zweifarbigen zentralen Kalibrationsmarkierungen und der vertikal angeordneten länglichen wood box des generierten Raums.

Nach der Qualitätsprüfung, die die verdeckten Bildanteile wirksam ausblendet, liegt der durchschnittliche Distanzbetrag für die verbleibenden Korrespondenzen zwar höher als bei Verwendung des deformationsfreien Modells in Lauf 5.1.1.3-h. Er sinkt jedoch signifikant unter die Werte der beiden Vorgängertests, weswegen zumindest ein partieller Ausgleich von Störungen in der 3D-Szene nach dem Matching auch mit realem Datenmaterial durchaus möglich erscheint.

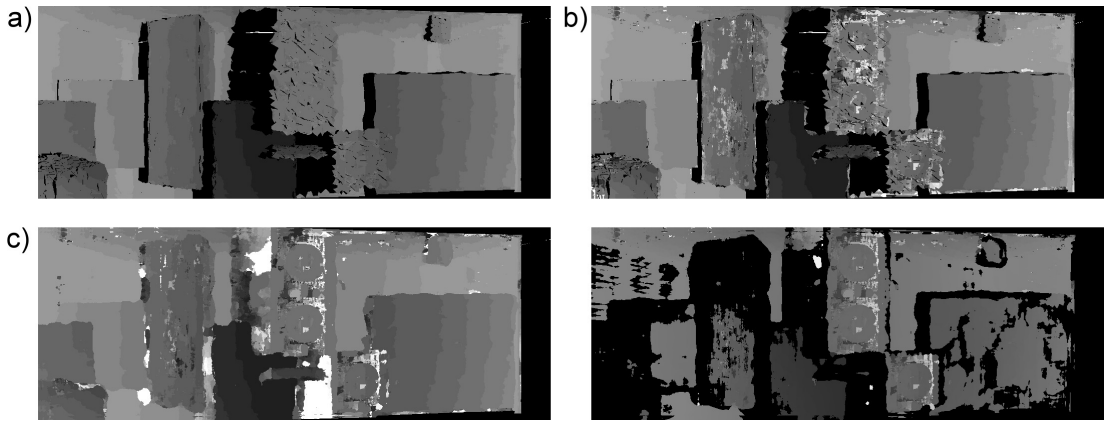


Abbildung 5.9.: Graustufen-Übersichtsbilder der Testläufe 5.1.2.2-a bis 5.1.2.2-c, bei letzterem ohne (links) und mit (rechts) aktivierter Qualitätsprüfung. Die Buchstaben vor den Bildern decken sich mit denen der Testbezeichnungen. Aus drucktechnischen Gründen sind die normalisierungsbedingt dunkleren und scheinbar zweifarbigen Originalbitmaps des Matchers mit nachträglich erhöhter Helligkeit und gesteigertem Kontrast dargestellt.

5.1.2.3. Korrektur des gestörten Modells

Mit den vom integrierten Matcher ausgegebenen Gleitkomma-Disparitätskarten des zylindrischen Stereopaars wird das deformierte 3D-Modell korrigiert. Die Raytracingeinstellungen für das DaTool-Plugin zur Modellverbesserung entsprechen denen der Zuordnungsbestimmung (5 Oktalbaumlevel, initiale Boxgröße 8 LE pro Raumdimension, inkrementelle Strahlverfolgung). Zur Interpolation der Verschiebungswerte an den Gleitkommakoordinaten der projizierten Polygoneckpunkte, die nicht zwangsläufig mit einer ganzzahligen Kartenpositionen zusammenfallen, wird stets der kubische B-Spline-Filter verwendet.

Die statistischen Informationen, die der Modellverbesserungsalgorithmus in Gestalt eines Konfigurationsdatenobjekts ausgibt, sind für die zur Untersuchung des Korrekturverhal-

tens durchgeführten Tests 5.1.2.3-a bis 5.1.2.3-d analog zur Korrespondenzanalyse in speziellen Tabellen protokolliert. Als Legende für die abgekürzten Messgrößen in den Kopfzeilen dient dabei die folgende Aufstellung, wobei sich die Numerierung an die Auflistung aus Abschnitt 4.7.4.3 anlehnt. Es bezeichnen

(1) *RTDuration* die Zeitdauer für das Raytracing zur Identifikation der Polygoneckpunkte des Modells, die in der ROI der ersten Stereoaufnahme liegen (raytracing duration),

(2) *RTMdlCrDuration* den Zeitbedarf für die Erstellung des Raytracingmodells (raytracing model creation duration),

(3) *CorrectDuration* die Ausführungszeit für die eigentliche Modellpunktkorrektur mittels Punktverschiebung (correction duration),

(4) *UniqueCorrPts* die Anzahl derjenigen vorhandenen Modellpunkte ohne Doubletten, die mit den vorliegenden Stereoaufnahmen korrigierbar sind,

(5) *CorrectedPts* die Anzahl der tatsächlich korrigierten vorhandenen Modellpunkte,

(6) *AddPhotoPts* die Anzahl der rein photogrammetrisch gewonnenen 3D-Punkte für die Verdichtung des Modells und

(7) *OutputPts* die Anzahl der Raumpunkte des resultierenden 3D-Modells nach der Korrektur und Verdichtung .

Test 5.1.2.3-a Test 5.1.2.3-a verwendet die horizontale und vertikale Disparitätskarte von Test 5.1.2.2-a. Die Verdichtung der Szene mit rein photogrammetrisch rekonstruierten Objektpunkten bleibt deaktiviert.

Test 5.1.2.3-b Test 5.1.2.3-b nutzt die Disparitäten von Test 5.1.2.2-b für die Modellverbesserung ohne Verdichtung.

Test 5.1.2.3-c Test 5.1.2.3-c korrigiert die vorhandenen Modellpunkte mit den qualitativ akzeptablen Disparitäten von Test 5.1.2.2-c. Eine Verdichtung der Szene findet nicht statt.

Test 5.1.2.3-d Test 5.1.2.3-d wird mit den gleichen Eingabedaten wie sein Vorgänger durchgeführt, die Modellverdichtung ist jedoch aktiviert. Es werden alle photogrammetrisch rekonstruierbaren zusätzlichen Raumpunke berücksichtigt. Eine Unterabtastung der doppelten Schnittpunkte des Sichtstrahls ausgehend von der ersten Kamera mit den Polygonen der Szene findet nicht statt.

Modell A (37.544 Polygone)

RTDuration	RTMdlCrDuration	CorrectDuration	UniqueCorrPts
18 s	<1 s	<1 s	19.929

CorrectedPts	AddPhotoPts	OutputPts
15.798	0	15.798

Tabelle 5.45.: Messdaten für Test 5.1.2.3-a

Modell A (37.544 Polygone)

RTDuration	RTMdlCrDuration	CorrectDuration	UniqueCorrPts
17 s	<1 s	1 s	19.929

CorrectedPts	AddPhotoPts	OutputPts
15.798	0	15.798

Tabelle 5.46.: Messdaten für Test 5.1.2.3-b

Modell A (37.544 Polygone)

RTDuration	RTMdlCrDuration	CorrectDuration	UniqueCorrPts
18 s	1 s	<1 s	19.929

CorrectedPts	AddPhotoPts	OutputPts
9.167	0	9.167

Tabelle 5.47.: Messdaten für Test 5.1.2.3-c

Modell A (37.544 Polygone)

RTDuration	RTMdlCrDuration	CorrectDuration	UniqueCorrPts
18 s	<1 s	1 s	19.929

CorrectedPts	AddPhotoPts	OutputPts
9.167	207.898	217.065

Tabelle 5.48.: Messdaten für Test 5.1.2.3-d

Die tabellierten Messwerte für die Modellverbesserung belegen, dass die Detektion der Modelloberfläche mittels der Strahlverfolgung von der ersten Stereokamera aus praktisch die gesamte benötigte Rechenzeit in Anspruch nimmt. Der Ausschluss von multiplen Schnittpunkten auf ein- und demselben Polygon verlangsamt allerdings den Vorgang von 14 s (vgl. Abschnitt 5.1.1.3, der dort aufgeführte Wert von 28 s bedarf aufgrund des lediglich einfach durchgeführten Raytracings der Halbierung) auf 17 ... 18 s trotz der Pufferung des jeweils zuletzt getroffenen Dreiecks.

Abbildung 5.10 veranschaulicht die vom Modellverbesserungsplugin ausgegebenen 3D-Punktwolken, die bei der visuellen Begutachtung erwartungsgemäß mit den Ergebnissen des Matchings korrespondieren. Die Störungen in der Tiefe respektive entlang der Z-Achse des Testraums werden mit dem Verschiebungsverfahren vermindert. Wie Abstandsmessun-

gen an repräsentativen Raumpunkten auf den Vorder- und Rückseiten der stone box- und man poster-Oberflächen ergeben, sinkt der Fehler dabei von 0,3 und 0,25 Längeneinheiten in der unkorrigierten Szene auf 0,06 und 0,17 Längeneinheiten nach der Verbesserung mit den Disparitätskarten aus Test 5.1.2.2-b . Die Darstellungen für die Tests 5.1.2.2-a, 5.1.2.2-c und 5.1.2.2-d deuten auf ähnliche Genauigkeitsgewinne hin. In den beiden letztgenannten Rekonstruktionen verlaufen die Kanten der zentral gelegenen Passpunktmarkierungen in der Frontalansicht aufgrund des verwendeten ausgedehnten horizontalen Suchbereichs überdies wiederum deutlich gerader.

Das Hinzufügen von zusätzlichen rein photogrammetrisch berechneten zu den verschobenen Raumpunkten im vierten Lauf dieses Abschnitts vervielfacht deren Gesamtzahl um mehr als das Zwanzigfache. Trotz der Auflösung der qualitativ selektierten Disparitäten von 0,25 Pixeln ist die diskrete Anordnung der neuen Szenenelemente in der Draufsicht von Abbildung 5.10(d) deutlich zu erkennen. Dies gilt vor allem für diejenigen 3D-Punkte, welche aus den Korrespondenzen im projizierten Deckenbereich des Testraums resultieren.

Obwohl eine Steigerung in der Modellqualität nach Anwendung des Verbesserungsverfahrens auszumachen ist, finden sich dennoch in allen Darstellungen auch Raumpunktgruppen, die auf fehlerhafte Zuordnungen zurückzuführen sind. Dies ist auch dann der Fall, wenn die Qualitätsprüfung (welche übrigens durchaus ihre mess- und sichtbare Berechtigung hat) wie im Lauf 5.1.2.2-c aktiviert ist. Eine weitere Nachbearbeitungsstufe (3D-Rauschfilter) kann die Zahl der mit Hilfe der photogrammetrischen Daten verschobenen oder neu hinzugefügten inexakten nulldimensionalen Raumelemente weiter reduzieren, wird aber im Allgemeinen nicht ohne ein gewisses Maß an Benutzerinteraktion auskommen. Eine vollständige Elimination ist aufgrund des inhärent schlecht gestellten Korrespondenzanalyseproblems praktisch nicht zu erwarten und nur für Szenenteile zu erreichen.

5.2. Reale Szene

Obschon die Untersuchung des integrierten Matchers mit der künstlichen Testwelt bislang aufschlussreiche Testergebnisse geliefert hat, steht die Korrespondenzanalyse und die Modellverbesserung mit realistischen Eingabedaten für eine abschließende Bewertung der im Rahmen dieser Arbeit entwickelten Verfahren noch aus. Um diese Lücke zu schließen, evaluieren die folgenden Abschnitte die beiden funktionalen Algorithmen unter Verwendung einer dreidimensionalen Repräsentation und eines Satzes von zylindrischen Stereopanoramen der Altarkuppel der Wallfahrtskirche zum Gegeißelten Heiland auf der Wies (Wieskirche) nahe Steingaden in Südbayern. Der Sakralbau stammt aus dem achtzehnten Jahrhundert und weist eine filigrane Innenraum- und Farbgestaltung auf, wie sie für die Rokokozeit typisch ist.

5.2.1. Matching

Das 3D-Modell der Kuppel für die modellgestützte Zuordnungsbestimmung setzt sich aus mehreren vermaschten Punktwolken zusammen, welche aus der berührungslosen Erfassung mit dem Laserscanner IMAGER 5003 des DLR resultieren. Tabelle 5.49 listet die Charakteristika der Szene auf, welche in verschiedenen Ansichten durch Abbildung 5.11 veranschaulicht wird.

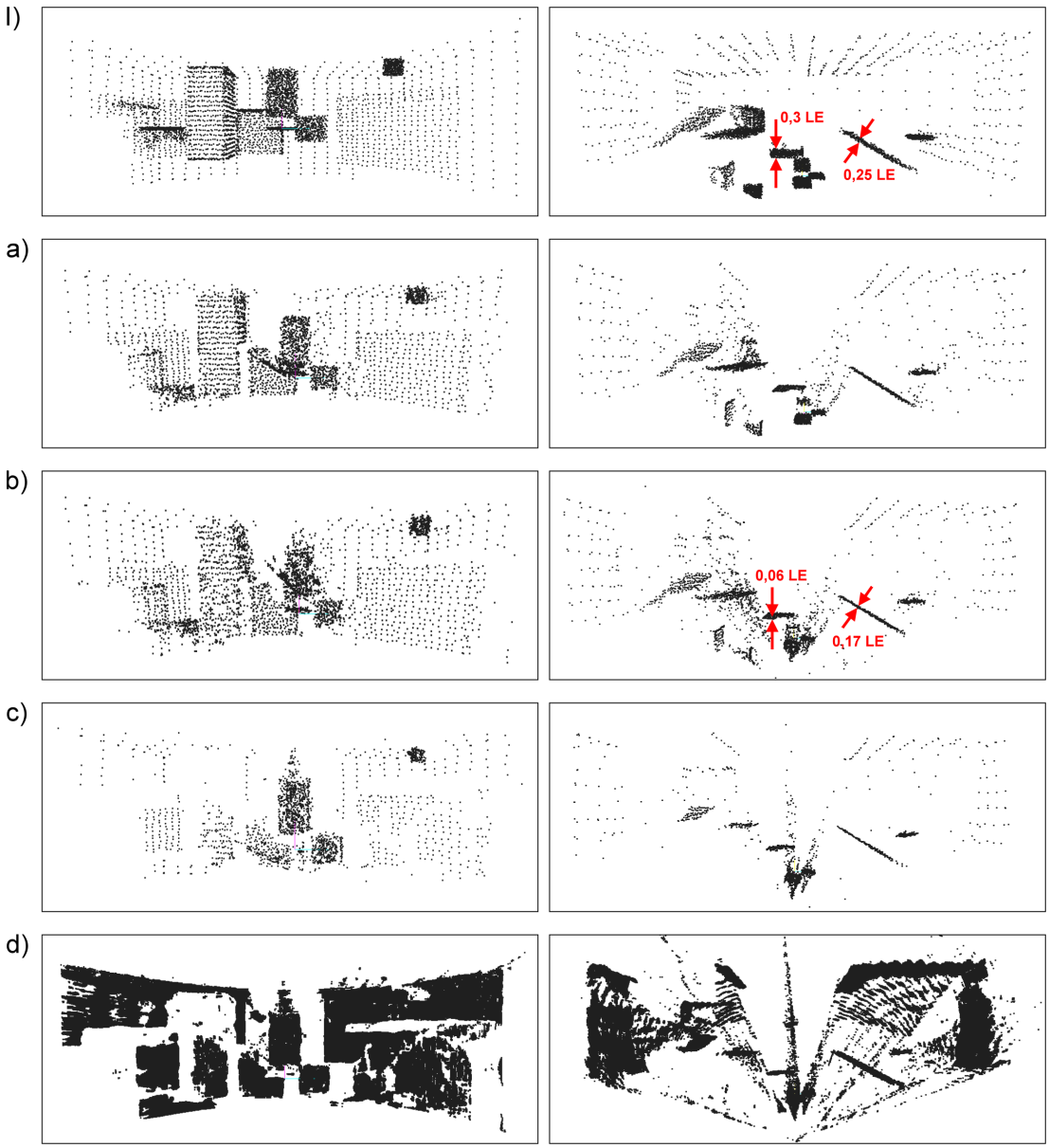


Abbildung 5.10.: 3D-Punktwolken des deformierten Testraums (I) vor der Modellverbesserung und (a-d) nach der Durchführung der Tests 5.1.2.3-a bis 5.1.2.3-d, jeweils in der Frontansicht (links) und Draufsicht (rechts)

Die zugehörigen Stereobilder mit einer Farbauflösung von acht Bit pro RGB-Komponente sind das Ergebnis einer Aufnahmenserie mit der hochauflösenden EyeScan-Kamera. Tabelle 5.50 enthält die Parameter der inneren und der nachträglich bestimmten relativen äußeren Orientierung zur LRF-Szene. Aufgrund von Ungenauigkeiten in den Drehwinkeln und dreidimensionalen Positionsangaben der beiden Stereokameras ergibt sich – gemittelt über vier verteilte und mit den berechneten Ausrichtungen projizierte Raumpunkte des Modells – eine durchschnittliche Abweichung der Suchbereichszentren zu deren vermuteter exakter Bildlage von etwa achtzehn Bildpunkten. Abbildung 5.12(a) zeigt die für die subpixelgenaue Zuordnungsbestimmung genutzte region of interest des Musterbildes mit insgesamt 56,6 Megapixeln, was etwa einem Fünftel der Auflösung des vollständigen ersten Panoramas entspricht. Die ROI überträgt sich beim Matching automatisch auf einen gleichgroßen korrespondierenden Ausschnitt im Suchbild, den Abbildung 5.12(b) darstellt.

Als Längenmaß für das Weltkoordinatensystem wird realitätsbezogen das Meter verwendet. Die SI-Basiseinheit, in der die vom Laserscanner ermittelten Distanzen vor der Modellerzeugung vorliegen, überträgt sich auf die „koppelnden“ internen Kameraparameter Brennweite und Pixelgröße und deckt sich mit deren metrischer Spezifikation im Datenblatt des photographischen Aufnahmegeräts.

Eigenschaft	**Modell der Altarkuppel**
Abmessungen	15,0657 x 10,58711 x 3,77826 m
Anzahl der Raumpunkte insgesamt	54.137
Anzahl der Dreiecke insgesamt	92.290
Speicherplatzbedarf (VRML-Datei)	5,1 MiB

Tabelle 5.49.: Eigenschaften des 3D-Modells von der Altarkuppel der Wieskirche.

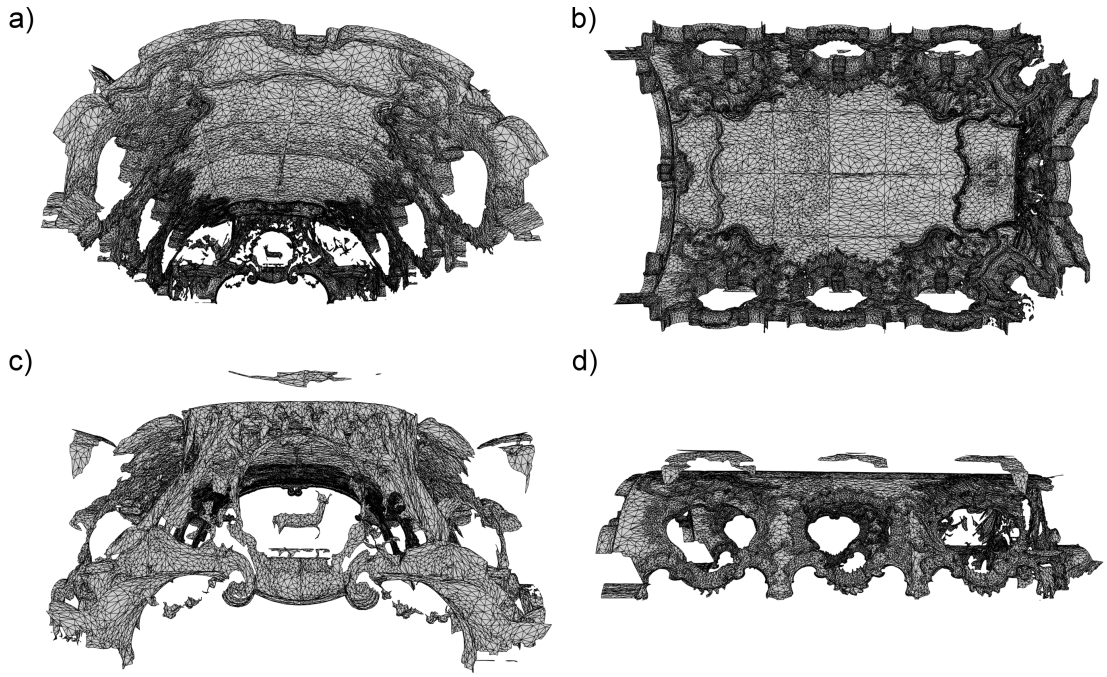


Abbildung 5.11.: 3D-Modell der Altarkuppel der Wieskirche (a) von vorn, (b) von unten, (c) in Rück- und (d) in Seitenansicht

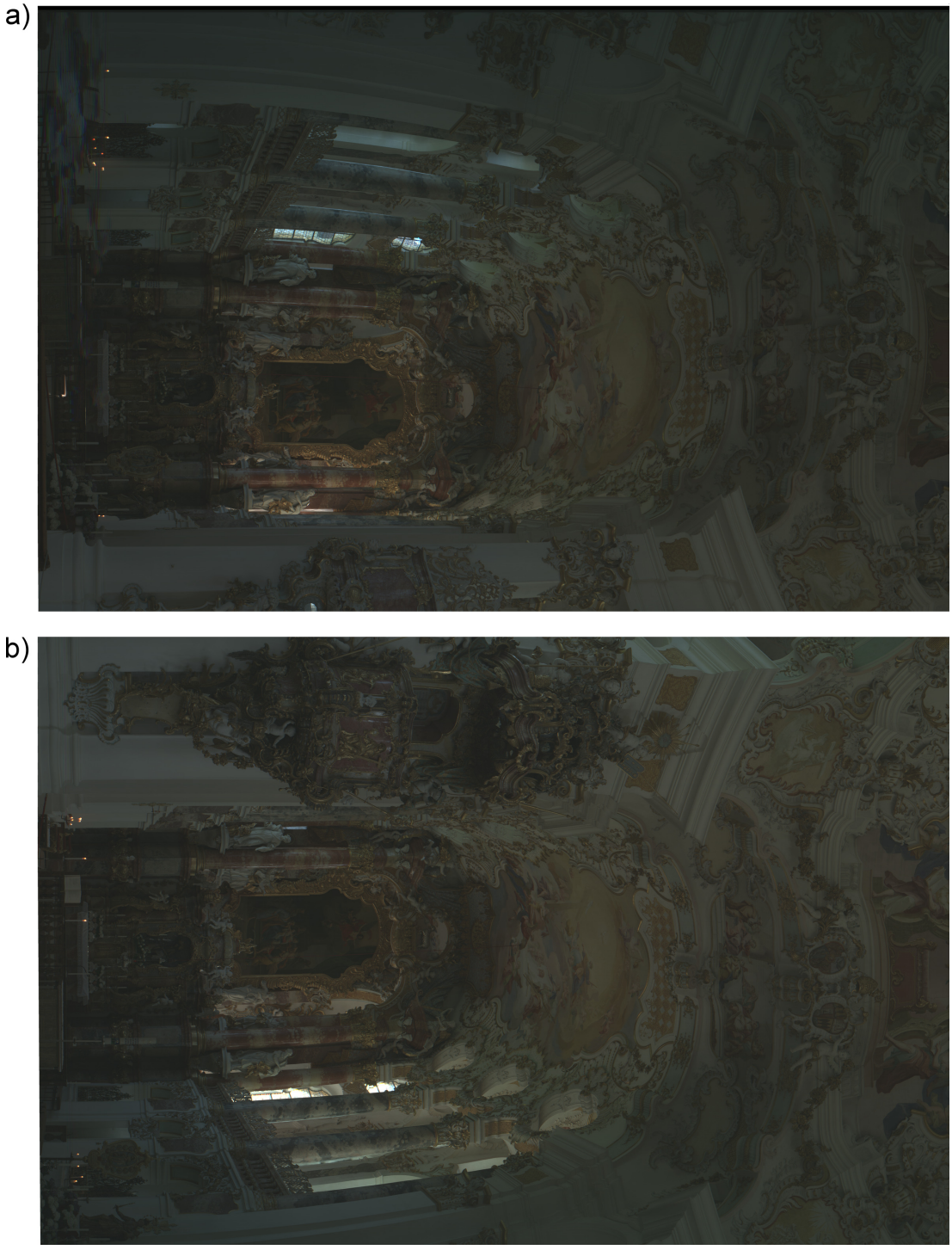


Abbildung 5.12.: (a) Muster- und (b) Suchbild-ROI aus den zylindrischen Stereopanoramen von der Altarkuppel der Wieskirche

Parameter	Stereobilder der Altarkuppel	
	Bild #1	**Bild #2**
Kamerazentrum	(-0,980011; -2,813414; -0,058147)	(-0,148670; 3,139229; -0,075410)
Rotation ω, ϕ, κ	(180,1°; 9,14°; 166,6°)	(180.795°; 7.975°, 169.595°)
Projektionstyp	zylindrisch	
Sensor-/Bildpixel	25.722 x 10.298	25.626 x 10.298
Pixelgröße	0,000007 x 0,000007 m	
Brennweite	0,06 m	
Kamerahauptpunkt	(12.860,5; 5.148,5)	(12.812,5; 5.148,5)
Sichtfeld	171,939° x 61,99°	171,297° x 61,988°

Tabelle 5.50.: Kameraparameter zu den photographischen Aufnahmen der Altarkuppel der Wieskirche

Aus den mit der auslagernden `CFastGridImage`-Klasse verwalteten Bildern (Pufferung von 512 Kacheln à 64 x 64 RGB-Pixel, insgesamt 6 MiB Daten), den vollständig im RAM vorliegenden Modellinformationen sowie den aufgeführten Kameraparametern und Matcheinstellungen in Form von insgesamt drei maschinenlesbaren XML-Konfigurationsdatensätzen resultieren die in Tabelle 5.52 aufgeführten Messergebnisse nach der Korrespondenzanalyse. Die Übereinstimmungssuche mit den in Tabelle 5.51 zusammengefassten Einstellungen erfolgt ausschließlich mit aktivierter Objektpunktschätzung, und die Qualitätsprüfung ist mit einem relativen Schwellwert von 0,25 aktiviert. Zur vollständigen visuellen Bewertung der Zuordnungsgüte wird parallel auch mit abgeschaltetem Selektionsmechanismus gematcht. Abbildung 5.13 gibt die ausgegebenen Graustufen-Übersichtsbilder wieder.

Modell und Stereopaar der Wieskirche (92.290 Polygone, 9.216 x 6.144 Pixel)

TreeClsDuration	**MatchDuration**	**BestPosSim**	**WorstPosSim**
2 s	267.014 s	0	289

AvgSimBefQC	**AvgSimAfterQC**	**AvgDispErrBefQC**	**AvgDispErrAftQC**
85,8876	31,4104	7,87617	6,56092

DispErrSDvBefQC	**DispErrSDvAftQC**	**RayHits1**	**RayHits2**
4,57395	4,23193	8.858.121	8.244.930

MAMatchingSteps	**EpiMatchingSteps**	**QualityMatches**	**MatchingErrors**
8.244.930	0	3.950.871	0

Tabelle 5.52.: Messdaten nach dem Matching mit dem Modell und den Stereoaufnahmen von der Altarkuppel der Wieskirche

Wie deutlich wird, ist der Zeitbedarf für die einmalige Klassifikation der Modellpolygone in die Boxen der sechsstufigen Oktalbaumstruktur praktisch vernachlässigbar. Dagegen nimmt der Matchingvorgang trotz der deaktivierten Zuordnungssuche entlang der Epipolarlinien und der optimistisch gewählten Abmessungen für die miteinander zu korrelierenden Bildausschnitte und den Suchbereich knapp über drei Tage in Anspruch. Diese lange Zeitspanne ist nach den Ergebnissen auf Basis des synthetischen Datenmaterials nur zu einem geringen Teil auf die Ein- und Auslagerung der jeweils in Bearbeitung befindlichen Bildausschnitte zurückzuführen. Sie stellt die praktische Anwendbarkeit des integrierten Korrespondenzanalyseverfahrens ohne den Einsatz von Spezialhardware oder die Paralle-

Objektpunktschätzung	
Level des 3D-Baums	6
Initiale Kantenlängen der Baumquader	8 x 8 x 8 m
Unterteilungsfaktor	2 x 2 x 2
Kleinste Boxabmessungen	0,25 x 0,25 x 0,25 m
Strahlverfolgungsmethode	rekursiv
Korrespondenzanalyse	
Region of Interest (ROI)	(0; 0) ... (9.215; 6.143)
Qualitätsschwellwert	0,25
Integer-Algorithmus	
Ähnlichkeitsfunktion	SAD
Verwendeter Farbkanal	RGB, perzeptuell gewichtet
Suchbereichsgröße	25 x 25 Pixel
Korrelationsfenstergröße	17 x 17 Pixel
Korrelationsschrittweite	1 x 1 Pixel
effektive Suchschrittweite	1 x 1 Pixel
Geman-McClure-Parameter σ	5,0
Gleitkomma-Algorithmus	
Ähnlichkeitsfunktion	SAD
Verwendete Farbkanäle	RGB, perzeptuell gewichtet
Subpixelinterpolation	bilinear (Dreiecksfilter)
Suchbereichsgröße	3 x 3 Pixel
Korrelationsfenstergröße	17 x 17 Pixel
Korrelationsschrittweite	1 x 1 Pixel
effektive Suchschrittweite	0,1 x 0,1 Pixel
Geman-McClure-Parameter σ	5,0

Tabelle 5.51.: Matchingparameter für die Korrespondenzanalyse auf den Panoramen der Wieskirche

lisierung der Berechnungen, welche aufgrund der funktionalen Unabhängigkeit der lokalen Matchingschritte prinzipiell realisierbar ist, teilweise in Frage.

Die Güte der Korrespondenzen ist, wenn man den durchschnittlichen SAD-Ähnlichkeitsgrad unreflektiert betrachtet, sowohl vor als auch nach der qualitativen Selektion höher als beim Matching mit den zylindrischen Aufnahmen des künstlichen Modells und ähnlichen Analyseparametern. Aufgrund der geringeren Helligkeit, die mit 53,8 bzw. 54,6 Intensitätsstufen für das Muster- bzw. Suchbild des Altars die Werte der generierten Zylinderprojektionen halbiert, verkehrt sich die getroffene Aussage jedoch ins Gegenteil. Nach der Verdoppelung der mittleren SAD-Beträge von ursprünglich 85,8876 und 31,4104 vor respektive nach der Qualitätsprüfung auf 171,7752 und 62,8208 , um vergleichbare Werte zu erhalten, liegt der durchschnittliche Übereinstimmungsgrad jeweils um ganze 67% bzw. ca. 18% über den Angaben des Tests i aus Abschnitt 5.31.

Verantwortlich für die geringe Zuordnungsgüte ist wohl primär die unzureichende Ausrichtung der Panoramen zur 3D-Szene, was partiell an der problematischen manuellen Auswahl von bekannten Modellpunkten in den Aufnahmen liegt. Wegen der hohen Bildauflösung treten selbst feine Details des Wieskirchenaltars als homogene Flächen in Erscheinung und lassen sich in den Projektionen nicht mit hinreichender Genauigkeit identifizieren. In der Folge sind die für die Herstellung der relativen Orientierung benötigten 3D-2D-Zuordnungen ungenau. Abgesehen davon führen auch Bildstörungen, wie sie Abbildung 5.14 exemplarisch darstellt, zu qualitativ unzureichenden Korrespondenzen für die Musterpixel. In den Graustufen-Übersichtsbildern äußern sich die genannten Ursachen bei genauer Betrachtung als Flecken und lokal begrenzte Intensitätswertstörungen.

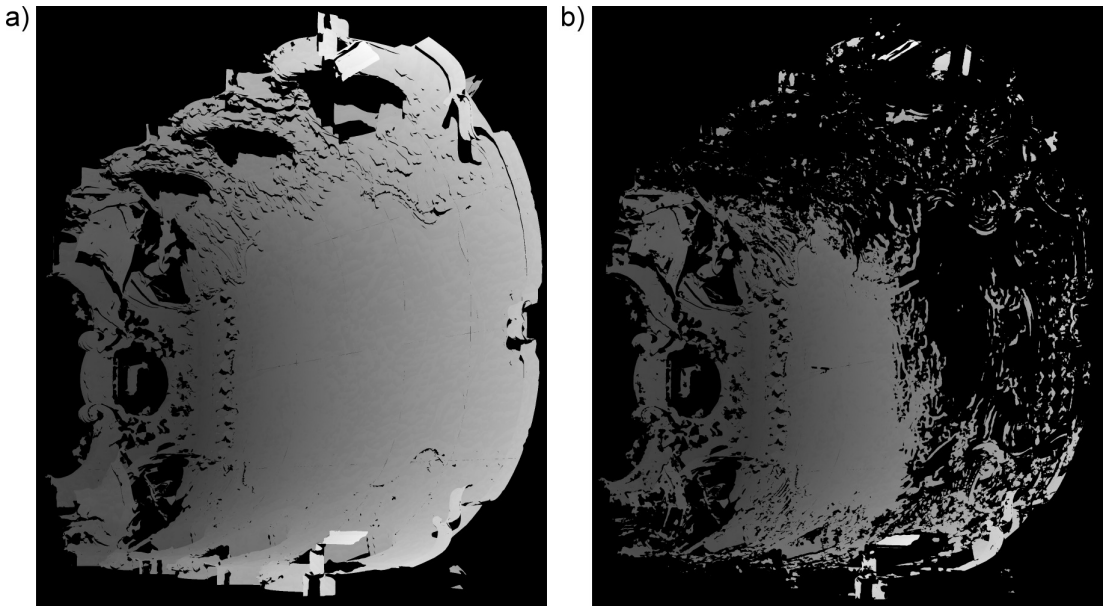


Abbildung 5.13.: Graustufen-Übersichtsbild von der Altarkuppel der Wieskirche nach dem Matching (a) ohne und (b) mit aktivierter Qualitätsprüfung. Aus Platzgründen ist die ROI nicht vollständig dargestellt, sondern auf den relevanten Bereich beschnitten.

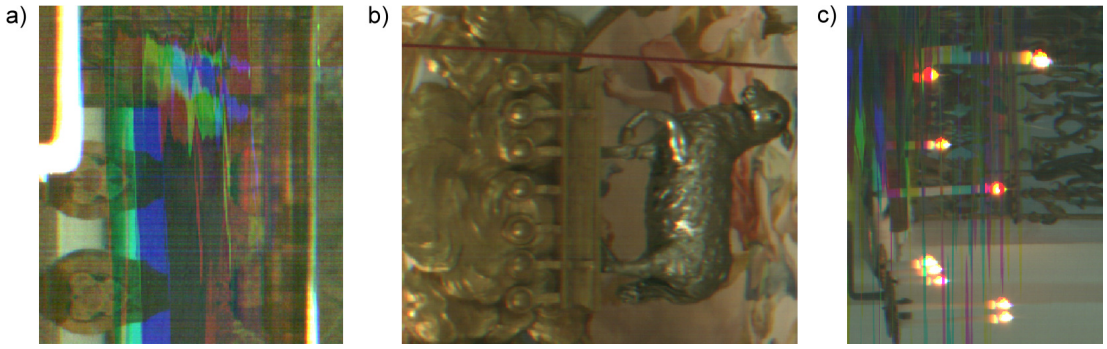


Abbildung 5.14.: Ausgewählte Störungen in den Aufnahmen von der Altarkuppel der Wieskirche (a) Fehlfarben und feine Querstreifen (b) monochrome Einzellinien (c) Lichtpunkte

5.2.2. Modellverbesserung

Die Disparitätskarten, die aus dem Matching auf den zylindrischen Stereopanoramen der Altarkuppel der Wieskirche mit und ohne aktivierter Qualitätsprüfung resultieren, bilden den Ausgangspunkt für die Verbesserung des LRF-Modells in den Tests 5.2.2-a und 5.2.2-b. Die statistischen Angaben der Ausführungsprotokolle des für die Berechnung eingesetzten DaTool-Plugins sind in den Tabellen 5.53 und 5.54 aufgeführt, wobei die Abkürzungen für die Messgrößen in der Kopfzeilen der Ergebnisübersicht in Abschnitt 5.1.2.3 erklärt werden.

Test 5.2.2-a Test 5.2.2-a verwendet für die Beschleunigung des Raytracings zur Bestimmung der im ersten Stereobild sichtbaren Dreieckseckpunkte einen Oktalbaum mit sechs Hierarchiestufen und einer initialen Boxgröße von 8 x 8 x 8 m. Die Strahlverfolgung läuft rekursiv ab, wobei die Halbgeraden ausschließlich für die Bildpunkte innerhalb der beim Matching verwendeten ROI konstruiert werden. Zur Interpolation der Disparitäten kommt der kubische B-Spline-Filter zum Einsatz. Es werden keine rein photogrammetrisch bestimmten 3D-Punkte der Ergebnisszene hinzugefügt.

Test 5.2.2-b Test 5.2.2-b wird mit nahezu den gleichen Einstellungen wie Test 5.2.2-a durchgeführt, jedoch verdichten rein photogrammetrisch wiedergewonnene Raumpunkte die Punktwolke nach der Anwendung des Verschiebungsverfahrens. Dabei findet ein Achtel der Sichtstrahltreffer Berücksichtigung, die pro Bildzeile auf ein und demselben Modellpolygon auftreten (Abtastrate=8).

Modell der Wieskirche (92.290 Polygone), alle Zuordnungen

RTDuration	RTMdlCrDuration	CorrectDuration	UniqueCorrPts
9.020 s	<1 s	31 s	204.492

CorrectedPts	AddPhotoPts	OutputPts
152.339	0	152.339

Modell der Wieskirche (92.290 Polygone), qualitativ akzeptable Zuordnungen

RTDuration	RTMdlCrDuration	CorrectDuration	UniqueCorrPts
8.922 s	<1 s	32 s	204.492

CorrectedPts	AddPhotoPts	OutputPts
82.911	0	82.911

Tabelle 5.53.: Messdaten für Test 5.2.2-a: Korrektur der originalen Modellpunkte

Modell der Wieskirche (92.290 Polygone), alle Zuordnungen

RTDuration	RTMdlCrDuration	CorrectDuration	UniqueCorrPts
9.630 s	2 s	89 s	204.492

CorrectedPts	AddPhotoPts	OutputPts
152.339	1.002.346	1.154.685

Modell der Wieskirche (92.290 Polygone), qualitativ akzeptable Zuordnungen

RTDuration	RTMdlCrDuration	CorrectDuration	UniqueCorrPts
9.103 s	1 s	69 s	204.492

CorrectedPts	AddPhotoPts	OutputPts
82.911	490.375	573.286

Tabelle 5.54.: Messdaten für Test 5.2.2-b: Korrektur der originalen Modellpunkte und Verdichtung

Nach den tabellierten Messwerten benötigt der Modellverbesserungsalgorithmus wie schon auf den Daten der künstlichen Szene erheblich weniger Zeit, als das subpixelgenau durchgeführte Matching. Der größte Anteil an der Ausführungsdauer wird wiederum vom Raytracing zur Oberflächendetektion in Anspruch genommen. Dabei macht sich der rechnerische Mehraufwand zur Verdichtung der korrigierten Punktwolke in einer geringfügigen Steigerung von ca. 150 auf 160 Minuten ohne und von ca. 148 auf 152 Minuten mit aktivierter Qualitätsprüfung (weniger gültige Disparitäten für die Rekonstruktion der zusätzlichen nulldimensionalen Raumelemente) bemerkbar.

Bedingt durch die Speicherung der jeweils 432 MiB umfassenden Gleitkomma-Disparitätskarten in den auslagernden `CFastGridImage`-Instanzen, die partiell für den Zeitbedarf der rekursiven Strahlverfolgung mitverantwortlich zeichnen, erfolgt auch der eigentliche Verbesserungsvorgang nicht mehr in „Nullzeit“. Vielmehr benötigt der Korrekturprozess im besten Fall (Test 5.2.2-a) rund eine halbe Minute für die Verschiebungsoperationen und die optionale Verdichtung der Szene, obwohl sich die Zahl der Raumpunkte im Vergleich zum grob aufgelösten Testraum kaum mehr als verdoppelt hat. Im Gegensatz dazu ist die Funktion für die Erstellung des Raytracingmodells, welches vollständig im RAM abgelegt wird, nur von marginalen Verzögerungen betroffen

In Bezug auf die Güte der korrigierten Raumpunkte übertragen sich die fehlerbehafteten Zuordnungen aus der Korrespondenzanalyse auf das dreidimensionale Endergebnis und führen zu einem sichtbaren Rauschen (Abbildung 5.15). Lediglich in einigen abgegrenzten Bereichen, wie der in Abbildung 5.16 dargestellten Tierfigur, ist – nach Aktivierung der Verdichtung – tatsächlich eine nennenswerte qualitative Verbesserung auszumachen. Die

Rundung am Bauch ist im Gegensatz zum Ausgangsmodell deutlicher nachgezeichnet. Die photogrammetrische Präzision, die am Ohr der Metallfigur bei knapp unter zwei Millimetern auf eine Entfernung von ca. 31 m vom Stereosystem liegt, dürfte die des LRF am gleichen Punkt weit übersteigen. Aufgrund der ungenauen Ausrichtung der Kameras bleibt ihr Wert für die Entfernungsmessung, wo die äußere Orientierung der Aufnahmegeräte möglichst exakt bekannt sein muss, im Unterschied zu reinen (Teil-)Rekonstruktionsaufgaben allerdings begrenzt.

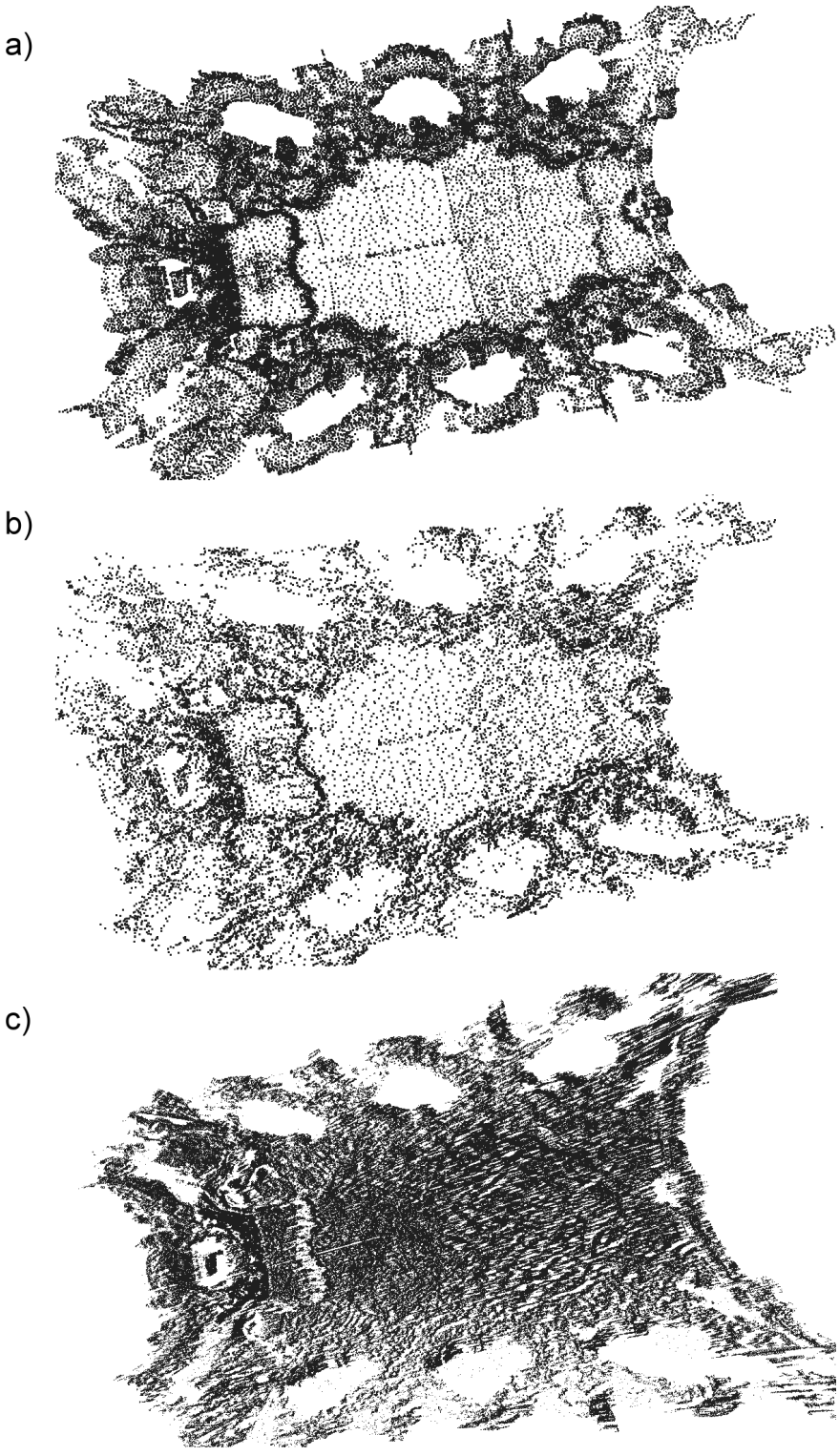


Abbildung 5.15.: 3D-Punktwolken von der Altarkuppel der Wieskirche vor und nach Anwendung des Verbesserungsverfahrens bei Berücksichtigung aller vorhandenen Zuordnungen (a) Raytracing-modell (b) Ursprüngliche LRF-Daten nach der Modellverbesserung ohne neue Raumpunkte (c) LRF-Daten nach der Modellverbesserung mit zusätzlichen rein photogrammetrisch rekonstruierten 3D-Punkten bei achtfacher Unterabtastung der multiplen Strahl-Dreieck-Schnittpunkte. Aus darstellungstechnischen Gründen ist die letztgenannte Punktwolke mit einer geringeren Punktgröße gerendert.

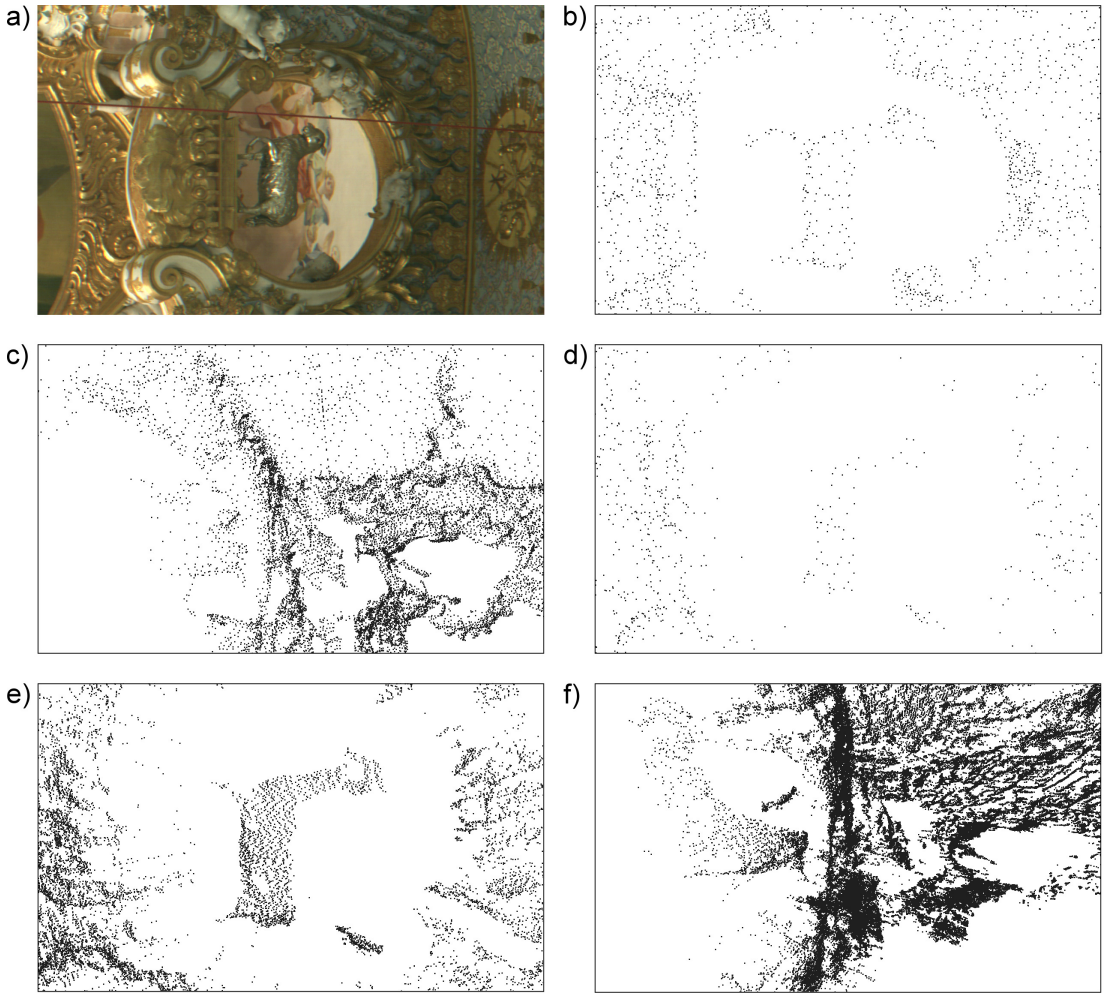


Abbildung 5.16.: 3D-Punktwolke der Tierfigur in der Altarkuppel der Wieskirche vor und nach Anwendung des Verbesserungsverfahrens auf Basis der mit aktivierter Qualitätsprüfung bestimmten Zuordnungen (a) Panoramaausschnitt der Figur (b) zugehöriger Ausschnitt des Raytracingmodells von vorn und (c) seitlich in der linken Bildhälfte (d) ursprüngliche LRF-Daten nach der Modellverbesserung ohne neue Raumpunkte und (e) nach der Modellverbesserung mit zusätzlichen rein photogrammetrisch rekonstruierten 3D-Punkten bei achtfacher Unterabtastung der multiplen Strahl-Dreieck-Schnittpunkte von vorn sowie (f) von der Seite in der linken Bildhälfte

6. Zusammenfassung und Ausblick

Die Auswertung der Messdaten hat ergeben, dass das in dieser Arbeit vorgestellte integrierte Matchingverfahren die Korrespondenzanalyse auf planaren und zylindrischen Stereoaufnahmen im Vergleich zur klassischen Übereinstimmungssuche entlang der Epipolarlinien stark beschleunigt. Der erreichte Zeitgewinn dürfte vor allem bei umfangreichem Bildmaterial den Aufwand für die Vorverarbeitungsschritte (Laserscanning, Registrierung/Vermaschung der 3D-Punkte, Bestimmung der relativen äußeren Orientierung), die sich partiell automatisieren lassen, überkompensieren. In absoluter Hinsicht müssen für die subpixelgenaue Übereinstimmungssuche auf realistischen Eingabedaten aber dennoch Laufzeiten im Bereich von Stunden bis zu mehreren Tagen veranschlagt werden.

Die Zuordnungsqualität erfährt durch die modellgestützte Eingrenzung des Suchraums gegenüber dem traditionellen Ansatz ebenfalls eine Steigerung. Dies gilt zumindest dann, wenn die Ausrichtung der Aufnahmegeräte zur Szene gewährleistet ist und die LRF-Erfassungsfehler primär in der Bildtiefe wirksam werden – eine Bedingung, welche die derzeit in der Entwicklung befindlichen kombinierten Erfassungseinheiten (LRF mit integriertem Stereokamerasystem) erfüllen. Eine ungenaue äußere Orientierung und überproportional ausgeprägte Modelldeformationen machen sowohl den Zeit- als auch Qualitätsgewinn, der durch den integrierten Matcher erzielt wird, teilweise zunichte und übertragen sich in Form von lokalen Ausreißern und eines Rauschteppichs auf die rekonstruierte Punktwolke.

Abgesehen vom Optimierungspotential bei den Speicherstrukturen und den sekundären 2D- und 3D-Algorithmen, das diese Arbeit an den entsprechenden Textstellen anspricht, existieren für die Beschleunigung des Matchings wie auch die qualitative Verbesserung der Korrespondenzen einige vielversprechende Fortentwicklungsansätze, deren Umsetzung aus Zeitgründen nicht mehr möglich war. Eine Option, die Ausführungsdauer der integrierten Zuordnungsbestimmung weiter zu reduzieren, besteht in der Implementierung des Korrelationsvorgangs mit rekursiven moving average-Filtern [11][52][69]. Dadurch wird die Laufzeit praktisch unabhängig von der eingestellten Muster- respektive Suchfenstergröße. Allerdings lassen sich nicht alle Ähnlichkeitskriterien mit dieser Methode realisieren, bei der das Resultat der Objektpunktschätzung mittels Raytracing wie bislang das designierte Suchbereichszentrum markiert. Für die SAD- und SSD-Funktion sollte eine Umsetzung prinzipiell jedoch machbar sein.

Die Qualität der Korrespondenzen lässt sich insbesondere bei der Verwendung von zylindrischen Stereoaufnahmen erhöhen, wenn die Form der jeweils miteinander zu korrelierenden Bildausschnitte an die projektive Verzerrung angepasst wird. Dabei stellt sich allerdings die Frage, wie man die von „ungeraden“ mathematischen Kurven begrenzten Muster- und Suchfenster implementierungstechnisch repräsentiert und effizient aufeinander abbildet. Die Anpassung der Gestalt der Bildareale, welche zeitraubende Interpolationsberechnungen erforderlich macht, kann abhängig von den Abmessungen die Laufzeit des Matchers merklich verlängern. Im Zweifelsfall gilt es unter Berücksichtigung des konkreten Anwen-

dungsszenarios abzuwägen, ob man der Güte der Zuordnungen und den damit rekonstruierten Raumpunkten oder der Geschwindigkeit der Korrespondenzsuche den Vorrang gibt.

A. Anhang

A.1. Künstliche Testwelt

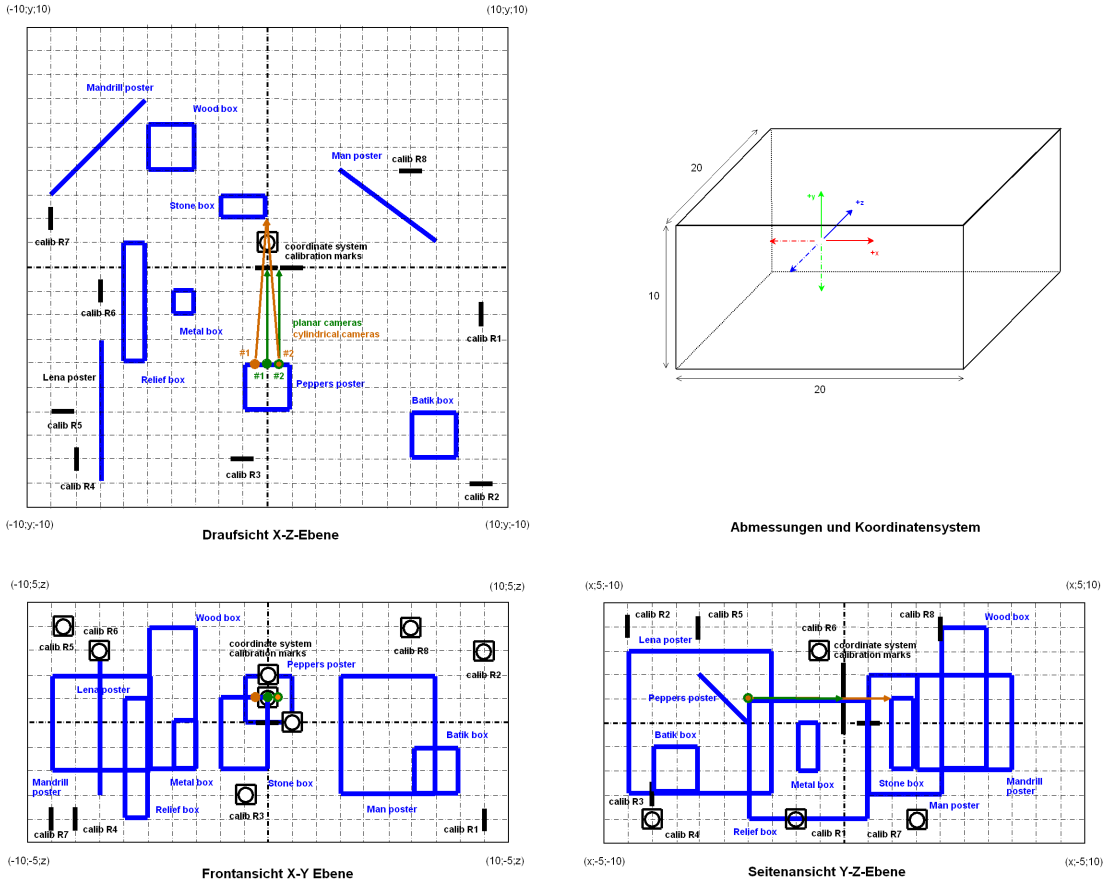
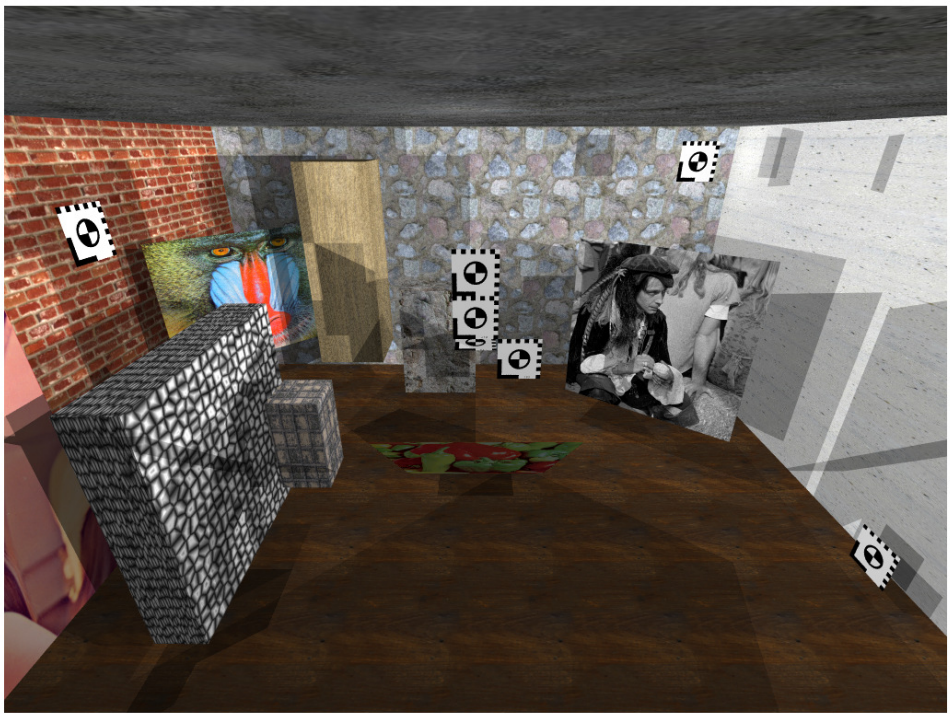


Abbildung A.1.: Karte der künstlichen Testwelt mit den Standorten und Blickrichtungen der Stereokameras bei der Evaluierung des Matchers

a)



b)

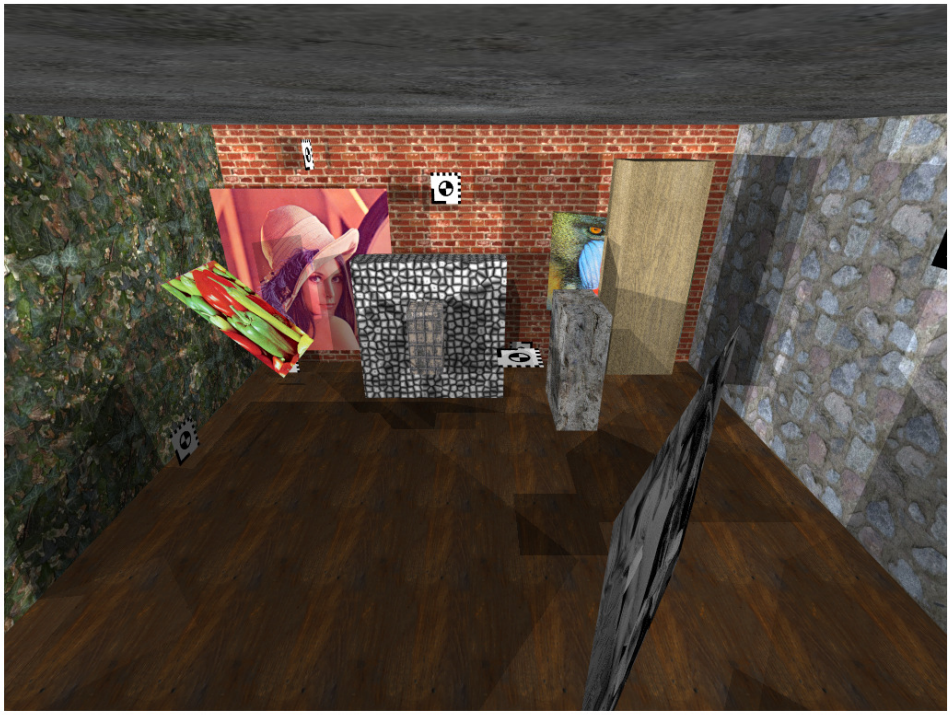
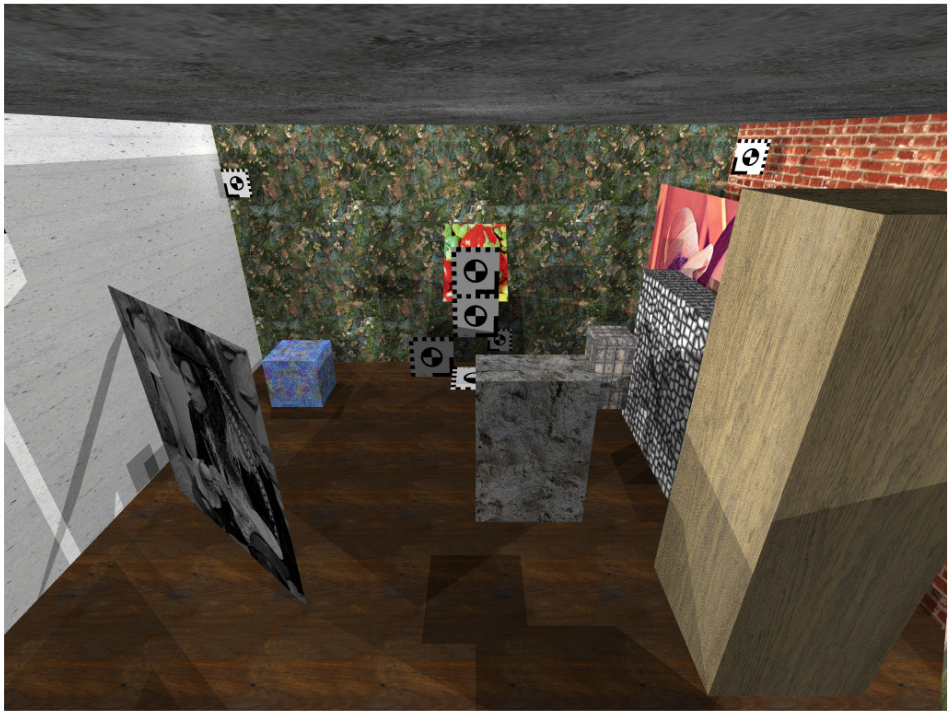


Abbildung A.2.: Darstellung der Testszene (a) von Position (0; 4,5; -9,5) und (b) von Position (9,5; 4,5; 0), jeweils mit Blick zum Weltkoordinatenursprung (0; 0; 0) im Zentrum

c)



d)

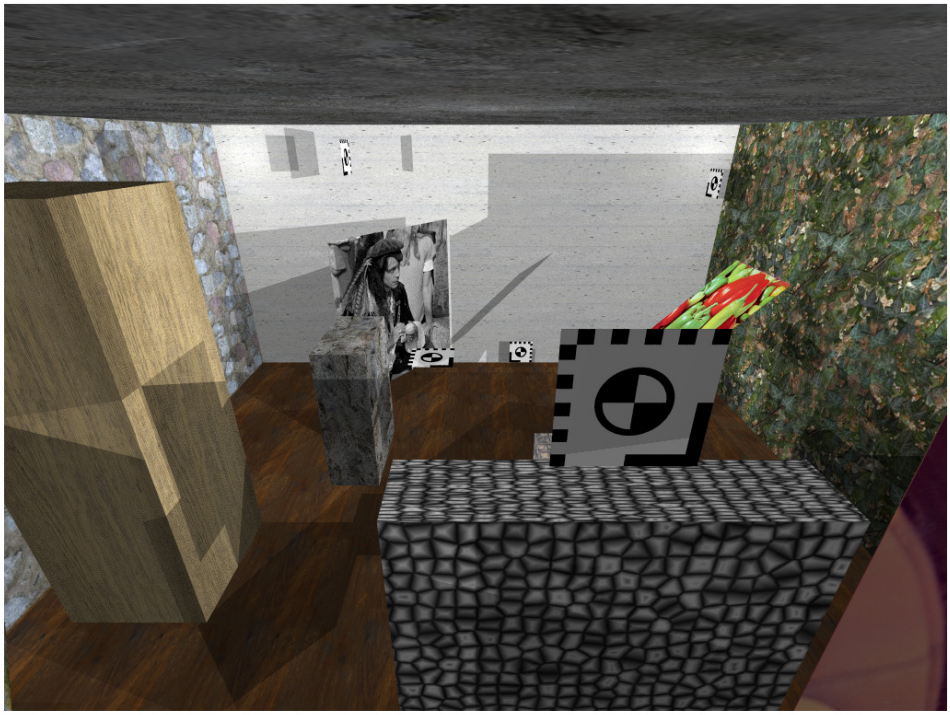


Abbildung A.2.: Darstellung der Testszene (Fortsetzung) (c) von Position (0; 4,5; 9,5) und (d) von Position (-9,5; 4,5; 0), jeweils mit Blick zum Weltkoordinatenursprung (0; 0; 0) im Zentrum

A.2. Struktur der VRML-Modellbeschreibungen

```
#VRML V2.0 utf8

# Shape 1 (=dataset 1)
Shape {

    appearance Appearance {
        # ImageTexture
        texture ImageTexture {url "C:/user/textureBitmap.png"}
    } # Appearance

    geometry IndexedFaceSet {

        coord Coordinate {
            point [
                # 3d points (vertices)
                0.5000000000  0.5000000000  0.0000000000   # (0; 0)
                0.5526315789  0.5000000000  0.0000000000   # (1; 0)
                0.6052631579  0.5000000000  0.0000000000   # (2; 0)
                ...
            ] # point
        } # Coordinate

        coordIndex [
            # surface triangles (indices to 3d point list above)
            0 20 21 -1
            0 21 1 -1
            1 21 22 -1
            ...
        ] # coordIndex

        texCoord TextureCoordinate {
            point [
                # normalized texture bitmap coordinates (0..1)
                0.0000000000 1.0000000000
                0.0526315789 1.0000000000
                0.1052631579 1.0000000000
                ...
            ] # point
        } # TextureCoordinate

    } # IndexedfaceSet
} # Shape 1


# Shape 2 (=dataset 2)
Shape {
    ...
} # Shape 2

    ...

# Shape n (=dataset 3)
Shape {
    ...
} # Shape n
```

Listing A.1: Struktur der VRML-Modellbeschreibungen

A.3. Struktur der XML-Konfigurationsdateien und Ausführungsprotokolle

```
<?xml version="1.0" standalone="no" ?>

<!-- DaTool configuration file -->
<Configuration>


    <!-- What's the configuration about? -->
    <ConfigurationType name="camera" version="1"/>


    <!-- The exterior orientation of the camera in the left-handed -->
    <!-- DaTool system (i.e. clockwise rotation -->
    <ExteriorOrientation>
        <Translation tx="-400" ty="-2550" tz="-500"/>
        <Rotation rx="2.9321" ry="1.0646" rz="2.9321"/>
    </ExteriorOrientation>


    <!-- The interior orientation of the camera -->
    <InteriorOrientation>
        <ProjectionType type="cylinder"/>
        <FocalLength f="0.06"/>
        <SensorDimension dimX="1024" dimY="614"/>
        <PixelDimension pixX="0.00007" pixY="0.00007"/>
        <PrincipalPoint px="511.5" py="306.5"/>
    </InteriorOrientation>


</Configuration>
```

Listing A.2: XML-Beschreibung der Kameraparameter mit innerer und äußerer Orientierung

```
<?xml version="1.0" standalone="no" ?>
<!-- DaTool configuration file -->
<Configuration>

    <!-- What's the configuration about? -->
    <ConfigurationType name="matcher" version="1"/>

    <!-- The raytracer settings -->
    <Raytracer>
        <Tree
            nrOfLevels="6"
            levelFactor="2"
            level0DimX="8"
            level0DimY="8"
            level0DimZ="8"
        />
        <TraceAlgorithm type="incremental"/>
    </Raytracer>
```

Listing A.3: XML-Beschreibung der Einstellungen für den integrierten Matcher

```
    <!-- The matcher settings -->
    <Matcher>

        <!-- The Region of interest (ROI) allows for subimage matching -->
            <!-- This tag is optional. If it is omitted, the whole image -->
        <!-- will be used. -->
        <ROI>
            <PatternImage
                startX="0"
                startY="0"
                dimX="1024"
                dimY="768"
            />
        </ROI>

        <!-- Algorithm settings -->
        <!-- Interpolation can be nn, triangle or cubic -->
        <!-- Channel can be 1 ... 3, or combine (weighted grayscale summing) -->
        <!-- The visibility check tag is optional, if omitted the test will -->
        <!-- be enabled -->
        <ModelAssistance enable="1" visibilityCheck="0"/>
        <EpipolarMatching enable="1"/>

        <Algorithm method="hybrid">

            <HybridStd
                enable                  = "1"
                metric                  = "sad"
                channel                 = "combine"

                searchDimX              = "9"
                searchDimY              = "9"
                pixelStepX              = "1"
                pixelStepY              = "1"
                corrDimX                = "17"
                corrDimY                = "17"
                searchStepRefinementX   = "1"
                searchStepRefinementY   = "1"

                gemanSigma              = "5.0"
            />

            <HybridPrecise
                enable                  = "1"
                channel                 = "combine"
                metric                  = "sad"
                interpolationFilter     = "triangle"

                searchDimX              = "3"
                searchDimY              = "3"
                pixelStepX              = "1"
                pixelStepY              = "1"
                corrDimX                = "17"
                corrDimY                = "17"
                searchStepRefinementX   = "2"
                searchStepRefinementY   = "5"

                gemanSigma              = "5.0"
            />
        </Algorithm>

        <!-- Result quality -->
        <Result qualityThreshold="0.25" />
    </Matcher>
</Configuration>
```

Listing A.3: XML-Beschreibung der Einstellungen für den integrierten Matcher (Fortsetzung)

```
<?xml version="1.0" standalone="no" ?>

<!-- DaTool configuration file -->
<Configuration>

    <!-- What's the configuration about? -->
    <ConfigurationType name="correction" version="1"/>

    <!-- The raytracer settings -->
    <Raytracer>
        <Tree
            nrOfLevels="6"
            levelFactor="2"
            level0DimX="8"
            level0DimY="8"
            level0DimZ="8"
        />
        <TraceAlgorithm type="recursive"/>
    </Raytracer>

    <Correction>

        <!-- The Region of interest (ROI) allows for subimage matching -->
        <!-- This tag is optional. If it is omitted, the whole image -->
        <!-- will be used. -->
        <ROI>
            <PatternImage
                startX="0"
                startY="0"
                dimX="1024"
                dimY="768"
            />
        </ROI>

        <Algorithm
            increasePointDensity         = "0"
            pointDensityIncSubsampling   = "1"

            interpolationFilter          = "cubic"
        />

    </Correction>
</Configuration>
```

Listing A.4: XML-Beschreibung der Parameter für den Modellverbesserungsalgorithmus

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<MatcherProtocol>
    <TreeClassification>
        <StartTime time="Tue Aug 21 22:04:24 2007&#x0A;" />
        <EndTime time="Tue Aug 21 21:04:43 2007&#x0A;" />
    </TreeClassification>
    <Matching>
        <StartTime time="Tue Aug 21 22:04:43 2007&#x0A;" />
        <EndTime time="Tue Aug 21 22:25:12 2007&#x0A;" />
        <Statistics>
            <BestPossibleSimilarity value="0" />
            <WorstPossibleSimilarity value="289" />
            <AvgSimilarityBeforeQC value="99.9478" />
            <AvgSimilarityAfterQC value="99.9478" />
            <AvgDisparityErrorToRTGTBeforeQC value="0.422068" />
            <AvgDisparityErrorToRTGTAfterQC value="0.422068" />
            <DisparityErrorToRTGTStdDeviationBeforeQC value="0.752941" />
            <DisparityErrorToRTGTStdDeviationAfterQC value="0.752941" />
            <RayHitsCamera1 value="761244" />
            <RayHitsCamera2 value="734795" />
            <ModelAssistedMatchingSteps value="734795" />
            <EpipolarMatchingSteps value="0" />
            <QualityMatches value="734795" />
            <MatchingErrors value="0" />
        </Statistics>
    </Matching>
    <Termination userAbort="0" />
</MatcherProtocol>
```

Listing A.5: XML-Ausführungsprotokoll des integrierten Matchers

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<CorrectionProtocol>
    <Raytracing>
        <StartTime time="Fri Nov 09 08:24:29 2007&#x0A;" />
        <EndTime time="Fri Nov 09 11:04:59 2007&#x0A;" />
    </Raytracing>
    <CreateRaytraceModel>
        <StartTime time="Fri Nov 09 11:04:59 2007&#x0A;" />
        <EndTime time="Fri Nov 09 11:05:01 2007&#x0A;" />
    </CreateRaytraceModel>
    <Correction>
        <StartTime time="Fri Nov 09 11:05:01 2007&#x0A;" />
        <EndTime time="Fri Nov 09 11:06:30 2007&#x0A;" />
    </Correction>
    <Statistics>
        <UniqueCorrectablePoints value="204492" />
        <CorrectedPoints value="152339" />
        <AdditionalPhotogrammetricPoints value="1002346" />
        <OutputPoints value="1154685" />
    </Statistics>
    <Termination userAbort="0" />
</CorrectionProtocol>
```

Listing A.6: XML-Ausführungsprotokoll des Modellverbesserungsalgorithmus

Abbildungsverzeichnis

2.1. Bilderfassung mit der digitalen Kamera 13

2.2. Panoramakamera KST EyeScan M3D metric 14

2.3. Planares Lochkameramodell 15

2.4. Zylindrisches Lochkameramodell 18

2.5. Stereogeometrie mit planaren Projektionsflächen 21

2.6. Schnitt der Epipolarebene mit der zylindrischen Bildfläche 23

2.7. Beispiel einer visualisierten horizontalen Disparitätskarte 26

2.8. Beispielkonfiguration zur Abschätzung der Auswirkung von fehlerhaften Punktkorrespondenzen 27

2.9. Zoller und Fröhlich Laserscanner IMAGER 5003 30

2.10. Erstellung eines Laserscans 31

2.11. Stufen der Modellerzeugung 34

3.1. Strahlunabhängige Modifikation der vorhandenen 3D-Daten 43

4.1. Bildschirmabzug der graphischen Benutzeroberfläche der DaTool-Anwendung 50

4.2. Bildzeilenwechsel bei Verwendung der blockorientierten Zugriffsfunktion . . 54

4.3. Verlagerung des Bezugspunkts für die Übereinstimmungsberechnung und Anpassung des Suchbereichs 61

4.4. GemanMcClure-Funktion 63

4.5. UML-Diagramm des Ein- und Ausgabesystems für die Bilddaten 67

4.6. Design der funktionalen Algorithmen als UML-Klassendiagramm 77

4.7. VRML- und DaTool-Koordinatensystem 79

4.8. Inkrementelles Raytracing und Bresenham-Linienrasterung 87

4.9. Distanzberechnung zur Auswahl der Nachbarbox beim inkrementellen Raytracing 88

4.10. Visualisierung des 3D-Baums und der inkrementellen Strahlverfolgung . . . 90

4.11. Verschiedene Objektschnittpunkte bei Verdeckungen 92

4.12. Verhältnisse beim Verschiebungsverfahren 99

4.13. Verdeckte Modellpunkte bei der Projektion zur Modellverbesserung 100

4.14. Konstruktion einer Viereckfläche durch den Testweltgenerator 106

5.1. Stereobildpaare der künstlichen Szene 109

5.2. Graustufen-Übersichtsbilder der Testläufe 5.1.1.1-a bis 5.1.1.1-e 117

5.3. Graustufen-Übersichtsbild von Test 5.1.1.2-g 126

5.4. Planare Graustufen-Übersichtsbilder der Testläufe 5.1.1.3-a bis 5.1.1.3-k . . 139

5.4. Planare Graustufen-Übersichtsbilder der Testläufe 5.1.1.3-a bis 5.1.1.3-k (Teil 2) 140

5.4. Planare Graustufen-Übersichtsbilder der Testläufe 5.1.1.3-a bis 5.1.1.3-k (Teil 3) 141

5.5. Zylindrische Graustufen-Übersichtsbilder der Testläufe 5.1.1.3-a bis 5.1.1.3-k 142

5.5. Zylindrische Graustufen-Übersichtsbilder der Testläufe 5.1.1.3-a bis 5.1.1.3-k (Teil 2) 143

5.6. Planare Graustufen-Übersichtsbilder der Testläufe 5.1.1.4-a bis 5.1.1.4-d . . 148

5.7. Deformiertes LRF-Modell aus Sicht des DaTool-Raytracers 149

5.8. Schätzung der Suchbereichsgröße für die Korrespondenzanalyse bei einem deformierten Modell 150

5.9. Graustufen-Übersichtsbilder der Testläufe 5.1.2.2-a bis 5.1.2.2-c 155

5.10. 3D-Punktwolken des deformierten Testraums vor und nach Anwendung des Modellverbesserungsalgorithmus 159

5.11. 3D-Modell der Altarkuppel der Wieskirche 160

5.12. ROIs aus den Stereopanoramen von der Altarkuppel der Wieskirche 161

5.13. Graustufen-Übersichtsbilder von der Altarkuppel der Wieskirche nach dem Matching 164

5.14. Ausgewählte Störungen in den Aufnahmen von der Altarkuppel der Wieskirche 165

5.15. 3D-Punktwolken von der Altarkuppel der Wieskirche vor und nach Anwendung des Verbesserungsverfahrens 168

5.16. 3D-Punktwolke der Tierfigur in der Altarkuppel der Wieskirche vor und nach Anwendung des Verbesserungsverfahrens 169

A.1. Karte der künstlichen Testwelt 172

A.2. Darstellung der Testszene 173

A.2. Darstellung der Testszene (Teil 2) 174

Tabellenverzeichnis

2.1. Technische Daten der Panoramakamera EyeScan M3D metric 14

2.2. Technische Daten des IMAGER 5003 mit Laser LARA 53500 30

5.1. Eigenschaften der für die Evaluierung verwendeten künstlichen 3D-Modelle 108

5.2. Kameraparameter für die Erzeugung der Stereoaufnahmen von der künstlichen Testwelt 108

5.3. Parameter des integrierten Matchingalgorithmus für die Eruierung der Projektionsarten, Speichermethoden und Zuordnungsalgorithmen 110

5.4. Messdaten für Test 5.1.1.1-a 112

5.5. Messdaten für Test 5.1.1.1-b 113

5.6. Messdaten für Test 5.1.1.1-c 113

5.7. Messdaten für Test 5.1.1.1-d 114

5.8. Messdaten für Test 5.1.1.1-e 114

5.9. Messdaten für Test 5.1.1.2-a 119

5.10. Messdaten für Test 5.1.1.2-b 120

5.11. Messdaten für Test 5.1.1.2-c 120

5.12. Messdaten für Test 5.1.1.2-d 121

5.13. Messdaten für Test 5.1.1.2-e 121

5.14. Messdaten für Test 5.1.1.2-f 122

5.15. Messdaten für Test 5.1.1.2-g 122

5.16. Messdaten für Test 5.1.1.2-h 123

5.17. Messdaten für Test 5.1.1.2-i 123

5.18. Messdaten für Test 5.1.1.2-j 124

5.19. Parameter für die Korrespondenzanalyse in Test 5.1.1.3-a 127

5.20. Modifizierte Matchingparameter für Test 5.1.1.3-g 128

5.23. Messdaten für Test 5.1.1.3-a 129

5.21. Modifizierte Matchingparameter für Test 5.1.1.3-h 130

5.22. Modifizierte Matchingparameter für Test 5.1.1.3-i 130

5.24. Messdaten für Test 5.1.1.3-b 131

5.25. Messdaten für Test 5.1.1.3-c 131

5.26. Messdaten für Test 5.1.1.3-d 132

5.27. Messdaten für Test 5.1.1.3-e 132

5.28. Messdaten für Test 5.1.1.3-f 133

5.29. Messdaten für Test 5.1.1.3-g 133

5.30. Messdaten für Test 5.1.1.3-h 134

5.31. Messdaten für Test 5.1.1.3-i 135

5.32. Messdaten für Test 5.1.1.3-j 135

5.33. Messdaten für Test 5.1.1.3-k 136

5.34. Geänderte äußere Kameraorientierung für Test 5.1.1.4-a 143

5.35. Geänderte äußere Kameraorientierung für Test 5.1.1.4-b 144

5.36. Messdaten für Test 5.1.1.4-a 145

5.37. Messdaten für Test 5.1.1.4-b 145

5.38. Messdaten für Test 5.1.1.4-c 145

5.39. Messdaten für Test 5.1.1.4-d 146

5.40. Parameter für die Modelldeformation zur Evaluierung des Modellverbesserungsalgorithmus 147

5.41. Parameter für den integrierten Matchingalgorithmus in Test 5.1.2.2-a 152

5.42. Messdaten für Test 5.1.2.2-a 153

5.43. Messdaten für Test 5.1.2.2-b 153

5.44. Messdaten für Test 5.1.2.2-c 154

5.45. Messdaten für Test 5.1.2.3-a 157

5.46. Messdaten für Test 5.1.2.3-b 157

5.47. Messdaten für Test 5.1.2.3-c 157

5.48. Messdaten für Test 5.1.2.3-d 157

5.49. Eigenschaften des 3D-Modells von der Altarkuppel der Wieskirche 160

5.50. Kameraparameter zu den photographischen Aufnahmen der Altarkuppel der Wieskirche 162

5.52. Messdaten nach dem Matching mit dem Modell und den Stereoaufnahmen von der Altarkuppel der Wieskirche 162

5.51. Matchingparameter für die Korrespondenzanalyse auf den Panoramen der Wieskirche 163

5.53. Messdaten für Test 5.2.2-a 166

5.54. Messdaten für Test 5.2.2-b 166

Listings

4.1. Beispielhafter Gebrauch der Ausgaben von `CLMArea` 60
4.2. Verschiebung des Musterfensters über den Suchbereich in der Klasse `COne-PassMatcherPrecise` 66
4.3. Integrierter Matchingalgorithmus im Pseudocode 94
4.4. Projektion der Modellpunkte bei der Modellverbesserung 103

A.1. Struktur der VRML-Modellbeschreibungen 175
A.2. XML-Beschreibung der Kameraparameter 176
A.3. XML-Beschreibung der Einstellungen für den integrierten Matcher 176
A.3. XML-Beschreibung der Einstellungen für den integrierten Matcher (Teil 2) 177
A.4. XML-Beschreibung der Parameter für den Modellverbesserungsalgorithmus 178
A.5. XML-Ausführungsprotokoll des integrierten Matchers 179
A.6. XML-Ausführungsprotokoll des Modellverbesserungsalgorithmus 179

Literaturverzeichnis

[1] Paul R. Wolf, Bon A. DeWitt: *Elements of Photogrammetry with Applications in GIS*, 3rd Revision, McGraw-Hill Publishing Co., Boston, 2000

[2] D. Göhring: *Digitalkameratechnologien - Eine vergleichende Betrachtung*, Ausarbeitung, Institut für Informatik der Humboldt-Universität Berlin, Berlin 2002

[3] KST GmbH: *Eyescan M3D metric*, Datenblatt, Kamera & System Technik GmbH, Pirna 2005

[4] R. Hartley, A. Zisserman: *Multiple View Geometry in Computer Vision*, 2nd edition, Cambridge University Press, Cambridge 2003

[5] L. McMillan. G. Bishop: *Plenoptic Modelling: An Image-Based Rendering System*, Proceedings of SIGGRAPH 95, pages 39-46, Los Angeles 1995

[6] F. Huang, S. K. Wei, R. Klette, G. Gimelfarb, R. Reulke, M. Scheele, K. Scheibe: *Cylindrical Panoramic Cameras - From Basic Design To Applications*, Technical Report No. 116, Centre for Image Technology and Robotics, University of Auckland, Auckland 2002

[7] O. Schreer: *Stereoanalyse und Bildsynthese*, Springer-Verlag, Berlin 2005

[8] N. Paragios, Y. Chen, O. Faugeras: *Handbook of Mathematical Models in Computer Vision*, Springer Science+Business Media Inc., New York, 2006

[9] D. A. Forsyth, J. Ponce: *Computer Vision: A Modern Approach*, Prentice Hall, New Jersey 2003

[10] K. Scheibe: *Design and Test of Algorithms for the Evaluation of Modern Sensors in Close-Range Photogrammetry*, Dissertation, Mathematische Fakultät der Georg-August Universität Göttingen, Göttingen 2006

[11] S. W. Smith: *The Scientist and Engineer's Guide to Digital Signal Processing*, 2nd Edition, California Technical Publishing, San Diego 1997

[12] J. Banks, P. Corke: *Quantitative Evaluation of Matching Methods and Validity Measures for Stereo Vision*, The International Journal of Robotics Research, Vol. 20, No. 7, pages 512-523, 2001

[13] W. Korth: *WP Laserscanning*, Vorlesungsskript Wahlpflichtveranstaltung Laserscanning, Version vom 04.04.2007, Technische Fachhochschule Berlin, Berlin 2007

[14] Zoller+Fröhlich GmbH: *Technische Daten IMAGER*, Datenblatt zum Laserscanner IMAGER 5003, Zoller+Fröhlich GmbH, Wangen 2005

[15] T. Schulz, H. Ingensand: *Laserscanning - Genauigkeitsbetrachtungen und Anwendungen*, In T. Luhmann (Hrsg.) Photogrammetrie Laserscanning Optische 3D-Messtechnik, Beiträge der Oldenburger 3D-Tage 2004, Herbert Wichmann Verlag, Heidelberg 2004

[16] T. Schulz, H. Ingensand: *Influencing Variables, Precision and Accuracy of Terrestrial Laser Scanners*, INGEO 2004 and FIG Regional Central and Eastern European Conference on Engineering Surveying Proceedings, Bratislava 2004

[17] P. J. Besl, N. D. McKay: *A Method for Registration of 3D Shapes*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 14, No. 2, pages 239-256, 1992

[18] S. Rusinkiewicz, M. Levoy: *Efficient Variants of the ICP Algorithm*, 3rd International Conference on 3D Digital Imaging and Modeling Proceedings, pages 145-152, 2001

[19] N. Amenta, M. Bern, M. Kamvysselis: *A New Voronoi-Based Surface Reconstruction Algorithm*, Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques, pages 415-421, Association for Computing Machinery ACM, New York 1998

[20] N. Amenta, S. Choi, T. K. Dey, N. Leekha: *A Simple Algorithm for Homeomorphic Surface Reconstruction*, Proceedings of the 16th Annual Symposium on Computational Geometry, Association for Computing Machinery ACM, New York 2000

[21] N. Amenta, S. Choi, R. Kolluri: *The Power Crust*, Proceedings of the 6th ACM Symposium on Solid Modeling and Applications, pages 249-260, Association for Computing Machinery ACM, New York 2001

[22] W. Kurth: *Computergrafik*, Kapitel 13 des Skripts zur Vorlesung im Wintersemester 2003/04, Brandenburgische Technische Universität Cottbus, Cottbus 2003

[23] H. Moravec: *Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover*, Technical Report CMU-RI-TR-80-03, Robotics Institute, Carnegie Mellon University & doctoral dissertation, Stanford University, 1980

[24] C. Harris, M. Stephens: *A Combined Corner and Edge Detector*, Proceedings of The Fourth Alvey Vision Conference, pages 147-151, Manchester 1988

[25] J. Shi, C. Tomasi: *Good Features to Track*, IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'94) Proceedings, pages 593-600, IEEE Computer Society, Seattle 1994

[26] D. G. Lowe: *Distinctive Image Features from Scale-Invariant Keypoints*, International Journal of Computer Vision, Vol. 60, Nr. 2, pages 91-110, 2004

[27] B. Stroustrup: *Die C++-Programmiersprache*, 3. aktualisierte und erweiterte Auflage, Addison-Wesley-Longman, Bonn 1998

[28] Microsoft Corporation: *Microsoft Developer Network*, Homepage des MSDN mit Informationen zum Visual Studio

`http://msdn2.microsoft.com/de-de/default.aspx`
(Stand vom 10.08.2007)

[29] GnuWin32-Projekt: *Flex for Windows*, Homepage des flex-Scannergenerators für Windows-Betriebssysteme

`http://gnuwin32.sourceforge.net/packages/flex.htm`
(Stand vom 10.08.2007)

[30] GnuWin32-Projekt: *Bison for Windows*, Homepage des bison-Parsergenerators für Windows-Betriebssysteme

`http://gnuwin32.sourceforge.net/packages/bison.htm`
(Stand vom 10.08.2007)

[31] Lee Thomason: *TinyXML*, Homepage der TinyXML-Bibliothek

`http://www.grinninglizard.com/tinyxml`
(Stand vom 20.08.2007)

[32] Trolltech: *Trolltech Homepage*, Homepage des Herstellers der Qt-GUI-Bibliothek

`http://trolltech.com`
(Stand vom 10.08.2007)

[33] Q.../Free-Projekt: *Q.../Free*, Homepage des Q.../Free-Projekts mit dem Advanced Compiler Support-Paket für die Qt-Bibliothek

`http://sourceforge.net/projects/qtwin`
(Stand vom 10.08.2007)

[34] R. Dementiev, J. Singler, A. Beckmann: *STXXL: Standard Template Library for Extra Large Data Sets*, Homepage der STXXL-Bibliothek

`http://stxxl.sourceforge.net/`
(Stand vom 17.08.2007)

[35] R. Dementiev: *Documentation for STXXL library 1.1.0*, Quelltextdokumentation der STXXL-Bibliothek Version 1.1.0 (liegt selbiger bei, vgl. [34])

[36] R. Dementiev, L. Kettner, P. Sanders: *STXXL: Standard Template Library for XXL Data Sets*, preprint, to appear in Software: Practice and Experience journal, John Wiley & Sons Ltd., Chichester 2007

[37] Boost-Projekt: *Boost C++ Libraries*, Homepage der Boost-Bibliothek

`http://www.boost.org/`
(Stand vom 22.08.2007)

[38] D. Shreiner, M. Woo, J. Neider, T. Davis: *OpenGL Programming Guide: The Official Guide to Learning OpenGL Version 2*, 5th Edition, Addison-Wesley Professional, 2005

[39] B. Henderson-Sellers: *Object-Oriented Metrics - Measures of Complexity*, Prentice

Hall PTR, New Jersey 1996

[40] International Organization for Standardization: *Programming Languages – C*, ISO/IEC 9899:TC2 Committee Draft Standard WG14/N1124, 2005

[41] International Electrotechnical Commission: *International Standard IEC 60027-2 – Letter Symbols to be used in Electrical Technology*, 3rd edition, IEC, Genf 2005

[42] International Telecommunication Union: *Information technology – Open Systems Interconnection – Procedures for the Operation of OSI Registration Authorities: Generation and Registration of Universally Unique Identifiers (UUIDs) and their use as ASN.1 Object Identifier Components*, ITU-T recommendation X.667, Genf 2004

[43] D. E. Knuth: *The Art of Computer Programming Volume 2 – Seminumerical Algorithms*, 2nd Edition, Addison-Wesley Series in Computer Science and Information Processing, Reading 1981

[44] E. Gamma, R. Helm, R. Johnson, J. Vlissides (Gang of Four): *Design Patterns CD - Elements of Reusable Object-Oriented Software*, Addison-Wesley-Longman, 1998

[45] Intel Corporation: *Intel 64 and IA-32 Architectures Software Developer's Manual Volume 3A - System Programming Guide Part 1*, Intel Corporation, Santa Clara 2007

[46] A. Fog: *The Microarchitecture of Intel and AMD CPU's - An Optimization Guide for Assembly Programmers and Compiler Makers*, Version vom 29.07.2007, Copenhagen University College of Engineering, Kopenhagen 2007

[47] A. Fog: *Instruction Tables - Lists of Instruction Latencies, Throughputs and Microoperation Breakdowns for Intel and AMD CPU's*, Version vom 06.05.2007, Copenhagen University College of Engineering, Kopenhagen 2007

[48] C. Poynton: *Frequently Asked Questions about Color*, Toronto 1997

`http://www.poynton.com/ColorFAQ.html`
(Stand vom 05.08.2007)

[49] A. V. Oppenheim, A. S. Willsky: *Signals and Systems*, Prentice Hall, New Jersey 1983

[50] T. M. Lehmann, C. Gönner, K. Spitzer: *Survey: Interpolation Methods in Medical Image Processing*, IEEE Transactions On Medical Imaging, Vol. 18, No. 11, pages 1046-1075, 1999

[51] N. C. Myers: *A New and Useful Template Technique: "Traits"*, in S. B. Lippman (editor): C++ Gems, pages 451-457, Cambridge University Press, New York 1998

[52] M. Gerrits, P. Bekaert: *Local Stereo Matching with Segmentation-based Outlier Rejection*, Proceedings of the The 3rd Canadian Conference on Computer and Robot Vision (CRV'06) - Volume 00, page 66, IEEE Computer Society, Washington 2006

[53] M. Fowler: *UML konzentriert: Eine kompakte Einführung in die Standard-Objektmodellierungssprache*, Addison-Wesley, München 2004

[54] Adobe Developers Association: *TIFF Revision 6.0*, Finale Version vom 3. Juni 1992, Adobe Developers Association, Adobe Systems Incorparated, Mountain View 1992

[55] LibTIFF-Projekt: *LibTIFF - TIFF Library and Utilities*, Homepage der LibTIFF-Bibliothek

`http://www.remotesensing.org/libtiff`
(Stand vom 13.08.2007)

[56] Web3D Consortium: *ISO/IEC 14772-1:1997 and ISO/IEC 14772-2:2004 – Virtual Reality Modeling Language (VRML)*, VRML-Spezifikation von 2003 online

`http://www.web3d.org/x3d/specifications/vrml/ISO-IEC-14772-VRML97`
(Stand vom 15.08.2007)

[57] D. Frommholz: *Neugestaltung des VHDL-Lademoduls und Konzeption eines Simulatorkerns für das Fehlerinjektionswerkzeug FIT zur Unterstützung bidirektionaler Busstrukturen*, Studienarbeit, Brandenburgische Technische Universität Cottbus, Cottbus 2006

[58] A. V. Aho, R. Sethi, J D. Ullman: *Compilers, Principles, Techniques and Tools*, Addison-Wesley, Reading 1986

[59] A. Isotton: *C++ dlopen mini HOWTO*, Anleitung zum dynamischen Laden von C++-Klassen, Revision 1.02

`http://www.faqs.org/docs/Linux-mini/C++-dlopen.html`
(Stand vom 13.08.2007)

[60] B. v. Braunmühl: *Kleines Skript „Effiziente Algorithmen“*, Skript zur gleichnamigen Vorlesung im Wintersemester 2001/02, Brandenburgische Technische Universität Cottbus, Cottbus 2001

[61] T. Akenine-Möller: *Fast 3D Triangle-Box Overlap Testing*, Journal of Graphics Tools, Vol. 6, No. 1, pages 29-33, 2001

[62] T. Möller, B. Trumbore: *Fast, Minimum Storage Ray-Triangle Intersection*, Journal of Graphics Tools, Vol. 2, No. 1, pages 21-28, 1997

[63] J. E. Bresenham: *Algorithm for Computer Control of a Digital Plotter*, IBM Systems Journal, Vol. 4, Issue 1, pages 25-30, 1965

[64] E. W. Weisstein: *Point-Line Distance – 3-Dimensional*, aus MathWorld – A Wolfram Web Resource

`http://mathworld.wolfram.com/Point-LineDistance3-Dimensional.html`
(Stand vom 20.08.2007)

[65] M. Hoemmen: *Computing the Standard Deviation Efficiently*, Parallel Programming for Multicore Course Tutorial, Electrical Engineering and Computer Science

Department, University of California, Berkeley 2007

[66] G. Cramer: *Introduction à L'Analyse des Lignes Courbes Algébriques*, Seiten 657-659, Genf 1750

[67] Persistence of Vision Raytracer Pty. Ltd.: *Introduction to POV-Ray*, Persistence of Vision Raytracer Pty. Ltd. Australia, Victoria 2007

`http://www.povray.org/redirect/www.povray.org/ftp/pub/povray/Official/Documentation/povdoc-3.6.1-a4-pdf.zip`
(Stand vom 16.08.2007)

[68] c't magazin für computertechnik: *H2bench für Windows V3.12*, Benchmarksoftware für Festplatten, Heise Zeitschriften Verlag GmbH, Hannover 2007

`http://www.heise.de/software/download/h2bench_h2benchw/3787`
(Stand vom 17.11.2007)

[69] O. Faugeras, B. Hotz, H. Mathieu u.v.m: *Real Time Correlation-based Stereo: Algorithm, Implementations and Applications*, Programme 4 – Robotique, Image et Vision, Projet Robotvis, Rapport de recherche de l'INRIA No. 2013, Sophia-Antipolis 1993